Loop freedom of the (untimed) AODV routing protocol

Timothy Bourke\textsuperscript{1} 	 Peter Höfner\textsuperscript{2}

April 18, 2020

\textsuperscript{1}Inria, École normale supérieure, and NICTA
\textsuperscript{2}NICTA and Computer Science and Engineering, UNSW
Abstract

The Ad hoc On-demand Distance Vector (AODV) routing protocol [6] allows the nodes in a Mobile Ad hoc Network (MANET) or a Wireless Mesh Network (WMN) to know where to forward data packets. Such a protocol is ‘loop free’ if it never leads to routing decisions that forward packets in circles.

This development mechanises an existing pen-and-paper proof of loop freedom of AODV [4]. The protocol is modelled in the Algebra of Wireless Networks (AWN), which is the subject of an earlier paper [3] and mechanization [1]. The proof relies on a novel compositional approach for lifting invariants to networks of nodes.

We exploit the mechanization to analyse several variants of AODV and show that Isabelle/HOL can re-establish most proof obligations automatically and identify exactly the steps that are no longer valid. Each of the variants is essentially a modified copy of the main development.

Further documentation is available in [2].
## Contents

0.1 Basic data types and constants ................................................. 5
0.2 Predicates and functions used in the AODV model ............................ 5
  0.2.1 Sequence Numbers .................................................. 5
  0.2.2 Modelling Routes .................................................. 6
  0.2.3 Routing Tables ..................................................... 6
  0.2.4 Updating Routing Tables ........................................... 9
  0.2.5 Route Requests .................................................... 19
  0.2.6 Queued Packets ................................................... 19
  0.2.7 Comparison with the original technical report ......................... 20
0.3 AODV protocol messages .................................................. 20
0.4 The AODV protocol ..................................................... 21
  0.4.1 Data state ........................................................ 22
  0.4.2 Auxilliary message handling definitions ............................. 23
  0.4.3 The protocol process ............................................. 24
0.5 Invariant assumptions and properties ..................................... 30
0.6 Quality relations between routes .......................................... 32
  0.6.1 Net sequence numbers ............................................. 32
  0.6.2 Comparing routes ................................................ 34
  0.6.3 Comparing routing tables ....................................... 35
  0.6.4 Strictly comparing routing tables ................................ 40
0.7 Invariant proofs on individual processes ................................ 44
0.8 The quality increases predicate .......................................... 54
0.9 The ‘open’ AODV model .................................................. 61
0.10 Global invariant proofs over sequential processes ......................... 61
0.11 Routing graphs and loop freedom ....................................... 79
0.12 Lift and transfer invariants to show loop freedom ....................... 81
  0.12.1 Lift to parallel processes with queues .......................... 81
  0.12.2 Lift to nodes ..................................................... 83
  0.12.3 Lift to partial networks ....................................... 84
  0.12.4 Lift to closed networks ....................................... 84
  0.12.5 Transfer into the standard model ................................ 85
  0.12.6 Loop freedom of AODV .......................................... 87

1 Variant A: Skipping the RREQ ID ........................................ 88
  1.1 Predicates and functions used in the AODV model ...................... 88
  1.1.1 Sequence Numbers ............................................... 88
  1.1.2 Modelling Routes ............................................... 88
  1.1.3 Routing Tables .................................................. 89
  1.1.4 Updating Routing Tables ....................................... 92
  1.1.5 Route Requests .................................................. 101
  1.1.6 Queued Packets ................................................ 102
  1.1.7 Comparison with the original technical report ..................... 103
  1.2 AODV protocol messages ............................................ 103
  1.3 The AODV protocol .................................................. 104
  1.3.1 Data state ....................................................... 104
  1.3.2 Auxilliary message handling definitions .......................... 105
  1.3.3 The protocol process .......................................... 107
2 Variant B: Forwarding the Route Reply

2.1 Predicates and functions used in the AODV model
   2.1.1 Sequence Numbers
   2.1.2 Modelling Routes
   2.1.3 Routing Tables
   2.1.4 Updating Routing Tables
   2.1.5 Route Requests
   2.1.6 Queued Packets
   2.1.7 Comparison with the original technical report

2.2 AODV protocol messages

2.3 The AODV protocol
   2.3.1 Data state
   2.3.2 Auxiliary message handling definitions
   2.3.3 The protocol process

2.4 Invariant assumptions and properties

2.5 Quality relations between routes
   2.5.1 Net sequence numbers
   2.5.2 Comparing routes
   2.5.3 Comparing routing tables
   2.5.4 Strictly comparing routing tables

2.6 Invariant proofs on individual processes

2.7 The quality increases predicate

2.8 The ‘open’ AODV model

2.9 Global invariant proofs over sequential processes

2.10 Routing graphs and loop freedom

2.11 Lift and transfer invariants to show loop freedom
   2.11.1 Lift to nodes
   2.11.2 Lift to partial networks
   2.11.3 Lift to closed networks
   2.11.4 Transfer into the standard model
   2.11.5 Loop freedom of AODV

3 Variant C: From Groupcast to Broadcast

3.1 Predicates and functions used in the AODV model
   3.1.1 Sequence Numbers
   3.1.2 Modelling Routes
   3.1.3 Routing Tables
   3.1.4 Updating Routing Tables
<table>
<thead>
<tr>
<th>Section</th>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.5</td>
<td>Route Requests</td>
<td>265</td>
</tr>
<tr>
<td>3.1.6</td>
<td>Queued Packets</td>
<td>265</td>
</tr>
<tr>
<td>3.1.7</td>
<td>Comparison with the original technical report</td>
<td>266</td>
</tr>
<tr>
<td>3.2</td>
<td>AODV protocol messages</td>
<td>266</td>
</tr>
<tr>
<td>3.3</td>
<td>The AODV protocol</td>
<td>267</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Data state</td>
<td>267</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Auxiliary message handling definitions</td>
<td>268</td>
</tr>
<tr>
<td>3.3.3</td>
<td>The protocol process</td>
<td>270</td>
</tr>
<tr>
<td>3.4</td>
<td>Invariant assumptions and properties</td>
<td>275</td>
</tr>
<tr>
<td>3.5</td>
<td>Quality relations between routes</td>
<td>277</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Net sequence numbers</td>
<td>277</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Comparing routes</td>
<td>279</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Comparing routing tables</td>
<td>280</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Strictly comparing routing tables</td>
<td>285</td>
</tr>
<tr>
<td>3.6</td>
<td>Invariant proofs on individual processes</td>
<td>289</td>
</tr>
<tr>
<td>3.7</td>
<td>The quality increases predicate</td>
<td>298</td>
</tr>
<tr>
<td>3.8</td>
<td>The ‘open’ AODV model</td>
<td>305</td>
</tr>
<tr>
<td>3.9</td>
<td>Global invariant proofs over sequential processes</td>
<td>306</td>
</tr>
<tr>
<td>3.10</td>
<td>Routing graphs and loop freedom</td>
<td>323</td>
</tr>
<tr>
<td>3.11</td>
<td>Lift and transfer invariants to show loop freedom</td>
<td>325</td>
</tr>
<tr>
<td>3.11.1</td>
<td>Lift to parallel processes with queues</td>
<td>325</td>
</tr>
<tr>
<td>3.11.2</td>
<td>Lift to nodes</td>
<td>327</td>
</tr>
<tr>
<td>3.11.3</td>
<td>Lift to partial networks</td>
<td>328</td>
</tr>
<tr>
<td>3.11.4</td>
<td>Lift to closed networks</td>
<td>329</td>
</tr>
<tr>
<td>3.11.5</td>
<td>Transfer into the standard model</td>
<td>329</td>
</tr>
<tr>
<td>3.11.6</td>
<td>Loop freedom of AODV</td>
<td>331</td>
</tr>
<tr>
<td>4</td>
<td>Variant D: Forwarding the Route Request</td>
<td>332</td>
</tr>
<tr>
<td>4.1</td>
<td>Predicates and functions used in the AODV model</td>
<td>332</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Sequence Numbers</td>
<td>332</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Modelling Routes</td>
<td>332</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Routing Tables</td>
<td>333</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Updating Routing Tables</td>
<td>336</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Route Requests</td>
<td>346</td>
</tr>
<tr>
<td>4.1.6</td>
<td>Queued Packets</td>
<td>346</td>
</tr>
<tr>
<td>4.1.7</td>
<td>Comparison with the original technical report</td>
<td>347</td>
</tr>
<tr>
<td>4.2</td>
<td>AODV protocol messages</td>
<td>347</td>
</tr>
<tr>
<td>4.3</td>
<td>The AODV protocol</td>
<td>348</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Data state</td>
<td>348</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Auxiliary message handling definitions</td>
<td>350</td>
</tr>
<tr>
<td>4.3.3</td>
<td>The protocol process</td>
<td>351</td>
</tr>
<tr>
<td>4.4</td>
<td>Invariant assumptions and properties</td>
<td>357</td>
</tr>
<tr>
<td>4.5</td>
<td>Quality relations between routes</td>
<td>359</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Net sequence numbers</td>
<td>359</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Comparing routes</td>
<td>361</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Comparing routing tables</td>
<td>362</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Strictly comparing routing tables</td>
<td>367</td>
</tr>
<tr>
<td>4.6</td>
<td>Invariant proofs on individual processes</td>
<td>371</td>
</tr>
<tr>
<td>4.7</td>
<td>The quality increases predicate</td>
<td>381</td>
</tr>
<tr>
<td>4.8</td>
<td>The ‘open’ AODV model</td>
<td>388</td>
</tr>
<tr>
<td>4.9</td>
<td>Global invariant proofs over sequential processes</td>
<td>389</td>
</tr>
<tr>
<td>4.10</td>
<td>Routing graphs and loop freedom</td>
<td>407</td>
</tr>
<tr>
<td>4.11</td>
<td>Lift and transfer invariants to show loop freedom</td>
<td>409</td>
</tr>
<tr>
<td>4.11.1</td>
<td>Lift to parallel processes with queues</td>
<td>409</td>
</tr>
<tr>
<td>4.11.2</td>
<td>Lift to nodes</td>
<td>410</td>
</tr>
<tr>
<td>4.11.3</td>
<td>Lift to partial networks</td>
<td>411</td>
</tr>
<tr>
<td>4.11.4</td>
<td>Lift to closed networks</td>
<td>412</td>
</tr>
</tbody>
</table>
5 Variants A–D: All proposed modifications

5.1 Predicates and functions used in the AODV model

5.1.1 Sequence Numbers

5.1.2 Modelling Routes

5.1.3 Routing Tables

5.1.4 Updating Routing Tables

5.1.5 Route Requests

5.1.6 Queued Packets

5.1.7 Comparison with the original technical report

5.2 AODV protocol messages

5.3 The AODV protocol

5.3.1 Data state

5.3.2 Auxilliary message handling definitions

5.3.3 The protocol process

5.4 Invariant assumptions and properties

5.5 Quality relations between routes

5.5.1 Net sequence numbers

5.5.2 Comparing routes

5.5.3 Comparing routing tables

5.5.4 Strictly comparing routing tables

5.6 Invariant proofs on individual processes

5.7 The quality increases predicate

5.8 The ‘open’ AODV model

5.9 Global invariant proofs over sequential processes

5.10 Routing graphs and loop freedom

5.11 Lift and transfer invariants to show loop freedom

5.11.1 Lift to parallel processes with queues

5.11.2 Lift to nodes

5.11.3 Lift to partial networks

5.11.4 Lift to closed networks

5.11.5 Transfer into the standard model

5.11.6 Loop freedom of AODV
0.1 Basic data types and constants

theory Aodv_Basic
imports Main AWN.AWN_SOS
begin

These definitions are shared with all variants.

type_synonym rreqid = nat

type_synonym sqn = nat
datatype k = Known | Unknown
abbreviation kno where "kno ≡ Known"
abbreviation unk where "unk ≡ Unknown"
datatype p = NoRequestRequired | RequestRequired
abbreviation noreq where "noreq ≡ NoRequestRequired"
abbreviation req where "req ≡ RequestRequired"
datatype f = Valid | Invalid
abbreviation val where "val ≡ Valid"
abbreviation inv where "inv ≡ Invalid"

lemma not_ks [simp]:
  "(x ≠ kno) = (x = unk)"
  "(x ≠ unk) = (x = kno)"
  by (cases x,clarsimp+)

lemma not_ps [simp]:
  "(x ≠ noreq) = (x = req)"
  "(x ≠ req) = (x = noreq)"
  by (cases x,clarsimp+)

lemma not_ffs [simp]:
  "(x ≠ val) = (x = inv)"
  "(x ≠ inv) = (x = val)"
  by (cases x,clarsimp+)

end

0.2 Predicates and functions used in the AODV model

theory Aodv_Data
imports Aodv_Basic
begin

0.2.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

definition inc :: "sqn ⇒ sqn"
  where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ inc x"
  unfolding inc_def by simp

lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp

lemma inc_never_one' [simp, intro]: "inc x ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
  by simp
0.2.2 Modelling Routes

A route is a 6-tuple, \((dsn, dsk, flag, hops, nhip, pre)\) where \(dsn\) is the 'destination sequence number', \(dsk\) is the 'destination-sequence-number status', \(flag\) is the route status, \(hops\) is the number of hops to the destination, \(nhip\) is the next hop toward the destination, and \(pre\) is the set of 'precursor nodes'—those interested in hearing about changes to the route.

\[
\text{proj2} :: \text{r} \Rightarrow \text{dsn} \quad \text{(}\pi_2\text{)}
\]

\[
\text{proj3} :: \text{r} \Rightarrow \text{dsk} \quad \text{(}\pi_3\text{)}
\]

\[
\text{proj4} :: \text{r} \Rightarrow \text{flag} \quad \text{(}\pi_4\text{)}
\]

\[
\text{proj5} :: \text{r} \Rightarrow \text{hops} \quad \text{(}\pi_5\text{)}
\]

\[
\text{proj6} :: \text{r} \Rightarrow \text{nhip} \quad \text{(}\pi_6\text{)}
\]

\[
\text{proj7} :: \text{r} \Rightarrow \text{pre} \quad \text{(}\pi_7\text{)}
\]

\[
\text{proj3\_pred} \quad \text{[intro]}: \quad \text{[} \quad \text{P kno; P unk} \quad \text{]} \quad \Rightarrow \quad \text{P} \quad \text{(}\pi_3\text{ x})
\]

\[
\text{proj4\_pred} \quad \text{[intro]}: \quad \text{[} \quad \text{P val; P inv} \quad \text{]} \quad \Rightarrow \quad \text{P} \quad \text{(}\pi_4\text{ x})
\]

\[
\text{proj6\_pair\_snd} \quad \text{[simp]}: \quad \text{fixes dsn' r}
\]

Routing tables map IP addresses to route entries.

\[
\text{type\_synonym} \quad \text{rt} = \quad \text{"ip} \quad \Rightarrow \quad \text{r}"
\]

\[
\text{syntax} \quad \text{"_Sigma\_route" :: "rt} \quad \Rightarrow \quad \text{ip} \quad \Rightarrow \quad \text{r} \quad \text{("\sigma\_route\'(\_, \_)\"})
\]

\[
\text{translations} \quad \text{"\sigma\_route(r, dip)" =⇒ "rt dip"}
\]

\[
\text{definition} \quad \text{sqn} :: \quad \text{"rt} \quad \Rightarrow \quad \text{ip} \quad \Rightarrow \quad \text{sqn} \quad \text{"}
\]

\[
\text{definition} \quad \text{sqnf} :: \quad \text{"rt} \quad \Rightarrow \quad \text{ip} \quad \Rightarrow \quad \text{k} \quad \text{"}
\]
abbreviation flag :: "rt ⇒ ip → f"
where "flag rt dip ≡ map_option π₄ (σ_route(rt, dip))"

abbreviation dhops :: "rt ⇒ ip → nat"
where "dhops rt dip ≡ map_option π₅ (σ_route(rt, dip))"

abbreviation nhop :: "rt ⇒ ip → ip"
where "nhop rt dip ≡ map_option π₆ (σ_route(rt, dip))"

abbreviation precs :: "rt ⇒ ip → ip set"
where "precs rt dip ≡ map_option π₇ (σ_route(rt, dip))"

definition vD :: "rt ⇒ ip set"
where "vD rt ≡ {dip. flag rt dip = Some val}"

definition iD :: "rt ⇒ ip set"
where "iD rt ≡ {dip. flag rt dip = Some inv}"

definition kD :: "rt ⇒ ip set"
where "kD rt ≡ {dip. rt dip ≠ None}"

lemma kD_is_vD_and_iD: "kD rt = vD rt ∪ iD rt"
unfolding kD_def vD_def iD_def by auto

lemma vD_iD_gives_kD [simp]:
"∀ ip rt. ip ∈ vD rt ⇒ ip ∈ kD rt"
"∀ ip rt. ip ∈ iD rt ⇒ ip ∈ kD rt"
unfolding kD_is_vD_and_iD by simp_all

lemma kD_Some [dest]:
fixes dip rt
assumes "dip ∈ kD rt"
shows "∃ dsn dsk flag hops nhip pre.
σ_route(rt, dip) = Some (dsn, dsk, flag, hops, nhip, pre)"
using assms unfolding kD_def by simp

lemma kD_None [dest]:
fixes dip rt
assumes "dip ∉ kD rt"
shows "σ_route(rt, dip) = None"
using assms unfolding kD_def
by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]:
fixes dip rt
assumes "dip ∈ vD rt"
shows "∃ dsn dsk hops nhip pre.
σ_route(rt, dip) = Some (dsn, dsk, val, hops, nhip, pre)"
using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
unfolding vD_def by simp

lemma iD_Some [dest]:
fixes dip rt
assumes "dip ∈ iD rt"
shows "∃ dsn dsk hops nhip pre.
σ_route(rt, dip) = Some (dsn, dsk, inv, hops, nhip, pre)"
using assms unfolding iD_def by simp

lemma val_is_vD [elim]:
fixes ip rt
assumes "ip ∈ kD(rt)"

and "the (flag rt ip) = val"
shows "ip∈vD(rt)"
using assms unfolding vD_def by auto

lemma inv_is_iD [elim]:
fixes ip rt
assumes "ip∈kD(rt)"
and "the (flag rt ip) = inv"
shows "ip∈iD(rt)"
using assms unfolding iD_def by auto

lemma iD_flag_is_inv [elim, simp]:
fixes ip rt
assumes "ip∈iD(rt)"
shows "the (flag rt ip) = inv"
proof -
from ⟨ip∈iD(rt)⟩ have "ip∈kD(rt)" by auto
with assms show ?thesis unfolding iD_def by auto
qed

lemma kD_but_not_vD_is_iD [elim]:
fixes ip rt
assumes "ip∈kD(rt)"
and "ip/∈vD(rt)"
shows "ip∈iD(rt)"
proof -
from ⟨ip∈kD(rt)⟩ obtain dsn dsk f hops nhop pre
where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop, pre)"
by (metis kD_Some)
from ⟨ip∈vD(rt)⟩ have "f≠val"
proof (rule contrapos_nn)
assume "f = val"
with rtip have "the (flag rt ip) = val" by simp
with ⟨ip∈kD(rt)⟩ show "ip∈vD(rt)" ..
qed
with rtip have "the (flag rt ip) = inv" by simp
with ⟨ip∈kD(rt)⟩ show "ip∈iD(rt)" ..
qed

lemma vD_or_iD [elim]:
fixes ip rt
assumes "ip∈kD(rt)"
and "ip∈vD(rt)⇒P rt ip"
and "ip∈iD(rt)⇒P rt ip"
shows "P rt ip"
proof -
from ⟨ip∈kD(rt)⟩ have "ip∈vD(rt)∪iD(rt)"
by (simp add: kD_is_vD_and_iD)
thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "⋀dip rt. dip∈kD(rt)⇒π₅(the (rt dip)) = the (dhops rt dip)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "⋀dip rt. dip∈kD(rt)⇒π₄(the (rt dip)) = the (flag rt dip)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "⋀dip rt. dip∈kD(rt)⇒π₂(the (rt dip)) = sqn rt dip"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
"⋀ip rt. ip∈kD(rt)⇒sqnf rt ip = π₃(the (rt ip))"
unfolding sqnf_def by auto
lemma vD_flag_val [simp]:
"∀ dip rt. dip ∈ vD (rt) → the (flag rt dip) = val"
unfolding vD_def by clarsimp

lemma kD_update [simp]:
"∀ nip v. kD (rtnip → v)) = insert nip (kD rt)"
unfolding kD_def by auto

lemma kD_empty [simp]: "kD Map.empty = {}"
unfolding kD_def by simp

lemma ip_equal_or_known [elim]:
  fixes rt ip ip’
  assumes "ip = ip’ ∨ ip ∈ kD(rt)"
  and "ip = ip’ → P rt ip ip’"
  and "[ [ip ≠ ip’; ip ∈ kD(rt)] ] → P rt ip ip’"
  shows "P rt ip ip’"
using assms by auto

0.2.4 Updating Routing Tables
Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

Updating Precursor Lists

definition addpre :: "r ⇒ ip set ⇒ r"
  where "addpre r npre ≡ let (dsn, dsk, flag, hops, nhip, pre) = r in
          (dsn, dsk, flag, hops, nhip, pre ∪ npre)"

lemma proj2_addpre:
  fixes v pre
  shows "π₂(addpre v pre) = π₂(v)"
unfolding addpre_def by (cases v) simp

lemma proj3_addpre:
  fixes v pre
  shows "π₃(addpre v pre) = π₃(v)"
unfolding addpre_def by (cases v) simp

lemma proj4_addpre:
  fixes v pre
  shows "π₄(addpre v pre) = π₄(v)"
unfolding addpre_def by (cases v) simp

lemma proj5_addpre:
  fixes v pre
  shows "π₅(addpre v pre) = π₅(v)"
unfolding addpre_def by (cases v) simp

lemma proj6_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "π₆(addpre v npre) = π₆(v)"
unfolding addpre_def by (cases v) simp

lemma proj7_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "π₇(addpre v npre) = π₇(v) ∪ npre"
unfolding addpre_def by (cases v) simp

lemma addpre_empty: "addpre r {} = r"
unfolding addpre_def by simp

lemma addpre_r:
"addpre (dsn, dsk, fl, hops, nhip, pre) npre = (dsn, dsk, fl, hops, nhip, pre ∪ npre)"

unfolding addpre_def by simp

lemmas addpre_simps [simp] = proj2_addpre proj3_addpre proj4_addpre proj5_addpre proj6_addpre proj7_addpre addpre_empty addpre_r

definition addpreRT :: "rt ⇒ ip ⇒ ip set ⇒ rt"
where "addpreRT rt dip npre ≡ map_option (λs. rt (dip ↦→ addpre s npre)) (σ_route(rt, dip))"

lemma snd_addpre [simp]:
"⋀dsn dsn' v pre. (dsn, snd(addpre (dsn', v) pre)) = addpre (dsn, v) pre"
unfolding addpre_def by clarsimp

lemma proj2_addpreRT [simp]:
fixes ip rt ip' npre
assumes "ip∈kD rt"
and "ip'∈kD rt"
shows "π_2(the (the (addpreRT rt ip' npre) ip)) = π_2(the (rt ip))"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj3_addpreRT [simp]:
fixes ip rt ip' npre
assumes "ip∈kD rt"
and "ip'∈kD rt"
shows "π_3(the (the (addpreRT rt ip' npre) ip)) = π_3(the (rt ip))"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj5_addpreRT [simp]:
"⋀rt dip ip npre. dip∈kD(rt) =⇒ π_5(the (the (addpreRT rt dip npre) ip)) = π_5(the (rt ip))"
unfolding addpreRT_def by auto

lemma flag_addpreRT [simp]:
fixes rt pre ip dip
assumes "dip ∈ kD rt"
shows "flag (the (addpreRT rt dip pre)) ip = flag rt ip"
unfolding addpreRT_def using assms [THEN kD_Some] by (clarsimp)

lemma kD_addpreRT [simp]:
fixes rt dip npre
assumes "dip ∈ kD rt"
shows "kD (the (addpreRT rt dip npre)) = kD rt"
unfolding kD_def addpreRT_def using assms [THEN kD_Some] by clarsimp blast

lemma vD_addpreRT [simp]:
fixes rt dip npre
assumes "dip ∈ kD rt"
shows "vD (the (addpreRT rt dip npre)) = vD rt"
unfolding vD_def addpreRT_def using assms [THEN kD_Some] by clarsimp auto

lemma iD_addpreRT [simp]:
fixes rt dip npre
assumes "dip ∈ kD rt"
shows "iD (the (addpreRT rt dip npre)) = iD rt"
unfolding iD_def addpreRT_def using assms [THEN kD_Some] by clarsimp auto

lemma nhop_addpreRT [simp]:
fixes rt pre ip dip
assumes "dip ∈ kD rt"
shows "nhop (the (addpreRT rt dip pre)) ip = nhop rt ip"
unfolding sqn_def addpreRT_def
using assms [THEN kd_Some] by (clarsimp)

lemma sqn_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "sqn (the (addpreRT rt dip pre)) ip = sqn rt ip"
  unfolding sqn_def addpreRT_def
  using assms [THEN kd_Some] by (clarsimp)

lemma dhops_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "dhops (the (addpreRT rt dip pre)) ip = dhops rt ip"
  unfolding addpreRT_def
  using assms [THEN kd_Some] by (clarsimp)

lemma sqnf_addpreRT [simp]:
  "∀ ip dip. ip ∈ kD(rt ξ) ⇒ sqnf (the (addpreRT (rt ξ) ip npre)) dip = sqnf (rt ξ) dip"
  unfolding sqnf_def addpreRT_def by auto

Updating route entries

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = es (the (rt dip))"
  using assms [THEN kd_Some] by auto

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip /∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = en"
  using assms [THEN kd_None] by auto

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip pre
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip /∈ kD(rt)"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def by simp

definition update_arg_wf :: "r ⇒ bool"
where "update_arg_wf r ≡ π_4(r) = val ∧
        (π_2(r) = 0) = (π_3(r) = unk) ∧
        (π_3(r) = unk → π_5(r) = 1)"

lemma update_arg_wf_gives_cases:
  "∀ r. update_arg_wf r =⇒ (π_2(r) = 0) = (π_3(r) = unk)"
  unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples [simp]:
  "∀ nhip pre. update_arg_wf (0, unk, val, Suc 0, nhip, pre)"
  "∀ n hops nhip pre. update_arg_wf (Suc n, kno, val, hops, nhip, pre)"
  unfolding update_arg_wf_def by auto

lemma update_arg_wf_tuples' [elim]:

11
\[ n \text{ hops nhip pre. Suc } 0 \leq n \implies \text{update_arg_wf } (n, \text{kno}, \text{val}, \text{hops}, \text{nhip}, \text{pre}) \]

unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  \[ \text{fixes } P \text{ r} \]
  assumes "update_arg_wf r"
  \[ \text{and } c1: \ "\text{nhip pre. P } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{nhip}, \text{pre})" \]
  \[ \text{and } c2: \ "\text{dsn hops nhip pre. dsn } > 0 \implies P (dsn, \text{kno}, \text{val}, \text{hops}, \text{nhip}, \text{pre})" \]
  shows "P r"
proof -
  obtain dsn dsk flag hops nhip pre
  where *: "r = (dsn, dsk, flag, hops, nhip, pre)"
  by (cases r)
  with ⟨update_arg_wf r⟩ have
  wf1: "flag = val"
  and
  wf2: "dsn = 0 \implies (dsk = unk)"
  and
  wf3: "dsk = unk \implies (hops = 1)"
  unfolding update_arg_wf_def
  by auto
  have "P (dsn, dsk, flag, hops, nhip, pre)"
  proof
    (cases dsk)
    assume "dsk = unk"
    moreover with wf2 wf3
    have "dsn = 0"
    and "hops = Suc 0"
    by (auto)
    ultimately show ?thesis
    using ⟨flag = val⟩
    by (simp (rule c1))
  next
    assume "dsk = kno"
    moreover with wf2
    have "dsn > 0"
    by (simp)
    ultimately show ?thesis
    using ⟨flag = val⟩
    by (simp (rule c2))
  qed
  with * show "P r" by simp
qed

definition update :: "rt ip r ⇒ rt"
where
"update rt ip r ≡ case σ_route(rt, ip) of
  None ⇒ rt (ip ↦→ r)
| Some s ⇒
  if π_2(s) < π_2(r) then rt (ip ↦→ addpre r (π_7(s)))
  else if π_2(s) = π_2(r) ∧ (π_5(s) > π_5(r) ∨ π_4(s) = inv)
    then rt (ip ↦→ addpre r (π_7(s)))
  else if π_3(r) = unk
    then rt (ip ↦→ (π_2(s), snd (addpre r (π_7(s)))))
  else rt (ip ↦→ addpre s (π_7(r)))"

lemma update_simps [simp]:
  \[ \text{fixes } r, s, nrt, nr, nr', ns, rt, ip \]
  \[ \text{defines } "s ≡ the σ_route(rt, ip)" \]
  \[ "nr ≡ addpre r (π_7(s))" \]
  \[ "nr' ≡ (π_3(s), π_3(nr), π_4(nr), π_5(nr), π_6(nr), π_7(nr))" \]
  \[ "ns ≡ addpre s (π_7(r))" \]
  shows
  "[ip \notin kD(rt)] \implies \text{update rt ip r = rt (ip ↦→ r)}" 
  "[ip \in kD(rt); sqn rt ip < π_2(r)] \implies \text{update rt ip r = rt (ip ↦→ nr)}" 
  "[ip \in kD(rt); sqn rt ip = π_2(r); the (dhops rt ip) > π_5(r)] \implies \text{update rt ip r = rt (ip ↦→ nr)}" 
  "[ip \in kD(rt); sqn rt ip = π_2(r); flag rt ip = Some inv] \implies \text{update rt ip r = rt (ip ↦→ nr)}" 
  "[ip \in kD(rt); π_3(r) = unk; (π_2(r) = 0) = (π_3(r) = unk)] \implies \text{update rt ip r = rt (ip ↦→ nr')}" 
  "[ip \in kD(rt); sqn rt ip ≥ π_2(r); π_3(r) = kno; sqn rt ip = π_2(r) \implies \text{the (dhops rt ip) ≤ π_5(r) \land the (flag rt ip) = val}] \implies \text{update rt ip r = rt (ip ↦→ ns)}"
proof -
  assume "ip ∉ kD(rt)"
  hence "σ_route(rt, ip) = None" ..
  thus "update rt ip r = rt (ip ↦→ r)"
  unfolding update_def by simp
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip < π₂(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with ⟨sqn rt ip < π₂(r)⟩ show "update rt ip r = rt (ip ↦ nr)"
    unfolding update_def nr_def s_def by auto

next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip = π₂(r)"
  and "the (dhops rt ip) > π₅(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with ⟨sqn rt ip = π₂(r)⟩ and ⟨the (dhops rt ip) > π₅(r)⟩
    show "update rt ip r = rt (ip ↦ nr)"
    unfolding update_def nr_def s_def by auto

next
  assume "ip ∈ kD(rt)"
  and "π₃(r) = unk"
  and "(π₂(r) = 0) = (π₃(r) = unk)"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with ⟨(π₂(r) = 0) = (π₃(r) = unk)⟩ and ⟨π₃(r) = unk⟩
    show "update rt ip r = rt (ip ↦ nr')"
    unfolding update_def nr'_def nr_def s_def
    by (cases r) simp

next
  assume "ip ∈ kD(rt)"
  and otherassms: "sqn rt ip ≥ π₂(r)"
    "π₃(r) = kno"
    "sqn rt ip = π₂(r) ⟹ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with otherassms show "update rt ip r = rt (ip ↦ ns)"
    unfolding update_def ns_def s_def by auto

qed

lemma update_cases [elim]:
  assumes "(π₂(r) = 0) = (π₃(r) = unk)"
  and c1: "[\[ \neg ip ∈ kD(rt) \] \] ⟹ P (rt (ip ↦ r))"

  and c2: "[\[ ip ∈ kD(rt); sqn rt ip < π₂(r) \] \] ⟹ P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip)))))"

  and c3: "[\[ ip ∈ kD(rt); sqn rt ip = π₂(r); the (dhops rt ip) > π₅(r) \] \] ⟹ P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip)))))"

  and c4: "[\[ ip ∈ kD(rt); sqn rt ip = π₂(r); the (flag rt ip) = inv \] \] ⟹ P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip)))))"

  and c5: "[\[ ip ∈ kD(rt); π₃(r) = unk \] \] ⟹ P (rt (ip ↦ (π₇(the σroute(rt, ip)), π₃(r), π₄(r), π₅(r), π₆(r), π₇(addpre r (π₇(the σroute(rt, ip)))))))"
and c6: "[[\text{ip} \in kD(\text{rt}); \text{sqn rt ip} \geq \pi_2(\text{r}); \pi_3(\text{r}) = \text{kno};
\text{sqn rt ip} = \pi_2(\text{r}) \implies (\text{dhops rt ip}) \leq \pi_5(\text{r}) \land (\text{flag rt ip}) = \text{val}]
\implies P (\text{rt (ip} \mapsto \text{addpre (the } \sigma_{\text{route}}(\text{rt, ip})) (\pi_7(\text{r}))))"
shows "(P (update rt ip r))"
proof (cases "ip \in kD(\text{rt})")
assume "ip \notin kD(\text{rt})"
with c1 show \text{?thesis}
by simp
next
assume "ip \in kD(\text{rt})"
moreover then obtain dsn dsk fl hops nhip pre
where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
moreover obtain dsn' dsk' fl' hops' nhip' pre'
where req: "\text{r} = (dsn', dsk', fl', hops', nhip', pre')"
by (cases \text{r}) metis
ultimately show \text{?thesis}
using ⟨(\pi_2(\text{r}) = 0) = (\pi_3(\text{r}) = \text{unk})⟩
c2 [OF \langle \text{ip} \in kD(\text{rt}) \rangle]
c3 [OF \langle \text{ip} \in kD(\text{rt}) \rangle]
c4 [OF \langle \text{ip} \in kD(\text{rt}) \rangle]
c5 [OF \langle \text{ip} \in kD(\text{rt}) \rangle]
c6 [OF \langle \text{ip} \in kD(\text{rt}) \rangle]
unfolding update_def sqn_def by auto
qed

lemma update_cases_kD:
assumes "(\pi_2(\text{r}) = 0) = (\pi_3(\text{r}) = \text{unk})"
and \text{ip} \in kD(\text{rt})"
and c2: "sqn \text{rt ip} < \pi_2(\text{r}) \implies P (\text{rt (ip} \mapsto \text{addpre (\pi_7(\text{the } \sigma_{\text{route}}(\text{rt, ip})))))})"
and c3: "[[\text{sqn rt ip} = \pi_2(\text{r}); \text{the (dhops rt ip)} > \pi_5(\text{r})]
\implies P (\text{rt (ip} \mapsto \text{addpre (\pi_7(\text{the } \sigma_{\text{route}}(\text{rt, ip})))))})"
and c4: "[[\text{sqn rt ip} = \pi_2(\text{r}); \text{the (flag rt ip)} = \text{inv}]
\implies P (\text{rt (ip} \mapsto \text{addpre (\pi_7(\text{the } \sigma_{\text{route}}(\text{rt, ip})))))})"
and c5: "\pi_3(\text{r}) = \text{unk} \implies P (\text{rt (ip} \mapsto ((\pi_2(\text{the } \sigma_{\text{route}}(\text{rt, ip}))), \pi_3(\text{r}), \pi_4(\text{r}), \pi_5(\text{r}), \pi_6(\text{r}),
\pi_7(\text{addpre r (\pi_7(\text{the } \sigma_{\text{route}}(\text{rt, ip}))))))))"
and c6: "[[\text{sqn rt ip} \geq \pi_2(\text{r}); \pi_3(\text{r}) = \text{kno};
\text{sqn rt ip} = \pi_2(\text{r}) \implies (\text{dhops rt ip}) \leq \pi_5(\text{r}) \land (\text{flag rt ip}) = \text{val}]
\implies P (\text{rt (ip} \mapsto \text{addpre (the } \sigma_{\text{route}}(\text{rt, ip})) (\pi_7(\text{r}))))"
shows "(P (update rt ip r))"
using assms(1) proof (rule update_cases)
assume "\text{sqn rt ip} < \pi_2(\text{r})"
thus "P (\text{rt (ip} \mapsto \text{addpre (\pi_7(\text{the (rt ip)}))}))" by (rule c2)
next
assume "\text{sqn rt ip} = \pi_2(\text{r})"
and "(\text{the (dhops rt ip)} > \pi_5(\text{r})"
thus "P (\text{rt (ip} \mapsto \text{addpre (\pi_7(\text{the (rt ip)}))}))" by (rule c3)
next
assume "\text{sqn rt ip} = \pi_2(\text{r})"
and "(\text{the (flag rt ip)} = \text{inv}"
thus "P (\text{rt (ip} \mapsto \text{addpre (\pi_7(\text{the (rt ip)}))}))" by (rule c4)
next
assume "\pi_3(\text{r}) = \text{unk}"
thus "P (\text{rt (ip} \mapsto ((\pi_2(\text{the } \sigma_{\text{route}}(\text{rt, ip}))), \pi_3(\text{r}), \pi_4(\text{r}), \pi_5(\text{r}), \pi_6(\text{r}),
\pi_7(\text{addpre r (\pi_7(\text{the (rt ip)}))))))" by (rule c5)
next
assume "\text{sqn rt ip} \geq \pi_2(\text{r})"
and "\pi_3(\text{r}) = \text{kno}"
and "\text{sqn rt ip} = \pi_2(\text{r}) \implies (\text{dhops rt ip}) \leq \pi_5(\text{r}) \land (\text{flag rt ip}) = \text{val}"
thus "P (\text{rt (ip} \mapsto \text{addpre (the (rt ip)) (\pi_7(\text{r}))}))"
by (rule c6)
qed (simp add: ip ∈ kD(rt))

lemma in_kD_after_update [simp]:
fixes rt nip dsn dsk flag hops nhip pre
shows "kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) = insert nip (kD rt)"
unfolding update_def
by (cases "rt nip") auto

lemma nhop_of_update [simp]:
fixes rt dip dsn dsk flag hops nhip
assumes "rt ≠ update rt dip (dsn, dsk, flag, hops, nhip, {})"
shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip, {})) dip) = nhip"
proof -
from assms have update_neq: "∀ v. rt dip = Some v →
update rt dip (dsn, dsk, flag, hops, nhip, {})
≠ rt (dip := addpre (the (rt dip)) (π7 (dsn, dsk, flag, hops, nhip, {})))"
by auto
show ?thesis
proof (cases "rt dip = None")
assume "rt dip = None"
thus ?thesis
unfolding update_def
by clarsimp
next
assume "rt dip ≠ None"
then obtain v where "rt dip = Some v" by (metis not_None_eq)
with update_neq [OF this] show ?thesis
unfolding update_def by auto
qed
qed

lemma sqn_if_updated:
fixes rip v rt ip
shows "sqn (λ x. if x = rip then Some v else rt x) ip
= (if ip = rip then π2(v) else sqn rt ip)"
unfolding sqn_def by simp

lemma update_sqn [simp]:
fixes rt dip rip dsn dsk hops nhip pre
assumes "(dsn = 0) = (dsk = unk)"
shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip, pre)) dip"
proof (rule update_cases)
show "(π2 (dsn, dsk, val, hops, nhip, pre) = 0) = (π3 (dsn, dsk, val, hops, nhip, pre) = unk)"
by simp (rule assms)
qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
fixes rt ip ip' dsn dsk flag hops nhip pre
assumes "1 ≤ hops"
shows "sqn rt ip ≤ sqn (update rt ip' (dsn, dsk, flag, hops, nhip, pre)) ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
fixes rt dsn dsk flag hops ip rip nhip pre
assumes "1 ≤ hops"
shows "∀ ip∈kD rt. the (dhops rt ip) ≥ 1"
and ip: "(ip = rip ∧ Suc 0 ≤ hops) ∨ (ip ≠ rip ∧ ip∈kD rt)"
shows "Suc 0 ≤ the (dhops (update rt rip (dsn, dsk, flag, hops, nhip, pre)) ip)"
using ip proof
assume "ip = rip ∧ Suc 0 ≤ hops" thus ?thesis
unfolding update_def using ex
by (cases "rip ∈ kD rt") (drule(1) bspec, auto)
next
assume "ip ≠ rip ∧ ip∈kD rt" thus ?thesis
using ex unfolding update_def
by (cases "rip∈kD rt") auto
qed

lemma update_another [simp]:
  fixes dip ip rt dan dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = rt ip"
using assms unfolding update_def
by (clarsimp split: option.split)

lemma nhop_update_another [simp]:
  fixes dip ip rt dan dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = nhop rt ip"
using assms unfolding update_def
by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dan dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = dhops rt ip"
using assms unfolding update_def
by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  "∀ rt ip dsn dsk flag hops nhip pre. sqn (rt(ip ↦→ v)) ip = π_2(v)"
unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip, {})) dip) = hops"
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma nhop_update_unk_val [simp]:
  "∀ rt dip ip dsn hops npre.
  the (nhop (update rt dip (dsn, unk, val, hops, ip, npre)) dip) = ip"
unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip, {}) ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
using assms unfolding update_def
by (clarsimp split: option.splits if_split_asm) auto

lemma update_rt_split_asm:
  "∀ rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip, {}))
  =
  (∼ rt = update rt ip (dsn, dsk, flag, hops, sip, {}) ∧ ∼ P rt
  ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip, {})
  ∧ ∼ P (update rt ip (dsn, dsk, flag, hops, sip, {}))))"
by auto

lemma sqn_update [simp]: "∀ rt dip dsn dsk flag hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
  ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsn"
unfolding update_def by (clarsimp split: option.split if_split_asm) auto

lemma sqnf_update [simp]: "∀ rt dip dsn dsk flag hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
  ⇒ sqnf (update rt dip (dsn, dsk, flg, hops, sip, {})) dip = dsn"
unfolding update_def by (clarsimp split: option.split if_split_asm) auto
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma update_kno_dsn_greater_zero:
"∀ rt dip ip dsn hops npre. 1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip, npre)) dip)"
unfolding update_def
by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∀ rt dip dsn dsk flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
⇒ π3 (the (update rt dip (dsn, dsk, flg, hops, sip, {}) dip)) = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma nhop_update_changed_kno_val [simp]: "∀ rt ip dsn dsk hops nhip.
rt ≠ update rt ip (dsn, kno, val, hops, nhip, {})
⇒ the (nhop (update rt ip (dsn, kno, val, hops, nhip, {})) ip) = nhip"
unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma flag_update [simp]: "∀ rt dip dsn flg hops sip.
rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
⇒ the (flag (update rt dip (dsn, kno, flg, hops, sip, {})) dip) = flg"
unfolding update_def
by (clarsimp split: option.split if_split_asm) auto

lemma the_flag_Some [dest!]:
fixes ip rt
assumes "the (flag rt ip) = x"
and "ip ∈ kD rt"
shows "flag rt ip = Some x"
using assms by auto

lemma kD_update_unchanged [dest]:
fixes rt dip dsn dsk flag hops nhip pre
assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip, pre)"
shows "dip ∈ kD(rt)"
proof -
  have "dip ∈ kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp
  with assms show ?thesis by simp
qed

lemma nhop_update [simp]: "∀ rt dip dsn dsk flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
⇒ the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma sqn_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip ≠ dip"
shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip ≠ dip"
shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:
dip ∈ vD(update rt dip’ (dsn, dsk, val, hops, nhip, pre)) \implies (dip∈vD(rt) \lor dip=dip’)

Invalidating route entries

definition invalidate :: "rt ⇒ (ip ⇒ sqn) ⇒ rt"
where "invalidate rt dests ≡ \lambda ip. case (rt ip, dests ip) of
  (None, _) ⇒ None
  | (Some s, None) ⇒ Some s
  | (Some (_, dsk, _, hops, nhip, pre), Some rsn) ⇒
    Some (rsn, dsk, inv, hops, nhip, pre)"

lemma proj3_invalidate [simp]:
"\forall dip. \pi_3 (the ((invalidate rt dests) dip)) = \pi_3 (the (rt dip))"

lemma proj5_invalidate [simp]:
"\forall dip. \pi_5 (the ((invalidate rt dests) dip)) = \pi_5 (the (rt dip))"

lemma proj6_invalidate [simp]:
"\forall dip. \pi_6 (the ((invalidate rt dests) dip)) = \pi_6 (the (rt dip))"

lemma proj7_invalidate [simp]:
"\forall dip. \pi_7 (the ((invalidate rt dests) dip)) = \pi_7 (the (rt dip))"

lemma invalidate_kD_inv [simp]:
"\forall rt dests. kD (invalidate rt dests) = kD rt"

lemma invalidate_sqn:
fixes rt dip dests
assumes "\forall rsn. dests dip = Some rsn \implies sqn rt dip \leq rsn"
shows "sqn rt dip \leq sqn (invalidate rt dests) dip"
proof (cases "dip \notin kD(rt)")
  assume "dip \notin kD(rt)"
  hence "dip\notin kD(rt)" by simp
  then obtain dsn dsk flag hops nhip pre where "rt dip = Some (dsn, dsk, flag, hops, nhip, pre)"
  by (metis kD_Some)
  with assms show "sqn rt dip \leq sqn (invalidate rt dests) dip"
  by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
qed simp

lemma sqn_invalidate_in_dests [simp]:
fixes dests ipa rsn rt
assumes "dests ipa = Some rsn" and "ipa\in kD(rt)"
shows "sqn (invalidate rt dests) ipa = rsn"
unfolding invalidate_def sqn_def
using assms(1) assms(2) [THEN kD_Some] by clarsimp

lemma dhops_invalidate [simp]:
"\forall dip. the (dhops (invalidate rt dests) dip) = the (dhops rt dip)"
unfolding invalidate_def by (clarsimp simp: option.split)

lemma sqnf_invalidate [simp]:
"\forall dip. sqnf (invalidate (rt \ξ) (dests \ξ)) dip = sqnf (rt \ξ) dip"
unfolding sqnf_def invalidate_def by (clarsimp simp: option.split)
lemma nhop_invalidate [simp]:
"∀ dip. the (nhop (invalidate (rt ξ) (dests ξ)) dip) = the (nhop (rt ξ) dip)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
  fixes rt dests dip
  assumes "dip ∉ dom(dests)"
  shows "invalidate rt dests dip = rt dip"
  using assms unfolding invalidate_def by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes rt dests dip
  assumes "dip ∉ kD(rt)"
  shows "invalidate rt dests dip = None"
  using assms unfolding invalidate_def by simp

lemma vD_invalidate_vD_not_dests:
"∀ dip rt dests. dip ∈ vD(invalidate rt dests) ⇒ dip ∈ vD(rt) ∧ dests dip = None"
unfolding invalidate_def vD_def by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
  fixes dests dip rt
  assumes "dip ∉ dom(dests)"
  shows "sqn (invalidate rt dests) dip = sqn rt dip"
  using assms unfolding sqn_def by simp

lemma invalidate_changes:
  fixes rt dests dip dsn dsk flag hops nhip pre
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "dsn = (case dests dip of None ⇒ π_2 (the (rt dip)) | Some rsn ⇒ rsn)
  ∧ dsk = π_3 (the (rt dip))
  ∧ flag = (if dests dip = None then π_4 (the (rt dip)) else inv)
  ∧ hops = π_5 (the (rt dip))
  ∧ nhip = π_6 (the (rt dip))
  ∧ pre = π_7 (the (rt dip))"
  using assms unfolding invalidate_def by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "∀ dip rt dests. dip ∈ kD (rt) ⇒ π_3 (the (invalidate rt dests dip)) = π_3 (the (rt dip))"
  by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
  assumes "dests ip = Some rsn"
  and "ip ∈ kD(rt)"
  shows "ip ∈ iD(invalidate rt dests)"
  using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
  by (clarsimp split: option.split)

0.2.5 Route Requests

Generate a fresh route request identifier.

definition nrreqid :: "(ip × rreqid) set ⇒ ip ⇒ rreqid"
  where "nrreqid reqs ip ≡ Max (?a. (ip, a) ∈ reqs ∪ {0}) + 1"

0.2.6 Queued Packets

Functions for sending data packets.

type_synonym store = "ip ⇒ (p × data list)"
definition \sigma_queue :: "store \Rightarrow ip \Rightarrow data list" ("\sigma_queue\('_, _'\)"
where \"\sigma_queue\(store, dip\) \equiv case store dip of None \Rightarrow [] \mid Some \(p, q\) \Rightarrow q"

definition \(qD \equiv dom"

definition \(add \equiv \"data \Rightarrow ip \Rightarrow store \Rightarrow\"
where \"add d dip store \equiv case store dip of
  None \Rightarrow store (dip \mapsto (req, \[d\]))
  \mid Some \(p, q\) \Rightarrow store (dip \mapsto (p, q @ \[d\]))"

lemma \(qD_add \ [simp]:\)
fixes \(d, dip, store\)
shows \"\(qD(add d dip store) = insert dip (qD store)\)"
unfolding add_def Let_def qD_def
by (clarsimp split: option.split)

definition \(drop \equiv \"ip \Rightarrow store \rightarrow store\"
where \"drop dip store \equiv map_option (\lambda (p, q). if tl q = [] then store (dip := None)
  else store (dip \mapsto (p, tl q))) (store dip)"

definition \(sigma_p_flag \equiv \"store \Rightarrow ip \Rightarrow p\" ("\sigma_p-flag\('_, _'\)"
where \"\sigma_p-flag\(store, dip\) \equiv map_option \(\lambda(p, q). (req, q)\) (store dip)"

definition \(unsetRRF \equiv \"store \Rightarrow ip \Rightarrow store\"
where \"unsetRRF store dip \equiv case store dip of
  None \Rightarrow store
  \mid Some \(p, q\) \Rightarrow store (dip \mapsto (noreq, q))"

definition \(setRRF \equiv \"store \Rightarrow \(ip \Rightarrow sqn\) \Rightarrow store\"
where \"setRRF store dests \equiv \lambda\(dip, \) if dests dip = None then store dip
  else map_option (\lambda(_, q). (req, q)) (store dip)"

0.2.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. \(nhop\) is partial, thus a ‘the’ is needed, similarly for \(dhops\) and \(addpreRT\).
2. \(precs\) is partial.
3. \(\sigma_p-flag\(store, dip\) is partial.
4. The routing table \(rt\) is modelled as a map \((ip \Rightarrow r\ option)\) rather than a set of 7-tuples, likewise, the \(r\)
is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address \(dip\) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the dests mapping passed to \(invalidate\), and \(store\).

end

0.3 AODV protocol messages

theory Aodv_Message
imports Aodv_Basic
begin

datatype msg =
  Rreq nat rreqid ip sqn k ip sqn ip
  / Rrep nat ip sqn ip
The \textit{msg} type models the different messages used within AODV. The instantiation as a \textit{msg} is a technicality due to the special treatment of \textit{newpkt} messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

\begin{lstlisting}[language=Isar]
definition rreq :: "nat × rreqid × ip × sqn × k × ip × sqn × ip ⇒ msg"
  where "rreq ≡ λ(hops, rreqid, dip, dsn, dsk, oip, osn, sip).
Rreq hops rreqid dip dsn dsk oip osn sip"

lemma rreq_simp [simp]:
"rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip) = Rreq hops rreqid dip dsn dsk oip osn sip"
unfolding rreq_def by simp

definition rrep :: "nat × ip × sqn × ip × ip ⇒ msg"
  where "rrep ≡ λ(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"

lemma rrep_simp [simp]:
"rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
unfolding rrep_def by simp

definition rerr :: "(ip ↦ sqn) × ip ⇒ msg"
  where "rerr ≡ λ(dests, sip). Rerr dests sip"

lemma rerr_simp [simp]:
"rerr(dests, sip) = Rerr dests sip"
unfolding rerr_def by simp

lemma not_eq_newpkt_rreq [simp]: "¬eq_newpkt (Rreq hops rreqid dip dsn dsk oip osn sip)"
unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rrep [simp]: "¬eq_newpkt (Rrep hops dip dsn oip sip)"
unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rerr [simp]: "¬eq_newpkt (Rerr dests sip)"
unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_pkt [simp]: "¬eq_newpkt (Pkt d dip sip)"
unfolding eq_newpkt_def by simp

definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"

lemma pkt_simp [simp]:
"pkt(d, dip, sip) = Pkt d dip sip"
unfolding pkt_def by simp
\end{lstlisting}

\subsection{The AODV protocol}

theory Aodv
imports Aodv_Data Aodv_Message
  AWN.AWN_SOS_Labels AWN.AWN_Invariants
begin

0.4.1 Data state

record state =
  ip :: "ip"
  sn :: "sn"
  rt :: "rt"
  rreqs :: "(ip × rreqid) set"
  store :: "store"
  msg :: "msg"
  data :: "data"
  dests :: "ip ⟷ sn"
  pre :: "ip set"
  rreqid :: "rreqid"
  dip :: "ip"
  oip :: "ip"
  hops :: "nat"
  dsn :: "sn"
  dsk :: "k"
  osn :: "sn"
  sip :: "ip"

abbreviation aodv_init :: "ip ⇒ state" where
  "aodv_init i ≡ (|
    ip = i,
    sn = 1,
    rt = Map.empty,
    rreqs = {},
    store = Map.empty,
    msg = (SOME x. True),
    data = (SOME x. True),
    dests = (SOME x. True),
    pre = (SOME x. True),
    rreqid = (SOME x. True),
    dip = (SOME x. True),
    oip = (SOME x. True),
    hops = (SOME x. True),
    dsn = (SOME x. True),
    dsk = (SOME x. True),
    osn = (SOME x. True),
    sip = (SOME x. x ≠ i)
  |)"

lemma some_neq_not_eq [simp]: "¬((SOME x :: nat. x ≠ i) = i)"
  by (subst some_eq_ex) (metis zero_neq_numeral)

definition clear_locals :: "state ⇒ state"
where "clear_locals ξ = ξ (|
  msg := (SOME x. True),
  data := (SOME x. True),
  dests := (SOME x. True),
  pre := (SOME x. True),
  rreqid := (SOME x. True),
  dip := (SOME x. True),
  oip := (SOME x. True),
  hops := (SOME x. True),
  dsn := (SOME x. True),
  dsk := (SOME x. True),
  osn := (SOME x. True),
  sip := (SOME x. x ≠ ip ξ)
  |)"

lemma clear_locals_sip_not_ip [simp]: "\neg(sip (clear_locals \xi) = ip \xi)"
unfolding clear_locals_def by simp

lemma clear_locals_but_notGlobals [simp]:
  "ip (clear_locals \xi) = ip \xi"
  "sn (clear_locals \xi) = sn \xi"
  "rt (clear_locals \xi) = rt \xi"
  "rreqs (clear_locals \xi) = rreqs \xi"
  "store (clear_locals \xi) = store \xi"
unfolding clear_locals_def by auto

0.4.2 Auxiliary message handling definitions

definition is_newpkt
where "is_newpkt \xi \equiv case msg \xi of
  Newpkt data' dip' \Rightarrow \{ \xi(data := data', dip := dip') \}
  _ \Rightarrow \{
"

definition is_pkt
where "is_pkt \xi \equiv case msg \xi of
  Pkt data' dip' oip' \Rightarrow \{ \xi(data := data', dip := dip', oip := oip' \)
  _ \Rightarrow \{
"

definition is_rreq
where "is_rreq \xi \equiv case msg \xi of
  Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' \Rightarrow
  \{ \xi(hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
    dsk := dsk', oip := oip', osn := osn', sip := sip' \)
  _ \Rightarrow \{
"

lemma is_rreq_asm [dest!]:
  assumes "\xi' \in is_rreq \xi"
  shows "(\exists hops' rreqid' dip' dsn' dsk' oip' osn' sip'.
    msg \xi = Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' \land
    \xi' = \xi(hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
    dsk := dsk', oip := oip', osn := osn', sip := sip' \))"
using assms unfolding is_rreq_def
by (cases "msg \xi") simp_all

definition is_rrep
where "is_rrep \xi \equiv case msg \xi of
  Rrep hops' dip' dsn' oip' sip' \Rightarrow
  \{ \xi(hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' \)
  _ \Rightarrow \{
"

lemma is_rrep_asm [dest!]:
  assumes "\xi' \in is_rrep \xi"
  shows "(\exists hops' dip' dsn' oip' sip'.
    msg \xi = Rrep hops' dip' dsn' oip' sip' \land
    \xi' = \xi(hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' \))"
using assms unfolding is_rrep_def
by (cases "msg \xi") simp_all

definition is_rerr
where "is_rerr \xi \equiv case msg \xi of
  Rerr dests' sip' \Rightarrow \{ \xi(dests := dests', sip := sip' \)
  _ \Rightarrow \{
"

lemma is_rerr_asm [dest!]:
  assumes "\xi' \in is_rerr \xi"
  shows "(\exists dests' sip'.
    msg \xi = Rerr dests' sip' \land
    \xi' = \xi(dests := dests', sip := sip' \))"

23
using assms unfolding is_rerr_def
by (cases "msg ξ") simp_all

lemmas is_msg_defs =
is_rerr_def is_rrep_def is_rqreq_def is_pkt_def is_newpkt_def

lemma is_msg_inv_ip [simp]:
"ξ' ∈ is_rerr ξ ⟹ ip ξ' = ip ξ"
"ξ' ∈ is_rrep ξ ⟹ ip ξ' = ip ξ"
"ξ' ∈ is_rqreq ξ ⟹ ip ξ' = ip ξ"
"ξ' ∈ is_pkt ξ ⟹ ip ξ' = ip ξ"
"ξ' ∈ is_newpkt ξ ⟹ ip ξ' = ip ξ"

unfolding is_msg_defs
by (cases "msg ξ",clarsimp+)+

lemma is_msg_inv_sn [simp]:
"ξ' ∈ is_rerr ξ ⟹ sn ξ' = sn ξ"
"ξ' ∈ is_rrep ξ ⟹ sn ξ' = sn ξ"
"ξ' ∈ is_rqreq ξ ⟹ sn ξ' = sn ξ"
"ξ' ∈ is_pkt ξ ⟹ sn ξ' = sn ξ"
"ξ' ∈ is_newpkt ξ ⟹ sn ξ' = sn ξ"

unfolding is_msg_defs
by (cases "msg ξ",clarsimp+)+

lemma is_msg_inv_rt [simp]:
"ξ' ∈ is_rerr ξ ⟹ rt ξ' = rt ξ"
"ξ' ∈ is_rrep ξ ⟹ rt ξ' = rt ξ"
"ξ' ∈ is_rqreq ξ ⟹ rt ξ' = rt ξ"
"ξ' ∈ is_pkt ξ ⟹ rt ξ' = rt ξ"
"ξ' ∈ is_newpkt ξ ⟹ rt ξ' = rt ξ"

unfolding is_msg_defs
by (cases "msg ξ",clarsimp+)+

lemma is_msg_inv_rreqs [simp]:
"ξ' ∈ is_rerr ξ ⟹ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_rrep ξ ⟹ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_rqreq ξ ⟹ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_pkt ξ ⟹ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_newpkt ξ ⟹ rreqs ξ' = rreqs ξ"

unfolding is_msg_defs
by (cases "msg ξ",clarsimp+)+

lemma is_msg_inv_store [simp]:
"ξ' ∈ is_rerr ξ ⟹ store ξ' = store ξ"
"ξ' ∈ is_rrep ξ ⟹ store ξ' = store ξ"
"ξ' ∈ is_rqreq ξ ⟹ store ξ' = store ξ"
"ξ' ∈ is_pkt ξ ⟹ store ξ' = store ξ"
"ξ' ∈ is_newpkt ξ ⟹ store ξ' = store ξ"

unfolding is_msg_defs
by (cases "msg ξ",clarsimp+)+

lemma is_msg_inv_sip [simp]:
"ξ' ∈ is_pkt ξ ⟹ sip ξ' = sip ξ"
"ξ' ∈ is_newpkt ξ ⟹ sip ξ' = sip ξ"

unfolding is_msg_defs
by (cases "msg ξ",clarsimp+)+

0.4.3 The protocol process

datatype pseqp =
| PAodv
| PNewPkt
| PPkt
| PRreq
fun nat_of_seqp :: "pseqp ⇒ nat"
where
  "nat_of_seqp PAodv = 1"
| "nat_of_seqp PPkt = 2"
| "nat_of_seqp PNewPkt = 3"
| "nat_of_seqp PRreq = 4"
| "nat_of_seqp PRrep = 5"
| "nat_of_seqp PRerr = 6"

instantiation "pseqp" :: ord begin
definition less_eq_seqp [iff]: "l1 ≤ l2 = (nat_of_seqp l1 ≤ nat_of_seqp l2)"
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
instance .. end

abbreviation AODV where
  "AODV ≡ λ_. [clear_locals] call(PAodv)"

abbreviation PKT where
  "PKT args ≡
  [ξ. let (data, dip, oip) = args ξ in
  (clear_locals ξ) [] data := data, dip := dip, oip := oip ]
call(PPkt)"

abbreviation NEWPKT where
  "NEWPKT args ≡
  [ξ. let (data, dip) = args ξ in
  (clear_locals ξ) [] data := data, dip := dip ]
call(PNewPkt)"

abbreviation RREQ where
  "RREQ args ≡
  [ξ. let (hops, rreqid, dip, dsn, dsk, oip, osn, sip) = args ξ in
  (clear_locals ξ) [] hops := hops, rreqid := rreqid, dip := dip,
  dsn := dsn, dsk := dsk, oip := oip,
  osn := osn, sip := sip ]
call(PRreq)"

abbreviation RREP where
  "RREP args ≡
  [ξ. let (hops, dip, dsn, oip, sip) = args ξ in
  (clear_locals ξ) [] hops := hops, dip := dip, dsn := dsn,
  oip := oip, sip := sip ]
call(PRrep)"

abbreviation RERR where
  "RERR args ≡
  [ξ. let (dests, sip) = args ξ in
  (clear_locals ξ) [] dests := dests, sip := sip ]
call(PRerr)"

fun Γ_AODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
  "Γ_AODV PAodv = labelled PAodv (
receive(\lambda msg'. \xi . \xi == msg \Rightarrow msg')

\text{receive}(\lambda msg'. \xi . \xi == msg \Rightarrow msg')

(\text{is_newpkt}) \text{NEWPKT}(\lambda \xi. (data \xi, ip \xi))

(\text{is_PKT}) \text{PKT}(\lambda \xi. (data \xi, dip \xi, oip \xi))

(\text{is_rreq})

[\xi, \xi (rt := update (rt \xi) (sip \xi) (0, unk, val, 1, sip \xi, \{\}) \Rightarrow]
\text{RREQ}(\lambda \xi. (hops \xi, rreqid \xi, dip \xi, dsn \xi, oip \xi, osn \xi, sip \xi))

(\text{is_rrep})

[\xi, \xi (rt := update (rt \xi) (sip \xi) (0, unk, val, 1, sip \xi, \{\}) \Rightarrow]
\text{RREP}(\lambda \xi. (hops \xi, dip \xi, dsn \xi, oip \xi, sip \xi))

(\text{is_rerr})

[\xi, \xi (rt := update (rt \xi) (sip \xi) (0, unk, val, 1, sip \xi, \{\}) \Rightarrow]
\text{RERR}(\lambda \xi. (dests \xi, sip \xi))

(\text{\lambda \xi. \{ \xi (\xi (dip := dip) \land dip \in qD(store \xi) \cap vD(rt \xi) \})
[\xi, \xi (data := \hd(\sigma_{\text{queuel}(store \xi, dip \xi)) \}][\muunicast(\lambda \xi. \{ (nhop (rt \xi) (dip \xi), \lambda \xi. \text{pkt}(data \xi, dip \xi, ip \xi)) .
[\xi, \xi (store := the (drop (dip \xi) (store \xi)) \}][\text{AODV}()])

(\text{\lambda \xi. \{ \xi (\xi (dip := dip) \land dip \in qD(store \xi) \land vD(rt \xi) \}][\xi, \xi (store := \text{unsetRRF}(store \xi, dip \xi)) \}][\xi, \xi (sn := \text{inc}(sn \xi)) \}][\xi, \xi (rreqid := \text{nrreqid}(rreqs \xi, ip \xi)) \}][\xi, \xi (rreqs := \text{rreqs}(\xi \cup \{(ip \xi, rreqid \xi)\})) \}][\mubroadcast(\lambda \xi. \text{rreq}(0, rreqid \xi, dip \xi, snq(rt \xi, dip \xi), sqnf(rt \xi, dip \xi), ip \xi, sn \xi, oip \xi)). \text{AODV}())]

/ "\text{\Gamma_{AODV} PNewPkt = labelled PNewPkt (}
\langle \xi, dip \xi = ip \xi \rangle
\text{deliver}(\lambda \xi. data \xi). \text{AODV}())

(\text{\xi. dip \xi \neq ip \xi})

[\xi, \xi (store := \text{add(data \xi, dip \xi, store \xi)) \}][\text{AODV}())]

/ "\text{\Gamma_{AODV} Ppkt = labelled Ppkt (}
\langle \xi, dip \xi = ip \xi \rangle
\text{deliver}(\lambda \xi. data \xi). \text{AODV}())

(\text{\xi. dip \xi \neq ip \xi})

(\text{\xi. dip \xi \in vD(rt \xi)})

\text{unicast(\lambda \xi. \{ (nhop (rt \xi) (dip \xi), \lambda \xi. \text{pkt}(data \xi, dip \xi, oip \xi)) \}. \text{AODV}())

(\text{\xi. dip \xi \in vD(rt \xi) \land nhop (rt \xi) (dip \xi)}
\text{\Rightarrow Some(inc(sqn(rt \xi, dip \xi))) else None}) \}][\xi, \xi (rt := \text{invalidate}(rt \xi) (dests \xi)) \}][\xi, \xi (store := \text{setRRF}(store \xi, dests \xi)) \}][\xi, \xi (pre := \bigcup \{ the (prec (rt \xi) (dip \xi)) \land rip \in \text{dom(dests \xi)} \}) \}]][\xi, \xi (dests := (\lambda rip. if ((dests \xi) (r) \neq None \land the (prec (rt \xi) (dip \xi)) \neq \}) \land (dests \xi) (rip else None)) \}][\mu groupcast(\lambda \xi. \text{pre}(\xi, \lambda \xi. \text{rerr}(dests \xi, ip \xi)). \text{AODV}())]

(\text{\xi. dip \xi \notin vD(rt \xi) \})

(\text{\xi. dip \xi \in id(rt \xi))
\text{unicast(\lambda \xi. \{ (nhop (rt \xi) (dip \xi), \lambda \xi. \text{pkt}(data \xi, dip \xi, oip \xi)) \}. \text{AODV}())

(\text{\xi. dip \xi \in id(rt \xi) \land nhop (rt \xi) (dip \xi) \Rightarrow Some(inc(sqn(rt \xi, dip \xi))) else None}) \}][\xi, \xi (rt := \text{invalidate}(rt \xi) (dests \xi)) \}][\xi, \xi (store := \text{setRRF}(store \xi, dests \xi)) \}][\xi, \xi (pre := \bigcup \{ the (prec (rt \xi) (dip \xi)) \land rip \in \text{dom(dests \xi)} \}) \}]][\xi, \xi (dests := (\lambda rip. if ((dests \xi) (r) \neq None \land the (prec (rt \xi) (dip \xi)) \neq \}) \land (dests \xi) (rip else None)) \}][\mu groupcast(\lambda \xi. \text{pre}(\xi, \lambda \xi. \text{rerr}(dests \xi, ip \xi)). \text{AODV}())]

(\text{\xi. dip \xi \notin id(rt \xi) \})

(\text{\xi. dip \xi \in iD(rt \xi))
\text{groupcast(\lambda \xi. \{ the (prec (rt \xi) (dip \xi)), 
\lambda \xi. \text{rerr}(\{dip \xi \mapsto sqn(rt \xi, dip \xi), ip \xi\}). \text{AODV}())

(\text{\xi. dip \xi \notin iD(rt \xi) \})

26
AODV(
)
)

Γ AODV PRreq = labelled PRreq ( ⟨ξ, (oip ξ, rreqid ξ) ∈ rreqs ξ⟩
AODV()
≥ ⟨ξ (oip ξ, rreqid ξ) ∈ rreqs ξ⟩

[ξ, ξ, (rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, hops ξ + 1, sip ξ, {}))]

[ξ, ξ, (rreqs := rreqs ξ ∪ {(oip ξ, rreqid ξ) (oip ξ) = ip ξ})]

(ξ, dip ξ = ip ξ)

unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rreq(0, dip ξ, sn ξ, oip ξ, ip ξ)).AODV()

> ⟨ξ, ξ, dests := (λrip. if ((dests ξ) (rip ξ) (ip ξ) ∧ nhop (rt ξ) (rip ξ) = nhop (rt ξ) (oip ξ))
then Some (inc (sqn (rt ξ) (rip ξ))) else None) ξ⟩

> ⟨ξ, ξ, (rt := invalidate (rt ξ) (dests ξ) ξ)⟩

> ⟨ξ, ξ, (store := setRFF (store ξ) (dests ξ) ξ)⟩

> ⟨ξ, ξ, (pre := dom (dests ξ) ξ)⟩

> ⟨ξ, ξ, dests := (λrip (dests ξ) rip ≠ None ∧ the (prec (rt ξ) rip) ξ)⟩

> ⟨ξ, ξ, dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (prec (rt ξ) rip) ξ)
then (dests ξ) rip else None) ξ)⟩

> groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()

≥ ⟨ξ, dip ξ ≠ ip ξ⟩

(ξ, dip ξ ∈ vD (rt ξ) ξ ∧ dsn ξ ≤ sqn (rt ξ) (oip ξ) ξ ∧ sqnf (rt ξ) (oip ξ) = kno)

[ξ, ξ, (rt := the (addpreRT (rt ξ) (oip ξ) (dip ξ) {sip ξ})) ξ]

[ξ, ξ, (rt := the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ)})) ξ]

unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, (nhop (rt ξ) (dip ξ)), dip ξ, sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).AODV()

> ⟨ξ, ξ, (dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (prec (rt ξ) rip) ξ)
then (dests ξ) rip else None) ξ)⟩

> ⟨ξ, ξ, store := setRFF (store ξ) (dests ξ) ξ)⟩

> ⟨ξ, (pre := dom (dests ξ) ξ)⟩

> ⟨ξ, (dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (prec (rt ξ) rip) ξ)
then (dests ξ) rip else None) ξ)⟩

> groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()

≥ ⟨ξ, dip ξ ≠ ip ξ⟩

(ξ, dip ξ ∈ vD (rt ξ) ξ ∧ dsn ξ ≤ sqn (rt ξ) (oip ξ) ξ ∧ sqnf (rt ξ) (oip ξ) = unk)

[ξ, ξ, (rt := the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ) (oip ξ)})) ξ]

unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, max (sqn (rt ξ) (dip ξ)), dsk ξ, oip ξ, osn ξ, ip ξ)).AODV()

)
declare \( \Gamma_{AODV} \) simps [simp del, code del]
lemmas \( \Gamma_{AODV} \_simps \) [simp, code] = \( \Gamma_{AODV} \) simps [simplified]

fun \( \Gamma_{AODV} \_skeleton \) where
  "\( \Gamma_{AODV} \_skeleton \) PAodv = seqp_skeleton (\( \Gamma_{AODV} \) PAodv)"
  / "\( \Gamma_{AODV} \_skeleton \) PNewPkt = seqp_skeleton (\( \Gamma_{AODV} \) PNewPkt)"
  / "\( \Gamma_{AODV} \_skeleton \) PPkt = seqp_skeleton (\( \Gamma_{AODV} \) PPkt)"
  / "\( \Gamma_{AODV} \_skeleton \) PRreq = seqp_skeleton (\( \Gamma_{AODV} \) PRreq)"
  / "\( \Gamma_{AODV} \_skeleton \) PRrep = seqp_skeleton (\( \Gamma_{AODV} \) PRrep)"
  / "\( \Gamma_{AODV} \_skeleton \) PRerr = seqp_skeleton (\( \Gamma_{AODV} \) PRerr)"

lemma \( \Gamma_{AODV} \_skeleton \) wf [simp]:
  "wellformed \( \Gamma_{AODV} \_skeleton \)"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') \notin ctermsl (\( \Gamma_{AODV} \_skeleton \) pn)"
  by (cases pn) simp_all
qed

declare \( \Gamma_{AODV} \_skeleton \) simps [simp del, code del]
lemmas \( \Gamma_{AODV} \_skeleton \) simps [simp, code] = \( \Gamma_{AODV} \) simps seqp_skeleton.simps

lemma aodv_proc_cases [dest]:
  fixes p pn
  shows "p \in ctermsl (\( \Gamma_{AODV} \) pn) \implies
   (p \in ctermsl (\( \Gamma_{AODV} \) PAodv) \lor
    p \in ctermsl (\( \Gamma_{AODV} \) PNewPkt) \lor
    p \in ctermsl (\( \Gamma_{AODV} \) PPkt) \lor
    p \in ctermsl (\( \Gamma_{AODV} \) PRreq) \lor
    p \in ctermsl (\( \Gamma_{AODV} \) PRrep) \lor
    p \in ctermsl (\( \Gamma_{AODV} \) PRerr))"
  by (cases pn) simp_all

definition \( \sigma_{AODV} \) :: "ip \Rightarrow (state \times (state, msg, pseqp, pseqp label) seqp) set"
where "\( \sigma_{AODV} \) i \equiv \{(aodv_init i, \( \Gamma_{AODV} \) PAodv)\}"
abbreviation paodv 
::\text{"ip}\Rightarrow (\text{state} \times \text{state}, \text{msg}, \text{seqp}, \text{pseqp label}) \text{ seqp, msg seq_action) automaton"}

where
\text{"paodv i \equiv (init = AODV i, trans = seqp sos AODV)"}

lemma aodv_trans: "trans (paodv i) = seqp sos AODV"
by simp

lemma aodv_control_within [simp]: "control_within AODV (init (paodv i))"
unfolding AODV_def by (rule control_withinI) (auto simp del: AODV_sims)

lemma aodv_wf [simp]:
"wellformed AODV"
proof (rule, intro allI)
fix pn pn'
show "call(pn') \notin \text{stermsl (AODV pn)"
by (cases pn) simp_all
qed

lemmas aodv_labels_not_empty [simp] = labels_not_empty [OF aodv_wf]

lemma aodv_ex_label [intro]: \(\exists l. l \in \text{labels AODV p}\)
by (metis aodv_labels_not_empty all_not_in_conv)

lemma aodv_ex_labelE [elim]:
assumes "\(\forall l \in \text{labels AODV p. P l p}\)"
and "\(\exists p l. P l p = \Rightarrow Q\)"
shows "Q"
using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "simple_labels AODV"
proof
fix pn p
assume "p \in \text{subterms(AODV pn)"
thus "\(\exists !l. \text{labels AODV p} = \{l\}\)"
by (cases pn) (simp_all cong: seqp_congs | elim disjE)+
qed

lemma AODV_labels [simp]: "(\xi, p) \in AODV i \Rightarrow \text{labels AODV p} = \{PAodv::0\}"
unfolding AODV_def by simp

lemma aodv_init_kD_empty [simp]:
"(\xi, p) \in AODV i \Rightarrow kD (rt \xi) = {}"
unfolding AODV_def kD_def by simp

lemma aodv_init_sip_not_ip [simp]: "(sip (aodv_init i) = i)" by simp

lemma aodv_init_sip_not_ip' [simp]:
assumes "(\xi, p) \in AODV i"
shows "sip \xi \neq ip \xi"
using assms unfolding AODV_def by simp

lemma aodv_init_sip_not_i [simp]:
assumes "(\xi, p) \in AODV i"
shows "sip \xi \neq i"
using assms unfolding AODV_def by simp

lemma clear_locals_sip_not_ip':
assumes "ip \xi = i"
shows "(sip (clear_locals \xi) = i)"
using assms by auto

Stop the simplifier from descending into process terms.
declare seqcongs [cong]

Configure the main invariant tactic for AODV.

declare Γ AODV_simps [cterms_env]
  aodv_proc_cases [ctermsl_cases]
  seq_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms_intros]
  seq_step_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms_intros]

end

0.5 Invariant assumptions and properties

theory Aodv_Predicates
imports Aodv
begin

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

abbreviation not_Pkt :: "msg ⇒ bool" where "not_Pkt m ≡ case m of Pkt _ _ _ ⇒ False | _ ⇒ True"

definition msg_sender :: "msg ⇒ ip" where "msg_sender m ≡ case m of Rreq _ _ _ _ _ _ _ ipc ⇒ ipc
  | Rrep _ _ _ _ ipc ⇒ ipc
  | Rerr _ ipc ⇒ ipc
  | Pkt _ _ ipc ⇒ ipc"

lemma msg_sender_simps [simp]:
  "∀hops rreqid dip dsn dsk oip osn sip. msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip) = sip"
  "∀hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) = sip"
  "∀dests sip. msg_sender (Rerr dests sip) = sip"
  "∀d dip sip. msg_sender (Pkt d dip sip) = sip"

unfolding msg_sender_def by simp_all

definition msg_zhops :: "msg ⇒ bool" where "msg_zhops m ≡ case m of Rreq hopsc _ dipc _ _ _ oipc _ sipc ⇒ hopsc = 0 −→ oipc = sipc
  | Rrep hopsc dipc _ _ sipc ⇒ hopsc = 0 −→ dipc = sipc
  | _ ⇒ True"

lemma msg_zhops_simps [simp]:
  "∀hops rreqid dip dsn dsk oip osn sip. msg_zhops (Rreq hops rreqid dip dsn dsk oip osn sip) = (hops = 0 −→ oip = sip)"
  "∀hops dip dsn oip sip. msg_zhops (Rrep hops dip dsn oip sip) = (hops = 0 −→ dip = sip)"
  "∀dests sip. msg_zhops (Rerr dests sip) = True"
  "∀d dip sip. msg_zhops (Pkt d dip sip) = True"

unfolding msg_zhops_def by simp_all

definition rreq_rrep_sn :: "msg ⇒ bool" where "rreq_rrep_sn m ≡ case m of Rreq _ _ _ _ _ _ osnc _ ⇒ osnc ≥ 1
  | Rrep _ _ dsnc _ ⇒ dsnc ≥ 1
  | _ ⇒ True"

lemma rreq_rrep_sn_simps [simp]:
  "∀hops rreqid dip dsn dsk oip osn sip. rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip) = (osn ≥ 1)"
  "∀hops dip dsn oip sip. rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn ≥ 1)"
  "∀dests sip. rreq_rrep_sn (Rerr dests sip) = True"
  "∀d dip. rreq_rrep_sn (Pkt d dip) = True"
definition rreq_rrep_fresh :: "rt ⇒ msg ⇒ bool"
where "rreq_rrep_fresh crt m ≡ case m of Rreq hopsc _ _ _ _ oipc osnc ipcc ⇒ (ipcc ≠ oipc ---→
\(\text{oipc} \in \text{kD}(\text{crt}) \land (\text{sqn} \text{crt} \text{ oipc} > \text{osnc})
\lor (\text{sqn} \text{crt} \text{ oipc} = \text{osnc}
\land (\text{dhops} \text{crt} \text{ oipc} \leq \text{hopsc}
\land (\text{flag} \text{crt} \text{ oipc} = val)))
\lor Rrep hopsc dipc dsnc _ ipcc ⇒ (ipcc ≠ dipc ---→
\(\text{dipc} \in \text{kD}(\text{crt})
\land (\text{sqn} \text{crt} \text{ dipc} = \text{dsnc}
\land (\text{dhops} \text{crt} \text{ dipc} = \text{hopsc}
\land (\text{flag} \text{crt} \text{ dipc} = val)))
\lor _ ⇒ True"

lemma rreq_rrep_fresh [simp]:
"\(\forall \text{hops} \text{rreqid} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}.
\text{rreq_rrep_fresh} \text{crt} (\text{Rreq} \text{ hopsc} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}) =
(\text{sip} ≠ \text{oip} ---→ \text{oipc} \in \text{kD}(\text{crt})
\lor (\text{sqn} \text{crt} \text{ oipc} = \text{osn}
\land (\text{dhops} \text{crt} \text{ oipc} \leq \text{hopsc}
\land (\text{flag} \text{crt} \text{ oipc} = val)))
\lor Rrep hopsc dipc dsnc _ ipcc ⇒ (ipcc ≠ dipc ---→
\(\text{dipc} \in \text{kD}(\text{crt})
\land (\text{sqn} \text{crt} \text{ dipc} = \text{dsnc}
\land (\text{dhops} \text{crt} \text{ dipc} = \text{hopsc}
\land (\text{flag} \text{crt} \text{ dipc} = val)))
\lor _ ⇒ True"

lemma rerr_invalid [simp]:
"\(\forall \text{hops} \text{rreqid} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}.
\text{rerr_invalid} \text{crt} (\text{Rreq} \text{ hopsc} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}) = True"
"\(\forall \text{hops} \text{ dip} \text{ dsn} \text{ oip} \text{ sip}.
\text{rerr_invalid} \text{crt} (\text{Rreq} \text{ hopsc} \text{ dip} \text{ dsn} \text{ oip} \text{ sip}) = True"
"\(\forall \text{dests} \text{ sip}.
\text{rerr_invalid} \text{crt} (\text{Rerr} \text{ dests} \text{ sip}) = True"
"\(\forall \text{d} \text{ dip}.
\text{rerr_invalid} \text{crt} (\text{Newpkt} \text{ d} \text{ dip}) = True"
"\(\forall \text{d} \text{ dip}.
\text{rerr_invalid} \text{crt} (\text{Pkt} \text{ d} \text{ dip} \text{ sip}) = True"

definition initmissing :: "(\text{nat} ⇒ \text{state} \text{option}) \times \text{a} ⇒ (\text{nat} ⇒ \text{state}) \times \text{a}"
where "initmissing σ = (\lambda i. \text{case} \ (\text{fst} σ) \ i \text{ of} \text{None} ⇒ \text{aodv_init} \ i | \text{Some} s ⇒ s, \text{snd} σ)"

lemma not_in_net_ips_fst_init_missing [simp]:
assumes "\text{i} \notin \text{net_ips} σ"
shows "\text{fst} (\text{initmissing} \ (\text{netgmap} \ \text{fst} σ)) \ i = \text{aodv_init} \ i"
using assms unfolding initmissing_def by simp

lemma fst_initmissing_netgmap_pair_fst [simp]:
"\text{fst} (\text{initmissing} \ (\text{netgmap} \ (\\lambda (p, q). \text{fst} \ (\text{id} \ p), \text{snd} \ (\text{id} \ p), \text{q}) \ s))
= \text{fst} (\text{initmissing} \ (\text{netgmap} \ \text{fst} \ s))"

unfolding initmissing_def by auto
We introduce a streamlined alternative to \texttt{initmissing} with \texttt{netgmap} to simplify invariant statements and thus facilitate their comprehension and presentation.

\textbf{lemma} \texttt{fst_initmissing_netgmap_default_aodv_init_netlift}:

"\texttt{fst (initmissing (netgmap \texttt{fst} \texttt{s}))) = default aodv_init (netlift \texttt{fst} \texttt{s})}"

unfolding \texttt{initmissing_def} \texttt{default_def}

by (simp add: \texttt{fst_netgmap_netlift} del: \texttt{One_nat_def})

\textbf{definition}

\texttt{netglobal :: "((nat ⇒ state) ⇒ bool) ⇒ ((state × 'b) × 'c) net_state ⇒ bool"}

where

"\texttt{netglobal \texttt{P} \equiv (λs. \texttt{P} (default aodv_init (netlift \texttt{fst} \texttt{s})))}"

end

\section{Quality relations between routes}

\textbf{theory} \texttt{Fresher}

\textbf{imports} \texttt{Aodv_Data}

\textbf{begin}

\subsection{Net sequence numbers}

\textbf{On individual routes}

\textbf{definition}

\texttt{nsqn_r :: "r ⇒ sqn"}

where

"\texttt{nsqn_r \equiv if π_4(r) = val ∨ π_2(r) = 0 then π_2(r) else (π_2(r) - 1)}"

\textbf{lemma} \texttt{nsqn_r_def'}:

"\texttt{nsqn_r \equiv (if π_4(r) = inv then π_2(r) - 1 else π_2(r))}"

unfolding \texttt{nsqn_r_def} by simp

\textbf{lemma} \texttt{nsqn_r_zero [simp]}:

"\texttt{dsn dsk flag hops nhip pre. nsqn_r (0, dsk, flag, hops, nhip, pre) = 0}"

unfolding \texttt{nsqn_r_def} by clarsimp

\textbf{lemma} \texttt{nsqn_r_val [simp]}:

"\texttt{dsn dsk hops nhip pre. nsqn_r (dsn, dsk, val, hops, nhip, pre) = dsn}"

unfolding \texttt{nsqn_r_def} by clarsimp

\textbf{lemma} \texttt{nsqn_r_inv [simp]}:

"\texttt{dsn dsk hops nhip pre. nsqn_r (dsn, dsk, inv, hops, nhip, pre) = dsn - 1}"

unfolding \texttt{nsqn_r_def} by clarsimp

\textbf{lemma} \texttt{nsqn_r_lte_dsn [simp]}:

"\texttt{dsn dsk flag hops nhip pre. nsqn_r (dsn, dsk, flag, hops, nhip, pre) ≤ dsn}"

unfolding \texttt{nsqn_r_def} by clarsimp

\textbf{On routes in routing tables}

\textbf{definition}

\texttt{nsqn :: "rt ⇒ ip ⇒ sqn"}

where

"\texttt{nsqn \equiv λrt dip. \texttt{case σRoute(rt, dip) of \texttt{None \Rightarrow 0 | \texttt{Some r ⇒ nsqn_r(r)\texttt{)}}}}"

\textbf{lemma} \texttt{nsqn_sqn_def}:

"\texttt{rt dip. nsqn rt dip = (if flag rt dip = Some val ∨ sqn rt dip = 0 then sqn rt dip else sqn rt dip - 1)}"

unfolding \texttt{nsqn_def} \texttt{sqn_def} by (clarsimp split: option.split)

\textbf{lemma} \texttt{not_in_kD_nsqn [simp]}:

assumes "\texttt{dip \notin kD(rt)}"

shows "\texttt{nsqn rt dip = 0}"
lemma kD_nsqn:
  assumes "dip ∈ kD(rt)"
  shows "nsqn rt dip = nsqn r (the (route rt dip))"
  using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

lemma nsqnr_r_flag_pred [simp, intro]:
  fixes dsn dsk flag hops nhip pre
  assumes "P (nsqn (dsn, dsk, val, hops, nhip, pre))"
  and "P (nsqn (dsn, dsk, inv, hops, nhip, pre))"
  shows "P (nsqn (dsn, dsk, flag, hops, nhip, pre))"
  using assms by (cases flag) auto

lemma nsqn_r_addpreRT_inv [simp]:
  \(\forall rt\ dip\ npre\ dip'.\ dip \in kD(rt) \implies nsqn (the (the (addpreRT rt dip npre) dip')) = nsqn (the (rt dip'))\)
  unfolding addpreRT_def nsqn_r_def
  by (frule kD_Some) (clarsimp split: option.split)

lemma sqn_nsqn:
  "\(\forall rt\ dip\. sqn rt dip - 1 \leq nsqn rt dip\)"
  unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma val_nsqn_sqn [elim, simp]:
  assumes "ip \in kD(rt)"
  and "the (flag rt ip) = val"
  shows "nsqn rt ip = sqn rt ip"
  using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip \in iD(rt)"
  and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
  using assms unfolding nsqn_sqn_def by auto

lemma nsqn_update_changed_kno_val [simp]:
  "\(\forall rt\ ip\ dsn\ dsk\ hops\ nhip.\ rt \neq update rt ip (dsn, kno, val, hops, nhip, {}) \implies nsqn (update rt ip (dsn, kno, val, hops, nhip, {})) ip = dsn\)"
  unfolding nsqn_def_update_def
  by (clarsimp simp: kD_nsqn split: option.split_asm option.split if_split_asm)

33
lemma nsqn_addpreRT_inv [simp]:
"∀ rt dip npre dip'. dip ∈ kD(rt) ⇒
nsqn (the (addpreRT rt dip npre)) dip' = nsqn rt dip'"
unfolding addpreRT_def nsqn_def nsqn_def
by (frule kD_Some) (clarsimp split: option.split)

lemma nsqn_update_other [simp]:
fixes dsn dsk flag hops dip nhip pre rt ip
assumes "dip ≠ ip"
shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip, pre)) dip = nsqn rt dip"
using assms unfolding nsqn_def
by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
assumes "dip ∈ kD(rt)"
and "dests dip = Some rsn"
shows "nsqn (invalidate rt dests) dip = rsn - 1"
using assms proof -
from assms obtain dsk hops nhip pre
where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip, pre)"
unfolding invalidate_def by auto
moreover from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp
ultimately show ?thesis
using ⟨dests dip = Some rsn⟩ by simp
qed

lemma nsqn_invalidate_other [simp]:
assumes "dip ∈ kD(rt)"
and "dip /∈ dom dests"
shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
using assms by (clarsimp simp add: kD_nsqn)

0.6.2 Comparing routes

definition
fresher :: "r ⇒ r ⇒ bool" ("(_/ ⊑ _)" [51, 51] 50)
where
"fresher r r' ≡ ((nsqn r < nsqn r') ∨ (nsqn r = nsqn r' ∧ π₅(r) ≥ π₅(r')))"

lemma fresherI1 [intro]:
assumes "nsqn r < nsqn r'"
shows "r ⊑ r'"
unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
assumes "nsqn r = nsqn r'"
and "π₅(r) ≥ π₅(r')"
shows "r ⊑ r'"
unfolding fresher_def using assms by simp

lemma fresherI [intro]:
assumes "(nsqn r < nsqn r') ∨ (nsqn r = nsqn r' ∧ π₅(r) ≥ π₅(r'))"
shows "r ⊑ r'"
unfolding fresher_def using assms .

lemma fresherE [elim]:
assumes "r ⊑ r'"
and "nsqn r < nsqn r'" "P r r'"
and "nsqn r = nsqn r' ∧ π₅(r) ≥ π₅(r')" "P r r'"
shows "P r r'"
using assms unfolding fresher_def by auto

lemma fresher_refl [simp]: "r ⊑ r"
lemma freshness_trans [elim, trans]:
  "[ x ⊑ y; y ⊑ z ] ⇒ x ⊑ z"
unfolding freshness_def by auto

lemma not_freshness_trans [elim, trans]:
  "[ ¬(x ⊑ y); ¬(z ⊑ x) ] ⇒ ¬(z ⊑ y)"
unfolding freshness_def by auto

lemma freshness_dsn_flag_hops_const [simp]:
  fixes dsn dsk dsk' flag hops nhip nhip' pre pre'
  shows "(dsn, dsk, flag, hops, nhip, pre) ⊑ (dsn, dsk', flag, hops, nhip', pre')"
unfolding freshness_def by (cases flag) simp_all

lemma addpre_freshness [simp]: "⋀ r npre. r ⊑ (addpre r npre)"
by clarsimp

0.6.3 Comparing routing tables

definition rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where "rt_fresher ≡ λ dip rt rt'. (the (σ_route(rt, dip))) ⊑ (the (σ_route(rt', dip)))"

abbreviation rt_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊑ _/)" [51, 999, 51] 50)
  where "rt1 ⊑ i rt2 ≡ rt_fresher i rt1 rt2"

lemma rt_fresher_def':
  "(rt1 ⊑ rt2) = (nsqn rt1 dip < nsqn rt2 dip
   ∨ nsqn rt1 dip = nsqn rt2 dip ∧ π_5 (the (dhops rt1 dip)) ≥ (the (dhops rt2 dip)))"
unfolding rt_fresher_def freshness_def by (rule refl)

lemma single_rt_fresher [intro]:
  assumes "the (rt1 ip) ⊑ the (rt2 ip)"
  shows "rt1 ⊑ ip rt2"
  using assms unfolding rt_fresher_def .

lemma rt_fresher_single [intro]:
  assumes "rt1 ⊑ ip rt2"
  shows "the (rt1 ip) ⊑ the (rt2 ip)"
  using assms unfolding rt_fresher_def .

lemma rt_fresher_def2:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  shows "(rt1 ⊑ dip rt2) = (nsqn rt1 dip < nsqn rt2 dip
   ∨ (nsqn rt1 dip = nsqn rt2 dip
   ∧ the (dhops rt1 dip) ≥ the (dhops rt2 dip)))"
  using assms unfolding rt_fresher_def freshness_def by (simp add: kD_nsqn proj5_eq_dhops)

lemma rt_fresherI1 [intro]:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1 ⊑ dip rt2"
  unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherI2 [intro]:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "nsqn rt1 dip = nsqn rt2 dip"
and "the (dhaps rt1 dip) ≥ the (dhaps rt2 dip)"
shows "rt1 ⊑ dip rt2"
unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt_fresherE [elim]:
  assumes "rt1 ⊑ dip rt2"
  and "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "[ nsqn rt1 dip < nsqn rt2 dip ] ⇒ P rt1 rt2 dip"
  and "[ nsqn rt1 dip = nsqn rt2 dip;
          the (dhaps rt1 dip) ≥ the (dhaps rt2 dip)] ⇒ P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
using assms(4-5) by auto

lemma rt_fresherE [elim]:
  assumes "rt1 ⊑ dip rt2"
  and "dip ∈ kD(rt2)"
  and "dip ∈ kD(rt1)"
  and "[ nsqn rt1 dip < nsqn rt2 dip ] ⇒ P rt1 rt2 dip"
  and "[ nsqn rt1 dip = nsqn rt2 dip;
          the (dhaps rt1 dip) ≥ the (dhaps rt2 dip)] ⇒ P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "rt ⊑ dip rt"
unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt3"
  shows "rt1 ⊑ dip rt3"
using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip) ⊑ r"
  shows "rt ⊑ dip (λip. if ip = dip then Some r else rt ip)"
using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip ⇒ rt ⇒ rt ⇒ bool" where
  "rt_fresh_as ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ (rt2 ⊑ dip rt1)"

abbreviation rt_fresh_as_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/≈_)") [51, 999, 51] 50
where
  "rt1 ≈_i rt2 ≡ rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: "⋀rt ip rt. rt ⊑ dip rt"
unfolding rt_fresher_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  "⋀rt1 rt2 rt3 dip. [ rt1 ⊑ dip rt2; rt2 ⊑ dip rt3 ] ⇒ rt1 ≈ dip rt3"
unfolding rt_fresh_as_def rt_fresher_def
by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt1"
  shows "rt1 ≈ dip rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "the (rt1 dip) ⊑ the (rt2 dip)"
  and "the (rt2 dip) ⊑ the (rt1 dip)"
  shows "rt1 ≈ dip rt2"
using assms unfolding rt_fresh_as_def
by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip ∈ kD(rt)"
  and "dip ∈ kD(rt')"
and "nsqn rt dip = nsqn rt' dip"
and "π_5(the (rt dip)) = π_5(the (rt' dip))"
shows "rt ≈_dip rt'"
proof
from assms(1-2,4) have dhops': "the (dhops rt' dip) ≤ the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
with assms(1-3) show "rt ⊑_dip rt'"
  by (rule rt_fresherI2)
next
from assms(1-2,4) have dhops: "the (dhops rt dip) ≤ the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
with assms(2,1) assms(3) [symmetric] show "rt' ⊑_dip rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1 ≈_dip rt2"
  and "[ rt1 ⊑_dip rt2; rt2 ⊑_dip rt1 ] ⟹ P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
  assumes "rt1 ≈_dip rt2"
  shows "rt1 ⊑_dip rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1 ≈_dip rt2"
  shows "rt2 ⊑_dip rt1"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1 ≈_dip rt2"
  shows "rt2 ≈_dip rt1"
using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes "¬ (rt1 ⊑_dip rt2)"
  shows "¬ (rt1 ≈_dip rt2)"
proof
  assume "rt1 ≈_dip rt2"
  hence "rt1 ⊑_dip rt2" ..
  with ⟨¬ (rt1 ⊑_dip rt2) ⟩ show False ..
qed

lemma not_rt_fresh_asI2 [intro]:
  assumes "¬ (rt2 ⊑_dip rt1)"
  shows "¬ (rt1 ≈_dip rt2)"
proof
  assume "rt1 ≈_dip rt2"
  hence "rt2 ⊑_dip rt1" ..
  with ⟨¬ (rt2 ⊑_dip rt1) ⟩ show False ..
qed

lemma not_single_rt_fresher [elim]:
  assumes "¬ (the (rt1 ip) ⊑ the (rt2 ip))"
  shows "¬(the (rt1 ⊑_ip rt2)"
proof
  assume "rt1 ⊑_ip rt2"
  hence "the (rt1 ip) ⊑ the (rt2 ip)" ..
  with ⟨¬(the (rt1 ⊑_ip rt2) ⟩ show False ..
qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
  assumes "¬ (rt1 ⊑ ip rt2)"
  shows "¬ (the (rt1 ip) ⊑ the (rt2 ip))"
proof
  assume "the (rt1 ip) ⊑ the (rt2 ip)"
  hence "rt1 ⊑ ip rt2" ..
  with ¬(rt1 ⊑ rt2) show False ..
qed

lemma rt_fresh_as_nsqnr:
  assumes "dip ∈ kD(rt1)" and "dip ∈ kD(rt2)" and "rt1 ≈ dip rt2"
  shows "nsqn_r (the (rt2 dip)) = nsqn_r (the (rt1 dip))"
using assms(3) unfolding rt_fresher_def
by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨dip ∈ kD(rt2)⟩]
  kD_nsqn [OF ⟨dip ∈ kD(rt1)⟩]
  kD_nsqn [OF ⟨dip ∈ kD(rt2)⟩])

lemma rt_fresher_mapupd [intro!]:
  assumes "dip ∈ kD(rt)" and "the (rt dip) ⊑ r"
  shows "rt ⊑ dip rt(dip ↦→ r)"
using assms unfolding rt_fresher_def
by simp

lemma rt_fresher_map_update_other [intro!]:
  assumes "dip ∈ kD(rt)" and "dip ≠ ip"
  shows "rt ⊑ dip rt(ip ↦→ r)"
using assms unfolding rt_fresher_def
by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip ∈ kD(rt)" and "dip ≠ ip"
  shows "rt ⊑ dip update rt ip r"
using assms unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip ∈ kD(rt)" and "the (dhops rt dip) ≥ 1" and "update_arg_wf r"
  shows "rt ⊑ dip update rt ip r"
proof (cases "dip = ip")
  assume "dip ≠ ip" with ⟨dip ∈ kD(rt)⟩ show ?thesis
  by (rule rt_fresher_update_other)
next
  assume "dip = ip"

  from ⟨dip ∈ kD(rt)⟩ obtain dsn, dsk, f, hops, nhip, pre
  where rt [simp]: "the (rt dip) = (dsn, dsk, f, hops, nhip, pre)"
  by (metis prod_cases6)
  with ⟨the (dhops rt dip) ≥ 1⟩ and ⟨dip ∈ kD(rt)⟩ have "hops ≥ 1"
  by (metis proj5_eq_dhops projects(4))
  from ⟨dip ∈ kD(rt)⟩ rt have [simp]: "sqn rt dip = dsn"
  and [simp]: "the (dhops rt dip) = hops"
  and [simp]: "the (flag rt dip) = f".
  by (simp add: sqn_def proj5_eq_dhops [symmetric]
    proj4_eq_flag [symmetric])

  from ⟨update_arg_wf r⟩ have "(dsn, dsk, f, hops, nhip, pre)"
proof (rule wf_r_cases)
  fix nhip pre
  from ⟨hops_n ≥ 1⟩ have "\pre'. (dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)
    ⊑ (dsn_n, unk, val, Suc 0, nhip, pre')"
  unfolding fresher_def sqn_def by (cases f_n) auto
thus 
  unfolding sqn_def
  by (cases f_n) auto
qed

next
  fix dsn :: sqn and hops nhip pre
  assume "0 < dsn"
  show "(dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)
    ⊑ (dsn_n, unk, val, hops, nhip, pre ∪ pre_n)"
  unfolding fresher_def by simp
next
  assume "dsn_n = dsn"
  with ⟨0 < dsn⟩ show "inv rt dip (dsn, dsk_n, f_n, hops_n, nhip_n, pre_n)
    ⊑ (dsn, dsk_n, f_n, hops_n, nhip_n, pre ∪ pre_n)"
  unfolding fresher_def by simp
qed

hence "rt ⊑ dip update rt dip r"
  by - (rule single_rt_fresher, simp)
with ⟨dip = ip⟩ show ?thesis by simp
qed

theorem rt_fresher_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
  and indests: "∀ rip ∈ dom(dests). rip ∈ vD(rt) ∧ sqn rt rip < the (dests rip)"
  shows "rt ⊑ dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)"
  assume "dip ∈ dom(dests)"
  thus ?thesis using ⟨dip ∈ kD(rt)⟩ by - (rule single_rt_fresher, simp)
next
  assume "dip ∈ dom(dests)"
  moreover with indests have "dip ∈ vD(rt)"
    and "sqn rt dip < the (dests dip)"
    by auto
  ultimately show ?thesis
    unfolding invalidate_def sqn_def
    by - (rule single_rt_fresher, auto simp: fresher_def)
qed

lemma nsqn_r_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
  and "dip ∈ dom(dests)"
  shows "nsqn_r (the (invalidate rt dests dip)) = the (dests dip) - 1"
  using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
  assumes "dip ∈ kD(rt)"

and "∀ rip∈dom(dests). rip∈vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
shows "rt ≈ dip invalidate rt dests"
proof (cases "dip∈dom(dests)"
  assume "dip∈dom(dests)"
  with ⟨dip∈kD(rt)⟩ have "dip∈kD(invalidate rt dests)" 
    by simp
  with ⟨dip∈kD(rt)⟩ show ?thesis 
    by (simp_all add: ⟨dip∉dom(dests)⟩)
next
  assume "dip∈dom(dests)"
  with ⟨dip∈kD(rt)⟩ have "dip∈kD(invalidate rt dests)" 
    by simp
  moreover then have "dip∈kD(invalidate rt dests)" 
    by simp
ultimately show ?thesis 
proof (rule nsqn_rt_fresh_asI)
  from ⟨dip∈vD(rt)⟩ have "nsqn rt dip = sqn rt dip" 
    by simp
  also have "sqn rt dip = nsqn (invalidate rt dests dip)" 
    proof
    from ⟨dip∈kD(rt)⟩ have "nsqn_r (the (invalidate rt dests dip)) = the (dests dip) - 1" 
      using ⟨dip∈dom(dests)⟩ by (rule nsqn_r_invalidate)
    with ⟨the (dests dip) = inc (sqn rt dip)⟩ 
      show "sqn rt dip = nsqn_r (the (invalidate rt dests dip))" 
        by simp
  qed
  also from ⟨dip∈kD(invalidate rt dests)⟩
    have "nsqn_r (the (invalidate rt dests dip)) = nsqn (invalidate rt dests dip)" 
      by simp
  finally show "nsqn rt dip = nsqn (invalidate rt dests dip)" 
    qed simp
qed

lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]

lemma rt_fresh_as_addpreRT [simp]: 
  assumes "ip∈kD(rt)"
  shows "rt ≈ dip the (addpreRT rt ip npre)" 
    using assms [THEN kD_Some] by (auto simp: addpreRT_def)
lemmas rt_fresher_addpreRT [simp] = rt_fresh_as_addpreRT [THEN rt_fresh_asD1]

0.6.4 Strictly comparing routing tables

definition rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool" 
where "rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ ¬(rt1 ≈ dip rt2)"

abbreviation rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_<_ \_)") [51, 999, 51] 50
where "rt1 < rt2 ≡ rt_strictly_fresher i rt1 rt2"

lemma rt_strictly_fresher_def'': 
  "rt1 < rt2 = ((rt1 < rt2) ∧ ¬(rt2 < rt1))"
unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

lemma rt_strictly_fresherI' [intro]: 
  assumes "rt1 < rt2" 
    and "¬(rt2 < rt1)"
  shows "rt1 < rt2" 
    using assms unfolding rt_strictly_fresher_def' by simp

lemma rt_strictly_fresherE' [elim]: 
  assumes "rt1 < rt2" 
    and "[ rt1 < rt2; ¬(rt2 < rt1) ] ⇒ P rt1 rt2 i"
shows "P rt1 rt2 i"
using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherI [intro]:
  assumes "rt1 ⊑ i rt2"
  and "¬ (rt1 ≈ i rt2)"
  shows "rt1 ⊏ i rt2"
unfolding rt_strictly_fresher_def using assms ..

lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]

lemma rt_strictly_fresherE [elim]:
  assumes "rt1 ⊏ i rt2"
  and "[ rt1 ⊑ i rt2; ¬ (rt1 ≈ i rt2) ] --- P rt1 rt2 i"
  shows "P rt1 rt2 i"
using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
  "rt1 ⊏ i rt2 =
  (nsqr_r (the (rt1 i)) < nsqr_r (the (rt2 i))
  ∨ (nsqr_r (the (rt1 i)) = nsqr_r (the (rt2 i)) ∧ π₅(the (rt1 i)) > π₅(the (rt2 i))))"
unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
  assumes "rt1 ⊏ dip rt2"
  shows "the(rt1 dip) ⊑ the(rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
  assumes "rt1 ⊏ dip rt2"
  shows "¬ (rt1 ≈ dip rt2)"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1 ⊏ dip rt2"
  and "rt2 ⊏ dip rt3"
  shows "rt1 ⊏ dip rt3"
using assms proof -
  from rt1 ⊏ dip rt2: obtain "the(rt1 dip) ⊑ the(rt2 dip)" by auto
  also from rt2 ⊏ dip rt3: obtain "the(rt2 dip) ⊑ the(rt3 dip)" by auto
  finally have "the(rt1 dip) ⊑ the(rt3 dip)" .
  moreover have "¬ (rt1 ≈ dip rt3)"
  proof -
    from rt1 ⊏ dip rt2: obtain "¬(the(rt2 dip) ⊑ the(rt1 dip))" by auto
    also from rt2 ⊏ dip rt3: obtain "¬(the(rt3 dip) ⊑ the(rt2 dip))" by auto
    finally have "¬(the(rt3 dip) ⊑ the(rt1 dip))" .
    thus ?thesis ..
  qed
ultimately show "rt1 ⊏ dip rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏ dip rt)"
unfolding rt_strictly_fresher_def
by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
  assumes "rt1 ⊏ dip rt2"
  and "rt2 ⊏ dip rt3"
  shows "rt1 ⊏ dip rt3"
proof -
  from rt1 ⊏ dip rt2: have "rt1 ⊏ dip rt2"
  and "¬(rt2 ⊏ dip rt1)"

41
unfolding \texttt{rt\_strictly\_fresher\_def''} by auto
from \texttt{this(1)} and \langle \texttt{rt2 \sqsubseteq dip rt3} \rangle have "rt1 \sqsubseteq dip rt3" ..

moreover from \langle \neg (rt2 \sqsubseteq dip rt1) \rangle have "\neg (rt3 \sqsubseteq dip rt1)"
proof (rule contrapos_nn)
assume "rt3 \sqsubseteq dip rt1"
with \langle rt2 \sqsubseteq dip rt3 \rangle show "rt2 \sqsubseteq dip rt1" ..
qed

ultimately show "rt1 \sqsubseteq dip rt3"
unfolding \texttt{rt\_strictly\_fresher\_def''} by auto
qed

lemma \texttt{rt\_fresher\_trans\_rt\_strictly\_fresher'} [elim, trans]:
assumes "rt1 \sqsubseteq dip rt2" and "rt2 \sqsubseteq dip rt3"
shows "rt1 \sqsubseteq dip rt3"
proof -
from \langle rt2 \sqsubseteq dip rt3 \rangle have "rt2 \sqsubseteq dip rt3" and "\neg (rt3 \sqsubseteq dip rt2)"
unfolding \texttt{rt\_strictly\_fresher\_def''} by auto
from \langle rt1 \sqsubseteq dip rt2 \rangle and \texttt{this(1)} have "rt1 \sqsubseteq dip rt3" ..
moreover from \langle \neg (rt3 \sqsubseteq dip rt2) \rangle have "\neg (rt3 \sqsubseteq dip rt1)"
proof (rule contrapos_nn)
assume "rt3 \sqsubseteq dip rt1"
thus "rt3 \sqsubseteq dip rt2" using \langle rt1 \sqsubseteq dip rt2 \rangle ..
qed

ultimately show "rt1 \sqsubseteq dip rt3"
unfolding \texttt{rt\_strictly\_fresher\_def''} by auto
qed

lemma \texttt{rt\_fresher\_imp\_nsqn\_le}:
assumes "rt1 \sqsubseteq dip rt2" and "ip \in kD rt1" and "ip \in kD rt2"
shows "nsqn rt1 ip \leq nsqn rt2 ip"
using \texttt{assms(1)}
by (auto simp add: \texttt{rt\_fresher\_def2 [OF \texttt{assms(2-3)}]})

lemma \texttt{rt\_strictly\_fresher\_ltI} [intro]:
assumes "dip \in kD(rt1)" and "dip \in kD(rt2)"
and "nsqn rt1 dip < nsqn rt2 dip"
shows "rt1 \sqsubseteq dip rt2"
proof
from \texttt{assms} show "rt1 \sqsubseteq dip rt2" ..
next
show "\neg (rt1 \approx dip rt2)"
proof
assume "rt1 \approx dip rt2"

"rt2 \sqsubseteq dip rt1" ..

hence "nsqn rt2 dip \leq nsqn rt1 dip"

using \langle dip \in kD(rt2) \rangle \langle dip \in kD(rt1) \rangle
by (rule \texttt{rt\_fresher\_imp\_nsqn\_le})

with \langle nsqn rt1 dip < nsqn rt2 dip \rangle show "False"

by simp
qed

lemma \texttt{rt\_strictly\_fresher\_eqI} [intro]:
assumes "i \in kD(rt1)" and "i \in kD(rt2)"

and "nsqn rt1 i = nsqn rt2 i"
and "π5(the (rt2 i)) < π5(the (rt1 i))"
shows "rt1 ⊏ rt2"

using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]:
"∀dests dip rt rt'. dests dip = None ⇒ (invalidate rt dests □ dip rt') = (rt □ dip rt')"
unfolding invalidate_def rt_strictly_fresher_def'
by (rule iffI) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
assumes "dip ∈ vD(invalidate rt1 dests)"
shows "(invalidate rt1 dests □ dip rt') = (rt1 □ dip rt')"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∈ dom(dests)"
  hence "dip /∈ vD(invalidate rt1 dests)"
    unfolding invalidate_def vD_def
    by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
  with ⟨dip ∈ vD(invalidate rt1 dests)⟩ show ?thesis by simp
next
  assume "dip /∈ dom(dests)"
  hence "dests dip = None" by auto
  moreover with ⟨dip ∈ vD(invalidate rt1 dests)⟩ have "dip ∈ vD(rt1)"
    unfolding invalidate_def vD_def
    by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
  ultimately show ?thesis
    unfolding invalidate_def rt_strictly_fresher_def'
    by auto
qed

lemma rt_strictly_fresher_update_other [elim!]:
"∀ dip ip rt r rt'. [ dip ≠ ip; rt ⊏ dip rt' ] = update rt ip r □ dip rt'"
unfolding rt_strictly_fresher_def' by clarsimp

lemma addpreRT_strictly_fresher [simp]:
assumes "dip ∈ kD(rt)"
shows "(the (addpreRT rt dip npre) □ dip rt2) = (rt □ dip rt2)"
using assms unfolding rt_strictly_fresher_def'
by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
assumes "dip ∈ vD (rt2 nhip)"
and *: "osn < sqn (rt2 nhip) dip" and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
shows "update rt dip (osn, kno, val, hops, nhip, {}) □ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
  from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip, {})) dip = osn"
    by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩ have "nsqn (the (update rt dip (osn, kno, val, hops, nhip, {})) dip) = osn"
    by (simp add: kD_nsqn)
  also have "osn < sqn (rt2 nhip) dip" by (rule *)
  also have "sqn (rt2 nhip) dip = nsqn (the (rt2 nhip dip))"
    unfolding nsqn_def using ⟨dip ∈ vD (rt2 nhip)⟩
  by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqn (the (update rt dip (osn, kno, val, hops, nhip, {})) dip) < nsqn (the (rt2 nhip dip))".
qed

lemma dhops_le_hops_imp_update_strictly_fresher:
assumes "dip ∈ vD(rt2 nhip)"
and sqn: "sqn (rt2 nhip) dip = osn"
and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip, {})"
shows "update rt dip (osn, kno, val, Suc hops, nhip, {}) □ dip rt2 nhip"
unfolding \texttt{rt\_strictly\_fresher\_def}'
proof (rule disjI2, rule conjI)
from ** have "\texttt{nsqn (update rt dip (osn, kno, val, Suc hops, nhip, \{})) dip = osn}"
  by (rule nsqn_update_changed_kno_val)
with \langle dip \in vD(rt2 nhip) \rangle
have "\texttt{nsqn_r (the (update rt dip (osn, kno, val, Suc hops, nhip, \{})) dip) = osn}"
  by (simp add: kD_nsqn)
also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
also have "sqn (rt2 nhip) dip = nsqn_r (the (rt2 nhip dip))"
  unfolding nsqn_r_def using \langle dip \in vD(rt2 nhip) \rangle
  by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally have "\texttt{nsqn_r (the (update rt dip (osn, kno, val, Suc hops, nhip, \{})) dip) = nsqn_r (the (rt2 nhip dip))}".
next
have "\texttt{(dhops (rt2 nhip) dip) \leq hops}" by (rule hop)
also have "hops < hops + 1" by simp
finally have "\texttt{(dhops (update rt dip (osn, kno, val, Suc hops, nhip, \{})) dip) < hops + 1}"
  using \langle dip \in vD(rt2 nhip) \rangle by (simp add: proj5_eq_dhops)
qed

\textbf{0.7 Invariant proofs on individual processes}

theory \textit{Seq\_Invariants}
imports AWN\_Invariants Aodv Aodv\_Data Aodv\_Predicates Fresher
begin

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

\textbf{lemma sequence_number_increases:}

\texttt{"paodv i ||\ A onll \Gamma_{AODV} (\lambda((\xi, \_), \_, ((\xi', \_)). sn \xi \leq sn \xi')")}
  by inv_cterms

\textbf{lemma sequence_number_one_or_bigger:}

\texttt{"paodv i ||\ A onll \Gamma_{AODV} (\lambda(\xi, \_). \_ \leq sn \xi")}
  by (rule onll_step_to_invariantI [OF sequence_number_increases])

\textbf{We can get rid of the onl/onll if desired...}

\textbf{lemma sequence_number_increases':}

\texttt{"paodv i ||\ A (\lambda((\xi, \_), \_, ((\xi', \_)). sn \xi \leq sn \xi')")}
  by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

44
lemma sequence_number_one_or_bigger':
"paodv i |= (λ(ξ, _). 1 ≤ sn ξ)"
by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

lemma sip_in_kD:
"paodv i |= onl Γ AODV (λ(ξ, _). 1 ≤ sn ξ) = (dsn ξ, kno, val, hops ξ + 1, sip ξ, {}))" by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

lemma rrep_1_update_changes:
"paodv i |= onl Γ AODV (λ(ξ, l). (l = PRrep-:1 → rt ξ ≠ update (rt ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})))" by inv_cterms

lemma addpreRT_partly_welldefined:
"paodv i |= onl Γ AODV (λ(ξ, l). (l ∈ {PRreq-:16..PRreq-:18} ∪ {PRrep-:2..PRrep-:6} → dip ξ ∈ kD (rt ξ)) ∧ (l ∈ {PRreq-:3..PRreq-:17} → oip ξ ∈ kD (rt ξ)))" by inv_cterms

Proposition 7.38

lemma includes_nhip:
"paodv i |= onl Γ AODV (λ(ξ, l). ∀ dip ∈ kD (rt ξ). the (nhop (rt ξ) dip) ∈ kD (rt ξ)))" proof -
{ fix ip and ξ :: state
  assume "∀ dip ∈ kD (rt ξ). the (nhop (rt ξ) dip) ∈ kD (rt ξ)"
  hence "∀ dip ∈ kD (rt ξ).
    the (nhop (update (rt ξ) ip (0, unk, val, Suc 0, ip, {})) dip) = ip
    ∨ the (nhop (update (rt ξ) ip (0, unk, val, Suc 0, ip, {})) dip) ∈ kD (rt ξ)"
    by clarsimp (metis nhop_update_unk_val update_another)
} note one_hop = this

{ fix ip sip sn hops and ξ :: state
  assume "∀ dip ∈ kD (rt ξ). the (nhop (rt ξ) dip) ∈ kD (rt ξ)"
  and "∀ dip ∈ kD (rt ξ).
    the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) dip) = ip
    ∨ the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) dip) ∈ kD (rt ξ)"
  and "∀ dip ∈ kD (rt ξ).
    the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) dip) = ip
    ∨ the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) dip) ∈ kD (rt ξ)"
  by (metis kD_update_unchanged nhop_update_changed update_another)
} note nhip_is_sip = this

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf sip_in_kD]
    onl_invariant_sterms [OF aodv_wf addpreRT_partly_welldefined]
    solve: one_hop nhip_is_sip)
qed

Proposition 7.22: needed in Proposition 7.4

lemma addpreRT_welldefined:
"paodv i |= onl Γ AODV (λ(ξ, l). (l ∈ {PRreq-:16..PRreq-:18} ∪ {PRrep-:2..PRrep-:6} → dip ξ ∈ kD (rt ξ)) ∧ (l = PRreq-:17 → oip ξ ∈ kD (rt ξ)) ∧ (l = PRrep-:5 → dip ξ ∈ kD (rt ξ)) ∧ (l = PRrep-:6 → (the (nhop (rt ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {}})) dip) ∈ kD (rt ξ)))" (is "_ |= onl Γ AODV ?P")

unfolding invariant_def
proof
  fix s
  assume "s ∈ reachable (paodv i) TT"
  then obtain ξ p where "s = (ξ, p)"
    and "(ξ, p) ∈ reachable (paodv i) TT"
  by (metis prod.exhaust)

45
have "onl Γ_{AODV} ?P (ξ, p)"
proof (rule onlI)
  fix l
  assume "l ∈ labels Γ_{AODV} p"
  with \langle (ξ, p) ∈ reachable (paodv i) TT \rangle 
  have I1: "l ∈ \{PRreq-:16..PRreq-:18\} → dip ξ ∈ kD(rt ξ)"
  and I2: "l = PRreq-:17 → oip ξ ∈ kD(rt ξ)"
  and I3: "l ∈ \{PRrep-:2..PRrep-:6\} → dip ξ ∈ kD(rt ξ)"
    by (auto dest!: invariantD [OF addpreRT_partly_welldefined])
moreover from \langle (ξ, p) ∈ reachable (paodv i) TT \rangle 
  and I3
  have "l = PRrep-:6 → (the (nhop (rt ξ) (dip ξ))) ∈ kD(rt ξ)"
    by (auto dest!: invariantD [OF includes_nhpl])
ultimately show "?P (ξ, l)"
  by simp
qed

with \langle s = (ξ, p) \rangle 
  show "onl Γ_{AODV} ?P s"
  by simp
qed

Proposition 7.4

lemma known_destinations_increase:
  "paodv i \|= onll Γ_{AODV} (λ((ξ, _), _, (ξ', _)). kD (rt ξ) ⊆ kD (rt ξ'))"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]
  simp add: subset_insertI)

Proposition 7.5

lemma rreqs_increase:
  "paodv i \|= onll Γ_{AODV} (λ((ξ, _), _, (ξ', _)). rreqs ξ ⊆ rreqs ξ')"
by (inv_cterms simp add: subset_insertI)

lemma dests_bigger_than_sqn:
  "paodv i \|= onl Γ_{AODV} (λ(ξ, 1). l ∈ \{PAodv-:15..PAodv-:19\}
    \cup \{PPkt-:7..PPkt-:11\}
    \cup \{PRreq-:9..PRreq-:13\}
    \cup \{PRreq-:21..PRreq-:25\}
    \cup \{PRrep-:10..PRrep-:14\}
    \cup \{PRerr-:1..PRerr-:5\}
    → (∀ip∈dom(dests ξ). ip∈kD(rt ξ) ∧ sqn (rt ξ) ip ≤ the (dests ξ ip)))"
proof -
  have sqninv:
    "\{dests rt rsn ip.
      [ ∀ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip); dests ip = Some rsn ]
    → sqn (invalidate rt dests) ip ≤ rsn"
  by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
  have indests:
    "\{dests rt rsn ip.
      [ ∀ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip); dests ip = Some rsn ]
    → ip∈kD(rt) ∧ sqn rt ip ≤ rsn"
  by (metis domI option.sel)
  show ?thesis
    by inv_cterms
      (clarsimp split: if_split_asm option.split_asm
        elim!: sqninv indests)+
  qed

Proposition 7.6

lemma sqns_increase:
  "paodv i \|= onll Γ_{AODV} (λ((ξ, _), _, (ξ', _)). ∀ip. sqn (rt ξ) ip ≤ sqn (rt ξ') ip)"
proof -
  { fix ξ :: state
    assume *: "∀ip∈dom(dests ξ). ip ∈ kD (rt ξ) ∧ sqn (rt ξ) ip ≤ the (dests ξ ip)"
    have "∀ip. sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip"
    proof
      fix ip
      }
from * have "ip ∈ dom(dests ξ) ∨ sqn (rt ξ) ip ≤ the (dests ξ ip)" by simp 
thus "sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip" 
by (metis domI invalidate_sqn option.sel)
qed

} note solve_invalidate = this
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn]
simp add: solve_invalidate)
qed

Proposition 7.7

lemma ip_constant:
"paodv i |= onl Γ AODV (λ(ξ, _). ip ξ = i)"
by (inv_cterms simp add: σ AODV _def)

Proposition 7.8

lemma sender_ip_valid':
"paodv i |= onll Γ AODV (λ((ξ, _), a, _). anycast (λm. not_Pkt m −→ msg_sender m = ip ξ) a)"
by inv_cterms

lemma sender_ip_valid:
"paodv i |= (recvmsg P →) onl Γ AODV (λ((ξ, _), a, _). anycast (λm. not_Pkt m −→ msg_sender m = i) a)"
by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
(auto dest!: onlD onllD)

lemma received_msg_inv:
"paodv i |= (recvmsg (λm. not_Pkt m −→ msg_sender m ≠ i) →) onl Γ AODV (λξ. l ∈ {PAodv-:1} → P (msg ξ))"
by inv_cterms

Proposition 7.9

lemma sip_not_ip':
"paodv i |= (recvmsg (λm. not_Pkt m −→ msg_sender m ≠ i) →) onl Γ AODV (λξ. sip ξ ≠ ip ξ)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]]
simp add: clear_locals_sip_not_ip') clarsimp+

lemma sip_not_ip:
"paodv i |= (recvmsg (λm. not_Pkt m −→ msg_sender m ≠ i) →) onl Γ AODV (λξ. sip ξ ≠ i)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]]
simp add: clear_locals_sip_not_ip') clarsimp+

Neither sip_not_ip' nor sip_not_ip is needed to show loop freedom.

Proposition 7.10

lemma hop_count_positive:
"paodv i |= onl Γ AODV (λξ. ∀ip ∈ kD (rt ξ). the (dhops (rt ξ) ip) ≥ 1)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]) auto

lemma rreq_dip_in_vD_dip_eq_ip:
"paodv i |= onl Γ AODV (λξ, l. (l ∈ {PRreq-:16..PRreq-:18} → dip ξ ∈ vD (rt ξ))
∧ (l ∈ {PRreq-:5, PRreq-:6} → dip ξ = ip ξ)
∧ (l ∈ {PRreq-:15..PRreq-:18} → dip ξ ≠ ip ξ))"
proof (inv_cterms, elim conjE)
fix l ξ pp p'
assume "(ξ, pp) ∈ reachable (paodv i) TT"
and "(PRreq-:17)[ξ, rt := the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ))})] p'
∈ sterms Γ AODV pp"
and "$1 = PRreq-:17"
and "dip ξ ∈ vD (rt ξ)"
from this(1-3) have "oip ξ ∈ kD (rt ξ)"
by (auto dest: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined, where l="PRreq-:17"])

47
with \( (\text{dip } \xi \in vD (\text{rt } \xi)) \)
show "\( \text{dip } \xi \in vD (\text{the (addpreRT (\text{rt } \xi) (\text{oip } \xi) (\text{the (nhop (\text{rt } \xi) (\text{dip } \xi))))})) \)" by simp
qed

Proposition 7.11

lemma anycast_msg_zhops:
"\( \wedge \text{rreqid dip dsn dsk oip osn sip} \).
\( \text{paodv i} \models \text{onll } \Gamma_{\text{AODV}} (\lambda (\xi, a, \_). \text{anycast msg_zhops a}) \)"
proof (inv_cterms inv add:
\text{onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip [THEN invariant_restrict_inD]]}
\text{onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]],}
elim conjE)
fix 1 \xi a pp p' pp'
assume "\( (\xi, pp) \in \text{reachable (paodv i) TT} \)"
and "\( \{\text{PRreq-:18} \} \text{unicast (} \lambda \xi. \text{the (nhop (\text{rt } \xi) (oip } \xi)) \), \lambda \xi. \text{Rrep (the (dhops (\text{rt } \xi) (dip } \xi))) \text{ (dip } \xi) \text{ (sqn (\text{rt } \xi) (dip } \xi)) \text{ (oip } \xi) \text{ (ip } \xi)) \).
p' \triangleright pp' \in \text{sterms } \Gamma_{\text{AODV}} pp"" and "\( l = \text{PRreq-:18} \)" and "\( a = \text{unicast (the (nhop (\text{rt } \xi) (oip } \xi))) \)
\( \text{Rrep (the (dhops (\text{rt } \xi) (dip } \xi))) \text{ (dip } \xi) \text{ (sqn (\text{rt } \xi) (dip } \xi)) \text{ (oip } \xi) \text{ (ip } \xi)) \)"" and "\( \forall \text{ip} \in kD (\text{rt } \xi). \text{Suc 0} \leq \text{the (dhops (rt } \xi) \text{ ip}) \)" and "\( \text{dip } \xi \in vD (\text{rt } \xi) \)"
from "\( \text{dip } \xi \in vD (\text{rt } \xi) \)"
** have "\( \text{Suc 0} \leq \text{the (dhops (rt } \xi) \text{ (dip } \xi)) \)" ..
thus "\( 0 < \text{the (dhops (rt } \xi) \text{ (dip } \xi)) \)" by simp
qed

lemma hop_count_zero_oip_dip_sip:
"\( \text{paodv i} \models \text{onll } \Gamma_{\text{AODV}} (\lambda (\xi, l). 0 \in \{PAodv-:4..PAodv-:5\} \cup \{PRreq-:n|n. \_\} \rightarrow (\text{hops } \xi = 0 \rightarrow \text{oip } \xi = \text{sip } \xi)) \)" and "\( \in \{PAodv-:6..PAodv-:7\} \cup \{PRrep-:n|n. \_\} \rightarrow (\text{hops } \xi = 0 \rightarrow \text{dip } \xi = \text{sip } \xi)) \)"
proof (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:
"\( \text{paodv i} \models \text{onll } \Gamma_{\text{AODV}} (\lambda (\xi, l). l \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n|n. True\} \rightarrow (\text{osn } \xi = 0 < \text{the (dhops (rt } \xi))) \)"
proof (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq':
"\( \text{paodv i} \models \text{onll } \Gamma_{\text{AODV}} (\lambda (\xi, l). l \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n|n. True\} \rightarrow (\text{osn } \xi = 0 < \text{the (dhops (rt } \xi))) \)"
proof (rule invariant_weakenE [OF osn_rreq])
fix a
assume "\( \text{recvmsg } (\lambda m. \text{rreq_rrep_sn } m) \text{ a} \)"
thus "\( \text{recvmsg } \text{rreq_rrep_sn a} \)" by (cases a) simp_all
qed

lemma dsn_rrep:
"\( \text{paodv i} \models \text{onll } \Gamma_{\text{AODV}} (\lambda (\xi, l). l \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-:n|n. True\} \rightarrow (\text{dsn } \xi = 0 < \text{the (dhops (rt } \xi))) \)"
proof (rule invariant_weakenE [OF osn_rreq])
fix a
assume "\( \text{recvmsg } (\lambda m. \text{rreq_rrep_sn } m) \text{ a} \)"
thus "\( \text{recvmsg } \text{rreq_rrep_sn a} \)" by (cases a) simp_all
qed
by (cases a) simp_all
qed

lemma hop_count_zero_oip_dip_sip':
  "paodv i ||= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) → onl Γ_AODV (λ(ξ, _).
  (l∈{PAodv-:4..PAodv-:5} ∪ {PRreq-:n|n. True} →
  (hops ξ = 0 → oip ξ = sip ξ))
  ∧
  ((l∈{PAodv-:6..PAodv-:7} ∪ {PRrep-:n|n. True} →
  (hops ξ = 0 → dip ξ = sip ξ))))"
proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
fix a
assume "recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
thus "recvmsg msg_zhops a" by (cases a) simp_all
qed

Proposition 7.12
lemma zero_seq_unk_hops_one':
  "paodv i ||= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) → onl Γ_AODV (λ(ξ, _).
  ∀ dip ∈ kD(rt). (sqn (rt ξ) dip = 0 → sqnf (rt ξ) dip = unk)
  ∧ (sqnf (rt ξ) dip = unk → the (dhops (rt ξ) dip) = 1)
  ∧ (the (dhops (rt ξ) dip) = 1 → the (nhop (rt ξ) dip) = dip)))"
proof -
{ fix dip and ξ :: state and P
  assume "sqn (invalidate (rt ξ) (dests ξ)) dip = 0"
  and all: "∀ ip. sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip"
  and *: "sqn (rt ξ) dip = 0 =⇒ P ξ dip"
  have "P ξ dip" proof -
  from all have "sqn (rt ξ) dip ≤ sqn (invalidate (rt ξ) (dests ξ)) dip" ..
  with ∀ ip. sqn (rt ξ) (dests ξ)) dip = 0 have "sqn (rt ξ) dip = 0" by simp
  thus "P ξ dip" by (rule *)
  qed
}
{ note sqn_invalidate_zero [elim!] = this

{ fix dip and ξ :: state and P
  assume "∀ dip ∈ kD(rt).
  (sqn rt dip = 0 → π₃(the (rt dip)) = unk) ∧
  (π₃(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
  and "hops = 0 → sip = dip"
  and "Suc 0 ≤ dsn"
  and "ip ≠ dip → ip∈kD(rt)"
  hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0 →
  the (nhop (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = ip"
  by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
  note prreq_ok1 [simp] = this
}
{ fix dip and ξ :: state and P
  assume "∀ dip ∈ kD(rt).
  (sqn rt dip = 0 → π₃(the (rt dip)) = unk) ∧
  (π₃(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
  and "Suc 0 ≤ dsn"
  and "ip ≠ dip → ip∈kD(rt)"
  hence "π₃(the (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = unk →
  the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0"
  by - (rule update_cases, auto simp add: sqn_def sqnf_dest!: bspec)
  note prreq_ok2 [simp] = this
}
{ fix dip and ξ :: state and P
  assume "∀ dip ∈ kD(rt).
  (sqn rt dip = 0 → π₃(the (rt dip)) = unk) ∧
  (π₃(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
  and "Suc 0 ≤ dsn"
  and "ip ≠ dip → ip∈kD(rt)"
  hence "π₃(the (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = unk →
  the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0"
  by - (rule update_cases, auto simp add: sqn_def sqnf_dest!: bspec)
  note prreq_ok3 [simp] = this

(π₃(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
(the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)

and "Suc 0 ≤ dsn"
and "ip ≠ dip → ip∈KD(rt)"

hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip = 0 → π₃ (the (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = unk"
by - (rule update_cases, auto simp add: sqn_def dest!: bspec)

} note prreq_ok3 [simp] = this

{ fix rt sip
  assume "∀ dip∈KD rt.
  (sqn rt dip = 0 → π₃(the (rt dip)) = unk) ∧
  (π₃(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"

hence "∀ dip∈KD rt.
  (sqn (update rt sip (0, unk, val, Suc 0, sip, {})) dip = 0 →
  π₃(the (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = unk)
∧ (π₃(the (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = unk →
  the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0)
∧ (the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0 →
  the (nhop (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = dip)"
by - (rule update_cases, simp_all add: sqn_def sqn_def)

} note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "∀ sip rt.
  π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {}) sip)) = unk →
  the (dhops (update rt sip (0, unk, val, Suc 0, sip, {}) sip)) = Suc 0"
by (rule update_cases) simp_all

have prreq_ok6 [simp]: "∀ sip rt.
  sqn (update rt sip (0, unk, val, Suc 0, sip, {})) sip = 0 →
  π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {}) sip)) = unk"
by (rule update_cases) simp_all

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip']
seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans]
onl_invariant_sterms [OF aodv_wf osn_rreq']
onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+

qed

lemma zero_seq_unk_hops_one:
  "paodv i |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) → onl Γ_AODV (λξ, _).
  ∀ dip∈KD(rt ξ). (sqn (rt ξ dip) = 0 → (sqn (rt ξ dip) = unk
  ∧ the (dhops (rt ξ dip)) = Suc 0)
  ∧ the (nhop (rt ξ dip)) = dip)))"  
by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
  "paodv i |= (recvmsg (rrq rrep sn →) onl Γ_AODV (λξ, 1).
  ∀ dip∈KD(rt ξ). π₃ (the (rt ξ dip)) = unk ∧ 1 ≤ π₂ (the (rt ξ dip)))"

proof -
  { fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
    assume "dsk1 = unk ∨ Suc 0 ≤ dsn2"
    hence "π₃ (the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1, pre1) sip)) = unk
    ∧ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2) sip)"
    unfolding update_def by (cases "dsk1 = unk") (clarsimp split: option.split)+
  } note fromsip [simp] = this

  { fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
    assume allkd: "∀ dip∈KD(rt). π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip"
    and "dsk1 = unk ∨ Suc 0 ≤ dsn2"
    have "∀ dip∈KD(rt). π₃ (the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1, pre1) dip)) = unk

\( \forall \text{Suc } 0 \leq \text{sqn (update } rt \text{ sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) dip} \)

(is "\( \forall dip \in \text{kD}(rt). \ ?\text{prop } dip \)"

proof
fix dip
assume "dip \in \text{kD}(rt)"
thus "\(?\text{prop dip}\)"
proof (cases "dip = \text{sip}")
assume "dip = \text{sip}"
with \(*\) show \(?\text{thesis}\)
by simp
next
assume "dip \not= \text{sip}"
with \(\langle dip \in \text{kD}(rt)\rangle\) \text{allkd} show \(?\text{thesis}\)
by simp
qed
qed

\} \text{note solve_update [simp] = this}

\{ fix rt dests
assume \(\ast\): "\(\forall ip \in \text{dom(dests) \& ip \in \text{kD}(rt) \& \text{sqn } rt \ ip \leq \text{the } (\text{dests } ip)\)"
and \(\ast\ast\): "\(\forall ip \in \text{kD}(rt) \& \pi_3(\text{the (rt ip)}) = \text{unk} \lor \text{Suc } 0 \leq \text{sqn } rt \ ip\)"
have "\(\forall dip \in \text{kD}(rt) \& \pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc } 0 \leq \text{sqn (invalidate } rt \text{ dests) dip}\)"
proof
fix dip
assume "dip \in \text{kD}(rt)"
with \(\ast\ast\) have "\(\pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc } 0 \leq \text{sqn (invalidate } rt \text{ dests) dip}\)" ..
thus "\(\pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc } 0 \leq \text{sqn (invalidate } rt \text{ dests) dip}\)"
proof
assume "\(\pi_3(\text{the (rt dip)}) = \text{unk}\)" thus \(?\text{thesis}\) ..
next
assume "\(\text{Suc } 0 \leq \text{sqn } rt \ dip\)"
have "\(\text{Suc } 0 \leq \text{sqn (invalidate } rt \text{ dests) dip}\)"
proof (cases "dip \in \text{dom(dests)}")
assume "dip \in \text{dom(dests)}"
with \(\ast\) have "\(\text{sqn } rt \ dip \leq \text{the } (\text{dests dip})\)" by simp
with \(\langle \text{Suc } 0 \leq \text{sqn } rt \ dip \rangle\) have "\(\text{Suc } 0 \leq \text{the } (\text{dests dip})\)" by simp
with \(\langle dip \in \text{dom(dests)}\rangle\) \(\langle dip \in \text{kD}(rt)\rangle\) \[THEN kD_Some\] show \(?\text{thesis}\)
unfolding \text{invalidate_def} sqn_def by auto
next
assume "dip \in \text{dom(dests)}"
with \(\langle \text{Suc } 0 \leq \text{sqn } rt \ dip \rangle\) \(\langle dip \in \text{kD}(rt)\rangle\) \[THEN kD_Some\] show \(?\text{thesis}\)
unfolding \text{invalidate_def} sqn_def by auto
qed
thus \(?\text{thesis}\) by \text{(rule disjI2)}
qed
\}
\text{note solve.invalidate [simp] = this}

show \(?\text{thesis}\)
by \(\text{inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]}\)
\text{onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn [THEN invariant_restrict_inD]]}
\text{onl_invariant_sterms [OF aodv_wf osn_rreq]}
\text{onl_invariant_sterms [OF aodv_wf dsn_rrep]}
simp add: proj3_inv proj2_eq_sqn)
qed

Proposition 7.13

**lemma rreq_rrep_sn any step invariant:**
"paodv i \|= \(\text{recvmsg } rreq_rrep_sn \rightarrow \text{onll } \Gamma_{AODV} (\lambda(\_, a, \_). \text{anycast rreq_rrep_sn } a)\)"

proof -
have sqnf_kno: "paodv i \|= \text{onl } \Gamma_{AODV} (\lambda(\xi, 1).
\(1 \in \{PRreq::16..PRreq::18\} \rightarrow \text{sqnf (rt } \xi \text{) (dip } \xi \text{ = kno)}\)"
by \(\text{inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]}\)
show thesis
  by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
     onl_invariant_sterms [OF aodv_wf sequence_number_one_or_bigger
          [THEN invariant_restrict_inD]]
     onl_invariant_sterms [OF aodv_wf kD_unk_or_atleast_one]
     onl_invariant_sterms_TT [OF aodv_wf sqnf_kno]
     onl_invariant_sterms [OF aodv_wf osn_rreq]
     onl_invariant_sterms [OF aodv_wf dsn_rrep])
(auto simp: proj2_eq_sqn)
qed

Proposition 7.14

lemma req_rep_fresh_any_step_invariant:
  "paodv i ⊨ onl Γ AODV (λ((ξ, _), a, _). ancast (req_rep_fresh (rt ξ)) a)"
proof -
  have req_oip: "paodv i ⊨ onl Γ AODV (λ((ξ, _), a, _). anycast (req_rep_fresh (rt ξ)) a)
          (l ∈ {PRreq:-3, PRreq:-4, PRreq:-15, PRreq:-27}
          ⟷ oip ξ ∈ kD(rt ξ)
          ∧ (sqn (rt ξ) (oip ξ) > (osn ξ)
          ∨ (sqn (rt ξ) (oip ξ) = (osn ξ)
          ∧ the (dhops (rt ξ) (oip ξ)) ≤ Suc (hops ξ)
          ∧ the (flag (rt ξ) (oip ξ) = val))))"
  proof
    fix l l' pp p'
    assume "((ξ, pp) ∈ reachable (paodv i) TT"
    and "{PRreq:-2}\[λξ. (ξ, l)]\ p' ∈ sterms Γ AODV pp"
    and "l' = PRreq:-3"
    show "osn ξ < sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})) (oip ξ)
          ∨ (sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})) (oip ξ) = osn ξ
          ∧ the (dhops (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})) (oip ξ))
          ≤ Suc (hops ξ)
          ∧ the (flag (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})) (oip ξ) = val)"
    unfolding update_def by (clarsimp split: option.split)
    (metis linorder_neqE_nat not_less)
  qed

have req_prrep: "paodv i ⊨ onl Γ AODV (λ(ξ, 1).
          (1 ∈ {PRrep:-2..PRrep:-7} ⟷ (dip ξ ∈ kD(rt ξ)
          ∧ sqn (rt ξ) (dip ξ) = dsn ξ
          ∧ the (dhops (rt ξ) (dip ξ)) = Suc (hops ξ)
          ∧ the (flag (rt ξ) (dip ξ)) = val
          ∧ the (nhop (rt ξ) (dip ξ)) ∈ kD (rt ξ))))"
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rrep_1_update_changes]
        onl_invariant_sterms [OF aodv_wf sip_in_kD])

show thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf req_oip]
        onl_invariant_sterms [OF aodv_wf req_dip_in_vD_dip_eq_ip]
        onl_invariant_sterms [OF aodv_wf req_prrep])
qed

Proposition 7.15

lemma err_invalidate_any_step_invariant:
  "paodv i ⊨ onl Γ AODV (λ((ξ, _), a, _). ancast (err_invalidate (rt ξ)) a)"
proof -
  have dests_inv: "paodv i ⊨ onl Γ AODV (λ(ξ, 1).
          (1 ∈ {PAodv:-15, PPkt:-7, PRreq:-9, PRreq:-21, PRrep:-10, PRerr:-1}
          ⟷ (∀ip ∈ dom(dests ξ). ip ∈ vD(rt ξ))
          ∧ (1 ∈ {PAodv:-16..PAodv:-19}
          ∪ {PPkt:-8..PPkt:-11}
          ∪ {PRreq:-10..PRreq:-13})
          ∧ (1 ∈ {PAodv:-15, PPkt:-7, PRreq:-9, PRreq:-21, PRrep:-10, PRerr:-1}
          ⟷ (∀ip ∈ dom(dests ξ). ip ∈ vD(rt ξ))
          ∧ (1 ∈ {PAodv:-16..PAodv:-19}
          ∪ {PPkt:-8..PPkt:-11}
          ∪ {PRreq:-10..PRreq:-13})}
∪ {PRreq:-22..PRreq:-25}
∪ {PRrep:-11..PRrep:-14}
∪ {PRerr:-2..PRerr:-5} → (∀ip∈dom(dests ξ). ip∈D(rt ξ)
∧ the (dests ξ ip) = sqn (rt ξ ip))
∧ (l = PPkt:-14 → dip ξ∈D(rt ξ))"
by inv_cterms (clarsimp split: if_split_asm option.split_asm simp: domIff) +
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_inv])
qed

Proposition 7.16
Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs \( (rip, rsn) \) is assigned to the variable \( dests \) of type \( ip \mapsto sqn \), or to the first argument of the function \( rerr \), this set is a partial function, i.e., there is at most one entry \( (rip, rsn) \) for each destination \( rip \): guaranteed by type.

lemma dests_vD_inc_sqn:
"paodv i ||
= onl ΓAODV (λ(ξ, _). (l ∈ {PAodv:-15, PPkt:-7, PRreq:-9, PRreq:-21, PRrep:-10} → (∀ip∈dom(dests ξ). ip∈vD(rt ξ) ∧ the (dests ξ ip) = inc (sqn (rt ξ ip))))
∧ (l = PRerr:-1 → (∀ip∈dom(dests ξ). ip∈vD(rt ξ) ∧ the (dests ξ ip) > sqn (rt ξ ip))))"
by inv_cterms (clarsimp split: if_split_asm option.split_asm) +

Proposition 7.27
lemma route_tables_fresher:
"paodv i ||
= onll ΓAODV (λ((ξ, _), _, ((ξ', _))). (∀ip∈kD(rt ξ). rt ξ ⊑ dip rt ξ'))"
proof (inv_cterms inv add:
onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
onl_invariant_sterms [OF aodv_wf addpreRT_welldefined [THEN invariant_restrict_inD]])
fix ξ pp p'
assume "((ξ, pp) ∈ reachable (paodv i) (recvmsg rreq_rrep_sn))"
and "{PRreq:-2}[(ξ, _), ((ξ', _))). (∀ip∈kD(rt ξ). rt ξ ⊑ dip rt ξ'))"
proof
fix ip
assume "ip∈kD (rt ξ)"
moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
moreover from Suc 0 ≤ osn ξ have "update_arg_wf (osn ξ, kno, val, Suc (hops ξ), sip ξ, {}, {})" ..
ultimately show "rt ξ ⊑ dip update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {}, {})"
by (rule rt_fresher_update)
qed
next
fix ξ pp p'
assume "((ξ, pp) ∈ reachable (paodv i) (recvmsg rreq_rrep_sn))"
and "{PRreq:-1}[(ξ, _), ((ξ', _))). (∀ip∈kD(rt ξ). rt ξ ⊑ dip rt ξ'))"
proof
fix ip
assume "ip∈kD (rt ξ)"
moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
moreover from Suc 0 ≤ osn ξ have "update_arg_wf (osn ξ, kno, val, Suc (hops ξ), sip ξ, {}, {})" ..
ultimately show "rt ξ ⊑ dip update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {}, {})"
by (rule rt_fresher_update)
qed
∀ip∈kD (rt ξ). rt ξ ⊑ ip update (rt ξ) (dsn ξ, kno, val, Suc(hops ξ), sip ξ, {})" 
proof 
fix ip 
assume "ip∈kD (rt ξ)"
moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
moreover from ⟨Suc 0 ≤ dsn ξ⟩ have "update_arg_wf (dsn ξ, kno, val, Suc(hops ξ), sip ξ, {})" ..
ultimately show "rt ξ ⊑ ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc(hops ξ), sip ξ, {})" 
by (rule rt_fresher_update) 
qed 
qed 

0.8 The quality increases predicate

theory Quality_Increases
imports Aodv_Predicates Fresher
begin

definition quality_increases :: "state ⇒ state ⇒ bool" 
where "quality_increases ξ ξ' ≡ (∀dip∈kD(rt ξ). dip ∈ kD(rt ξ') ∧ rt ξ ⊑ dip rt ξ') 
∧ (∀dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip)"

lemma quality_increasesI [intro!]: 
assumes "⋀dip. dip ∈ kD(rt ξ) =⇒ dip ∈ kD(rt ξ')" 
and "⋀dip. [ dip ∈ kD(rt ξ); dip ∈ kD(rt ξ') ] =⇒ rt ξ ⊑ dip rt ξ'"
and "⋀dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip"
shows "quality_increases ξ ξ'"
unfolding quality_increases_def using assms by clarsimp

lemma quality_increasesE [elim]: 
fixes dip 
assumes "quality_increases ξ ξ'" 
and "dip∈kD(rt ξ)"
and "[ dip ∈ kD(rt ξ); rt ξ ⊑ dip rt ξ'; sqn (rt ξ) dip ≤ sqn (rt ξ') dip ] =⇒ R dip ξ ξ'"
shows "R dip ξ ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rt_fresherD [dest]: 
fixes ip 
assumes "quality_increases ξ ξ'" 
and "ip∈kD(rt ξ)"
shows "rt ξ ⊑ ip rt ξ'"
using assms by auto

lemma quality_increases_sqnE [elim]: 
fixes dip 
assumes "quality_increases ξ ξ'" 
and "sqn (rt ξ) dip ≤ sqn (rt ξ') dip =⇒ R dip ξ ξ'"
shows "R dip ξ ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases ξ ξ"
by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]: 
fixes σ σ' dip 
assumes "rt (σ i) ⊑ dip rt (σ' nhip)"
and qinc: "quality_increases (σ nhip) (σ' nhip)" 
and "dip∈kD(rt (σ' nhip))"
shows "rt (σ i) ⊑ dip rt (σ' nhip)"
proof -
from qinc have "rt (σ nhip) ⊑ dip rt (σ' nhip)" using ⟨dip∈kD(rt (σ nhip))⟩
lemma kD_quality_increases [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality Increases ξ ξ'"
shows "i ∈ kD(rt ξ')"
using assms by auto

lemma kD_nsqn_quality_increases [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality Increases ξ ξ'"
shows "i ∈ kD(rt ξ') ∧ nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
proof -
  from assms have "i ∈ kD(rt ξ')" ...
  moreover with assms have "rt ξ ⊑ i rt ξ'" by auto
  ultimately have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  using i ∈ kD(rt ξ') by (erule(2) rt_fresher_imp_nsqn_le)
  with i ∈ kD(rt ξ') show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality Increases ξ ξ'"
shows "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
assumes "i ∈ kD(rt ξ)"
and "s ≤ nsqn (rt ξ) i"
and "quality Increases ξ ξ'"
shows "i ∈ kD(rt ξ') ∧ s ≤ nsqn (rt ξ') i"
proof
  from ⟨i ∈ kD(rt ξ)⟩ and ⟨qualityIncreases ξ ξ'⟩ have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  next
  from ⟨i ∈ kD(rt ξ)⟩ and ⟨qualityIncreases ξ ξ'⟩ have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with ⟨s ≤ nsqn (rt ξ) i⟩ show "s ≤ nsqn (rt ξ') i" by (rule le_trans)
qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality Increases ξ ξ'"
and "s = nsqn (rt ξ) i"
shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

lemma nsqn_quality_increases_dhops [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality Increases ξ ξ'"
and "nsqn (rt ξ) i = nsqn (rt ξ') i"
shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
using assms unfolding quality_increases_def
by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)
lemma quality_increases_rreq_rrep_props [elim]:
  fixes sn ip hops sip
assumes qinc: "quality_increases (σ sip) (σ' sip)"
  and "1 ≤ sn"
  and "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip ⋀
    (nsqn (rt (σ sip)) ip = sn "→ (the (dhops (rt (σ sip)) ip) ≤ hops π)
    ∨ the (flag (rt (σ sip)) ip) = inv))"
shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip ⋀
    (nsqn (rt (σ' sip)) ip = sn "→ (the (dhops (rt (σ' sip)) ip) ≤ hops π)
    ∨ the (flag (rt (σ' sip)) ip) = inv))"
(is "... ∧ ?nsqnafter")

proof -
  from * obtain "ip ∈ kD(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto

  from "quality_increases (σ sip) (σ' sip)"
  have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
  from "quality_increases (σ sip) (σ' sip)" and "ip ∈ kD (rt (σ sip))" have "ip ∈ kD (rt (σ' sip))" ..

  from "sn ≤ nsqn (rt (σ sip)) ip" have ?nsqnafter
  proof
    assume "sn < nsqn (rt (σ sip)) ip"
    also from "ip ∈ kD(rt (σ sip))" and "quality_increases (σ sip) (σ' sip)"
    have "... ≤ nsqn (rt (σ' sip)) ip" ..
    finally have "sn < nsqn (rt (σ' sip)) ip".
    thus ?thesis by simp
  next
    assume "sn = nsqn (rt (σ sip)) ip"
    with "ip ∈ kD(rt (σ sip))" and "quality_increases (σ sip) (σ' sip)"
    have "sn < nsqn (rt (σ' sip)) ip ⋀
      the (dhops (rt (σ' sip)) ip) ≤ hops π ∨ the (flag (rt (σ' sip)) ip) = inv))"
    hence "sn < nsqn (rt (σ' sip)) ip ⋀
      (nsqn (rt (σ' sip)) ip = sn ⋀ (the (dhops (rt (σ' sip)) ip) ≤ hops π)
      ∨ the (flag (rt (σ' sip)) ip) = inv))"
    proof
      assume "sn < nsqn (rt (σ' sip)) ip" thus ?thesis ..
    next
      assume "sn = nsqn (rt (σ' sip)) ip ⋀
        the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)"
      hence "sn = nsqn (rt (σ' sip)) ip ⋀
        "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)" by auto
      from * and "sn = nsqn (rt (σ sip)) ip" have "the (dhops (rt (σ sip)) ip) ≤ hops π ∨ the (flag (rt (σ sip)) ip) = inv)"
      by simp
      thus ?thesis
    proof
      assume "the (dhops (rt (σ sip)) ip) ≤ hops π"
      with "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)"
      have "the (dhops (rt (σ' sip)) ip) ≤ hops π by simp
      with "sn = nsqn (rt (σ' sip)) ip" show ?thesis by simp
    next
      assume "the (flag (rt (σ sip)) ip) = inv"
      with "ip ∈ kD(rt (σ sip))" have "sqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1" ..
      with "sn ≥ 1" and "sn = nsqn (rt (σ sip)) ip" have "sqn (rt (σ sip)) ip > 1" by simp
      from "ip ∈ kD(rt (σ' sip))" show ?thesis
      proof (rule vD_or_iD)
assume "ip ∈ D(rt (σ’ sip))"
hence "the (flag (rt (σ’ sip)) ip) = inv" ..
with ⟨sn = nsqn (rt (σ’ sip)) ip⟩ show ?thesis
by simp
next
assume "ip ∈ vD(rt (σ’ sip))"
hence "nsqn (rt (σ’ sip)) ip = sqn (rt (σ’ sip)) ip" ..
with ⟨sqn (rt (σ’ sip)) ip⟩ show "ip ∈ kD (rt (σ’ sip)) ∧ ?nsqafter" ..
qed

lemma quality_increases_rreq_rrep_props'::
  fixes sn ip hops sip
  assumes "∀ j. quality_increases (σ j) (σ’ j)"
  and "1 ≤ sn" and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
  ∧ (nsqn (rt (σ sip)) ip = sn
  --> (the (dhops (rt (σ sip)) ip) ≤ hops
  ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip ∈ kD(rt (σ’ sip)) ∧ sn ≤ nsqn (rt (σ’ sip)) ip
  ∧ (nsqn (rt (σ’ sip)) ip = sn
  --> (the (dhops (rt (σ’ sip)) ip) ≤ hops
  ∨ the (flag (rt (σ’ sip)) ip) = inv))"
proof -
  from assms(1) have "quality_increases (σ sip) (σ’ sip)" ..
  thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props)
qed

lemma rteq_quality_increases:
  assumes "∀ j. j ≠ i --> quality_increases (σ j) (σ’ j)"
  and "rt (σ’ i) = rt (σ i)"
  shows "∀ j. quality_increases (σ j) (σ’ j)"
using assms by clarsimp (metis order_refl quality_increasesI rt_fresher_refl)

definition msg_fresh :: "'a ⇒ state ⇒ msg ⇒ bool"
where "msg_fresh σ m ≡
case m of Rreq hopsc _ _ _ oipc osnc sipc ⇒ osnc ≥ 1 ∧ (sipc ≠ oipc -->
oipc ∈ D(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc
∧ (nsqn (rt (σ sipc)) oipc = osnc
  --> (hopsc ≥ the (dhops (rt (σ sipc)) oipc)
    ∨ the (flag (rt (σ sipc)) oipc) = inv)))
Rrep dipc dsnc _ sipc ⇒ dsnc ≥ 1 ∧ (sipc ≠ dipc -->
dipc ∈ D(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc
∧ (nsqn (rt (σ sipc)) dipc = dsnc
  --> (hopsc ≥ the (dhops (rt (σ sipc)) dipc)
    ∨ the (flag (rt (σ sipc)) dipc) = inv)))
Rerr destsc sipc ⇒ (∀ ripc ∈ dom(destsc). (ripc ∈ D(rt (σ sipc))
  --> the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))
| _ ⇒ True"
lemma msg_fresh [simp]:
  "\( \forall \text{hops rreqid dip dsns} \) \( \text{msg_fresh} \sigma (Rreq \text{hops rreqid dip dsns} oips osns) = \)
  \( \quad \text{osn} \geq 1 \land (\text{sip} \neq \text{oip} \rightarrow \text{oip} \in \text{kD}(rt (\sigma \text{sip})) \)
  \quad \land \text{nsqn} (rt (\sigma \text{sip})) \text{oip} \geq \text{osn} \)
  \quad \land (\text{nsqn} (rt (\sigma \text{sip})) \text{oip} = \text{osn} \rightarrow (\text{hops} \geq \text{the \dhops} (rt (\sigma \text{sip})) \text{oip}) \)
  \quad \lor \text{the (flag (rt (\sigma \text{sip})) oip) = inv}))" 

"\( \forall \text{hops dip dsns} \) \( \text{msg_fresh} \sigma (Rrep \text{hops dip dsns} oips) = \)
  \( \quad \text{dsn} \geq 1 \land (\text{sip} \neq \text{dip} \rightarrow \text{dip} \in \text{kD}(rt (\sigma \text{sip})) \)
  \quad \land \text{nsqn} (rt (\sigma \text{sip})) \text{dip} \geq \text{dsn} \)
  \quad \land (\text{nsqn} (rt (\sigma \text{sip})) \text{dip} = \text{dsn} \rightarrow (\text{hops} \geq \text{the \dhops} (rt (\sigma \text{sip})) \text{dip})) \)
  \quad \lor \text{the (flag (rt (\sigma \text{sip})) \text{dip}) = inv}))" 

"\( \forall \text{dests sip} \) \( \text{msg_fresh} \sigma (Rerr \text{dests sip}) = \)
  \( \quad \forall \text{ripc} \in \text{dom(dests)} \). \text{ripc} \in \text{kD}(rt (\sigma \text{sip})) \land \text{the (dests \text{ripc}) - 1} \leq \text{nsqn} (rt (\sigma \text{sip}) \text{ripc})" 

\( \forall \text{d dip} \). \text{msg_fresh} \sigma (Newpkt \text{d dip}) = \text{True}" 

\( \forall \text{d dip sip} \). \text{msg_fresh} \sigma (Pkt \text{d dip sip}) = \text{True}"

unfolding msg_fresh_def by simp_all

lemma msg_fresh_inc_sn [simp, elim]:
  "\( \text{msg_fresh} \sigma m \implies \text{rreq_rrep_sn} \text{m} \)"
by (cases m) simp_all

lemma recv_msg_fresh_inc_sn [simp, elim]:
  "\( \text{orecvmsg (msg_fresh)} \sigma m \implies \text{recvmsg rreq_rrep_sn} \text{m} \)"
by (cases m) simp_all

lemma rreq_msg_is_fresh [simp]:
  fixes \( \sigma \) \( \text{msg hops rreqid dip dsns} \) \( \text{osns} \) \( \text{sips} \)
assumes "\( \text{rreq_rrep_fresh} (rt (\sigma \text{sip})) \) \( \text{Rreq hops rreqid dip dsns} \) \( \text{oips osns} \)"
and "\( \text{rreq_rrep_sn} \) \( \text{Rreq hops rreqid dip dsns} \) \( \text{oips osns} \)"
shows "\( \text{msg_fresh} \sigma \) \( \text{Rreq hops rreqid dip dsns} \) \( \text{oips osns} \)"
(is "\( \text{msg_fresh} \sigma ?msg \)"

proof -
  let \( ?rt = \text{rt (\sigma \text{sip})} \)
  from \( \text{assms(2)} \) have "\( 1 \leq \text{osn} \)" by simp
  thus \( \text{?thesis} \)
  unfolding msg_fresh_def
  proof (simp only: \( \text{msg.case} \), intro conjI impI)
    assume "\( \text{\text{sip} \neq \text{oip}} \)"
    with \( \text{assms(1)} \) show "\( \text{oip} \in \text{kD(\text{rt})} \)" by simp
  next
    assume "\( \text{\text{sip} \neq \text{oip}} \)"
    and "\( \text{\text{nsqn \text{?rt oip} = \text{osn}} \)"
    show "\( \text{the (\dhops \text{?rt oip})} \leq \text{\text{hops}} \lor \text{the (\text{flag \text{?rt oip}) = \text{inv}}} \)"
    proof (cases "\( \text{\text{oip} \in \text{vD}(\text{\text{rt})}} \)"
      assume "\( \text{oip} \in \text{vD(\text{\text{rt})}} \)"
      hence "\( \text{\text{nsqn \text{\text{rt oip} = \text{sqn \text{\text{rt oip}}}}} \)" ..
      with \( \text{\text{\text{nsqn \text{\text{rt oip} = \text{osn}}}}} \) have "\( \text{\text{sqn \text{\text{rt oip} = \text{osn}}} \)" by simp
      with \( \text{\text{assms(1)}} \) and \( \text{\text{\text{sip} \neq \text{oip}}} \) have "\( \text{the (\dhops \text{\text{rt oip})} \leq \text{\text{hops}}} \)"
      by simp
      thus \( \text{?thesis} \) ..
  next
    assume "\( \text{oip} \in \text{vD(\text{\text{rt})}} \)"
    moreover from \( \text{\text{assms(1)}} \) and \( \text{\text{\text{sip} \neq \text{oip}}} \) have "\( \text{\text{oip} \in \text{kD(\text{\text{rt})}}} \)" by simp
    ultimately have "\( \text{\text{oip} \in \text{\text{iD(\text{\text{rt})}}} \)" by auto
    hence "\( \text{\text{the (\text{\text{flag \text{\text{rt oip}) = \text{inv}}} \)" ..
    thus \( \text{?thesis} \) ..
  qed
next
  assume "\( \text{\text{\text{\text{\text{\text{\text{sip}}} = \text{\text{\text{\text{\text{\text{oip}}}}} \)"}

58
with assms(1) have "osn ≤ sqn ?rt oip" by auto
thus "osn ≤ nsqn (rt (σ sip)) oip" by auto

proof (rule nat_le_eq_or_lt)
  assume "osn < sqn ?rt oip" hence "osn ≤ sqn ?rt oip - 1" by simp
  also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
  finally show "osn ≤ nsqn ?rt oip".

next
  assume "osn = sqn ?rt oip"
  with assms(1) and ⟨sip ≠ oip⟩ have "oip∈kD(?rt)"
  and "the (flag ?rt oip) = val" by auto
  hence "nsqn ?rt oip = sqn ?rt oip" ..
  thus "osn ≤ nsqn ?rt oip" by simp
qed simp

definition rrep_nsqn_is_fresh [simp]:
  fixes σ msg hops dip dsn oip sip
  assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
  and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
  shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
  (is "msg_fresh σ ?msg")

proof -
  let ?rt = "rt (σ sip)"
  from assms have "sip ≠ dip → dip∈kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val" by simp
  hence "sip ≠ dip → dip∈kD(?rt) ∧ nsqn ?rt dip ≥ dsn" by clarsimp
  with assms show "msg_fresh σ ?msg" by clarsimp
qed simp

definition rerr_nsqn_is_fresh [simp]:
  fixes σ msg dests sip
  assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
  shows "msg_fresh σ (Rerr dests sip)"
  (is "msg_fresh σ ?msg")

proof -
  let ?rt = "rt (σ sip)"
  from assms have *: "(∀ rip∈dom(dests). (rip∈iD(rt (σ sip))
              ∧ the (dests rip) = sqn (rt (σ sip)) rip))"
    by clarsimp
  have "(∀ rip∈dom(dests). (rip∈kD(rt (σ sip))
              ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip))"
  proof
    fix rip
    assume "rip ∈ dom dests"
    with * have "rip∈iD(rt (σ sip))" and "the (dests rip) = sqn (rt (σ sip)) rip"
      by auto
    from this(2) have "the (dests rip) - 1 = sqn (rt (σ sip)) rip - 1" by simp
    also have "... ≤ nsqn (rt (σ sip)) rip" by (rule sqn_nsqn)
    finally have "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip".
    with ⟨rip∈iD(rt (σ sip))⟩
    show "rip∈kD(rt (σ sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
      by clarsimp
  qed
  thus "msg_fresh σ ?msg" by simp
qed simp
lemma quality_increases_msg_fresh [elim]:
  assumes qinc: "∀ j. quality_increases (σ j) (σ' j)"
  and "msg_fresh σ m"
  shows "msg_fresh σ' m"
using assms(2)
proof (cases m)
  fix hops rreqid dip dsn dsk oip osn sip
  assume [simp]: "m = Rreq hops rreqid dip dsn dsk oip osn sip"
  and "msg_fresh σ m"
  then have "osn ≥ 1" and "sip = oip ∨ (oip∈kD(rt (σ sip)) ∧ osn ≤ nsqn (rt (σ sip)) oip
  ∧ (nsqn (rt (σ sip)) oip = osn
  → (the (dhops (rt (σ sip)) oip) ≤ hops
  ∨ the (flag (rt (σ sip)) oip) = inv)))"
by auto
from this(2) show ?thesis
proof
  assume "sip = oip" with ⟨osn ≥ 1⟩ show ?thesis by simp
next
  assume "oip∈kD(rt (σ sip)) ∧ osn ≤ nsqn (rt (σ sip)) oip
  ∧ (nsqn (rt (σ sip)) oip = osn
  → (the (dhops (rt (σ sip)) oip) ≤ hops
  ∨ the (flag (rt (σ sip)) oip) = inv))"
moreover from qinc have "quality_increases (σ sip) (σ' sip)" ..
ultimately have "oip∈kD(rt (σ' sip)) ∧ osn ≤ nsqn (rt (σ' sip)) oip
  ∧ (nsqn (rt (σ' sip)) oip = osn
  → (the (dhops (rt (σ' sip)) oip) ≤ hops
  ∨ the (flag (rt (σ' sip)) oip) = inv))"
  using ⟨osn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2])
with ⟨osn ≥ 1⟩ show "msg_fresh σ' m"
  by (clarsimp)
qed
next
  fix hops dip dsn oip sip
  assume [simp]: "m = Rrep hops dip dsn oip sip"
  and "msg_fresh σ m"
  then have "dsn ≥ 1" and "sip = dip ∨ (dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip
  ∧ (nsqn (rt (σ sip)) dip = dsn
  → (the (dhops (rt (σ sip)) dip) ≤ hops
  ∨ the (flag (rt (σ sip)) dip) = inv))"
by auto
from this(2) show "?thesis"
proof
  assume "sip = dip" with ⟨dsn ≥ 1⟩ show ?thesis by simp
next
  assume "dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip
  ∧ (nsqn (rt (σ sip)) dip = dsn
  → (the (dhops (rt (σ sip)) dip) ≤ hops
  ∨ the (flag (rt (σ sip)) dip) = inv))"
moreover from qinc have "quality_increases (σ sip) (σ' sip)" ..
ultimately have "dip∈kD(rt (σ' sip)) ∧ dsn ≤ nsqn (rt (σ' sip)) dip
  ∧ (nsqn (rt (σ' sip)) dip = dsn
  → (the (dhops (rt (σ' sip)) dip) ≤ hops
  ∨ the (flag (rt (σ' sip)) dip) = inv))"
  using ⟨dsn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2])
with ⟨dsn ≥ 1⟩ show "msg_fresh σ' m"
  by clarsimp
qed
next
  fix dests sip
  assume [simp]: "m = Rerr dests sip"
  and "msg_fresh σ m"
  then have "∀ rip∈dom(dests). rip∈kD(rt (σ sip))
  ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
by simp
have "∀ rip∈dom(dests). rip∈kD(rt (σ' sip))
∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip"
proof
fix rip
assume "rip∈dom(dests)"
with * have "rip∈kD(rt (σ sip))" and "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
by - (drule(1) bspec, clarsimp)+
moreover from qinc have "quality_increases (σ sip) (σ' sip)" by simp
ultimately show "rip∈kD(rt (σ' sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" ..
qed
thus ?thesis by simp
qed simp_all
end

0.9 The ‘open’ AODV model

theory OAODV
imports AodvAWN.OAWN_SOS_LabelsAWN.OAWN_Convert
begin
Definitions for stating and proving global network properties over individual processes.
definition σAODV' :: "((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp)) set"
where "σAODV' ≡ {λ i. aodv_init i, ΓAODV PAodv)}"
abbreviation opaodv
:: "ip ⇒ ((ip ⇒ state) × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where "opaodv i ≡ (| init = σAODV', trans = oseqp_sos ΓAODV i |)"
lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
unfolding σAODV'_def
by rule simp_all
lemma oaadv_control_within [simp]: "control_within ΓAODV (init (opaodv i))"
unfolding σAODV'_def by (rule control_withinI) (auto simp del: ΓAODV'_simps)
lemma σAODV'_labels [simp]: "(σ, p) ∈ σAODV' ⇒ labels ΓAODV p = {PAodv-:0}"
unfolding σAODV'_def by simp
lemma oaadv_init_kD_empty [simp]: "(σ, p) ∈ σAODV' ⇒ kD (rt (σ i)) = {}"
unfolding σAODV'_def kD_def by simp
lemma oaadv_init_vD_empty [simp]: "(σ, p) ∈ σAODV' ⇒ vD (rt (σ i)) = {}"
unfolding σAODV'_def vD_def by simp
lemma oaadv_trans: "trans (opaodv i) = oseqp_sos ΓAODV i"
by simp
declare
oseq_invariant_ctermsI [OF aodv_wf oaadv_control_within aodv_simple_labels aodv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf oaadv_control_within aodv_simple_labels aodv_trans, cterms_intros]
end

0.10 Global invariant proofs over sequential processes

theory Global_Invariants
imports Seq_Invariants Aodv_Predicates Fresher


lemma other_quality_increases [elim]:

assumes "other quality_increases I σ σ'"

shows "∀ j. quality_increases (σ j) (σ' j)"

using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:

fixes m

assumes *: "otherwith P I (orecvmsg Q) σ σ' a"

and weakenP: "∀ σ m. P σ m =⇒ P' σ m"

and weakenQ: "∀ σ m. Q σ m =⇒ Q' σ m"

shows "otherwith P' I (orecvmsg Q') σ σ' a"

proof

fix j

assume "j /∈ I"

with * have "P (σ j) (σ' j)" by auto

thus "P' (σ j) (σ' j)" by (rule weakenP)

next

from * have "orecvmsg Q σ a" by auto

thus "orecvmsg Q' σ a" by rule (erule weakenQ)

qed

lemma oreceived_msg_inv:

assumes other: "∀ σ σ' m. [| P σ m; other Q {i} σ σ' |] =⇒ P σ' m"

and local: "∀ σ m. P σ m =⇒ P (σ (i := σ i (msg := m)) m)"

shows "opaodv i ||= (otherwith Q {i} (orecvmsg P), other Q {i} →)

onl Γ_{AODV} (λ(σ, l). l ∈ {PAodv-:1} → P σ (msg (σ i)))"

proof (inv_cterms, intro impI)

fix σ σ' l

assume "l = PAodv-:1 → P σ (msg (σ i))"

and "l = PAodv-:1"

and "other Q {i} σ σ'"

from this(1-2) have "P σ (msg (σ i))" ..

hence "P σ' (msg (σ i))" using 'other Q {i} σ σ'

by (rule other)

moreover from 'other Q {i} σ σ' have "σ' i = σ i" ..

ultimately show "P σ' (msg (σ' i))" by simp

next

fix σ σ' msg

assume "otherwith Q {i} (orecvmsg P) σ σ' (receive msg)"

and "σ' i = σ i[[msg := msg]]"

from this(1) have "P σ msg"

and "∀ j ≠ i Q (σ j) (σ' j)" by auto

from this(1) have "P (σ (i := σ i[[msg := msg]]) msg)" by (rule local)

thus "P σ' msg"

proof (rule other)

from "σ' i = σ i[[msg := msg]]" and "∀ j ≠ i Q (σ j) (σ' j)"

show "other Q {i} (σ (i := σ i[[msg := msg]]) σ' i)"

by - (rule otherI, auto)

qed

qed

(Equivalent to) Proposition 7.27

lemma local_quality_increases:

"paodv i ||= (recvmsg rreq_rrep_sn → onll Γ_{AODV} (λ((ξ, _), (ξ', _)). quality_increases ξ ξ'))"

proof (rule step_invariantI)

fix s a s'

assume sr: "s ∈ reachable (paodv i) (recvmsg rreq_rrep_sn)"

and tr: "(s, a, s') ∈ trans (paodv i)"

62
and rm: "recvmsg rreq_rrep_sn a"
from sr have srTT: "s ∈ reachable (paodv i) TT" ..

from route_tables_fresher sr tr rm
have "onll Γ_AODV (λ((ξ, ...), (ξ', ...)). ∀ dip∈kD (rt ξ). rt ξ ⊑ dip rt ξ') (s, a, s')"
  by (rule step_invariantD)
moreover from known_destinations_increase srTT tr TT_True
have "onll Γ_AODV (λ((ξ, ...), (ξ', ...)). kD (rt ξ) ⊆ kD (rt ξ')) (s, a, s')"
  by (rule step_invariantD)
moreover from sqns_increase srTT tr TT_True
have "onll Γ_AODV (λ((ξ, ...), (ξ', ...)). ∀ dip. sqn (rt ξ) ≤ sqn (rt ξ')) (s, a, s')"
  by (rule step_invariantD)
ultimately show "onll Γ_AODV (λ((ξ, ...), (ξ', ...)). quality_increases ξ ξ') (s, a, s')"
unfolding onll_def by auto
qed

lemmas olocal_quality_increases =
open_seq_step_invariant [OF local_quality_increases initiali_aodv oaodv_trans aodv_trans,
simplified seqll_onll_swap]

lemma quality_increases:
"opaodv i |=A (otherwith quality_increases {i} (orecvmsg (λ_. rreq_rrep_sn)),
other quality_increases {i} →)
  onll Γ_AODV (λ((ξ, ...), (ξ', ...)). ∀ j. quality_increases (σ j) (σ' j))"
(is "_=A (?S, _ →)")
proof (rule onll_ostep_invariantI, simp)
  fix σ p a σ' p' l'
  assume or: "((σ, p), a) ∈ oreachable (opaodv i) ?S (other quality_increases {i})"
  and ll: "l ∈ labels Γ_AODV p"
  and ?S σ σ' a
  and tr: "((σ, p), a, (σ', p')) ∈ oseqp_sos Γ_AODV i"
  and ll': "l' ∈ labels Γ_AODV p''"
  from this(1-3) have "orecvmsg (λ_. rreq_rrep_sn) σ a"
    by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
and QU="other quality_increases {i}"], otherwith_actionD)
  with or have crw: "((σ, p), a) ∈ oreachable (opaodv i) (act (recvmsg rreq_rrep_sn))
  (other quality_increases {i})"
    by (erule oreachable_weakenE, auto)
  with tr ll ll' and orecvmsg (λ_. rreq_rrep_sn) σ a have "quality_increases (σ i) (σ' i)"
    by (erule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def)
  with ?S σ σ' a show "∀ j. quality_increases (σ j) (σ' j)"
    by (auto dest!: otherwith_syncD)
qed

lemma rreq_rrep_nsqn_fresh_any_step_invariant:
"opaodv i |=A (act (recvmsg rreq_rrep_sn), other A {i} →)
  onll Γ_AODV (λ(σ, a). anycast (msg_fresh σ) a)"
proof (rule ostep_invariantI, simp del: act_simp)
  fix σ p a σ' p' l'
  assume or: "((σ, p), a, (σ', p')) ∈ oseqp_sos Γ_AODV i"
  and recv: "act (recvmsg rreq_rrep_sn) τ σ σ' a"
  obtain l l' where "l ∈ labels Γ_AODV p" and "l' ∈ labels Γ_AODV p''"
    by (metis aodv_ex_label)
  from "((σ, p), a, (σ', p')) ∈ oseqp_sos Γ_AODV i"
  have tr: "((σ, p), a, (σ', p')) ∈ trans (opaodv i)" by simp
  have "anycast (rreq_rrep_fresh (rt (σ i))) a"
    proof -
      have "opaodv i |=A (act (recvmsg rreq_rrep_sn), other A {i} →)"
onll $\Gamma_{AODV}$ (seqll i ($\lambda((\xi, _), a, _)$. anycast (req_rrep_fresh (rt $\xi$)) a))
by (rule ostep_invariant_weakenE [OF
open_seq_step_invariant [OF req_rrep_fresh_any_step_invariant initiali_aodv,
simplified seqll_onll_swap]]) auto
hence "onll $\Gamma_{AODV}$ (seqll i ($\lambda((\xi, _), a, _)$. anycast (req_rrep_fresh (rt $\xi$)) a))
((\sigma, p), a, (\sigma', p'))"
using or tr recv by - (erule(4) ostep_invariantE)
thus ?thesis
using $l \in \text{labels } \Gamma_{AODV}$ p and $l' \in \text{labels } \Gamma_{AODV}$ p' by auto
qed

moreover have "anycast (rerr_invalid (rt (\sigma i))) a"
proof -
have "opaodv i $\models_{A} (\text{act } (\text{recvmsg } \text{req_rrep_sn}), \text{other } A \{i\} \rightarrow)
onll \Gamma_{AODV}$ (seqll i ($\lambda((\xi, _), a, _)$. anycast (rerr_invalid (rt $\xi$)) a))"
by (rule ostep_invariant_weakenE [OF
open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aodv,
simplified seqll_onll_swap]]) auto
hence "onll $\Gamma_{AODV}$ (seqll i ($\lambda((\xi, _), a, _)$. anycast (rerr_invalid (rt $\xi$)) a))
((\sigma, p), a, (\sigma', p'))"
using or tr recv by - (erule(4) ostep_invariantE)
thus ?thesis
using $l \in \text{labels } \Gamma_{AODV}$ p and $l' \in \text{labels } \Gamma_{AODV}$ p' by auto
qed

moreover have "anycast \text{req_rrep_sn a}"
proof -
from or tr recv
have "onll $\Gamma_{AODV}$ (seqll i ($\lambda(_, a, _). \text{anycast } \text{req_rrep_sn a}$))
((\sigma, p), a, (\sigma', p'))"
by (rule ostep_invariantE [OF
open_seq_step_invariant [OF req_rrep_sn_any_step_invariant initiali_aodv,
aodv_trans aodv_trans,
simplified seqll_onll_swap]])
thus ?thesis
using $l \in \text{labels } \Gamma_{AODV}$ p and $l' \in \text{labels } \Gamma_{AODV}$ p' by auto
qed

moreover have "anycast ($\lambda m. \text{not_Pkt } m \rightarrow \text{msg_sender } m = i$) a"
proof -
have "opaodv i $\models_{A} (\text{act } (\text{recvmsg } \text{req_rrep_sn}), \text{other } A \{i\} \rightarrow)
onll \Gamma_{AODV}$ (seqll i ($\lambda((\xi, _), a, _)$. anycast ($\lambda m. \text{not_Pkt } m \rightarrow \text{msg_sender } m = i$) a))"
by (rule ostep_invariant_weakenE [OF
open_seq_step_invariant [OF sender_ip_valid initiali_aodv,
simplified seqll_onll_swap]]) auto
hence "onll $\Gamma_{AODV}$ (seqll i ($\lambda((\xi, _), a, _)$. anycast ($\lambda m. \text{not_Pkt } m \rightarrow \text{msg_sender } m = i$) a))
((\sigma, p), a, (\sigma', p'))"
using or tr recv by - (drule(3) onll_ostep_invariantD, auto)
thus ?thesis using $l \in \text{labels } \Gamma_{AODV}$ p and $l' \in \text{labels } \Gamma_{AODV}$ p' by auto
qed

ultimately have "anycast (msg_fresh $\sigma$) a"
by (simp_all add: anycast_def
del: msg_fresh
split: seq.action.split_asm msg.split_asm) simp_all
thus "onll $\Gamma_{AODV}$ ($\lambda((\sigma, _), a, _). \text{anycast } (\text{msg_fresh } \sigma) a$)
((\sigma, p), a, (\sigma', p'))"
by auto
qed

lemma oreceived_req_rrep_msg_sn_fresh_inv:
"opaodv i $\models_{A} (\text{otherwith } \text{quality_increases } \{i\} (\text{orecvmsg } \text{msg_fresh}),\text{other } \text{quality_increases } \{i\} \rightarrow)
onll \Gamma_{AODV}$ ($\lambda(\sigma, 1). 1 \in \{\text{PAodv-:1}\} \rightarrow \text{msg_fresh } \sigma (\text{msg } (\sigma i))$)"
proof (rule oreceived_msg_inv)
fix $\sigma \sigma'$ m
assume *: "\text{msg_fresh } \sigma \ m"
and "\text{other } \text{quality_increases } \{i\} \sigma \sigma'"

64
from this(2) have "∀ j. quality_increases (σ j) (σ’ j)" ..
thus "msg_fresh σ’ m" using * ..

next
fix σ m
assume "msg_fresh σ m"
thus "msg_fresh (σ i := σ i(msg := m)) m"

proof (cases m)
  fix dests sip
  assume "m = Rerr dests sip"
  with msg_fresh σ m show ?thesis
    by auto
  qed auto

qed

lemma quality_increases_nsqn_fresh:
"opaodv i ⊨ A (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onl Γ AODV (λ((σ, _), _, (σ’, _)). ∀ j. quality_increases (σ j) (σ’ j))"
by (rule ostep_invariant_weakenE [OF quality_increases]) auto

lemma osn_rreq:
"opaodv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onl Γ AODV (seq l (λ(τ, l). 1 ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-:0 ln. True} → 1 ≤ osn (ξ)))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq_initiali_aodv]])
(auto simp: seq1_onl_swap)

lemma rreq_sip:
"opaodv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onl Γ AODV (λ(σ, 1).
(1 ∈ {PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2} ∨ nip (σ i) ≠ oip (σ i))
→ oip (σ i) ∈ kD (rt (σ (sip (σ i))))
∧ nsqn (rt (σ (sip (σ i)))) (oip (σ i)) ≥ osn (σ i)
∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i))
→ l ∈ {PRreq-:0 ln. True} ∨ (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (oip (σ i)))
∨ (the (flag (rt (σ (sip (σ i)))) (oip (σ i)))) = inv))"
(is "_ ⊨ (?S, ?U → _)")

proof (inv cterms inv add: oseq_step_invariant_stems [OF quality_increases_nsqn_fresh
aadv_wf aodv_trans]
onl oinvariant_stems [OF aodv_wf orecvmsg_msg_fresh_rrep]
onl oinvariant_stems [OF aodv_wf osn_rreq]
simp add: seqlsimp
simp del: One_nat_def, rule impI)

fix σ σ’ p l
assume "("σ, p") ∈ oreachable (opaodv i) ?S ?U"
and "1 ∈ labels Γ AODV p"

and pre:
"(1 = PAodv-:4 ∨ 1 = PAodv-:5 ∨ 1 = PRreq-:0 ∨ 1 = PRreq-:2) ∨ nip (σ i) ≠ oip (σ i)
→ oip (σ i) ∈ kD (rt (σ (sip (σ i))))
∧ osn (σ i) ≤ nsqn (rt (σ (sip (σ i)))) (oip (σ i))
∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i))
→ the (dhops (rt (σ (sip (σ i)))) (oip (σ i))) ≤ hops (σ i)
∨ (the (flag (rt (σ (sip (σ i)))) (oip (σ i)))) = inv"

and "other quality_increases {i} σ σ’" and hyp:
"(1 = PAodv-:4 ∨ 1 = PAodv-:5 ∨ 1 = PRreq-:0 ∨ 1 = PRreq-:2) ∨ nip (σ’ i) ≠ oip (σ’ i)"
(is "labels ∧ nip (σ’ i) ≠ oip (σ’ i)"
from this(4) have "σ’ i = σ i" ..
with hyp have hyp': "labels ∧ nip (σ i) ≠ oip (σ i)" by simp
show "oip (σ’ i) ∈ kD (rt (σ’ (sip (σ’ i))))
∧ osn (σ’ i) ≤ nsqn (rt (σ’ (sip (σ’ i)))) (oip (σ’ i))
∧ (nsqn (rt (σ’ (sip (σ’ i)))) (oip (σ’ i)) = osn (σ’ i))
→ the (dhops (rt (σ’ (sip (σ’ i)))) (oip (σ’ i))) ≤ hops (σ’ i)
∨ (the (flag (rt (σ’ (sip (σ’ i)))) (oip (σ’ i)))) = inv"
proof (cases "sip (σ i) = i")

assume "sip (σ i) ≠ i"
from other quality increases {i} σ σ' have "quality_increases (σ (sip (σ i))) (σ' (sip (σ' i)))"
by (rule otherE) (clarsimp simp: sip (σ i) ≠ i)
moreover from ⟨σ, p⟩ ∈ oreachable (opaodv i) ?S ?U l ∈ labels Γ_AODV p and hyp have "1 ≤ osn (σ' i)"
by (auto dest!: onl_oinvariant_weakenD [OF oosn_rreq]
simp add: seqlsimp ⟨σ, σ⟩)
moreover from :sip (σ i) ≠ i hyp' and pre have "oip (σ' i) ∈ kD (rt (σ (sip (σ i)))) ∧ osn (σ' i) ≤ nsqn (rt (σ (sip (σ i)))) (oip (σ' i)) ∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ' i)) = osn (σ' i)) → (hops (σ i) ≥ hops (σ' i)) ∨ (the (flag (rt (σ (sip (σ i)))) (oip (σ' i))) = inv)"
by (auto simp: σ' i = σ i)
ultimately show ?thesis
by (rule quality_increases_rreq_rrep_props)

next
assumes "sip (σ i) = i" thus ?thesis
using ⟨σ' i = σ i⟩ hyp and pre by auto

qed (auto elim!: quality_increases_rreq_rrep_props')

lemma odsn_rrep:
"opaodv i = (otherwith quality_increases {i} (orecvmsg msg_fresh), other quality increases {i} →)
onl Γ_AODV (seql (λ(τ, l). l ∈ {PAodv-:6, PAodv-:7, PRrep-:0, PRrep-:1}) ∧ sip (σ i) ≠ dip (σ i)) → dip (σ i) ∈ kD (rt (σ (sip (σ i)))) ∧ nsqn (rt (σ (sip (σ i)))) (dip (σ i)) ≥ dsn (σ i) ∧ (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dsn (σ i) → (hops (σ i) ≥ hops (σ i)) ∨ (the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv))"
by (rule onl_invariant_weakenE [OF open_seq_invariant [OF odsn_rrep_initial_aodv]])

(auto simp: seql_seql_swap)

lemma rrep_sip:
"opaodv i = (otherwith quality_increases {i} (orecvmsg msg_fresh), other quality increases {i} →)
onl Γ_AODV (λ(σ, i).
(1 ∈ {PAodv-:6, PAodv-:7, PRrep-:0, PRrep-:1}) ∧ sip (σ i) ≠ dip (σ i) → dip (σ i) ∈ kD (rt (σ (sip (σ i)))) ∧ nsqn (rt (σ (sip (σ i)))) (dip (σ i)) ≥ dsn (σ i) ∧ (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dsn (σ i) → (hops (σ i) ≥ hops (σ i)) ∨ (the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv))"

(is "_ = (?:, ?U →) _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF quality_increases NSAQ_FRESH AODV_wf]
oaadv_trans]
onl_onl_invariant_sterms [OF aoavg_wf oreceived_rreq_rrep_NSQ_FRESH_INV] 
onl_onl_invariant_sterms [OF aoavg_wf odsn_rrep]
simp del: One_nat_def, ruleImpl)

fix σ σ' p l
assume "⟨σ, p⟩ ∈ oreachable (opaodv i) ?S ?U"
and "l ∈ labels Γ_AODV p"
and pre: "(1 = PAodv-:6 ∨ 1 = PAodv-:7 ∨ 1 = PRrep-:0 ∨ 1 = PRrep-:1) ∧ sip (σ i) ≠ dip (σ i) → dip (σ i) ∈ kD (rt (σ (sip (σ i)))) ∧ dsn (σ i) ≤ nsqn (rt (σ (sip (σ i)))) (dip (σ i)) ∧ (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dsn (σ i) → (hops (σ i) ≥ hops (σ i)) ∨ (the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv))"
and "other quality increases {i} σ σ'"
and hyp: "(1 = PAodv-:6 ∨ 1 = PAodv-:7 ∨ 1 = PRrep-:0 ∨ 1 = PRrep-:1) ∧ sip (σ' i) ≠ dip (σ' i)"
(is "labels ∧ sip (σ' i) ≠ dip (σ' i)"
from this(4) have "σ' i = σ i" ..
with hyp have hyp': "labels ∧ sip (σ i) ≠ dip (σ i)" by simp
show "dip (σ' i) ∈ kD (rt (σ' (sip (σ' i)))) ∧ dsn (σ' i) ≤ nsqn (rt (σ' (sip (σ' i)))) (dip (σ' i))"
lemma rerr_sip:
"opaodv i \models (otherwith quality_increases \{i\} \ (orecvmsg msg_fresh),
other_quality_increases \{i\} \rightarrow)
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
(clarsimp simp del: One_nat_def split: if_split_asm option.split_asm, erule(2) partial)+
qed

lemma prerr_guard: "paodv i \|= 
  onl \Gamma_{AODV} (\lambda(\xi, l). (l = \text{PRerr-}:1 
  \rightarrow (\forall ip:\text{dom(dests} \xi). \text{ip\in}vD(rt \xi) 
    \land \text{the} (\text{nhop} (rt \xi) \text{ip}) = sip \xi 
    \land \text{sqn} (rt \xi) \text{ip} < \text{the} (\text{dests} \xi \text{ip})))"
by (inv_cterms) (clarsimp split: option.split_asm if_split_asm)

lemmas oaddpreRT_welldefined = 
  open_seq_invariant [OF addpreRT_welldefined initiali_aodv oaodv_trans aodv_trans, 
  simplified seql_onl_swap, 
  THEN oinvariant_anyact]
lemmas odests_vD_inc_sqn = 
  open_seq_invariant [OF dests_vD_inc_sqn initiali_aodv oaodv_trans aodv_trans, 
  simplified seql_onl_swap, 
  THEN oinvariant_anyact]
lemmas oprerr_guard = 
  open_seq_invariant [OF prerr_guard initiali_aodv oaodv_trans aodv_trans, 
  simplified seql_onl_swap, 
  THEN oinvariant_anyact]

Proposition 7.28
lemma seq_compare_next_hop':
  "opaodv i \|= (otherwith quality_increases {i} (orecmsg msg_fresh), 
  other quality_increases {i} \rightarrow) onl \Gamma_{AODV} (\lambda(\sigma, _). 
  \forall dip. let nhip = \text{the} (\text{nhop} (rt (\sigma i)) dip) 
  in dip \in kD(rt (\sigma i)) \land nhip \neq dip \rightarrow 
  dip \in kD(rt (\sigma nhip)) \land sqn (rt (\sigma i)) dip \leq sqn (rt (\sigma nhip)) dip)"
(is "_ \|= (?S, ?U \rightarrow) _")
proof - 
{ fix nhop and \sigma \sigma' :: "ip \Rightarrow state"
  assume pre: "\forall dip\in kD(rt (\sigma i)). nhop dip \neq dip \rightarrow 
    dip\in kD(rt (\sigma (nhop dip))) \land sqn (rt (\sigma i)) dip \leq sqn (rt (\sigma (nhop dip))) dip"
  and qinc: "\forall j. quality_increases (\sigma j) (\sigma' j)"
  have "\forall dip\in kD(rt (\sigma i)). nhop dip \neq dip \rightarrow 
    dip\in kD(rt (\sigma' (nhop dip))) \land sqn (rt (\sigma i)) dip \leq sqn (rt (\sigma' (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip 
    assume "dip\in kD(rt (\sigma i))"
    and "nhop dip \neq dip"
    with pre have "dip\in kD(rt (\sigma (nhop dip)))"
    and "sqn (rt (\sigma i)) dip \leq sqn (rt (\sigma (nhop dip))) dip"
    by auto 
    from qinc have qinc_nhop: "quality_increases (\sigma (nhop dip)) (\sigma' (nhop dip))" ..
    with "dip\in kD(rt (\sigma (nhop dip)))" have "dip\in kD(rt (\sigma' (nhop dip)))" ..
    moreover have "sqn (rt (\sigma i)) dip \leq sqn (rt (\sigma' (nhop dip))) dip"
    proof -
      from "dip\in kD(rt (\sigma (nhop dip)))" qinc_nhop 
      have "sqn (rt (\sigma (nhop dip))) dip \leq sqn (rt (\sigma' (nhop dip))) dip" ..
      with "sqn (rt (\sigma i)) dip \leq sqn (rt (\sigma (nhop dip))) dip" show ?thesis
      by simp 
    qed 
    ultimately show "dip\in kD(rt (\sigma' (nhop dip))) 
    \land sqn (rt (\sigma i)) dip \leq sqn (rt (\sigma' (nhop dip))) dip" ..
  qed 
} note basic = this
{ fix nhop and σ σ' :: "ip ⇒ state"
 assume pre: "∀dikD(rt (σ i)). nhop dip ≠ dip → dikD(rt (σ (nhop dip)))
     ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
   and ndest: "∀ripc dom (dests (σ i)). ripc ∈ kD (rt (σ (sip (σ i)))))
     ∧ the (dests (σ i) ripc) - 1 ≤ nsqn (rt (σ (sip (σ i))) ripc"
   and issip: "∀ip dom (dests (σ i)). nhop ip = sip (σ i)"
   and qinc: "∀j. quality_increases (σ j) (σ' j)"
 have "∀dip dikD(rt (σ i)). nhop dip ≠ dip → dip ∈ kD (rt (σ' (nhop dip)))
     ∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
 proof (intro ballI impI)
 fix dip
 assume "dip dikD(rt (σ i))"
 and "nhop dip ≠ dip"
 with pre and qinc have "dip dikD(rt (σ' (nhop dip)))"
   and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
 by (auto dest!: basic)
 have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
 proof (cases "dip dikD (dests (σ i))")
   assume "dip dikD (dests (σ i))"
   with dip dikD(rt (σ i)): obtain dsn where "dests (σ i) dip = Some dsn"
   by auto
   with dip dikD(rt (σ i)): have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = dsn - 1"
     by (rule nsqn_invalidate_eq)
   moreover have "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
   proof -
     from dests (σ i) dip = Some dsn have "the (dests (σ i) dip) = dsn" by simp
     with ndest and dip dikD (dests (σ i)) have "dip ∈ kD (rt (σ (sip (σ i))))"
     "dsn - 1 ≤ nsqn (rt (σ (sip (σ i))) dip)"
     by auto
   moreover from issip and dip dikD (dests (σ i)) have "nhop dip = sip (σ i) ..." ultimately have "dip ∈ kD (rt (σ (nhop dip)))"
     and "dsn - 1 ≤ nsqn (rt (σ (nhop dip))) dip" by auto
   with qinc show "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
     by simp (metis kD nsqn_quality_increases_trans)
 qed
 ultimately show ?thesis by simp
 next
 assume "dip ∉ dom (dests (σ i))"
 with dip dikD(rt (σ i)): have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = nsqn (rt (σ i)) dip"
   by (rule nsqn_invalidate_other)
 with nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip: show ?thesis by simp
 qed
 with dip dikD(rt (σ' (nhop dip)))):
   show "dip ∈ kD (rt (σ' (nhop dip)))"
     ∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ...
 qed
 } note basic_prerr = this

{ fix σ σ' :: "ip ⇒ state"
 assume a1: "∀dip dikD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
     → dikD(rt (σ (the (nhop (rt (σ i)) dip))))
     ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
 and a2: "∀j. quality_increases (σ j) (σ' j)"
 have "∀dip dikD(rt (σ i)).
     the (nhop (update (rt (σ i)) (sip (σ i)) (0, unk, val, Suc 0, sip (σ i), {})) dip) ≠ dip →
     dikD(rt (σ' (the (nhop (update (rt (σ i)) (sip (σ i))
       (0, unk, val, Suc 0, sip (σ i), {})) dip))))
     ∧
     nsqn (update (rt (σ i)) (sip (σ i)) (0, unk, val, Suc 0, sip (σ i), {})) dip ≤
     nsqn (rt (σ' (the (nhop (update (rt (σ i)) (sip (σ i))
       (0, unk, val, Suc 0, sip (σ i), {})) dip))))
     ∧
\(\text{dip} \) (is \(\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \ ?P \ \text{dip}\))

proof

fix dip

assume "\(\text{dip} \in \text{kD}(\text{rt}(\sigma i))\)"

with \(a1\) and \(a2\)

have "\(\text{the (nhop (rt (\sigma i)) dip) \neq dip \implies \text{dip} \in \text{kD}(\text{rt}(\sigma' (\text{the (nhop (rt (\sigma i)) dip))))) \land \text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))) dip)}\)"

by \(-\) (drule(1) basic, auto)

thus "\(?P \ \text{dip}\)" by (cases "\(\text{dip} = \text{sip} (\sigma i)\") auto

qed

} note nhop_update_sip = this

{ fix \(\sigma\) \(\sigma'\) oip sip osn hops

assume pre: "\(\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \ \text{the (nhop (rt (\sigma i)) dip) \neq dip} \implies \text{dip} \in \text{kD}(\text{rt}(\sigma' (\text{the (nhop (rt (\sigma i)) dip))))) \land \text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))) dip})\)"

and qinc: "\(\forall j. \ \text{quality_increases (\sigma j)} (\sigma' j)\)"

and *: "\(\text{sip} \neq \text{oip} \implies \text{oip} \in \text{kD}(\text{rt} (\sigma \text{sip})) \land \text{osn} \leq \text{nsqn (rt (\sigma \text{sip})) oip} \land (\text{nsqn (rt (\sigma \text{sip})) oip} = \text{osn} \implies (\text{the (dhops (rt (\sigma \text{sip})) oip}) \leq \text{hops} \lor (\text{the (flag (rt (\sigma \text{sip})) oip}) = \text{inv})\)"

from pre and qinc

have pre': "\(\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \ \text{the (nhop (rt (\sigma i)) dip) \neq dip} \implies \text{dip} \in \text{kD}(\text{rt}(\sigma' (\text{the (nhop (rt (\sigma i)) dip))))) \land \text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))) dip})\)"

by (rule basic)

have "\((\text{the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) oip) \neq oip} \implies \text{oip} \in \text{kD}(\text{rt}(\sigma' (\text{the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) oip)))) \land \text{nsqn (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) oip} \leq \text{nsqn (rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) oip)))) oip})\)"

(is "\(?\text{nop_not_oip} \implies \text{oip_in_kD} \land \text{nsqn_le_nsqn}\)"

proof (rule, split update_rt_split_asm)

assume "\(\text{rt (\sigma i)} = \text{update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})}\)"

and the (nhop (rt (\sigma i)) oip) \neq oip"

with pre' show "\(?\text{oip_in_kD} \land \text{nsqn_le_nsqn}\)" by auto

next

assume rtnot: "\(\text{rt (\sigma i)} \neq \text{update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})}\)"

and notoip: \(?\text{nop_not_oip}\)

with * qinc have \(?\text{oip_in_kD}\)

by auto

moreover with * pre qinc rtnot notoip have ?nsqn_le_nsqn

by simp (metis kD_nsqn_quality_increases_trans)

ultimately show "\(?\text{oip_in_kD} \land \text{nsqn_le_nsqn}\)" ..

qed

} note update1 = this

{ fix \(\sigma\) \(\sigma'\) oip sip osn hops

assume pre: "\(\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \ \text{the (nhop (rt (\sigma i)) dip) \neq dip} \implies \text{dip} \in \text{kD}(\text{rt}(\sigma' (\text{the (nhop (rt (\sigma i)) dip))))) \land \text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))) dip})\)"

and qinc: "\(\forall j. \ \text{quality_increases (\sigma j)} (\sigma' j)\)"

and *: "\(\text{sip} \neq \text{oip} \implies \text{oip} \in \text{kD}(\text{rt} (\sigma \text{sip})) \land \text{osn} \leq \text{nsqn (rt (\sigma \text{sip})) oip} \land (\text{nsqn (rt (\sigma \text{sip})) oip} = \text{osn} \implies (\text{the (dhops (rt (\sigma \text{sip})) oip}) \leq \text{hops} \lor (\text{the (flag (rt (\sigma \text{sip})) oip}) = \text{inv})\)"

from pre and qinc

have pre': "\(\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \ \text{the (nhop (rt (\sigma i)) dip) \neq dip} \implies \text{dip} \in \text{kD}(\text{rt}(\sigma' (\text{the (nhop (rt (\sigma i)) dip))))) \land \text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))) dip})\)"

by (rule basic)
have "∀ dip ∈ kD(rt (σ i)).
    the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip
    → dip ∈ kD(rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, {})) dip)))
    ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip
    ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, {})) dip))) dip"

(is "∀ dip ∈ kD(rt (σ i)). _ → ?dip in kD dip ∧ ?nsqn le nsqn dip")

proof
  (intro ballI impI, split update_rt_split_asm)
  fix dip
  assume "dip ∈ kD(rt (σ i))"
  and "the (nhop (rt (σ i)) dip) ≠ dip"
  and "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
  with pre' show "?dip in kD dip ∧ ?nsqn le nsqn dip" by simp

next
  fix dip
  assume "dip ∈ kD(rt (σ i))"
  and notdip: "the (nhop (update (rt (σ i)) oip
  (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"
  and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
  show "?dip in kD dip ∧ ?nsqn le nsqn dip"
  proof (cases "dip = oip")
    assume "dip ≠ oip"
    with pre' ⟨dip ∈ kD(rt (σ i))⟩ notdip *
    have "?dip in kD dip" by (simp (metis kD_quality_increases))
    moreover from ⟨dip = oip⟩ rtnot qinc ⟨dip ∈ kD(rt (σ i))⟩ notdip *
    have "?nsqn le nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)
    ultimately show "?thesis ..
    qed
  qed

note update2 = this

have "opaodv i | (??S, ??U →) onl Γ_{AODV} (λ(σ, _)."
  "∀ dip ∈ kD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
   → dip ∈ kD(rt (σ (the (nhop (rt (σ i)) dip))))
   ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"

by (inv_cterms inv add: oseq_stepInvariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf
  oaddpreRT_welldefined]
oaddpreRT_welldefined]
onl_oivariant_sterms [OF aodv_wf odests_vD_inc_sqn]
onl_oivariant_sterms [OF aodv_wf oprerr_guard]
onl_oivariant_sterms [OF aodv_wf rreq_sip]
onl_oivariant_sterms [OF aodv_wf rrep_sip]
onl_oivariant_sterms [OF aodv_wf rerr_sip]
oaddpreRT_welldefined]
oquality_increases
other_localD

solve: basic basic_prerr
simp add: seqlsimp nsqn_invalidate nhop_update_sip
simp del: One_nat_def

(rule conjI, erule(2) update1, erule(2) update2)"

thus "?thesis unfolding Let_def by auto
  qed

Proposition 7.30

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aodv,
simplified seql_onl_swap]

71
lemmas ozero_seq_unk_hops_one =
open_seq_invariant [OF zero_seq_unk_hops_one initiali_aodv,
simplified seq1_unk_swap]

lemma orachachable_fresh_okD_unk_or_atleast_one:
  fixes dip
  assumes "(σ, p) ∈ orachable (opaodv i)
            (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
                                  ∧ msg_zhops m)))
      (other quality_increases {i})"
  and "dip ∈ kD (rt (σ i))"
  shows "π3 (the (rt (σ i) dip)) = unk ∨ 1 ≤ π2 (the (rt (σ i) dip))"
  (is "?P dip")
proof -
  have "∃ l. l ∈ labels Γ AODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaodv_trans aodv_trans]],
          auto dest!: onlD otherwith_actionD simp: seqlsimp)
  with ⟨dip ∈ kD (rt (σ i))⟩ show ?thesis by simp
qed

lemma orachachable_fresh_ozero_seq_unk_hops_one:
  fixes dip
  assumes "(σ, p) ∈ orachable (opaodv i)
            (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
                                  ∧ msg_zhops m)))
      (other quality_increases {i})"
  and "dip ∈ kD (rt (σ i))"
  shows "sqn (rt (σ i)) dip = 0 −→
         sqnf (rt (σ i)) dip = unk
         ∧ the (dhops (rt (σ i)) dip) = 1
         ∧ the (nhop (rt (σ i)) dip) = dip"
  (is "?P dip")
proof -
  have "∃ l. l ∈ labels Γ AODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]],
          auto dest!: onlD otherwith_actionD simp: seqlsimp)
  with ⟨dip ∈ kD (rt (σ i))⟩ show ?thesis by simp
qed

lemma seq_nhlop_quality_increases':
  shows "opaodv i \models (otherwith ((=)) {i})
          (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
          other quality_increases {i} →)
      onl Γ AODV (λσ, _. ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
              in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip))
              ∧ nhip ≠ dip
              −→ (rt (σ i) \sqsubset dip (rt (σ nhip))))"
  (is "\models (?S i, _ →) _")
proof -
  have weaken:
  "∀ l Q R P. P \models (otherwith quality_increases I (orecvmsg Q),
                  other quality_increases I →) P
                  → P \models (otherwith ((=)) I (orecvmsg (λσ m. Q σ m ∧ R σ m)),
                  other quality_increases I →) P"
  by auto
  { fix i a and σ σ' :: "ip ⇒ state"
    assume ai: "∀ dip. dip∈vD(rt (σ i))
                    ∧ dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))
                    ∧ (the (nhop (rt (σ i)) dip)) ≠ dip
                    −→ rt (σ i) \sqsubset dip rt (σ (the (nhop (rt (σ i)) dip)))"
    and ov: "?S i σ σ' a"
    have "∀ dip. dip∈vD(rt (σ i))
                  ∧ dip∈vD (rt (σ' (the (nhop (rt (σ i)) dip))))"
proof clarify

fix dip

assume a2: "dip \in VD(rt (\sigma i))"
and a3: "dip \in VD (rt (\sigma' (the (nhop (rt (\sigma i)) dip))))"
and a4: "(the (nhop (rt (\sigma i)) dip)) \neq dip"

from ow have "\forall j. j \neq i \rightarrow \sigma j = \sigma' j" by auto

show "rt (\sigma i) \sqsubseteq dip rt (\sigma' (the (nhop (rt (\sigma i)) dip)))"

proof (cases "(the (nhop (rt (\sigma i)) dip)) = i")

assume "(the (nhop (rt (\sigma i)) dip)) = i"

with \sigma, \sigma' \in VD(rt (\sigma i)) have "dip \in VD(rt (\sigma (the (nhop (rt (\sigma i)) dip))))" by simp

with a1 a2 a4 have "rt (\sigma i) \sqsubseteq dip rt (\sigma' (the (nhop (rt (\sigma i)) dip)))" by simp

with (the (nhop (rt (\sigma i)) dip)) = i have "rt (\sigma i) \sqsubseteq dip rt (\sigma i)" by simp

hence False by simp

thus \thesis ..

next

assume "(the (nhop (rt (\sigma i)) dip)) \neq i"

with \forall j. j \neq i \rightarrow \sigma j = \sigma' j

have *: "\sigma (the (nhop (rt (\sigma i)) dip)) = \sigma' (the (nhop (rt (\sigma i)) dip))" by simp

with \sigma, \sigma' \in VD (rt (\sigma' (the (nhop (rt (\sigma i)) dip)))) have "dip \in VD(rt (\sigma' (the (nhop (rt (\sigma i)) dip))))" by simp

with a1 a2 a4 have "rt (\sigma i) \sqsubseteq dip rt (\sigma' (the (nhop (rt (\sigma i)) dip)))" by simp

with * show \thesis by simp

qed

{

| fix \sigma \sigma' a dip sip i |
| assume a1: "\forall dip. dip \in VD(rt (\sigma i))
\land dip \in VD (rt (\sigma (the (nhop (rt (\sigma i)) dip))))
\land the (nhop (rt (\sigma i)) dip) \neq dip
\rightarrow rt (\sigma i) \sqsubseteq dip rt (\sigma (the (nhop (rt (\sigma i)) dip))))"
| and ow: "?S i \sigma \sigma' a"
|
| have "\forall dip. dip \in VD(update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {}))
\land dip \in VD (rt (\sigma' (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip))))
\land the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip) \neq dip
\rightarrow update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {}) 
\sqsubseteq dip rt (\sigma' (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip))))"
|

proof clarify

fix dip

assume a2: "dip \in VD (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {}))"
and a3: "dip \in VD (rt (\sigma' (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip))))"
and a4: "the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip) \neq dip"

show "update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {}) 
\sqsubseteq dip rt (\sigma' (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip))))"

proof (cases "dip = sip")

assume "dip = sip"

with (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip) \neq dip)
have False by simp

thus \thesis ..

next

assume [simp]: "dip \neq sip"

from a2 have "dip \in VD(rt (\sigma i)) \lor dip = sip"

by (rule vD_update_val)

with \sigma, \sigma' \in VD(rt (\sigma i)) have "dip \in VD(rt (\sigma i))" by simp

moreover from a3 have "dip \in VD (rt (\sigma' (the (nhop (rt (\sigma i)) dip))))" by simp

moreover from a4 have "the (nhop (rt (\sigma i)) dip) \neq dip" by simp

ultimately have "rt (\sigma i) \sqsubseteq dip rt (\sigma' (the (nhop (rt (\sigma i)) dip)))"

using a1 ow by - (drule(1) basic, simp)

with \sigma, \sigma' \in VD (rt (\sigma i)) have "dip \neq sip" show \thesis

by - (erule rt_strictly_fresher_update_other, simp)

qed

73
\[
\text{proof clarify}
\]
\[
\text{fix dip}
\]
\[
\text{assume } \forall \text{ dip}. \text{ dip} \in vD (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i)))
\]
\[
\wedge \text{ dip} \in vD (\text{rt} (\sigma (\text{nhop dip})))
\]
\[
\wedge \text{ nhop dip} \neq \text{ dip}
\]
\[
\rightarrow \text{ rt} (\sigma i) \sqsubseteq \text{ dip} \text{ rt} (\sigma (\text{nhop dip}))
\]
\[
\text{and ow: } "?S \ a \ \sigma' a"
\]
\[
\text{have } \forall \text{ dip}. \text{ dip} \in vD (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i)))
\]
\[
\wedge \text{ dip} \in vD (\text{rt} (\sigma' (\text{nhop dip})))
\]
\[
\wedge \text{ nhop dip} \neq \text{ dip}
\]
\[
\rightarrow \text{ rt} (\sigma i) \sqsubseteq \text{ dip} \text{ rt} (\sigma' (\text{nhop dip}))
\]
\[
\text{qed}
\]
\[
\text{note invalidate = this}
\]
from a2 have "dip ∈ vD (rt (σ i))" by simp
moreover from a3 have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))"
  using nochange and (∃ j. σ j = σ' j) by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) ⊑ dip rt (σ (the (nhop (rt (σ i)) dip)))"
  using pre by simp

hence "rt (σ i) ⊑ dip rt (σ' (the (nhop (rt (σ i)) dip)))"
  using (∃ j. σ j = σ' j) by simp
thus "?thesis" by simp

next
  assume change: "?rt1 ≠ rt (σ i)"
  from after a2 have "dip ∈ kD (rt (σ sip))" by auto
show ?thesis
proof (cases "dip = oip")
  assume "dip ≠ oip"
  with a2 have "dip ∈ vD (rt (σ i))" by auto
  moreover with a3 a5 after and ⟨dip ≠ oip⟩ have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))"
    by simp metis
  moreover from a4 and ⟨dip ≠ oip⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
    using pre by simp
  with after and a5 and ⟨dip ≠ oip⟩ show ?thesis
    by simp (metis rt_strictly_fresher_update_other rt_strictly_fresher_irefl)
next
  assume "dip = oip"
  with a4 and change have "sip ≠ oip" by simp
  with a6 have "oip ∈ kD (rt (σ sip))"
    and "osn ≤ nsqn (rt (σ sip)) oip" by auto
  from a3 change ⟨dip = oip⟩ have "oip ∈ vD (rt (σ sip))" by simp
  hence "the (flag (rt (σ sip)) oip) = val" by simp
  from ⟨oip ∈ kD (rt (σ sip))⟩ have "osn < nsqn (rt (σ sip)) oip ∨ (osn = nsqn (rt (σ sip)) oip"
    ∧ the (dhops (rt (σ sip)) oip) ≤ hops)"
  proof
    assume "oip ∈ vD (rt (σ sip))"
    hence "the (flag (rt (σ sip)) oip) = val" by simp
  with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops"
    by simp
  show ?thesis
proof (cases "sip = i")
  assume "sip ≠ i"
  with a5 have "σ sip = σ' sip" by simp
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
    and ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  show ?thesis by auto
next
  — alternative to using sip_not_ip
  assume [simp]: "sip = i"
  have "?rt1 = rt (σ i)"
    proof (rule update_cases_kD, simp_all)
      from Suc 0 ≤ osn: show "0 < osn" by simp
    next
      from ⟨oip ∈ kD (rt (σ sip))⟩ and ⟨sip = i⟩ show "oip ∈ kD (rt (σ i))"
        by simp
    next

next

75
assume "sqn (rt (σ i)) oip < osn"
also from οsn ≤ nsqn (rt (σ sip)) oip
  have ". . . ≤ nsqn (rt (σ i)) oip" by simp
also have "... ≤ sqn (rt (σ i)) oip"
  by (rule nsqn_sqn)
finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip" .
hence False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i, π_7 (the (rt (σ i) oip)))
  else rt (σ i) a) = rt (σ i)" ..

next
assume "sqn (rt (σ i)) oip = osn"
  and "Suc hops < the (dhops (rt (σ i)) oip)"
from this(1) and oip ∈ νD (rt (σ sip)) have "nsqn (rt (σ i)) oip = osn" by simp
with nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops
  have "the (dhops (rt (σ i)) oip) ≤ hops" by simp
with \Suc hops < the (dhops (rt (σ i)) oip)\ have False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i, π_7 (the (rt (σ i) oip)))
  else rt (σ i) a) = rt (σ i)" ..

next
assume "the (flag (rt (σ i)) oip) = inv"
with (the (flag (rt (σ sip)) oip) = val) have False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i, π_7 (the (rt (σ i) oip)))
  else rt (σ i) a) = rt (σ i)" ..

next
from oip∈kD(rt (σ sip))
  show "(λa. if a = oip then Some (the (rt (σ i) oip)) else rt (σ i) a) = rt (σ i)"
    by (auto dest!: kD_Some)
  with change have False ..
thus ?thesis ..

qed

next
assume "oip∈iD(rt (σ sip))"
with (the (flag (rt (σ’ sip)) oip) = val) and a5 have "sip = i"
    by (metis f.distinct(1) iD_flag_is_inv)
from oip∈iD(rt (σ sip)) have "the (flag (rt (σ sip)) oip) = inv" by auto
with (sip = i ; Suc 0 ≤ osn) change after oip∈kD(rt (σ sip))
  have "nsqn (rt (σ sip)) oip < nsqn (rt (σ’ sip)) oip"
    unfolding update_def
    by (clarsimp split: option.split_asm if_split_asm)
  (auto simp: sqn_def)
  with (osn ≤ nsqn (rt (σ sip)) oip) have "osn < nsqn (rt (σ’ sip)) oip"
    by simp
  thus ?thesis ..
qed

thus ?thesis ..

proof
assume osnlt: "osn < nsqn (rt (σ’ sip)) oip"
from dip∈kD(rt (σ’ i)); and dip = oip have "dip ∈ kD (?rt1)" by simp
moreover from a3 have "dip ∈ kD(?rt2 dip)" by simp
moreover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"
  proof
    have "nsqn ?rt1 oip = osn"
      by (simp add: dip = oip; nsqn_update_changed_kno_val [OF change [THEN not_sym]])
    also have "... < nsqn (rt (σ’ sip)) oip" using osnlt .
    also have "... = nsqn (?rt2 oip) oip" by (simp add: change)
    finally show ?thesis
      using dip = oip by simp
  qed
ultimately show ?thesis
by (rule rt_strictly_fresher_ltI)

next

assume osneq: "osn = nsqn (rt (σ' sip)) oip ∧ the (dhops (rt (σ' sip)) oip) ≤ hops"

have "oip∈kD(?rt1)" by simp

moreover have from a3 ⟨dip = oip⟩ have "oip∈kD(?rt2 oip)" by simp

moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"

proof -
  from osneq have "osn = nsqn (rt (σ' sip)) oip" ..
  also have "osn = nsqn ?rt1 oip"
    by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "nsqn (rt (σ' sip)) oip = nsqn (?rt2 oip) oip"
    by (simp add: change)
  finally show ?thesis .

qed

moreover have "π₅(the (?rt2 oip oip)) < π₅(the (?rt1 oip))"

proof -
  from osneq have "the (dhops (rt (σ' sip)) oip) ≤ hops" ..
  moreover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip∈kD(rt (σ' sip))" by auto
  ultimately have "π₅(the (rt (σ' sip) oip)) ≤ hops"
    by (auto simp add: proj5_eq_dhops)
  also from change after have "hops < π₅(the (rt (σ' i) oip))"
    by (metis dhops_update_changed lessI)
  finally have "π₅(the (rt (σ' sip) oip)) < π₅(the (rt (σ' i) oip))" .
  with change after show ?thesis by simp

qed

ultimately have "?rt1 ⊏ oip ?rt2 oip"

by (rule rt_strictly_fresher_eqI)

with ⟨dip = oip⟩ show ?thesis by simp

qed

qed

note rreq_rrep_update = this

have "opaodv i ⊨ (otherwith ((=)) {i} (orecmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)), other quality_increases {i} →)
  onl Γ_{AODV}
  (λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip))))"

proof (inv_cterms inv add: onl_oinvariant_sterms [OF aodv_wf rreq_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aodv_wf rerr_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]]
  onl_oinvariant_sterms [OF aodv_wf odsn_rrep [THEN weaken]]
  onl_oinvariant_sterms [OF aodv_wf oaddpreRT_welldefined]
  solve: basic update_0_unk invalidate rreq_rrep_update
t
  simp add: seqlsimp)

fix σ σ' p l

assume or: "(σ, p) ∈ oreachable (opaodv i) (?S i) (other quality_increases {i})"
  and "other quality_increases {i} σ σ'"
  and ll: "l ∈ labels Γ_{AODV} p"

and pre: "∀dip. dip∈vD (rt (σ i))
  ∧ dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))
  ∧ the (nhop (rt (σ i)) dip) ≠ dip
  → rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip)))"

from this(1-2)

have or': "(σ', p) ∈ oreachable (opaodv i) (?S i) (other quality_increases {i})"
  by - (rule oreachable_other')
from or and ll have next_hop: "∀ dip. let nhip = the (nhop (rt (σ i)) dip) 
    in dip ∈ kD(rt (σ i)) \(\land\) nhip ≠ dip 
    ⇒ dip ∈ kD(rt (σ nhip))  
    \(\land\) nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip"
by (auto dest!: onl_oinvariant_weakenD [OF seq_compare_next_hop'])

from or and ll have unk_hops_one: "∀ dip ∈ kD (rt (σ i)). sqn (rt (σ i)) dip = 0 
    ⇒ sqnf (rt (σ i)) dip = unk  
    \(\land\) the (dhops (rt (σ i)) dip) = 1  
    \(\land\) the (nhop (rt (σ i)) dip) = dip"
by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]
otherwith_actionD simp: seqlsimp)

from ⟨other quality_increases {i} σ σ'⟩ have "σ' i = σ i" by auto
hence "quality_increases (σ i) (σ' i)" by auto
with ⟨other quality_increases {i} σ σ'⟩ have "∀ j. quality_increases (σ j) (σ' j)"
by -(erule otherE, metis singleton_iff)
show "∀ dip. dip ∈ vD (rt (σ' i)) 
    \(\land\) dip ∈ vD (rt (σ' (the (nhop (rt (σ' i)) dip))))  
    \(\land\) the (nhop (rt (σ' i)) dip) ≠ dip  
    ⇒ rt (σ' i) ⊏ dip rt (σ' (the (nhop (rt (σ' i)) dip)))"
proof clarify
fix dip
assume "dip ∈ vD(rt (σ' i))"
and "dip ∈ vD (rt (σ' (the (nhop (rt (σ' i)) dip))))"  
and "the (nhop (rt (σ' i)) dip) ≠ dip"
from this(1) and ⟨σ' i = σ i⟩ have "dip ∈ vD (rt (σ i))"
and "dip ∈ kD (rt (σ i))"
by auto
from ⟨the (nhop (rt (σ' i)) dip) ≠ dip⟩ and ⟨σ' i = σ i⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" (is "?nhip ≠ _") by simp
with ⟨dip ∈ kD (rt (σ i))⟩ and next_hop have "dip ∈ kD (rt (σ ?nhip))"
and nsqns: "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
by (auto simp: Let_def)
have "0 < sqn (rt (σ i)) dip"
proof (rule neq0_conv [THEN iffD1, OF notI])
assume "sqn (rt (σ i)) dip = 0"
with ⟨dip ∈ kD (rt (σ i))⟩ and unk_hops_one have "?nhip = dip" by simp
with ⟨?nhip ≠ dip⟩ show False ..
qed
also have "... = nsqn (rt (σ i)) dip"
by (rule vD nsqn_sqn [OF ⟨dip ∈ vD (rt (σ i))⟩, THEN sym])
also have "... ≤ sqn (rt (σ ?nhip)) dip"
by (rule nsqns)
also have "... ≤ sqn (rt (σ ?nhip)) dip"
by (rule nsqn_sqn)
finally have "0 < sqn (rt (σ ?nhip)) dip".

have "rt (σ i) ⊏ dip rt (σ' ?nhip)"
proof (cases "dip ∈ vD(rt (σ ?nhip))")
assume "dip ∈ vD (rt (σ ?nhip))"
with ⟨dip ∈ vD (rt (σ i))⟩ and ⟨σ' i = σ i⟩ have "rt (σ i) ⊏ dip rt (σ' ?nhip)" by auto
moreover from "∀ j. quality_increases (σ j) (σ' j)" have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
ultimately show ?thesis
using \(\{dip \in kD(rt (\sigma \ nhip))\}\) 
by (rule strictly_fresher_quality_increases_right)

next 
assume "\(dip \in D(rt (\sigma \ nhip))\)"
with \(\{dip \in kD(rt (\sigma \ nhip))\}\) have "\(dip \in D(rt (\sigma \ nhip))\)" ..

hence "the (flag (rt (\sigma \ nhip)) dip) = inv"
by auto

have "nsqn (rt (\sigma \ i)) dip \leq nsqn (rt (\sigma \ nhip)) dip"
by (rule nsqns)

also from \(\{dip \in kD(rt (\sigma \ nhip))\}\) 

have "\(\ldots = sqn (rt (\sigma \ nhip)) dip - 1\)" ..
also have "\(\ldots < sqn (rt (\sigma \ nhip)) dip\)"
proof
- 
  from \(\forall j. \text{quality increases} (\sigma j) (\sigma j')\)

    have "\(\text{quality increases} (\sigma \ nhip) (\sigma \ nhip')\)" ..
    hence "\(\forall ip. \text{sqn} (rt (\sigma \ nhip)) ip \leq \text{sqn} (rt (\sigma \ nhip)) ip\)" by auto
    hence "\(\text{sqn} (rt (\sigma \ nhip)) dip \leq \text{sqn} (rt (\sigma \ nhip)) dip\)" ..
    with \(0 < \text{sqn} (rt (\sigma \ nhip)) dip\) show ?thesis by auto

qed

also have "\(\ldots = nsqn (rt (\sigma \ nhip)) dip\)"
proof (rule vD_nsqn_sqn [THEN sym])

from \(\{dip \in kD(rt (\sigma \ (\text{the (nhop (rt (\sigma \ i)) dip)})))\}\) and \(\sigma \ i = \sigma \ i\)

show "\(dip \in kD(rt (\sigma \ nhip))\)" by simp

qed

finally have "\(\text{nsqn} (rt (\sigma \ i)) dip < \text{nsqn} (rt (\sigma \ nhip)) dip\)" .

moreover from \(\{dip \in D(rt (\sigma \ (\text{the (nhop (rt (\sigma \ i)) dip)})))\}\) and \(\sigma \ i = \sigma \ i\)

have "\(dip \in D(rt (\sigma \ nhip))\)" by auto
ultimately show "\(rt (\sigma \ i) \sqsubseteq dip rt (\sigma \ nhip)\)"
using \(\{dip \in kD(rt (\sigma \ i))\}\) by - (rule rt_strictly_fresher_ltI)

with \(\sigma \ i = \sigma \ i\) show "\(rt (\sigma \ i) \sqsubseteq dip rt (\sigma \ (\text{the (nhop (rt (\sigma \ i)) dip)}))\)"
  by simp

qed

qed

thus ?thesis unfolding Let_def .

lemma seq_nhopt_quality_increases:
shows "\(\text{opaodv} \ i \supseteq \\text{otherwith} ((\sigma)) \{i\}\)
  \(\text{orecvmsg (\sigma m. \text{msg_fresh} \sigma m \wedge \text{msg_zhops} m)},\)
  \(\text{other quality increases} (i) \rightarrow\)
  \(\text{global (\lambda \sigma. \forall dip. let nhip = the (nhop (rt (\sigma \ i)) dip)}\)

      in dip \in VD (rt (\sigma \ i)) \cap VD (rt (\sigma \ nhip)) \wedge nhip \neq dip

      \rightarrow \langle rt (\sigma \ i) \sqsubseteq dip rt (\sigma \ nhip)\rangle)"

by (rule oinvariant_weakenE [OF seq_nhopt_quality_increases']) (auto dest!: onlD)

end

0.11 Routing graphs and loop freedom

theory Loop_Freedom
imports Aodv_Predicates Fresher
begin

Define the central theorem that relates an invariant over network states to the absence of loops in the associate
routing graph.

definition
  \textbf{rt_graph} :: 
  "\(\langle \text{ip} \Rightarrow \text{state} \rangle \Rightarrow \text{ip} \Rightarrow \text{ip rel}\)"
where
  \textbf{rt_graph} \(\sigma = (\lambda dip.\)

  \{(ip, ip') | ip \neq dip \wedge rt (\sigma ip) dip = Some (dsn, dsk, val, hops, ip', pre)\)"
Given the state of a network \( \sigma \), a routing graph for a given destination ip address dip abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

**lemma rt_graphE [elim]:**

fixes n dip ip ip'

assumes "\((ip, ip') \in rt_graph \sigma dip\)"

shows "ip \(\neq\) dip \(\wedge\) (\(\exists\) r. rt \((\sigma ip) = r\) 
\(\wedge\) (\(\exists\) den dsk hops pre. r dip = Some (dsn, dsk, val, hops, ip', pre)))"

using asms unfolding rt_graph_def by auto

**lemma rt_graph_vD [dest]:**

\[\forall ip ip' \sigma dip. (ip, ip') \in rt_graph \sigma dip \implies dip \in vD(rt \((\sigma ip)\))\]

unfolding rt_graph_def vD_def by auto

**lemma rt_graph_vD_trans [dest]:**

\[\forall ip ip' \sigma dip. (ip, ip') \in (rt_graph \sigma dip)^+ \implies dip \in vD(rt \((\sigma ip)\))\]

by (erule converse_tranclE) auto

**lemma rt_graph_not_dip [dest]:**

\[\forall ip ip' \sigma dip. (ip, ip') \in rt_graph \sigma dip \implies ip \neq dip\]

unfolding rt_graph_def by auto

**lemma rt_graph_not_dip_trans [dest]:**

\[\forall ip ip' \sigma dip. (ip, ip') \in (rt_graph \sigma dip)^+ \implies ip \neq dip\]

by (erule converse_tranclE) auto

NB: the property below cannot be lifted to the transitive closure

**lemma rt_graph_nhip_is_nhop [dest]:**

\[\forall ip ip' \sigma dip. (ip, ip') \in rt_graph \sigma dip \implies ip' = the (nhop (rt \((\sigma ip)) dip)\]

unfolding rt_graph_def by auto

**theorem inv_to_loop_freedom:**

assumes 
\[\forall i dip. let nhip = the (nhop (rt \((\sigma i)) dip) \in dip \in vD(rt \((\sigma i)) \cap vD(rt \((\sigma nhip)) \land nhip \neq dip \implies (rt \((\sigma i)) \sqsubseteq dip rt \((\sigma nhip))\)

shows "\(\forall dip. irrefl ((rt_graph \sigma dip)^+)\)"

using asms proof (intro allI)

fix \(\sigma::\) "ip \(\Rightarrow\) state" and dip

assume inv: "\(\forall ip dip. \)

let nhip = the (nhop (rt \((\sigma ip)) dip) \in dip \in vD(rt \((\sigma ip)) \cap vD(rt \((\sigma nhip)) \land nhip \neq dip \implies rt \((\sigma ip) \sqsubseteq dip rt \((\sigma nhip))\)"

{ fix ip ip'

assume "\((ip, ip') \in (rt_graph \sigma dip)^+\)"

and "dip \in vD(rt \((\sigma ip'))\)"

and "ip' \(\neq\) dip"

hence "rt \((\sigma ip) \sqsubseteq dip rt \((\sigma ip'))\)"

proof induction

fix nhip

assume "\((ip, nhip) \in rt_graph \sigma dip\)"

and "dip \in vD(rt \((\sigma nhip))"

and "nhip \(\neq dip)"

from \((ip, nhip) \in rt_graph \sigma dip) have "dip \in vD(rt \((\sigma ip))"

and "nhip = the (nhop (rt \((\sigma ip)) dip)"

by auto

from \(dip \in vD(rt \((\sigma ip))\) and \(dip \in vD(rt \((\sigma nhip))\)

have "dip \in vD(rt \((\sigma ip)) \cap vD(rt \((\sigma nhip))\) ..

with nhip = the (nhop (rt \((\sigma ip)) dip)\)

and nhip \(\neq dip)"

and inv

show "rt \((\sigma ip) \sqsubseteq dip rt \((\sigma nhip))"

by (clarsimp simp: Let_def)

next

fix nhip nhip'

assume "\((ip, nhip) \in (rt_graph \sigma dip)^+\)"
and "(nhip, nhip') ∈ rt_graph σ dip"
and IH: "[ dip ∈ vD(rt (σ nhip)); nhip ≠ dip ] ⊢ rt (σ ip) ⊑ dip rt (σ nhip)"
and "dip ∈ vD(rt (σ nhip'))"
and "nhip' ≠ dip"
from ⟨(nhip, nhip') ∈ rt_graph σ dip⟩ have 1: "dip ∈ vD(rt (σ nhip))"
and 2: "nhip ≠ dip"
and "nhip' = the (nhop (rt (σ nhip)) dip)"
by auto
from 1 2 have "rt (σ ip) ⊑ dip rt (σ nhip)"
by (rule IH)
also have "rt (σ nhip) ⊑ dip rt (σ nhip')" proof -
from ⟨dip ∈ vD(rt (σ nhip))⟩ and ⟨dip ∈ vD(rt (σ nhip'))⟩
have "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
with ⟨nhip' ≠ dip⟩ and ⟨nhip' = the (nhop (rt (σ nhip)) dip)⟩
and inv
show "rt (σ nhip) ⊑ dip rt (σ nhip')"
by (clarsimp simp: Let_def)
qed
finally show "rt (σ ip) ⊑ dip rt (σ nhip')".
qed 
} note fresher = this
show "irrefl ((rt_graph σ dip)⁺)"
unfolding irrefl_def proof (intro allI notI)
fix ip
assume "(ip, ip) ∈ (rt_graph σ dip)⁺" moreover then have "dip ∈ vD(rt (σ ip))"
and "ip ≠ dip"
by auto
ultimately have "rt (σ ip) ⊑ dip rt (σ ip)" by (rule fresher)
thus False by simp
qed
qed
end

0.12 Lift and transfer invariants to show loop freedom

theory Aodv_Loop_Freedom
imports AODV_OClosed_Transfer AODV_Qmsg_Lifting Global_Invariants Loop_Freedom begin

0.12.1 Lift to parallel processes with queues

lemma par_step_no_change_on_send_or_receive:
  fixes σ s a σ' s'
  assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG)
  and "a ≠ τ"
  shows "σ' i = σ i"
using assms by (rule qmsg_no_change_on_send_or_receive)

lemma par_nhop_quality_increases:
  shows "opaodv i (⟨i qmsg |= (otherwith ((=)) {i} (orecmsg (λσ m.
    msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →
  ) (orecmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)))
  ) (seqp_sos Γ_QMSG)" proof (rule lift_into_qmsg [OF seq_nhop_quality_increases])
  show "opaodv i (⟨i qmsg |= (otherwith ((=)) {i} (orecmsg (λσ m.
    msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →
  ) (orecmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)))
  ) (seqp_sos Γ_QMSG)" proof (rule ostep_invariant_weakenE [OF oquality_increases], simp_all)
  qed
  qed
end
lemma \textbf{par\_rreq\_rrep\_sn\_quality\_increases:} \\
\textbf{par\_rreq\_rrep\_nsq\_f\_\_\_fresh\_any\_step:} \\
\textbf{lemma \textbf{par\_anycast\_msg\_zhops:}} \\
\textbf{proof -} \\
\textbf{next} \\
\textbf{fix } \sigma \sigma' \ a \\
\textbf{assume "otherwith } \{\i\} \ \{(\lambda a. \text{msg\_fresh } a) m \wedge \text{msg\_zhops } m\} \ \sigma' \ a" \\
\textbf{thus "otherwith } \{\i\} \ \{(\lambda a. \text{rreq\_rrep\_sn})\} \ \sigma' \ a" \\
\textbf{by } \text{(erule weaken\_otherwith, auto)} \\
\textbf{qed auto} \\
\textbf{lemma \textbf{par\_rreq\_rrep\_sn\_quality\_increases:}} \\
"\text{opaodv } i \ \{(\i \ \text{qmsg } \Rightarrow \lambda\sigma. \ \text{orecvmsg } (\lambda. \ \text{rreq\_rrep\_sn}) \ \sigma, \ \text{other } (\lambda. \ \text{True}) \ \{\i\} \ \rightarrow \ \text{globala } (\lambda(\sigma, \ _, \ \sigma'). \ \text{quality\_increases } (\sigma \ i) (\sigma' \ i))" \\
\textbf{proof -} \\
\textbf{have "opaodv } i \ \Rightarrow \ (\lambda a. \ \text{orecvmsg } (\lambda. \ \text{rreq\_rrep\_sn}) \ \sigma, \ \text{other } (\lambda. \ \text{True}) \ \{\i\} \ \rightarrow \ \text{globala } (\lambda(\sigma, a, \sigma'). \ \text{anycast } (\text{msg\_fresh } a) a)" \\
\textbf{proof } \text{(rule ostep\_invariant\_weakenE \[OF \text{local\_quality\_increases}\])} \\
\textbf{fix } t \\
\textbf{assume "onll } \Gamma_{\text{AO\_D}} \ (\lambda((\sigma, \ _, \ a, \ _). \ \text{anycast } (\text{msg\_fresh } a) a) \ t" \\
\textbf{thus "globala } (\lambda(\sigma, a, \sigma'). \ \text{anycast } (\text{msg\_fresh } a) a) \ t" \\
\textbf{by } \text{(cases } t \text{) } \text{(clarsimp dest!: olocal\_quality\_increases \[OF olocal\_quality\_increases\])} \\
\textbf{qed auto} \\
\textbf{lemma \textbf{par\_rreq\_rrep\_nsq\_f\_\_\_fresh\_any\_step:}} \\
\textbf{shows "opaodv } i \ \{(\i \ \text{qmsg } \Rightarrow \lambda\sigma. \ \text{orecvmsg } (\lambda. \ \text{rreq\_rrep\_sn}) \ \sigma, \ \text{other } (\lambda. \ \text{True}) \ \{\i\} \ \rightarrow \ \text{globala } (\lambda(\sigma, a, \ underscores \ a). \ \text{anycast } (\text{msg\_fresh } a) a)" \\
\textbf{proof -} \\
\textbf{have "opaodv } i \ \Rightarrow \ (\lambda a. \ \text{orecvmsg } (\lambda. \ \text{rreq\_rrep\_sn}) \ \sigma, \ \text{other } (\lambda. \ \text{True}) \ \{\i\} \ \rightarrow \ \text{globala } (\lambda(\sigma, a, \sigma'). \ \text{anycast } (\text{msg\_fresh } a) a)" \\
\textbf{proof } \text{(rule ostep\_invariant\_weakenE \[OF ostep\_invariant\_weakenE \text{any\_step\_invariant}\])} \\
\textbf{fix } t \\
\textbf{assume "onll } \Gamma_{\text{AO\_D}} \ (\lambda(\_, \ a, \ _). \ \text{anycast } (\text{msg\_fresh } a) a) \ t" \\
\textbf{thus "globala } (\lambda(\_, a, \sigma'). \ \text{anycast } (\text{msg\_fresh } a) a) \ t" \\
\textbf{by } \text{(cases } t \text{) } \text{(clarsimp dest!: olocal\_quality\_increases \[OF olocal\_quality\_increases\])} \\
\textbf{qed auto} \\
\textbf{lemma \textbf{par\_anycast\_msg\_zhops:}} \\
\textbf{shows "opaodv } i \ \{(\i \ \text{qmsg } \Rightarrow \lambda\sigma. \ \text{orecvmsg } (\lambda. \ \text{rreq\_rrep\_sn}) \ \sigma, \ \text{other } (\lambda. \ \text{True}) \ \{\i\} \ \rightarrow \ \text{globala } (\lambda(\_, a, \ _). \ \text{anycast } \text{msg\_zhops } a)" \\
\textbf{proof -} \\
\textbf{from anycast\_msg\_zhops } \text{initial\_ao\_d } \text{ao\_d\_trans } \text{ao\_d\_trans} \\
\textbf{have "opaodv } i \ \Rightarrow \ (\text{act } \text{TT}, \ \text{other } (\lambda. \ \text{True}) \ \{\i\} \ \rightarrow \ \text{seq\_l\_l } (\text{onll } \Gamma_{\text{AO\_D}} \ (\lambda(\_, a, \ _). \ \text{anycast } \text{msg\_zhops } a))" \\
\textbf{by } \text{(rule open\_seq\_step\_invariant)} \\
\textbf{hence "opaodv } i \ \Rightarrow \ (\lambda a. \ \text{orecvmsg } (\lambda. \ \text{rreq\_rrep\_sn}) \ \sigma, \ \text{other } (\lambda. \ \text{True}) \ \{\i\} \ \rightarrow \ \text{globala } (\lambda(\_, a, \ _). \ \text{anycast } \text{msg\_zhops } a)" \\
\textbf{proof } \text{(rule ostep\_invariant\_weakenE \[OF ostep\_invariant\_weakenE \text{any\_step\_invariant}\])} \\
\textbf{fix } t :: "((\text{nat } \Rightarrow \text{state}) \times (\text{state, msg, pseq\_p, pseq\_label, seq\_p}), \ \text{msg\_seq\_action}) \ \text{transition}" \\
\textbf{assume "seq\_l\_l } (\text{onll } \Gamma_{\text{AO\_D}} \ (\lambda(\_, a, \ _). \ \text{anycast } \text{msg\_zhops } a)) \ t" \\
\textbf{thus "globala } (\lambda(\_, a, \ _). \ \text{anycast } \text{msg\_zhops } a) \ t" \\
\textbf{by } \text{(cases } t \text{) } \text{(clarsimp dest!: seq\_l\_l \text{onll\_l} \ \text{metis } \text{ao\_d\_ex\_label})} \\
\textbf{qed simp\_all} \\
\textbf{hence "opaodv } i \ \{(\i \ \text{qmsg } \Rightarrow \lambda\sigma. \ \text{orecvmsg } (\lambda. \ \text{rreq\_rrep\_sn}) \ \sigma, \ \text{other } (\lambda. \ \text{True}) \ \{\i\} \ \rightarrow \ \text{globala } (\lambda(\_, a, \ _). \ \text{anycast } \text{msg\_zhops } a)"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

0.12.2 Lift to nodes

lemma node_step_no_change_on_send_or_receive:
assumes "((σ, NodeS i P R), a, (σ', NodeS i' P' R')) ∈ onode_sos
(oparp_sos i (oseqp_sos Γ \text{AODV} i) (seqp_sos Γ \text{QMSG}))"
and "a ≠ \tau"
shows "σ' i = σ i"
using assms
by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

lemma node_nhop_quality_increases:
shows "⟨ i : opaodv i ⟨⟨ i qmsg : R ⟩⟩o ∣=
(otherwith ((=)) {i})
(oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
other_quality_increases {i} ⟩ \rightarrow\ global (λσ. ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
→ (rt (σ i)) \square_{dip} (rt (σ nhip)))"
by (rule node_lift [OF par_nhop_quality_increases]) auto

lemma node_quality_increases:
shows "⟨ i : opaodv i ⟨⟨ i qmsg : R ⟩⟩o ∣=
λσ_. oarrivemsg (λ_._ \text{rrreq_rrep_sn} σ),
other (λ_._ True) {i} ⟩ \rightarrow\ globala (λ(σ, a, σ'). quality_increases (σ i) (σ' i))"
by (rule node_lift_step_statelessassm [OF par_rrreq_rrep_sn_quality_increases]) simp

lemma node_rreq_rrep_nsqn_fresh_any_step:
shows "⟨ i : opaodv i ⟨⟨ i qmsg : R ⟩⟩o ∣=
λσ_. oarrivemsg (λ_._ \text{rrreq_rrep_sn} σ),
other (λ_._ True) {i} ⟩ \rightarrow\ globala a (λ(_, a, _). castmsg (msg_fresh σ) a)"
by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step])

lemma node_anycast_msg_zhops:
shows "⟨ i : opaodv i ⟨⟨ i qmsg : R ⟩⟩o ∣=
λσ_. oarrivemsg (λ_._ \text{rrreq_rrep_sn} σ),
other (λ_._ True) {i} ⟩ \rightarrow\ globala (λ(_._ a, _)._ castmsg msg_zhops a)"
by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

lemma node_silent_change_only:
shows "⟨ i : opaodv i ⟨⟨ i qmsg : R_i ⟩⟩o ∣=
λσ_. oarrivemsg (λ_._ True) σ,
other (λ_._ True) {i} ⟩ \rightarrow\ globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i)"
proof (rule ostep_invariantI, simp (no_asm), rule impl)
fix σ ζ a σ' ζ'
assume or: "⟨σ, ζ⟩ ∈ oreachable ⟨⟨ i : opaodv i ⟨⟨ i qmsg : R_i ⟩⟩o ⟩⟩
(λσ_. oarrivemsg (λ_._ True) σ)
(other (λ_._ True) {i})"
and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans ⟨⟨ i : opaodv i ⟨⟨ i qmsg : R_i ⟩⟩o ⟩⟩"
and "a ≠ τ_n"
from or obtain P R where "ζ = NodeS i p R"
by (drule node_net_state, metis)
with tr have "((σ, NodeS i p R), a, (σ', ζ')) ∈ onode_sos (oparp_sos i (trans (opaodv i)) (trans qmsg))"
by simp
thus "σ' i = σ i" using a ≠ τ_n
by (cases rule: onode_sos.cases)
(auto elim: qmsg_no_change_on_send_or_receive)
qed
0.12.3 Lift to partial networks

lemma arrive_rreq_rrep_nsqn_fresh_inc_sn [simp]:
  assumes "oarrivemsg (λσ m. msg_fresh σ m ∧ P σ m) σ m"
  shows "oarrivemsg (λ_. rreq_rrep_sn) σ m"
  using assms by (cases m) auto

lemma opnet_nhop_quality_increases:
  shows "opnet (λi. opaodv i ⟨⟨i qmsg : R⟩⟩ p) |=
    (otherwith ((=)) (net_tree_ips p))
    (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other_quality_increases (net_tree_ips p) →
    global (λσ. ∀i∈net_tree_ips p. ∀dip. let nhip = the (nhop (rt (σ i))) dip
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊆ dip (rt (σ nhip)))"

proof (rule pnet_lift [OF node_nhop_quality_increases])
  fix i R
  have ""(i : opaodv i ⟨⟨i qmsg : R⟩⟩ o =_A (λσ_. oarrivemsg (λ_. rreq_rrep_sn) σ),
    other (λ_. True) {i} →) globala (λ(σ, a, σ').
    castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
  proof (rule ostep_invariantI, simp (no_asm))
    fix σ s a σ' s'
    assume or: "((σ, s) ∈ oreachable (i : opaodv i ⟨⟨i qmsg : R⟩⟩ o)
      (λσ_. oarrivemsg (λ_. rreq_rrep_sn) σ)
      (other (λ_. True) {i})"
    and tr: "((σ, s), a, (σ', s')) ∈ trans (i : opaodv i ⟨⟨i qmsg : R⟩⟩ o)"
    and am: "oarrivemsg (λ_. rreq_rrep_sn) σ a"
    from or tr am have "castmsg (msg_fresh σ) a" by (auto dest!: ostep_invariantD [OF node_rreq_rrep_nsqn_fresh_any_step])
    moreover from or tr am have "castmsg (msg_zhops m) a" by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
    ultimately show "castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a" by (case_tac a) auto
  qed

thus ""(i : opaodv i ⟨⟨i qmsg : R⟩⟩ o =_A (λσ_. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
  other_quality_increases {i} →) globala (λ(σ, a, _).
  castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
  by rule auto
next
  fix i R
  show ""(i : opaodv i ⟨⟨i qmsg : R⟩⟩ o =_A (λσ_. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
    other_quality_increases {i} →) globala (λ(σ, a, σ').
    a ≠ τ ∧ (∀d. a ≠ i:deliver(d)) → σ i = σ' i)"
  by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto
next
  fix i R
  show ""(i : opaodv i ⟨⟨i qmsg : R⟩⟩ o =_A (λσ_. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
    other_quality_increases {i} →) globala (λ(σ, a, σ').
    a = τ ∨ (∃d. a = i:deliver(d)) → quality_increases (σ i) (σ' i))"
  by (rule ostep_invariant_weakenE [OF node_quality_increases]) auto
qed simp_all

0.12.4 Lift to closed networks

lemma onet_nhop_quality_increases:
  shows "onet (open (onet (λi. opaodv i ⟨⟨i qmsg⟩⟩ p))
    |= (λ_. True, other_quality_increases (net_tree_ips p) →)
    global (λσ. ∀i∈net_tree_ips p. ∀dip. let nhip = the (nhop (rt (σ i))) dip
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊆ dip (rt (σ nhip)))"
is "_ |= (_ , ?U ⇒ ) ?inv"

proof (rule inclosed_closed)
from opnet_nhop_quality_increases
show "opnet (λi. opaodv i ⟨⟨i qmsg⟩ p | = (otherwith (=)) (net_tree_ips p) inclosed, ?U ⇒ ) ?inv"
proof (rule oinvariant_weakenE)
fix σ σ' :: "ip ⇒ state" and a :: "msg node_action"
assume "otherwith (=) (net_tree_ips p) inoclosed σ σ' a"
thus "otherwith (=) (net_tree_ips p) (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"
proof (rule oinvariant_weakenE)
fix σ :: "ip ⇒ state" and a :: "msg node_action"
assume "inoclosed σ a"
thus "oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ a"
proof (cases a)
fix ii ni ms
assume "a = ii ¬ ni:arrive(ms)"
moreover with ⟨inoclosed σ a⟩ obtain d di where "ms = newpkt(d, di)"
ultimately show ?thesis by simp
qed simp_all
qed

0.12.5 Transfer into the standard model
interpretation aodv_openproc: openproc paodv opaodv id
rewrites "aodv_openproc.initmissing = initmissing"
proof -
show "openproc paodv opaodv id"
proof unfold_locales
fix i :: ip
have "{(σ, ζ). (σ i, ζ) ∈ σ_AODV i ∧ (∀j. j ≠ i ⇒ σ j ∈ fst ' σ_AODV j)} ⊆ σ_AODV'" unfolding σ_AODV_def σ_AODV'_def
proof (rule equalityD1)
show "∀p. {(σ, ζ). (σ i, ζ) ∈ {(f i, p)} ∧ (∀j. j ≠ i ⇒ σ j ∈ fst ' {(f j, p)})} = {(f, p)}" by simp
qed
thus "{ (σ, ζ) | σ ζ s. s ∈ init (paodv i) ∧ (σ i, ζ) = id s ∧ (∀j. j ≠ i ⇒ σ j ∈ (fst o id) ' init (paodv j)) } ⊆ init (opaodv i)" by simp
next
show "∀j. init (paodv j) ≠ {}"
unfolding σ_AODV_def by simp
next
fix i s a s' σ σ'
assume "σ i = fst (id s)"
and "σ' i = fst (id s')"
and "(s, a, s') ∈ trans (paodv i)"
then obtain q q' where "s = (σ i, q)"
and "s' = (σ' i, q')"
and "((σ i, q), a, (σ' i, q')) ∈ trans (paodv i)"
by (cases s, cases s')
from this(3) have "((σ, q), a, (σ', q')) ∈ trans (opaodv i)"
by simp (rule open_seqp_action [OF aodv_wf])
with ⟨s = (σ i, q)⟩ and ⟨s' = (σ' i, q')⟩
show "((σ, snd (id s)), a, (σ', snd (id s')))) ∈ trans (opaodv i)"
by simp
qed
then interpret opn: openproc paodv opaodv id .
interpretation aodv_openproc_par_qmsg: openproc_parq paodv opaodv id qmsg
rewrites "aodv_openproc_par_qmsg.netglobal = netglobal"
and "aodv_openproc_par_qmsg.initmissing = initmissing"
proof -
show "openproc_parq paodv opaodv id qmsg"
by (unfold_locales) simp
then interpret opq: openproc_parq paodv opaodv id qmsg.

have im: "∀σ. openproc.initmissing (λi. paodv i qmsg) (λ(p, q). (fst (id p), snd (id p), q)) σ = initmissing σ"
unfolding opq.initmissing_def opq.someinit_def initmissing_def
unfolding σ AODV_def σ QM SG_def by (clarsimp cong: option.case_cong)
thus "openproc.initmissing (λi. paodv i qmsg) (λ(p, q). (fst (id p), snd (id p), q)) = initmissing" by (rule ext)

have nm: "∀P σ. openproc.netglobal (λi. paodv i qmsg) (λ(p, q). (fst (id p), snd (id p), q)) P σ = netglobal P σ"
unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
unfolding σ AODV_def σ QM SG_def by (clarsimp cong: option.case_cong simp del: One_nat_def simp add: fst_initmissing_netgmap_default_aodv_init_netlift [symmetric, unfolded initmissing_def])
thus "openproc.netglobal (λi. paodv i qmsg) (λ(p, q). (fst (id p), snd (id p), q)) = netglobal" by auto
qed
hence "(fst (initmissing (netgmap fst σ))) i = aodv_init i"
  by simp
thus ?thesis by simp
qed metis
qed

0.12.6 Loop freedom of AODV

theorem aodv_loop_freedom:
  assumes "wf_net_tree n"
  shows "closed (pnet (λi. paodv i (⟨⟨ qmsg⟩⟩ n) ||= netglobal (λσ. ∀dip. irrefl ((rt_graph σ dip)⁺)))"
  using assms by (rule aodv_openproc_par_qmsg.netglobal_weakenE
  [OF net_nhop_quality_increases inv_to_loop_freedom])
end
Chapter 1

Variant A: Skipping the RREQ ID

Explanation [4, §10.1]: AODV does not need the route request identifier. This number, in combination with the IP address of the originator, is used to identify every RREQ message in a unique way. This variant shows that the combination of the originator’s IP address and its sequence number is just as suited to uniquely determine the route request to which the message belongs. Hence, the route request identifier field is not required. This can then reduce the size of the RREQ message.

1.1 Predicates and functions used in the AODV model

theory A_Aodv_Data
imports A_Norreqid
begin

1.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

definition inc :: "sqn ⇒ sqn"
  where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ inc x"
  unfolding inc_def by simp

lemma inc_minus_suc_0 [simp]: "inc x - Suc 0 = x"
  unfolding inc_def by simp

lemma inc_never_one' [simp, intro]: "inc x ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
  by simp

1.1.2 Modelling Routes

A route is a 6-tuple, \((dsn, dsk, flag, hops, nhip, pre)\) where \(dsn\) is the ‘destination sequence number’, \(dsk\) is the ‘destination-sequence-number status’, \(flag\) is the route status, \(hops\) is the number of hops to the destination, \(nhip\) is the next hop toward the destination, and \(pre\) is the set of ‘precursor nodes’—those interested in hearing about changes to the route.

type_synonym r = "sqn × k × f × nat × ip × ip set"

definition proj2 :: "r ⇒ sqn" ("\pi_2")
  where "\pi_2 ≡ λ(dsn, _, _, _, _, _). dsn"

definition proj3 :: "r ⇒ k" ("\pi_3")
  where "\pi_3 ≡ λ(_, dsk, _, _, _, _). dsk"

definition proj4 :: "r ⇒ f" ("\pi_4")
where \[ \pi_4 \equiv \lambda(_, _, flag, _, _). flag \]

definition proj5 :: "r ⇒ nat" ("\(\pi_5\)"
 where \[ \pi_5 \equiv \lambda(_, _, hops, _, _). hops \]

definition proj6 :: "r ⇒ ip" ("\(\pi_6\)"
 where \[ \pi_6 \equiv \lambda(_, _, _, nhip, _). nhip \]

definition proj7 :: "r ⇒ ip set" ("\(\pi_7\)"
 where \[ \pi_7 \equiv \lambda(_, _, _, _, pre). pre \]

lemma projs [simp]:
\[
\begin{align*}
\pi_2&(dsn, dsk, flag, hops, nhip, pre) = dsn \\
\pi_3&(dsn, dsk, flag, hops, nhip, pre) = dsk \\
\pi_4&(dsn, dsk, flag, hops, nhip, pre) = flag \\
\pi_5&(dsn, dsk, flag, hops, nhip, pre) = hops \\
\pi_6&(dsn, dsk, flag, hops, nhip, pre) = nhip \\
\pi_7&(dsn, dsk, flag, hops, nhip, pre) = pre
\end{align*}
\]
by (clarsimp simp: proj2_def proj3_def proj4_def
 proj5_def proj6_def proj7_def)+

lemma proj3_pred [intro]: "[ P kno; P unk ] ⇒ P (\(\pi_3\) x)"
by (rule k.induct)

lemma proj4_pred [intro]: "[ P val; P inv ] ⇒ P (\(\pi_4\) x)"
by (rule f.induct)

lemma proj6_pair_snd [simp]:
fixes dsn' r
shows "\(\pi_6\) (dsn', snd (r)) = \(\pi_6\) (r)"
by (cases r) simp

1.1.3 Routing Tables

Routing tables map ip addresses to route entries.

type_synonym rt = "ip ⇒ r"

syntax
"_Sigma_route" :: "rt ⇒ ip ⇒ r" ("\(\sigma_{route}\)(\_, \_, \_, \_)")

translations
"\(\sigma_{route}\)(rt, dip)" ⇒ "rt dip"

definition sqn :: "rt ⇒ ip ⇒ sqn"
where "sqn rt dip ≡ case \(\sigma_{route}\)(rt, dip) of Some r ⇒ \pi_2(r) | None ⇒ 0"

definition sqnf :: "rt ⇒ ip ⇒ k"
where "sqnf rt dip ≡ case \(\sigma_{route}\)(rt, dip) of Some r ⇒ \pi_3(r) | None ⇒ unk"

abbreviation flag :: "rt ⇒ ip ⇒ f"
where "flag rt dip ≡ map_option \pi_4 (\(\sigma_{route}\)(rt, dip))"

abbreviation dhops :: "rt ⇒ ip ⇒ nat"
where "dhops rt dip ≡ map_option \pi_5 (\(\sigma_{route}\)(rt, dip))"

abbreviation nhop :: "rt ⇒ ip ⇒ ip"
where "nhop rt dip ≡ map_option \pi_6 (\(\sigma_{route}\)(rt, dip))"

abbreviation precs :: "rt ⇒ ip ⇒ ip set"
where "precs rt dip ≡ map_option \pi_7 (\(\sigma_{route}\)(rt, dip))"

definition vD :: "rt ⇒ ip set"
where "vD rt ≡ {dip. flag rt dip = Some val}"
definition \( iD :: \text{"rt} \Rightarrow \text{ip set} \) where \( iD \ rt \equiv \{\text{dip. flag rt dip = Some inv}\} \)

definition \( kD :: \text{"rt} \Rightarrow \text{ip set} \) where \( kD \ rt \equiv \{\text{dip. rt dip \neq None}\} \)

lemma \( \text{kD_is_vD_and_iD: kD rt = vD rt \cup iD rt} \)
unfolding kD_def vD_def iD_def by auto

lemma \( \text{vD_iD_gives_kD \[simp\]:} \)
\( \\forall ip \ rt. ip \in vD rt \Rightarrow ip \in kD rt \)
\( \\forall ip \ rt. ip \in iD rt \Rightarrow ip \in kD rt \)
unfolding kD_is_vD_and_iD by simp_all

lemma \( \text{kD_Some \[dest\]:} \)
fixes dip rt
assumes \( \text{\"dip \in kD rt\"} \)
shows \( \exists dsn dsk flag hops nhip pre. \sigma\text{route}(rt, dip) = \text{Some (dsn, dsk, flag, hops, nhip, pre)}\)" using assms unfolding kD_def by simp

lemma \( \text{kD_None \[dest\]:} \)
fixes dip rt
assumes \( \text{\"dip \notin kD rt\"} \)
shows \( \sigma\text{route}(rt, dip) = \text{None} \)
using (metis (mono_tags) mem_Collect_eq)

lemma \( \text{vD_Some \[dest\]:} \)
fixes dip rt
assumes \( \text{\"dip \in vD rt\"} \)
shows \( \exists dsn dsk hops nhip pre. \sigma\text{route}(rt, dip) = \text{Some (dsn, dsk, val, hops, nhip, pre)}\)" using assms unfolding vD_def by simp

lemma \( \text{vD_empty \[simp\]:} \) vD Map.empty = {}" unfolding vD_def by simp

lemma \( \text{iD_Some \[dest\]:} \)
fixes dip rt
assumes \( \text{\"dip \in iD rt\"} \)
shows \( \exists dsn dsk hops nhip pre. \sigma\text{route}(rt, dip) = \text{Some (dsn, dsk, inv, hops, nhip, pre)}\)" using assms unfolding iD_def by simp

lemma \( \text{val_is_vD \[elim\]:} \)
fixes ip rt
assumes \( \text{\"ip \in kD(rt)\"} \)
and \( \text{\"the (flag rt ip) = val\"} \)
shows \( \text{ip \in vD(rt)} \)
using assms unfolding vD_def by auto

lemma \( \text{inv_is_iD \[elim\]:} \)
fixes ip rt
assumes \( \text{\"ip \in kD(rt)\"} \)
and \( \text{\"the (flag rt ip) = inv\"} \)
shows \( \text{ip \in iD(rt)} \)
using assms unfolding iD_def by auto

lemma \( \text{iD_flag_is_inv \[elim, simp\]:} \)
fixes ip rt
assumes \( \text{\"ip \in iD(rt)\"} \)
shows \( \text{\"the (flag rt ip) = inv\"} \)
proof -
from \( \{ \text{ip} \in \text{ID}(\text{rt}) \} \) have "\( \text{ip} \in \text{ID}(\text{rt}) \)" by auto
with assms show ?thesis unfolding id_def by auto
qed

lemma kD_but_not_vD_is_iD \[elim\]:
  fixes ip rt
  assumes "\( \text{ip} \in \text{ID}(\text{rt}) \)"
  shows "\( \text{ip} \in \text{ID}(\text{rt}) \)"
proof -
  from \( \{ \text{ip} \in \text{ID}(\text{rt}) \} \) obtain dsn dsk f hops nhop pre
    where rtip: "\( \text{rt} \; \text{ip} \) = \text{Some} \; (dsn, dsk, f, hops, nhop, pre)" by (metis kD_Some)
  from \( \{ \text{ip} \notin \text{vD}(\text{rt}) \} \) have "\( f \neq \text{val} \)"
proof (rule contrapos_nn)
    assume "\( f = \text{val} \)"
    with rtip have "\( \text{the } (\text{flag } \text{rt} \; \text{ip}) = \text{val} \)" by simp
    with \( \{ \text{ip} \in \text{ID}(\text{rt}) \} \) show "\( \text{ip} \in \text{vD}(\text{rt}) \)" ..
  qed
  with rtip have "\( \text{the } (\text{flag } \text{rt} \; \text{ip}) = \text{inv} \)"
  with \( \{ \text{ip} \in \text{ID}(\text{rt}) \} \) show "\( \text{ip} \in \text{ID}(\text{rt}) \)" ..
  qed

lemma vD_or_iD \[elim\]:
  fixes ip rt
  assumes "\( \text{ip} \in \text{ID}(\text{rt}) \)"
  and "\( \text{ip} \in \text{vD}(\text{rt}) \)"
  shows "\( \text{P } \text{rt} \; \text{ip} \)"
proof -
  from \( \{ \text{ip} \in \text{ID}(\text{rt}) \} \) have "\( \text{ip} \in \text{vD}(\text{rt}) \cup \text{ID}(\text{rt}) \)"
  by (simp add: kD_is_vD_and_ID)
  thus ?thesis by (auto elim: assms(2-3))
  qed

lemma proj5_eq_dhops: "\( \forall \text{dip } \text{rt}. \text{dip} \in \text{ID}(\text{rt}) \implies \pi_5(\text{the } (\text{rt} \; \text{dip})) = \text{the } (\text{dhops } \text{rt} \; \text{dip}) \)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "\( \forall \text{dip } \text{rt}. \text{dip} \in \text{ID}(\text{rt}) \implies \pi_4(\text{the } (\text{rt} \; \text{dip})) = \text{the } (\text{flag } \text{rt} \; \text{dip}) \)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "\( \forall \text{dip } \text{rt}. \text{dip} \in \text{ID}(\text{rt}) \implies \pi_2(\text{the } (\text{rt} \; \text{dip})) = \text{sqn } \text{rt} \; \text{dip} \)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 \[simp\]:
  "\( \forall \text{ip } \text{rt}. \text{ip} \in \text{ID}(\text{rt}) \implies \text{sqnf } \text{rt} \; \text{ip} = \pi_3(\text{the } (\text{rt} \; \text{ip})) \)"
  unfolding sqnf_def by auto

lemma vD_flag_val \[simp\]:
  "\( \forall \text{dip } \text{rt}. \text{dip} \in \text{vD} (\text{rt}) \implies \text{the } (\text{flag } \text{rt} \; \text{dip}) = \text{val} \)"
  unfolding vD_def by clarsimp

lemma kD_update \[simp\]:
  "\( \forall \text{rt } \text{nip } \text{v}. \text{kD } (\text{rt}(\text{nip} \mapsto \text{v})) = \text{insert } \text{nip } (\text{kD } \text{rt}) \)"
  unfolding kD_def by auto

lemma kD_empty \[simp\]: "\( \text{kD } \text{Map.empty} = \{\} \)"
  unfolding kD_def by simp

lemma ip_equal_or_known \[elim\]:
  fixes rt ip ip'
  assumes "\( \text{ip} = \text{ip}' \vee \text{ip} \in \text{kD}(\text{rt}) \)"
  and "\( \text{ip} = \text{ip}' \implies \text{P } \text{rt} \; \text{ip} \; \text{ip}' \)"
  and "\( \lceil \text{ip} \neq \text{ip}' ; \text{ip} \in \text{kD}(\text{rt}) \rceil \implies \text{P } \text{rt} \; \text{ip} \; \text{ip}' \)"
1.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

Updating Precursor Lists

definition addpre :: "r ⇒ ip set ⇒ r"
  where "addpre r npre ≡ let \(dsn, dsk, flag, hops, nhip, pre\) = r in
  \(dsn, dsk, flag, hops, nhip, pre \cup npre\)"

lemma proj2_addpre:
  fixes v pre
  shows "\(\pi_2(addpre v pre) = \pi_2(v)\)"
  unfolding addpre_def by (cases v) simp

lemma proj3_addpre:
  fixes v pre
  shows "\(\pi_3(addpre v pre) = \pi_3(v)\)"
  unfolding addpre_def by (cases v) simp

lemma proj4_addpre:
  fixes v pre
  shows "\(\pi_4(addpre v pre) = \pi_4(v)\)"
  unfolding addpre_def by (cases v) simp

lemma proj5_addpre:
  fixes v pre
  shows "\(\pi_5(addpre v pre) = \pi_5(v)\)"
  unfolding addpre_def by (cases v) simp

lemma proj6_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "\(\pi_6(addpre v npre) = \pi_6(v)\)"
  unfolding addpre_def by (cases v) simp

lemma proj7_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "\(\pi_7(addpre v npre) = \pi_7(v) \cup npre\)"
  unfolding addpre_def by (cases v) simp

lemma addpre_empty: "addpre r {} = r"
  unfolding addpre_def by simp

lemma addpre_r:
  "addpre (dsn, dsk, f1, hops, nhip, pre) npre = (dsn, dsk, f1, hops, nhip, pre \cup npre)"
  unfolding addpre_def by simp

lemmas addpre_simps [simp] = proj2_addpre proj3_addpre proj4_addpre proj5_addpre proj6_addpre proj7_addpre addpre_empty addpre_r

definition addpreRT :: "rt ⇒ ip set ⇒ rt"
  where "addpreRT rt dip npre ≡
  \(\sigma_{route}(rt, dip)\)"

lemma snd_addpre [simp]:
  "\(dsn, dsk\) v pre. (dsn, snd(addpre (dsn', v) pre)) = addpre (dsn, v) pre"
  unfolding addpre_def by clarsimp

lemma proj2_addpreRT [simp]:
  fixes ip rt ip' npre
assumes "ip \in kD rt" and "ip' \in kD rt"
shows "\pi_2((the (addpreRT rt ip' npre) ip)) = \pi_2((the (rt ip)))"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj3_addpreRT [simp]:
  fixes ip rt ip' npre
assumes "ip \in kD rt" and "ip' \in kD rt"
shows "\pi_3((the (addpreRT rt ip' npre) ip)) = \pi_3((the (rt ip)))"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj5_addpreRT [simp]:
"\forall rt dip ip npre. dip \in kD(rt) \implies \pi_5((the (addpreRT rt dip npre) ip)) = \pi_5((the (rt ip)))"
unfolding addpreRT_def by auto

lemma flag_addpreRT [simp]:
  fixes rt pre ip dip
assumes "dip \in kD rt"
shows "flag((the (addpreRT rt dip pre)) ip) = flag(rt ip)"
unfolding addpreRT_def by (clarsimp)

lemma kD_addpreRT [simp]:
  fixes rt dip npre
assumes "dip \in kD rt"
shows "kD((the (addpreRT rt dip npre)) ip) = kD(rt)"
unfolding kD_def addpreRT_def by clarsimp auto

lemma vD_addpreRT [simp]:
  fixes rt dip npre
assumes "dip \in kD rt"
shows "vD((the (addpreRT rt dip npre)) ip) = vD(rt)"
unfolding vD_def addpreRT_def by clarsimp auto

lemma iD_addpreRT [simp]:
  fixes rt dip npre
assumes "dip \in kD rt"
shows "iD((the (addpreRT rt dip npre)) ip) = iD(rt)"
unfolding iD_def addpreRT_def by clarsimp auto

lemma nhop_addpreRT [simp]:
  fixes rt pre ip dip
assumes "dip \in kD rt"
shows "nhop((the (addpreRT rt dip pre)) ip) = nhop(rt ip)"
unfolding sqn_def addpreRT_def by clarsimp

lemma sqn_addpreRT [simp]:
  fixes rt pre ip dip
assumes "dip \in kD rt"
shows "sqn((the (addpreRT rt dip pre)) ip) = sqn(rt ip)"
unfolding sqn_def addpreRT_def by clarsimp

lemma dhops_addpreRT [simp]:
  fixes rt pre ip dip
assumes "dip \in kD rt"
shows "dhops((the (addpreRT rt dip pre)) ip) = dhops(rt ip)"
unfolding addpreRT_def
lemma sqnf_addpreRT [simp]:
"∀ ip dip. ip ∈ kD(rt ξ) ⇒ sqnf (the (addpreRT (rt ξ) ip npre)) dip = sqnf (rt ξ) dip"
unfolding sqnf_def addpreRT_def by auto

Updating route entries

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = es (the (rt dip))"
  using assms [THEN kD_Some] by auto

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip /∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = en"
  using assms [THEN kD_None] by auto

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip pre
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip /∈ kD(rt)"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def by simp

definition update_arg_wf :: "r ⇒ bool"
where "update_arg_wf r ≡ π 4 (r) = val ∧
       (π 2 (r) = 0) = (π 3 (r) = unk) ∧
       (π 3 (r) = unk −→ π 5 (r) = 1)"

lemma update_arg_wf_gives_cases:
  "∀ r. update_arg_wf r =⇒ (π 2 (r) = 0) = (π 3 (r) = unk)"
unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples [simp]:
  "∀ nhip pre. update_arg_wf (0, unk, val, Suc 0, nhip, pre)"
  "∀ n hops nhip pre. update_arg_wf (Suc n, kno, val, hops, nhip, pre)"
unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples' [elim]:
  "∀ n hops nhip pre. Suc 0 ≤ n ⇒ update_arg_wf (n, kno, val, hops, nhip, pre)"
unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r" and cl1: "∀ nhip pre. P (0, unk, val, Suc 0, nhip, pre)" and cl2: "∀ dsn hops nhip pre. dsn > 0 ⇒ P (dsn, kno, val, hops, nhip, pre)"
  shows "P r"
proof -
  obtain dsn dsk flag hops nhip pre
    where *: "r = (dsn, dsk, flag, hops, nhip, pre)" by (cases r)
  with (update_arg_wf r) have wf1: "flag = val" and wf2: "dsn = 0 = (dsk = unk)" and wf3: "dsk = unk −→ (hops = 1)"
  unfolding update_arg_wf_def by auto

94
have "P (dsn, dsk, flag, hops, nhip, pre)"

proof (cases dsk)
  assume "dsn = unk"
  moreover with wf2 wf3 have "dsn = 0" and "hops = Suc 0" by auto
  ultimately show ?thesis using (flag = val) by simp (rule c1)
next
  assume "dsk = kno"
  moreover with wf2 have "dsn > 0" by simp
  ultimately show ?thesis using (flag = val) by simp (rule c2)
qed

with * show "P r" by simp

qed

definition update :: "rt ⇒ ip ⇒ r ⇒ rt"
where
  "update rt ip r ≡ case σ route (rt, ip) of
    None ⇒ rt (ip ↦ r)
    Some s ⇒
    if π2(s) < π2(r) then rt (ip ↦ addpre r (π7(s)))
    else if π2(s) = π2(r) ∧ (π5(s) > π5(r) ∨ π3(s) = inv)
    then rt (ip ↦ addpre r (π7(s)))
    else if π3(r) = unk
    then rt (ip ↦ (π2(s), snd (addpre r (π7(s)))))
    else rt (ip ↦ addpre s (π7(r)))"

lemma update_simps [simp]:
  fixes r s nr nr' ns rt ip
  defines "s ≡ the σ route (rt, ip)"
  and "nr ≡ addpre r (π7(s))"
  and "nr' ≡ (π2(s), π3(nr), π4(nr), π5(nr), π6(nr), π7(nr))"
  and "ns ≡ addpre s (π7(r))"
  shows
  "[ip /∈ kD(rt)] ⇒ update rt ip r = rt (ip ↦ r)"
  "[ip ∈ kD(rt); sqn rt ip < π2(r)] ⇒ update rt ip r = rt (ip ↦ nr)"
  "[ip ∈ kD(rt); sqn rt ip = π2(r);
    the (dhops rt ip) > π5(r)] ⇒ update rt ip r = rt (ip ↦ nr)"
  "[ip ∈ kD(rt); sqn rt ip = π2(r);
    flag rt ip = Some inv] ⇒ update rt ip r = rt (ip ↦ nr)"
  "[ip ∈ kD(rt); π3(r) = unk; (π2(r) ≠ 0) = (π3(r) = unk)] ⇒ update rt ip r = rt (ip ↦ nr')"
  "[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
    sqn rt ip = π2(r)] ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val ]⇒ update rt ip r = rt (ip ↦ ns)"

proof -
  assume "ip /∈ kD(rt)"
  hence "σ route (rt, ip) = None" ..
  thus "update rt ip r = rt (ip ↦ r)"
  unfolding update_def by simp
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip < π2(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with (sqn rt ip < π2(r)) show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip = π2(r)"
  and "the (dhops rt ip) > π5(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with (sqn rt ip = π2(r)) and (the (dhops rt ip) > π5(r))
show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr_def s_def by auto
next
assume "ip ∈ kD(rt)"
  and "sqn rt ip = π₂(r)"
  and "flag rt ip = Some inv"
from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
with ⟨sqn rt ip = π₂(r)⟩ and ⟨flag rt ip = Some inv⟩
  show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr_def s_def by auto
next
assume "ip ∈ kD(rt)"
  and "π₃(r) = unk"
  and "(π₂(r) = 0) = (π₃(r) = unk)"
from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
with ⟨(π₂(r) = 0) = (π₃(r) = unk)⟩ and ⟨π₃(r) = unk⟩
  show "update rt ip r = rt (ip ↦ nr')"
  unfolding update_def nr'_def nr_def s_def by (cases r) simp
next
assume "ip ∈ kD(rt)"
  and otherassms: "sqn rt ip ≥ π₂(r)"
  "π₃(r) = kno"
  "sqn rt ip = π₂(r) =⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val"
from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
with otherassms show "update rt ip r = rt (ip ↦ ns)"
  unfolding update_def ns_def s_def by auto
qed

definition update_cases [elim]:
  assumes c1: "(π₂(r) = 0) = (π₃(r) = unk)"
  and c2: "[\[ \begin{array}{l}
  ip \notin kD(rt) \\wedge \\ \\ \sqn rt ip < π₂(r) \\
  \quad \Rightarrow P \ (rt \ (ip \mapsto r))
  \end{array} \]"
  and c3: "[\[ \begin{array}{l}
  \quad \Rightarrow P \ (rt \ (ip \mapsto \text{addpre} \ (\sigma_{\\text{route}(rt, ip)))))
  \end{array} \]"
  and c4: "[\[ \begin{array}{l}
  \quad \Rightarrow P \ (rt \ (ip \mapsto \text{addpre} \ (\sigma_{\\text{route}(rt, ip})))))
  \end{array} \]"
  and c5: "[\[ \begin{array}{l}
  \quad \Rightarrow P \ (rt \ (ip \mapsto (\sigma_{\\text{route}(rt, ip)}), π₃(r),
      π₄(r), π₅(r), π₆(r), π₇(\text{addpre} \ \pi₇(\sigma_{\\text{route}(rt, ip)))))))
  \end{array} \]"
  and c6: "[\[ \begin{array}{l}
  \quad \Rightarrow P \ (rt \ (ip \mapsto \text{addpre} \ (\sigma_{\\text{route}(rt, ip)}) (π₇(\pi₇(\pi₇(\pi₇))))))
  \end{array} \]"
  shows "(P (update rt ip r))"
proof (cases "ip ∈ kD(rt)")
  assume "ip ∈ kD(rt)"
  with c1 show thesis
  by simp
next
assume "ip ∈ kD(rt)"
moreover then obtain dsn dsk fl hops nhip pre
  where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
moreover obtain dsn' dsk' fl' hops' nhip' pre'
  where req: "r = (dsn', dsk', fl', hops', nhip', pre')"
  by (cases r) metis
ultimately show ?thesis
using ((π2(r) = 0) = (π3(r) = unk));
c2 [OF ip∈kD(rt);]
c3 [OF ip∈kD(rt);]
c4 [OF ip∈kD(rt);]
c5 [OF ip∈kD(rt);]
c6 [OF ip∈kD(rt);]

unfolding update_cases_kD by auto
qed

lemma update_cases_def sqn_def by auto
shows "(π2(r) = 0) = (π3(r) = unk)"
and "ip ∈ kD(rt)"
and c2: "sqn rt ip < π2(r) → P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
and c3: "[sqn rt ip = π2(r); the (dhops rt ip) > π5(r)] → P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
and c4: "[sqn rt ip = π2(r); the (flag rt ip) = inv] → P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
and c5: "π3(r) = unk → P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r), π4(r), π5(r), π6(r),
π7(addpre r (π7(the σroute(rt, ip)))))" #1

shows "(P (update rt ip r))"
using assms(1) proof (rule update_cases)
assume "sqn rt ip < π2(r)"
thus "P (rt (ip ↦ addpre r (π7(the (rt ip)))))" by (rule c2)
next
assume "sqn rt ip = π2(r)"
and "the (dhops rt ip) > π5(r)"
thus "P (rt (ip ↦ addpre r (π7(the (rt ip)))))" by (rule c3)
next
assume "sqn rt ip = π2(r)"
and "the (flag rt ip) = inv"
thus "P (rt (ip ↦ addpre r (π7(the (rt ip)))))" by (rule c4)
next
assume "π3(r) = unk"
thus "P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r), π4(r), π5(r), π6(r),
π7(addpre r (π7(the (rt ip)))))" by (rule c5)
next
assume "sqn rt ip ≥ π2(r)"
and "π3(r) = kno"
and "sqn rt ip = π2(r) → the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
thus "P (rt (ip ↦ addpre (the (rt ip)) (π7(r))))" by (rule c6)
qed (simp add: #:ip ∈ kD(rt));

lemma in_kD_after_update [simp]:
fixes rt nip dsn dsk flag hops nhip pre
shows "kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) = insert nip (kD rt)"
unfolding update_def
by (cases "rt nip") auto

lemma nhop_of_update [simp]:
fixes rt dip dsn dsk flag hops nhip
assumes "rt ≠ update rt dip (dsn, dsk, flag, hops, nhip, {})"
shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip, {})) dip) = nhip"
proof -
  from assms
  have update_neq: "∀ v. rt dip = Some v →→
update rt dip (dsn, dsk, flag, hops, nhip, {}) 
  ≠ rt(dip ↦ addpre (the (rt dip)) (π₂ (dsn, dsk, flag, hops, nhip, {})))

by auto

show ?thesis

proof (cases "rt dip = None")
  assume "rt dip = None"
  thus "?thesis" unfolding update_def by clarsimp

next
  assume "rt dip ≠ None"
  then obtain v where "rt dip = Some v" by (metis not_None_eq)
  with update_neq [OF this] show ?thesis
    unfolding update_def by auto

qed

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn (λx. if x = rip then Some v else rt x) ip
            = (if ip = rip then π₂(v) else sqn rt ip)"

unfolding sqn_def by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip pre
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip, pre)) dip"

proof (rule update_cases)
  show "(π₂ (dsn, dsk, val, hops, nhip, pre) = 0) = (π₃ (dsn, dsk, val, hops, nhip, pre) = unk)"
    by simp (rule assms)

qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip pre
  assumes "1 ≤ hops"
  shows "sqn rt ip ≤ sqn (update rt ip' (dsn, dsk, flag, hops, nhip, pre)) ip"

using assms unfolding update_def sqn_def
by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops ip rip nhip pre
  assumes ex: "∀ip ∈ kD rt. the (dhops rt ip) ≥ 1"
    and ip: "(ip = rip ∧ Suc 0 ≤ hops) ∨ (ip ≠ rip ∧ ip ∊ kD rt)"
  shows "Suc 0 ≤ the (dhops (update rt rip (dsn, dsk, flag, hops, nhip, pre)) ip)"

using ip proof
  assume "ip = rip ∧ Suc 0 ≤ hops" thus ?thesis
  unfolding update_def using ex
  by (cases "rip ∈ kD rt") (drule(1) bspec, auto)

next
  assume "ip ≠ rip ∧ ip ∊ kD rt" thus ?thesis
  using ex unfolding update_def
  by (cases "rip ∊ kD rt") auto

qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = rt ip"

using assms unfolding update_def
by (clarsimp split: option.split)

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhhip, pre)) ip = nhop rt ip"

using assms unfolding update_def
by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dan dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  "∀ rt ip dsn dsk flag hops nhip pre. sqn (rt(ip ↦ v)) ip = π₂(v)"
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})")"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip, {})) dip) = hops"
  using assms unfolding update_def
  by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma nhop_update_unk_val [simp]:
  "∀ rt dip dsn hops npre.
  the (nhop (update rt dip (dsn, unk, val, hops, ip, npre)) dip) = ip"
  unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip, {} ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.splits if_split_asm) auto

lemma update_rt_split_asm:
  "∀ rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip, {})) =
  (¬ (rt = update rt ip (dsn, dsk, flag, hops, sip, {}) ∧ ¬P rt
    ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip, {})
    ∧ ¬P (update rt ip (dsn, dsk, flag, hops, sip, {}))))"
  by auto

lemma sqn_update [simp]: "∀ rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
  ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsn"
  unfolding update_def by (clarsimp split: option.split if_split_asm) auto

lemma sqnf_update [simp]: "∀ rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
  ⇒ sqnf (update rt dip (dsn, dsk, flg, hops, sip, {})) dip = dsk"
  unfolding sqnf_def
  by (clarsimp split: option.splits if_split_asm) auto

lemma update_kno_dsn_greater_zero:
  "∀ rt dip dsn hops npre. 1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip, npre)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∀ rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
  ⇒ π₃ (the (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits if_split_asm) auto

lemma nhop_update_changed_kno_val [simp]: "∀ rt ip dsn dsk hops nhip.

99
\ril  \rt \neq \text{update } \rt \text{ ip } (dsn, \text{kno}, \text{val}, \text{hops}, \text{nhip}, \{\}) \\
\quad \Rightarrow \text{the } \text{ (nhop } (\text{update } \rt \text{ ip } (dsn, \text{kno}, \text{val}, \text{hops}, \text{nhip}, \{\})) \text{ ip) } = \text{nhip}"
\unfolding \text{update_def} \\
\by (\text{clarsimp split: option.split_asm option.split if_split_asm}) \text{auto}

\lemma \text{flag_update [simp]: } "\rt \text{ dip dsn flg hops sip.} \\
\rt \neq \text{update } \rt \text{ dip } (dsn, \text{kno}, \text{flg}, \text{hops}, \text{sip}, \{\}) \\
\Rightarrow \text{the } \text{ (flag } (\text{update } \rt \text{ dip } (dsn, \text{kno}, \text{flg}, \text{hops}, \text{sip}, \{\})) \text{ dip) } = \text{flg}"
\unfolding \text{update_def} \\
\by (\text{clarsimp split: option.split if_split_asm}) \text{auto}

\lemma \text{the_flag_Some [dest!]: } \\
\fixes ip \rt \\
\assumes "the \text{ (flag } \rt \text{ ip) } = \text{x}" \\
\quad and \text{ } "ip \in kD \rt" \\
\shows "flag \text{ rt ip } = \text{Some x}"
\using \text{assms} \by \text{clarsimp split: option.split if_split_asm} \text{auto}

\lemma \text{kD_update_unchanged [dest]: } \\
\fixes rt dip dsn dsk flag hops nhip pre \\
\assumes "rt = \text{update } \rt \text{ dip } (dsn, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}, \text{pre})" \\
\shows "dip\in kD(\rt)"
\proof - \\
\have "dip\in kD(\text{update } \rt \text{ dip } (dsn, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}, \text{pre}))" \by simp \with \text{assms show ?thesis by simp}
\qed

\lemma \text{nhop_update [simp]: } "\rt \text{ dip dsn dsk flg hops sip.} \\
\rt \neq \text{update } \rt \text{ dip } (dsn, \text{dsk}, \text{flg}, \text{hops}, \text{sip}, \{\}) \\
\Rightarrow \text{the } \text{ (nhop } (\text{update } \rt \text{ dip } (dsn, \text{dsk}, \text{flg}, \text{hops}, \text{sip}, \{\})) \text{ dip) } = \text{sip}"
\unfolding \text{update_def sqnf_def} \\
\by (\text{clarsimp split: option.splits if_split_asm}) \text{auto}

\lemma \text{sqn_update_another [simp]: } \\
\fixes dip ip rt dsn dsk flag hops nhip pre \\
\assumes "ip \neq dip" \\
\shows "sqn \text{ (update rt dip } (dsn, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}, \text{pre})) ip = sqn \text{ rt ip}"
\using \text{assms unfolding update_def sqn_def} \\
\by (\text{clarsimp split: option.splits}) \text{auto}

\lemma \text{sqnf_update_another [simp]: } \\
\fixes dip ip rt dsn dsk flag hops nhip pre \\
\assumes "ip \neq dip" \\
\shows "sqnf \text{ (update rt dip } (dsn, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}, \text{pre})) ip = sqnf \text{ rt ip}"
\using \text{assms unfolding update_def sqnf_def} \\
\by (\text{clarsimp split: option.splits}) \text{auto}

\lemma \text{vD_update_val [dest]: } \\
"dip \text{ rt dip'} dsn dsk hops nhip pre. \\
\text{dip} \in \text{vD(} \text{update rt dip'} (dsn, \text{dsk}, \text{val}, \text{hops}, \text{nhip}, \text{pre})\text{) } \Rightarrow \text{ (dip} \in \text{vD(} \rt \text{) } \lor \text{ dip'=dip')}"
\unfolding \text{update_def vD_def} \by (\text{clarsimp split: option.split_asm if_split_asm}) \text{auto}

\textbf{Invalidating route entries}

definition \text{invalidate :: } "\rt \Rightarrow (ip \rightarrow sqn) \Rightarrow rt" \\
\where "\text{invalidate } \rt \text{ dests } \equiv \lambda \text{ dip. case (rt ip, dests ip) of} \\
(\text{None, } _) \Rightarrow \text{None} \\
| (\text{Some } s, \text{None}) \Rightarrow \text{Some } s \\
| (\text{Some } (_, \text{dsk, } (_, \text{hops}, \text{nhip}, \text{pre}), \text{Some rsn}) \Rightarrow \text{Some (rsn, dsk, inv, hops, nhip, pre)}"

\lemma \text{proj3_invalidate [simp]: } "\forall \text{ dip. } \pi_3(\text{the } (\text{invalidate rt dests} \text{ dip})) = \pi_3(\text{the (rt dip))}"
lemma proj5_invalidate [simp]:
  "\(\forall \text{dip}. \pi_5(\text{the } (\text{invalidate } \text{rt } \text{dests} \text{ dip})) = \pi_5(\text{the } (\text{rt } \text{dip}))\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
  "\(\forall \text{dip}. \pi_6(\text{the } (\text{invalidate } \text{rt } \text{dests} \text{ dip})) = \pi_6(\text{the } (\text{rt } \text{dip}))\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj7_invalidate [simp]:
  "\(\forall \text{dip}. \pi_7(\text{the } (\text{invalidate } \text{rt } \text{dests} \text{ dip})) = \pi_7(\text{the } (\text{rt } \text{dip}))\)"
unfolding invalidate_def by (clarsimp split: option.split)

1.1.5 Route Requests

lemma invalidate_kD_inv [simp]:
  "\(\forall \text{rt } \text{dests}. kD (\text{invalidate } \text{rt } \text{dests}) = kD \text{rt}\)"
unfolding invalidate_def kD_def
by (simp split: option.split)

lemma invalidate_sqn:
  fixes \(\text{rt } \text{dip } \text{dests}\)
  assumes \("\forall \text{rsn}. \text{dests dip } = \text{Some rsn }\rightarrow \text{sqn } \text{rt dip } \leq \text{rsn}\"
  shows \("\text{sqn } \text{rt dip } \leq \text{sqn } (\text{invalidate } \text{rt } \text{dests} \text{ dip})\"
proof (cases "\text{dip} \notin kD(\text{rt})")
  assume "\neg \text{dip} \notin kD(\text{rt})"
  hence "\text{dip} \in kD(\text{rt})" by simp
  then obtain \(\text{dsn } \text{dsk } \text{flag hops nhip pre}\) where \("\text{rt dip } = \text{Some } (\text{dsn }, \text{dsk }, \text{flag }, \text{hops }, \text{nhip }, \text{pre})\"
  by (metis kD_Some)
  with assms show "\text{sqn } \text{rt dip } \leq \text{sqn } (\text{invalidate } \text{rt } \text{dests} \text{ dip})"
  by (cases "\text{dests dip}\)" (auto simp add: invalidate_def sqn_def)
qed simp

lemma sqn_invalidate_in_dests [simp]:
  fixes \(\text{dests } \text{ipa } \text{rsn } \text{rt}\)
  assumes \("\text{dests } \text{ipa } = \text{Some rsn}\"
  and \("\text{ipa} \in kD(\text{rt})\"
  shows \("\text{sqn } (\text{invalidate } \text{rt } \text{dests} \text{/ipa} = \text{rsn}\"
unfolding invalidate_def sqn_def
using assms(1) assms(2) [THEN kD_Some]
by clarsimp

lemma dhops_invalidate [simp]:
  "\(\forall \text{dip}. \text{the } (\text{dhops } (\text{invalidate } \text{rt } \text{dests} \text{ dip})) = \text{the } (\text{dhops } \text{rt } \text{dip})\"
unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
  "\(\forall \text{dip}. \text{sqnf } (\text{invalidate } (\text{rt } \xi ) (\text{dests } \xi )) \text{ dip } = \text{sqnf } (\text{rt } \xi ) \text{ dip}\"
unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
  "\(\forall \text{dip}. \text{the } (\text{nhop } (\text{invalidate } (\text{rt } \xi ) (\text{dests } \xi )) \text{ dip}) = \text{the } (\text{nhop } (\text{rt } \xi ) \text{ dip})\"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
  fixes \(\text{rt } \text{dests } \text{dip}\)
  assumes \("\text{dip} \notin \text{dom}(\text{dests})\"
  shows \("\text{invalidate rt dests dip } = \text{rt dip}\"
using assms unfolding invalidate_def
by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes \(\text{rt } \text{dests } \text{dip}\)
  101
assumes "dip ∈ kD(rt)"
shows "invalidate rt dests dip = None"
using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
"∀ dip rt dests. dip ∈ vD(invalidate rt dests) ⇒ ∃ dip ∈ vD(rt) \land dests dip = None"
unfolding invalidate_def vD_def
by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
fixes dests dip rt
assumes "dip ∈ dom(dests)"
shows "sqn (invalidate rt dests) dip = sqn rt dip"
using assms unfolding sqn_def
by simp

lemma invalidate_changes:
fixes rt dests dip dsn dsk flag hops nhip pre
assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip, pre)"
shows "dsn = (case dests dip of None ⇒ π_2(the (rt dip)) \lor Some rsn ⇒ rsn)\land dsk = π_3(the (rt dip))\land flag = (if dests dip = None then π_4(the (rt dip)) else inv)\land hops = π_5(the (rt dip))\land nhip = π_6(the (rt dip))\land pre = π_7(the (rt dip))"
using assms unfolding invalidate_def
by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "∀ dip rt dests. dip ∈ kD (rt)\implies π_3(the (invalidate rt dests dip)) = π_3(the (rt dip))"
by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
assumes "dests ip = Some rsn" and "ip ∈ kD(rt)"
shows "ip ∈ iD(invalidate rt dests)"
using assms(1) assms(2) [THEN kD_Some]
unfolding invalidate_def iD_def
by (clarsimp split: option.split)

1.1.6 Queued Packets
Functions for sending data packets.

type_synonym store = "ip ⇒ (p × data list)"

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("σ_queue(_,_,_)")
where "σ_queue(store, dip) = case store dip of None ⇒ [] \lor Some (p, q) ⇒ q"

definition qD :: "store ⇒ ip set"
where "qD ≡ dom"

definition add :: "data ⇒ ip ⇒ store ⇒ store"
where "add d dip store = case store dip of None ⇒ store (dip := None) \lor Some (p, q) ⇒ store (dip := (p, q @ d))"

lemma qD_add [simp]:
fixes d dip store
shows "qD(add d dip store) = insert dip (qD store)"
unfolding add_def Let_def qD_def
by (clarsimp split: option.split)

definition drop :: "ip ⇒ store ⇒ store"
where "drop dip store = map_option (λ(p, q). if tl q = [] then store (dip := None)"
else store (dip ↦ (p, tl q))) (store dip)"

definition sigma_p_flag :: "store ⇒ ip ↦ p" ("σ_p-flag(_, _')")
  where "σ_p-flag(store, dip) ≡ map_option fst (store dip)"

definition unsetRRF :: "store ⇒ ip ⇒ store"
  where "unsetRRF store dip ≡ case store dip of
    None ⇒ store
  | Some (p, q) ⇒ store (dip ↦ (noreq, q))"

definition setRRF :: "store ⇒ (ip ↦ sqn) ⇒ store"
  where "setRRF store dests ≡ λdip. if dests dip = None then store dip
  else map_option (λ(_, q). (req, q)) (store dip)"

1.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. nhop is partial, thus a ‘the’ is needed, similarly for dhops and addpreRT.

2. precs is partial.

3. σ_p-flag(store, dip) is partial.

4. The routing table (rt) is modelled as a map (ip ⇒ r option) rather than a set of 7-tuples, likewise, the r is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address (dip) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.

5. Similar remarks hold for the dests mapping passed to invalidate, and store.

end

1.2 AODV protocol messages

theory A_Aodv_Message
  imports A_Norreqid
begin

datatype msg =
  Rreq nat ip sqn k ip sqn ip
  | Rrep nat ip sqn ip ip
  | Rerr "ip ↦ sqn" ip
  | Newpkt data ip
  | Pkt data ip ip

instantiation msg :: msg
begin
  definition newpkt_def [simp]: "newpkt ≡ λd, dip. Newpkt d dip"
  definition eq_newpkt_def: "eq_newpkt m ≡ case m of Newpkt d dip ⇒ True | _ ⇒ False"

  instance by intro_classes (simp add: eq_newpkt_def)
end

The msg type models the different messages used within AODV. The instantiation as a msg is a technicality due to the special treatment of newpkt messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

definition rreq :: "nat × ip × sqn × k × ip × sqn × ip ⇒ msg"
  where "rreq ≡ λ(hops, dip, dns, dsk, oip, osn, sip).
    Rreq hops dip dns dsk oip osn sip"

lemma rreq_simp [simp]:
  "rreq(hops, dip, dns, dsk, oip, osn, sip) = Rreq hops dip dns dsk oip osn sip"
1.3 The AODV protocol

theory A_Aodv
imports A_Aodv_Data A_Aodv_Message
  AWN.AWN_SOS_Labels AWN.AWN_Invariants
begin

1.3.1 Data state

record state =
  ip :: "ip"
  sn :: "sqn"
  rt :: "rt"
  rreqs :: "(ip × sqn) set"
  store :: "store"
  msg :: "msg"
  data :: "data"
  dests :: "ip → sqn"
  pre :: "ip set"
  dip :: "ip"
  oip :: "ip"
  hops :: "nat"
  dsn :: "sqn"
  dsk :: "k"
  osn :: "sqn"
  sip :: "ip"
end
abbreviation aodv_init :: "ip ⇒ state"
where "aodv_init i ≡ |
  ip = i,
  sn = 1,
  rt = Map.empty,
  rreqs = {},
  store = Map.empty,
  msg = (SOME x. True),
  data = (SOME x. True),
  dests = (SOME x. True),
  pre = (SOME x. True),
  dip = (SOME x. True),
  oip = (SOME x. True),
  hops = (SOME x. True),
  dsn = (SOME x. True),
  dsk = (SOME x. True),
  osn = (SOME x. True),
  sip = (SOME x. x ≠ i)
|"
lemma some_neq_not_eq [simp]: "¬((SOME x :: nat. x ≠ i) = i)"
  by (subst some_eq_ex) (metis zero_neq_numeral)

 definition clear_locals :: "state ⇒ state"
where "clear_locals ξ = ξ |
  msg := (SOME x. True),
  data := (SOME x. True),
  dests := (SOME x. True),
  pre := (SOME x. True),
  dip := (SOME x. True),
  oip := (SOME x. True),
  hops := (SOME x. True),
  dsn := (SOME x. True),
  dsk := (SOME x. True),
  osn := (SOME x. True),
  sip := (SOME x. x ≠ ip ξ)
|"

lemma clear_locals_sip_not_ip [simp]: "¬(sip (clear_locals ξ) = ip ξ)"
  unfolding clear_locals_def by simp

lemma clear_locals_but_not_globals [simp]:
  "ip (clear_locals ξ) = ip ξ"
  "sn (clear_locals ξ) = sn ξ"
  "rt (clear_locals ξ) = rt ξ"
  "rreqs (clear_locals ξ) = rreqs ξ"
  "store (clear_locals ξ) = store ξ"
  unfolding clear_locals_def by auto

1.3.2 Auxilliary message handling definitions

definition is_newpkt
where "is_newpkt ξ ≡ case msg ξ of
  Newpkt data’ dip’ ⇒ \{ξ[] data := data’, dip := dip’\} 
| _ ⇒ {}"

definition is_pkt
where "is_pkt ξ ≡ case msg ξ of
  Pkt data’ dip’ oip’ ⇒ \{ξ[] data := data’, dip := dip’, oip := oip’ \} 
| _ ⇒ {}"

definition is_rreq
where "is_rreq \( \xi \) \( \equiv \) case msg \( \xi \) of
  Rreq hops' dip' dsn' dsk' oip' osn' sip' \( \Rightarrow \)
  \{ \( \xi \)\( \mid \) hops := hops', dip := dip', dsn := dsn',
  dsk := dsk', oip := oip', osn := osn', sip := sip' \}\}
| - \( \Rightarrow \) \{\}

lemma is_rreq_asm [dest!]:
assumes "\( \xi ' \in is_rreq \( \xi \)\)"
shows "(\( \exists \) hops' rreqid' dip' dsn' dsk' oip' osn' sip'.
  msg \( \xi \) = Rreq hops' dip' dsn' dsk' oip' osn' sip' \wedge
  \( \xi '\) = \( \xi \)\( \mid \) hops := hops', dip := dip', dsn := dsn',
  dsk := dsk', oip := oip', osn := osn', sip := sip' \}\)"
using assms unfolding is_rreq_def
by (cases "msg \( \xi \)") simp_all

definition is_rrep
where "is_rrep \( \xi \) \( \equiv \) case msg \( \xi \) of
  Rrep hops' dip' dsn' oip' sip' \( \Rightarrow \)
  \{ \( \xi \)\( \mid \) hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' \}\}
| - \( \Rightarrow \) \{\}

lemma is_rrep_asm [dest!]:
assumes "\( \xi ' \in is_rrep \( \xi \)\)"
shows "(\( \exists \) hops' dip' dsn' oip' sip'.
  msg \( \xi \) = Rrep hops' dip' dsn' oip' sip' \wedge
  \( \xi '\) = \( \xi \)\( \mid \) hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' \}\)"
using assms unfolding is_rrep_def
by (cases "msg \( \xi \)") simp_all

definition is_rerr
where "is_rerr \( \xi \) \( \equiv \) case msg \( \xi \) of
  Rerr dests' sip' \( \Rightarrow \)
  \{ \( \xi \)\( \mid \) dests := dests', sip := sip' \}\}
| - \( \Rightarrow \) \{\}

lemma is_rerr_asm [dest!]:
assumes "\( \xi ' \in is_rerr \( \xi \)\)"
shows "(\( \exists \) dests' sip'.
  msg \( \xi \) = Rerr dests' sip' \wedge
  \( \xi '\) = \( \xi \)\( \mid \) dests := dests', sip := sip' \}\)"
using assms unfolding is_rerr_def
by (cases "msg \( \xi \)") simp_all

lemmas is_msg_defs =
is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def

lemma is_msg_inv_ip [simp]:
"\( \xi ' \in is_rerr \( \xi \) \Rightarrow ip \( \xi '\) = ip \( \xi \)"
"\( \xi ' \in is_rrep \( \xi \) \Rightarrow ip \( \xi '\) = ip \( \xi \)"
"\( \xi ' \in is_rreq \( \xi \) \Rightarrow ip \( \xi '\) = ip \( \xi \)"
"\( \xi ' \in is_pkt \( \xi \) \Rightarrow ip \( \xi '\) = ip \( \xi \)"
"\( \xi ' \in is_newpkt \( \xi \) \Rightarrow ip \( \xi '\) = ip \( \xi \)"
unfolding is_msg_defs
by (cases "msg \( \xi \)", clarsimp)+

lemma is_msg_inv_sn [simp]:
"\( \xi ' \in is_rerr \( \xi \) \Rightarrow sn \( \xi '\) = sn \( \xi \)"
"\( \xi ' \in is_rrep \( \xi \) \Rightarrow sn \( \xi '\) = sn \( \xi \)"
"\( \xi ' \in is_rreq \( \xi \) \Rightarrow sn \( \xi '\) = sn \( \xi \)"
"\( \xi ' \in is_pkt \( \xi \) \Rightarrow sn \( \xi '\) = sn \( \xi \)"
"\( \xi ' \in is_newpkt \( \xi \) \Rightarrow sn \( \xi '\) = sn \( \xi \)"
unfolding is_msg_defs
by (cases "msg \( \xi \)", clarsimp)+

lemma is_msg_inv_rt [simp]:

106
\begin{quote}
\parbox{\linewidth}{
\begin{verbatim}
"ξ' ∈ is_rerr ξ  ⇒ rt ξ' = rt ξ"
"ξ' ∈ is_rrep ξ  ⇒ rt ξ' = rt ξ"
"ξ' ∈ is_rreq ξ  ⇒ rt ξ' = rt ξ"
"ξ' ∈ is_pkt ξ  ⇒ rt ξ' = rt ξ"
"ξ' ∈ is_newpkt ξ  ⇒ rt ξ' = rt ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

lemma is_msg_inv_rreqs [simp]:
"ξ' ∈ is_rerr ξ  ⇒ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_rrep ξ  ⇒ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_rreq ξ  ⇒ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_pkt ξ  ⇒ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_newpkt ξ  ⇒ rreqs ξ' = rreqs ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

lemma is_msg_inv_store [simp]:
"ξ' ∈ is_rerr ξ  ⇒ store ξ' = store ξ"
"ξ' ∈ is_rrep ξ  ⇒ store ξ' = store ξ"
"ξ' ∈ is_rreq ξ  ⇒ store ξ' = store ξ"
"ξ' ∈ is_pkt ξ  ⇒ store ξ' = store ξ"
"ξ' ∈ is_newpkt ξ  ⇒ store ξ' = store ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

lemma is_msg_inv_sip [simp]:
"ξ' ∈ is_pkt ξ  ⇒ sip ξ' = sip ξ"
"ξ' ∈ is_newpkt ξ  ⇒ sip ξ' = sip ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

1.3.3 The protocol process

datatype pseqp =
  PAodv |
  PNewPkt |
  PPkt |
  PRreq |
  PRrep |
  PRerr

fun nat_of_seqp :: "pseqp ⇒ nat" where
  "nat_of_seqp PAodv = 1"
|  "nat_of_seqp PPkt = 2"
|  "nat_of_seqp PNewPkt = 3"
|  "nat_of_seqp PRreq = 4"
|  "nat_of_seqp PRrep = 5"
|  "nat_of_seqp PRerr = 6"

instantiation "pseqp" :: ord begin
definition less_eq_seqp [iff]: "l1 ≤ l2 = (nat_of_seqp l1 ≤ nat_of_seqp l2)"
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
instance ..
end

abbreviation AODV
where
  "AODV ≡ λ_. [clear_locals] call(PAodv)"

abbreviation PKT
where
\end{verbatim}
}\end{quote}
"PKT args ≡
[ξ. let (data, dip, oip) = args ξ in
(clear_locals ξ) (| data := data, dip := dip, oip := oip |)
call(PPkt)"

abbreviation NEWPKT
where
"NEWPKT args ≡
[ξ. let (data, dip) = args ξ in
(clear_locals ξ) (| data := data, dip := dip |)
call(PNewPkt)"

abbreviation RREQ
where
"RREQ args ≡
[ξ. let (hops, dip, dsn, dsk, oip, osn, sip) = args ξ in
(clear_locals ξ) (| hops := hops, dip := dip,
    dsn := dsn, dsk := dsk, oip := oip,
    osn := osn, sip := sip |)
call(PRreq)"

abbreviation RREP
where
"RREP args ≡
[ξ. let (hops, dip, dsn, oip, sip) = args ξ in
(clear_locals ξ) (| hops := hops, dip := dip, dsn := dsn,
    oip := oip, sip := sip |)
call(PRrep)"

abbreviation RERR
where
"RERR args ≡
[ξ. let (dests, sip) = args ξ in
(clear_locals ξ) (| dests := dests, sip := sip |)
call(PRerr)"

fun Γ_{AODV} :: "(state, msg, pseqp, pseqp label) seqp_env"
where
"Γ_{AODV} PAodv = labelled PAodv (λmsg' ξ. ξ () msg := msg' |).
( (is_newpkt) NEWPKT(λξ. (data ξ, ip ξ))
  (is_pkt) PKT(λξ. (data ξ, dip ξ, oip ξ))
  (is_rreq)
    [ξ. ξ () rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {}))]
    RREQ(λξ. (hops ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, sip ξ))
  (is_rrep)
    [ξ. ξ () rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {}))]
    RREP(λξ. (hops ξ, dip ξ, dsn ξ, oip ξ, sip ξ))
  (is_rerr)
    [ξ. ξ () rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {}))]
    RERR(λξ. (dests ξ, sip ξ))
)
  (λξ. { dip := dip | dip. dip ∈ qD(store ξ) \ vD(rt ξ)})
    [ξ. ξ () data := hd(σqueue(store ξ, dip ξ))]
  unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, ip ξ)).
  [ξ. ξ () store := the (drop (dip ξ) (store ξ))]
  AODV()}
( [ξ. ξ () dests := (λrip. if (rip ∈ vD (rt ξ) \ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
    then Some (inc (sqn (rt ξ) rip)) else None))]
  ( [ξ. ξ () rt := invalidate (rt ξ) (dests ξ)]
    [ξ. ξ () store := setRRF (store ξ) (dests ξ)]
  [ξ. ξ () pre := { the (prec (rt ξ) rip) | rip. rip ∈ dom (dests ξ)}]
  [ξ. ξ () dests := (λrip. if ((dests ξ) rip ≠ None \ the (prec (rt ξ) rip) ≠ {})}
    then (dests ξ) rip else None)]}
groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV()

Ο {ξ. dip ξ = dip | 
  ! dip. dip ∈ qD(store ξ) ∧ vD(rt ξ) ∧ the (α_p-flag(store ξ, dip)) = req }
[ξ. ξ | store := unsetRRF (store ξ) (dip ξ)]
[ξ. ξ | sn := inc (sn ξ)]
[ξ. ξ | reqs := reqs ξ ∪ {(ip ξ, sn ξ)}]
broadcast(λξ. rreq(0, dip ξ, sqn (rt ξ) (dip ξ), sqnf (rt ξ) (dip ξ), ip ξ, sn ξ, ip ξ)). AODV()"

"Γ_AODV PNewPkt = labelled PNewPkt ( 
  Σ ξ. dip ξ = ip ξ 
  deliver(λξ. data ξ).AODV()
Ο {ξ. dip ξ ≠ ip ξ }
  [ξ. ξ | store := add (data ξ) (dip ξ) (store ξ)]
AODV()"

"Γ_AODV PPkt = labelled PPkt ( 
  Σ ξ. dip ξ = ip ξ 
  deliver(λξ. data ξ).AODV()
Ο {ξ. dip ξ ≠ ip ξ }
  ⟨ξ. dip ξ ∈ vD (rt ξ)⟩
  unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, oip ξ)).AODV()
Ο {ξ. dip ξ ≠ vD (rt ξ)⟩
  ⟨ξ. dip ξ ∈ iD (rt ξ)⟩
  groupcast(λξ. the (precs (rt ξ) (dip ξ)), 
  λξ. rerr([dip ξ = sqn (rt ξ) (dip ξ)], ip ξ)). AODV()
Ο {ξ. dip ξ ≠ iD (rt ξ)⟩
AODV()
  )
)"

"Γ_AODV PRreq = labelled PRreq ( 
  Σ ξ. (oip ξ, osn ξ) ∈ rreqs ξ)
AODV()
Ο {ξ. (oip ξ, osn ξ) ≠ rreqs ξ }
[ξ. ξ | rt := update (rt ξ) (oip ξ, osn ξ, kno, val, hops ξ + 1, sip ξ, {})]
[ξ. ξ | rreqs := rreqs ξ ∪ {(oip ξ, osn ξ)}]
( 
  ⟨ξ. dip ξ = ip ξ⟩
  [ξ. ξ | sn := max (sn ξ) (dsn ξ)]
  unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, oip ξ, ip ξ)).AODV()
Ο {ξ. dip ξ ≠ ip ξ⟩
[ξ. ξ | dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ)) 
  then Some (inc (sqn (rt ξ) rip)) else None) ]
[ξ. ξ | rt := invalidate (rt ξ) (dests ξ)]
[ξ. ξ | store := setRRF (store ξ) (dests ξ)]
[ξ. ξ | pre := \{ the (precs (rt ξ) rip) \ rip. rip ∈ dom (dests ξ) \}]
[ξ. ξ | dests := (λrip. if ((dests ξ rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})) 
  then (dests ξ rip else None) ]
  groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV()
Ο {ξ. dip ξ ≠ ip ξ⟩

Γ0

AODV()

> "ΓAODV PRrep = labelled PRrep (κ. rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})
κ. oip ξ = ip ξ AODV()
k. oip ξ ≠ ip ξ
κ. oip ξ ∈ vD (rt ξ)
κ. rrep (pre ξ) (dip ξ) (dsn ξ, oip ξ, ip ξ)"

AODV()

> "ΓAODV PRerr = labelled PRerr (κ. dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
then Some (inc (dsn (rt ξ) rip)) else None)
κ. rt := the (addpreRT (rt ξ) (dip ξ) {the (nhop (rt ξ) (dip ξ))})
κ. store := setRRF (store ξ) (dests ξ)
κ. pre := ∪ {the (precs (rt ξ) rip) | rip rip ∈ dom (dests ξ)
κ. dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
then (dests ξ) rip else None)
κ. groupcast (λξ. pre ξ, λξ. rrep (dests ξ, ip ξ))
κ. groupcast (λξ. pre ξ, λξ. rrep (dests ξ, ip ξ))
κ. rrep (hops ξ + 1, dip ξ, max (dsn (rt ξ) (dip ξ)) (dsn ξ),
dsk ξ, oip ξ, osn ξ, ip ξ))
κ. update (rt ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})
κ. oip ξ = ip ξ AODV()
k. oip ξ ≠ ip ξ
κ. oip ξ ∈ vD (rt ξ)
κ. rrep (pre ξ) (dip ξ) (dsn ξ, oip ξ, ip ξ)"

AODV()"
declare \( \Gamma_{AODV}.\text{simps} \) [simp del, code del]
lemmas \( \Gamma_{AODV} \text{\_simps} \) [simp, code] = \( \Gamma_{AODV}.\text{simps} \) [simplified]

fun \( \Gamma_{AODV}.\text{skeleton} \) where

\[ \begin{align*}
\Gamma_{AODV}.\text{skeleton} \ PAodv & = \text{seqp\_skeleton} (\Gamma_{AODV} \ PAodv) \\
\Gamma_{AODV}.\text{skeleton} \ PNewPkt & = \text{seqp\_skeleton} (\Gamma_{AODV} \ PNewPkt) \\
\Gamma_{AODV}.\text{skeleton} \ PPkt & = \text{seqp\_skeleton} (\Gamma_{AODV} \ PPkt) \\
\Gamma_{AODV}.\text{skeleton} \ PRreq & = \text{seqp\_skeleton} (\Gamma_{AODV} \ PRreq) \\
\Gamma_{AODV}.\text{skeleton} \ PRrep & = \text{seqp\_skeleton} (\Gamma_{AODV} \ PRrep) \\
\Gamma_{AODV}.\text{skeleton} \ PRerr & = \text{seqp\_skeleton} (\Gamma_{AODV} \ PRerr)
\end{align*} \]

lemma \( \Gamma_{AODV}.\text{skeleton}\_wf \) [simp]:

\[ \text{wellformed } \Gamma_{AODV}.\text{skeleton} \]
proof (rule, intro allI)
fix pn pn'
show \( \text{call(pn')} \notin \text{stermsl} (\Gamma_{AODV}.\text{skeleton} \ pn) \)
by (cases pn) simp_all
qed

declare \( \Gamma_{AODV}.\text{skeleton}.\text{simps} \) [simp del, code del]
lemmas \( \Gamma_{AODV}.\text{skeleton}\_\text{simps} \) [simp, code] = \( \Gamma_{AODV}.\text{skeleton}.\text{simps} \) seqp_skeleton.simps

lemma aodv\_proc\_cases [dest]:
fixes p pn
shows \( p \in \text{ctermsl} (\Gamma_{AODV} \ pn) \Longrightarrow \)
\[ (p \in \text{ctermsl} (\Gamma_{AODV} \ PAodv) \lor p \in \text{ctermsl} (\Gamma_{AODV} \ PNewPkt) \lor p \in \text{ctermsl} (\Gamma_{AODV} \ PPkt) \lor p \in \text{ctermsl} (\Gamma_{AODV} \ PRreq) \lor p \in \text{ctermsl} (\Gamma_{AODV} \ PRrep) \lor p \in \text{ctermsl} (\Gamma_{AODV} \ PRerr)) \]
by (cases pn) simp_all

definition \( \sigma_{AODV} \) :: \( \text{ip} \Rightarrow (\text{state} \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp\_label}) \text{ seqp}) \text{ set} \)
where \( \sigma_{AODV} i \equiv \{(\text{aodv\_init} \ i, \Gamma_{AODV} \ PAodv)\} \)

abbreviation paodv :: \( \text{ip} \Rightarrow (\text{state} \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp\_label}) \text{ seqp}, \text{msg} \text{ seq\_action}) \text{ automaton} \)
where
\( \text{paodv} i \equiv (\\text{init} = \sigma_{AODV} i, \text{trans} = \text{seqp\_sos} \ \Gamma_{AODV}) \)

lemma aodv\_trans: \( \text{trans} (\text{paodv} \ i) = \text{seqp\_sos} \ \Gamma_{AODV} \)
by simp

lemma aodv\_control\_within [simp]: \( \text{control\_within} \ \Gamma_{AODV} (\text{init} (\text{paodv} \ i)) \)
    unfolding \( \sigma_{AODV}\_\text{def} \) by (rule control\_withinI) (auto simp del: \( \Gamma_{AODV}\_\text{simps} \))
assumes \( \forall l \in \text{labels } \Gamma_{AODV} \ p. \ P l p \) and \( \exists p l. \ P l p \implies Q \) shows \( "Q" \)
using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "simple_labels \( \Gamma_{AODV} \)"
proof
  fix pn p
  assume \( p \in \text{subterms}(\Gamma_{AODV} \ pn) \)
  thus \( \exists l. \ \text{labels } \Gamma_{AODV} \ p = \{l\} \)
  by (cases pn) (simp_all cong: seqp_congs | elim disjE)+
qed

lemma \( \sigma_{AODV \_labels} \) [simp]: "(\xi, p) \in \sigma_{AODV \ i} \implies \text{labels } \Gamma_{AODV} \ p = \{\text{PAodv-}:0\}"
unfolding \( \sigma_{AODV \_def} \) by simp

lemma aodv_init_kD_empty [simp]: "(\xi, p) \in \sigma_{AODV \ i} \implies kD (rt \xi) = {}"
unfolding \( \sigma_{AODV \_def} \) kD_def by simp

lemma aodv_init_sip_not_ip [simp]: "\neg (sip (aodv_init i) = i)" by simp

lemma aodv_init_sip_not_ip' [simp]:
  assumes "(\xi, p) \in \sigma_{AODV \ i}"
  shows "sip \xi \neq ip \xi"
  using assms unfolding \( \sigma_{AODV \_def} \) by simp

lemma aodv_init_sip_not_i [simp]:
  assumes "(\xi, p) \in \sigma_{AODV \ i}"
  shows "sip \xi \neq i"
  using assms unfolding \( \sigma_{AODV \_def} \) by simp

lemma clear_locals_sip_not_ip':
  assumes "ip \xi = i"
  shows "\neg (sip (clear_locals \xi) = i)"
  using assms by auto

Stop the simplifier from descending into process terms.

declare seqp_congs [cong]

Configure the main invariant tactic for AODV.

declare \( \Gamma_{AODV \_simps} \) [cterm_env]
  aodv_proc_cases [cterm1_cases]
  seq_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms_intros]
  seq_step_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms_intros]

end

1.4 Invariant assumptions and properties

theory A_Aodv_Predicates
imports A_Aodv
begin

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

abbreviation not_Pkt :: "msg \Rightarrow bool"
where "not_Pkt m \equiv \text{case m of Pkt _ _ _} \Rightarrow \text{False} | \_ \Rightarrow \text{True}"

definition msg_sender :: "msg \Rightarrow ip"
where "msg_sender m \equiv \text{case m of Rreq _ _ _ _ _ _ ipc} \Rightarrow \text{ipc}"

112
lemma msg_sender_simps [simp]:
  "\(\exists hops dip dsn dsk oip osn sip. msg_sender (Rreq hops dip dsn dsk oip osn sip) == sip\)"
  "\(\exists hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) == sip\)"
  "\(\exists dests sip. msg_sender (Rerr dests sip) == sip\)"
  "\(\exists d dip sip. msg_sender (Pkt d dip sip) == sip\)"

unfolding msg_sender_def by simp_all

definition msg_zhops :: "msg \Rightarrow bool"
where "msg_zhops m == case m of
  Rreq hopsc dipc _ _ _ oipc _ sipc => hopsc = 0 \rightarrow oipc = sipc
  Rrep hopsc dipc _ _ sipc => hopsc = 0 \rightarrow dipc = sipc
  _ => True"

lemma msg_zhops_simps [simp]:
  "\(\exists hops dip dsn dsk oip osn sip. msg_zhops (Rreq hops dip dsn dsk oip osn sip) == (hops = 0 \rightarrow oip = sip)\)"
  "\(\exists hops dip dsn oip sip. msg_zhops (Rrep hops dip dsn oip sip) == (hops = 0 \rightarrow dip = sip)\)"
  "\(\exists d dip sip. msg_zhops (Rerr dests sip) == True\)"
  "\(\exists d dip sip. msg_zhops (Pkt d dip sip) == True\)"

unfolding msg_zhops_def by simp_all

definition rreq_rrep_sn :: "msg \Rightarrow bool"
where "rreq_rrep_sn m == case m of
  Rreq _ _ _ _ _ osnc _ => osnc \geq 1
  Rrep _ _ dsnc _ _ => dsnc \geq 1
  _ => True"

lemma rreq_rrep_sn_simps [simp]:
  "\(\exists hops dip dsn dsk oip osn sip. rreq_rrep_sn (Rreq hops dip dsn dsk oip osn sip) == (osn \geq 1)\)"
  "\(\exists hops dip dsn oip sip. rreq_rrep_sn (Rrep hops dip dsn oip sip) == (dsn \geq 1)\)"
  "\(\exists dests sip. rreq_rrep_sn (Rerr dests sip) == True\)"
  "\(\exists d dip. rreq_rrep_sn (Newpkt d dip) == True\)"

unfolding rreq_rrep_sn_def by simp_all

definition rreq_rrep_fresh :: "rt \Rightarrow msg \Rightarrow bool"
where "rreq_rrep_fresh crt m == case m of
  Rreq hopsc dipc _ _ _ oipc osnc ipcc => (ipcc \neq oipc \rightarrow
    (ipcc \in kD(crt) \land (sqn crt oipc > osnc
      \lor (sqn crt oipc = osnc
        \land the (dhops crt oipc) \leq hopsc
        \land the (flag crt oipc) = val)))
    \lor Rreq hopsc dipc dsnc _ _ => dipc \in kD(crt)
    \land sqn crt dipc = dsnc
    \land the (dhops crt dipc) = hopsc
    \land the (flag crt dipc) = val)
  _ => True"

lemma rreq_rrep_fresh [simp]:
  "\(\exists hops dip dsn dsk oip osn sip. rreq_rrep_fresh crt (Rreq hops dip dsn dsk oip osn sip) == (sip \neq oip \rightarrow
    (sip \in kD(crt) \land (sqn crt oip = osn
      \lor (sqn crt oip > osn
        \land the (dhops crt oip) \leq hops
        \land the (flag crt oip) = val)))\)"

  "\(\exists hops dip dsn oip sip. rreq_rrep_fresh crt (Rrep hops dip dsn oip sip) == (sip \neq dip \rightarrow dip \in kD(crt))\)"
∧ sqn crt dip = dsn
∧ the (dhops crt dip) = hops
∧ the (flag crt dip) = val"
∧ \(d\) dip.
∧ \(rreq\_rrep\_fresh\) crt (\(Rerr\) \(dests\) dip) = True"
∧ \(d\) dip.
∧ \(rreq\_rrep\_fresh\) crt (\(Newpkt\) \(d\) dip) = True"
∧ \(d\) dip dip.
∧ \(rreq\_rrep\_fresh\) crt (\(Pkt\) \(d\) dip dip) = True"

unfolding \(rreq\_rrep\_fresh\) def by simp_all

definition \(rerr\_invalid\) :: "rt \Rightarrow msg \Rightarrow bool"
where "\(rerr\_invalid\) \(crt\) \(m\) \(\equiv\) \(\text{case } m \text{ of } Rerr destin\_sc\ _ \Rightarrow (\forall ripc \in \text{dom}(destin\_sc) . \ripc \in iD(crt) \land \text{the } (destin\_sc \ripc) = sqn \crt \ripc)\) |
\_ \Rightarrow \text{True}"

lemma \(rerr\_invalid\) [simp]:
"\(d\) hops dip dsn dsk oip osn sip.
\(rerr\_invalid\) \(crt\) (\(Rreq\) \(hops\) dip dsn dsk oip osn sip) = True"
"\(d\) hops dip dsn oip sip.
\(rerr\_invalid\) \(crt\) (\(Rreq\) \(hops\) dip dsn oip sip) = True"
"\(d\) dests sip.
\(rerr\_invalid\) \(crt\) (\(Rerr\) \(dests\) sip) = (\(\forall rip \in \text{dom}(dests) . \ri \in \text{iD}(crt) \land \text{the } (dests \ri) = sqn \crt \ri)"
"\(d\) dip.
\(rerr\_invalid\) \(crt\) (\(Newpkt\) \(d\) dip) = True"
"\(d\) dip dip.
\(rerr\_invalid\) \(crt\) (\(Pkt\) \(d\) dip dip) = True"

unfolding \(rerr\_invalid\) def by simp_all

definition \(initmissing\) :: "(\(nat \Rightarrow \text{state option}\) \times \text{'a} \Rightarrow (\(nat \Rightarrow \text{state}\) \times \text{'a})"
where "\(initmissing\) \(\sigma\) = (\(\lambda i . \text{case } (\text{fst } \sigma) \text{ i of } \text{None } \Rightarrow aodv\_init \ i \ | \ \text{Some } s \Rightarrow s, \text{snd } \sigma)\)"

lemma not_in_net_ips_fst_init_missing [simp]:
assumes "\(i\) \notin \text{net_ips} \(\sigma\)"
shows "\(\text{fst } (\text{initmissing } (\text{netgmap } \text{fst } \sigma)) \text{ i} = aodv\_init \text{ i}\)"
using \(\text{assms}\) unfolding \(\text{initmissing}\) \(\text{def}\) by simp

lemma \(\text{fst}\_\text{initmissing}\_\text{netgmap}\_\text{pair}\_\text{fst}\) [simp]:
"\(\text{fst } (\text{initmissing } (\text{netgmap } (\lambda (p, q). (\text{fst } (\text{id } p), \text{snd } (\text{id } p), q)) \text{ s}))\) = \(\text{fst } (\text{initmissing } (\text{netgmap } \text{fst } \text{s}))\)"
unfolding \(\text{initmissing}\) \(\text{def}\) by auto

We introduce a streamlined alternative to \(\text{initmissing}\) with \(\text{netgmap}\) to simplify invariant statements and thus facilitate their comprehension and presentation.

lemma \(\text{fst}\_\text{initmissing}\_\text{netgmap}\_\text{default}\_\text{aodv}\_\text{init}\_\text{netlift}\):
"\(\text{fst } (\text{initmissing } (\text{netgmap } \text{fst } \text{s})) = \text{default } aodv\_\text{init } (\text{netlift } \text{fst } \text{s})\)"

unfolding \(\text{initmissing}\) \(\text{def}\) \(\text{default}\) \(\text{def}\)
by (simp add: \(\text{fst}\_\text{netgmap}\_\text{netlift}\) del: One_nat_def)

definition \(\text{netglobal}\) :: "((\(nat \Rightarrow \text{state}\) \Rightarrow \text{bool}) \Rightarrow ((\text{state} \times \text{'b}) \times \text{'c}) \text{ net_state} \Rightarrow \text{bool})"
where "\(\text{netglobal}\) \(\text{P}\) \(\equiv\) (\(\lambda s . \text{P} (\text{default } aodv\_\text{init } (\text{netlift } \text{fst } \text{s})))\)"

end

1.5 Quality relations between routes

theory \(\text{A\_Fresher}\)
imports \(\text{A\_Aodv\_Data}\)
begin

1.5.1 Net sequence numbers

On individual routes

definition \(\text{nsqn}\_r\) :: "\(r \Rightarrow \text{sqn}\)"
"nsqn, r ≡ if \( \pi_4(r) = \text{val} \lor \pi_2(r) = 0 \) then \( \pi_2(r) - 1 \) else \( \pi_2(r) \)"

**Lemma nsqr_def':**
"nsqn, r = (if \( \pi_4(r) = \text{inv} \) then \( \pi_2(r) - 1 \) else \( \pi_2(r) \))"

unfolding nsqr_def by simp

**Lemma nsqn_zero [simp]:**
"\( \forall dsn \text{ dsk flag hops nhip pre. nsqn_r (0, dsk, flag, hops, nhip, pre) = 0} \)"

unfolding nsqr_def by clarsimp

**Lemma nsqn_val [simp]:**
"\( \forall dsn \text{ dsk hops nhip pre. nsqn_r (dsn, dsk, val, hops, nhip, pre) = dsn} \)"

unfolding nsqr_def by clarsimp

**Lemma nsqn_inv [simp]:**
"\( \forall dsn \text{ dsk hops nhip pre. nsqn_r (dsn, dsk, inv, hops, nhip, pre) = dsn - 1} \)"

unfolding nsqr_def by clarsimp

**Lemma nsqn_lte_dsn [simp]:**
"\( \forall dsn \text{ dsk flag hops nhip pre. nsqn_r (dsn, dsk, flag, hops, nhip, pre) ≤ dsn} \)"

unfolding nsqr_def by clarsimp

### On routes in routing tables

**Definition**
\[ \text{nsqn} :: "rt \Rightarrow ip \Rightarrow sqn" \]

where
"\( \text{nsqn} ≡ \lambda rt \text{ dip. case } \sigma_{route}(rt, dip) \text{ of None } \Rightarrow 0 \mid \text{Some } r \Rightarrow \text{nsqn_r}(r) \)"

**Lemma nsqn_sqn_def:**
"\( \forall rt \text{ dip. nsqn rt dip} = (if flag rt dip = \text{Some val} \lor \text{sqn rt dip} = 0 \) then \text{sqn rt dip} else \text{sqn rt dip} - 1) \)"

unfolding nsqn_def by clarsimp

**Lemma not_in_kD_nsqn [simp]:**
assumes "\( \text{dip } \notin \text{kD}(rt) \)"
shows "\( \text{nsqn rt dip} = 0 \)"
using assms unfolding nsqn_def by simp

**Lemma kD_nsqn:**
assumes "\( \text{dip } \in \text{kD}(rt) \)"
shows "\( \text{nsqn rt dip} = \text{nsqn_r} \text{(the (} \sigma_{route}(rt, dip) \text{))} \)"
using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

**Lemma nsqr_r_flag_pred [simp, intro]:**
fixes dsn dsk flag hops nhip pre
assumes "\( \text{P (nsqn_r (dsn, dsk, val, hops, nhip, pre))} \)"
and "\( \text{P (nsqn_r (dsn, dsk, inv, hops, nhip, pre))} \)"
shows "\( \text{P (nsqn_r (dsn, dsk, flag, hops, nhip, pre))} \)"
using assms by (cases flag) auto

**Lemma nsqr_r_flag_pred [simp, intro]:**
fixes dsn dsk flag hops nhip pre
assumes "\( \text{P (nsqn_r (dsn, dsk, val, hops, nhip, pre))} \)"
and "\( \text{P (nsqn_r (dsn, dsk, inv, hops, nhip, pre))} \)"
shows "\( \text{P (nsqn_r (dsn, dsk, flag, hops, nhip, pre))} \)"
using assms by (cases flag) auto

**Lemma nsqr_r_flag_pred [simp, intro]:**
fixes dsn dsk flag hops nhip pre
assumes "\( \text{P (nsqn_r (dsn, dsk, val, hops, nhip, pre))} \)"
and "\( \text{P (nsqn_r (dsn, dsk, inv, hops, nhip, pre))} \)"
shows "\( \text{P (nsqn_r (dsn, dsk, flag, hops, nhip, pre))} \)"
using assms by (cases flag) auto

**Lemma nsqr_r_flag_pred [simp, intro]:**
fixes dsn dsk flag hops nhip pre
assumes "\( \text{P (nsqn_r (dsn, dsk, val, hops, nhip, pre))} \)"
and "\( \text{P (nsqn_r (dsn, dsk, inv, hops, nhip, pre))} \)"
shows "\( \text{P (nsqn_r (dsn, dsk, flag, hops, nhip, pre))} \)"
using assms by (cases flag) auto

**Lemma sqn_nsqn:**
"\( \forall rt \text{ dip. sqn rt dip} - 1 ≤ \text{nsqn rt dip} \)"
unfolding sqn_def nsqn_def by (clarsimp split: option.split)

**Lemma nsqn_sqn:**
"\( \forall rt \text{ dip. nsqn rt dip} ≤ \text{sqn rt dip} \)"
unfolding sqn_def nsqn_def by (cases "rt dip") auto

115
lemma val_nsqn_sqn [elim, simp]:
  assumes "ip∈kD(rt)"
  and "the (flag rt ip) = val"
  shows "nsqn rt ip = sqn rt ip"
  using assms unfolding nsqn_sqn_def by (auto)

lemma vD_nsqn_sqn [elim, simp]:
  assumes "ip∈vD(rt)"
  shows "nsqn rt ip = sqn rt ip"
  proof -
    from ⟨ip∈vD(rt)⟩ have "ip∈kD(rt)"
    and "the (flag rt ip) = val" by (auto)
    thus ?thesis ..
  qed

lemma inv_nsqn_sqn [elim, simp]:
  assumes "ip∈kD(rt)"
  and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
  using assms unfolding nsqn_sqn_def by (auto)

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip∈iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
  proof -
    from ⟨ip∈iD(rt)⟩ have "ip∈kD(rt)"
    and "the (flag rt ip) = inv" by (auto)
    thus ?thesis ..
  qed

lemma nsqn_update_changed_kno_val [simp]: 
  "∀ rt ip dsn dsk hops nhip.
   rt ≠ update rt ip (dsn, dsk, flag, hops, nhip, {}) 
   ⟹ nsqn (update rt ip (dsn, dsk, val, hops, nhip, {})) ip = dsn"
  unfolding nsqn_def update_def
  by (clarsimp simp: kD_nsqn split: option.split_asm option.split if_split_asm)
  (metis fun_upd_triv)

lemma nsqn_addpreRT_inv [simp]:
  "∀ rt dip npre dip'. dip ∈ kD(rt) ⟹
   nsqn (the (addpreRT rt dip npre)) dip' = nsqn rt dip'"
  unfolding addpreRT_def nsqn_def nsqns_def
  by (frule kD_Some) (clarsimp simp: split: option.split)

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip pre rt ip
  assumes "dip ≠ ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip, pre)) dip = nsqn rt dip"
  using assms unfolding nsqn_def
  by (clarsimp simp: split: option.split)

lemma nsqn_invalidate_eq:
  assumes "dip∈kD(rt)"
  and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
  using assms
  proof -
    from assms obtain dsk hops nhip pre
    where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip, pre)"
    unfolding invalidate_def
    by auto
    moreover from ⟨dip∈kD(rt)⟩ have "dip∈kD(invalidate rt dests)" by simp
    ultimately show ?thesis
    using ⟨dests dip = Some rsn⟩ by simp

116
lemma nsqn_invalidate_other [simp]:
assumes "dip \in kD(rt)"
and "dip \notin dom dests"
shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
using assms by (clarsimp simp add: kD_nsqn)

1.5.2 Comparing routes

definition
fresher :: "r ⇒ r ⇒ bool" ("(_/ ≺_/ _)"
where
"fresher r r' ≡ ((nsqn, r < nsqn, r') ∨ (nsqn, r = nsqn, r' ∧ π_5(r) ≥ π_5(r')))"

lemma fresherI1 [intro]:
assumes "nsqn, r < nsqn, r'"
shows "r ≺ r'"
unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
assumes "nsqn, r = nsqn, r'" and "π_5(r) ≥ π_5(r')"
shows "r ≺ r'"
unfolding fresher_def using assms by simp

lemma fresherI [intro]:
assumes "(nsqn, r < nsqn, r') ∨ (nsqn, r = nsqn, r' ∧ π_5(r) ≥ π_5(r'))"
shows "r ≺ r'"
unfolding fresher_def using assms.

lemma fresherE [elim]:
assumes "r ≺ r'" and "nsqn, r < nsqn, r'" ⇒ P r r'" and "nsqn, r = nsqn, r' ∧ π_5(r) ≥ π_5(r')" ⇒ P r r'"
shows "P r r'"
using assms unfolding fresher_def by auto

lemma fresher_refl [simp]: "r ≺ r"
unfolding fresher_def by simp

lemma fresher_trans [elim, trans]:
"[ x ≺ y; y ≺ z ] ⇒ x ≺ z"
unfolding fresher_def by auto

lemma not_fresher_trans [elim, trans]:
"[ ¬(x ≺ y); ¬(z ≺ x)] ⇒ ¬(z ≺ y)"
unfolding fresher_def by auto

lemma fresher_dsn_flag_hops_const [simp]:
fixes dsk dsk' flag hops nhip nhip' pre pre'
shows "(dsn, dsk, flag, hops, nhip, pre) ⊆ (dsn, dsk', flag, hops, nhip', pre')"
unfolding fresher_def by (cases flag) simp_all

lemma addpre_fresher [simp]: "∀r npre. r ≺ (addpre r npre)"
by clarsimp

1.5.3 Comparing routing tables

definition
rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
"rt_fresher ≡ \lambda dip rt rt'. (the (sigma rt (dip))) ⊆ (the (sigma rt' (dip)))"
abbreviation
  \texttt{rt\_fresher\_syn} :: "rt \Rightarrow \text{ip} \Rightarrow rt \Rightarrow bool" ("\_/
  \_\_\_) [51, 999, 51] 50
where
  "rt1 \sqsubseteq rt2 \equiv rt\_fresher \ i \ rt1 \ rt2"

lemma \texttt{rt\_fresher\_def'}:
  "\((rt1 \sqsubseteq rt2) = (\text{nsqn} \ (\text{the} \ (rt1 \ i)) < \text{nsqn} \ (\text{the} \ (rt2 \ i)) \lor
  \text{nsqn} \ (\text{the} \ (rt1 \ i)) = \text{nsqn} \ (\text{the} \ (rt2 \ i)) \land \pi_5 \ (\text{the} \ (rt1 \ i)) \leq \pi_5 \ (\text{the} \ (rt2 \ i)))"
unfolding rt\_fresher\_def fresher\_def by \text{(rule refl)}

lemma single\_rt\_fresher [intro]:
  assumes "the (rt1 ip) \sqsubseteq the (rt2 ip)"
  shows "rt1 \sqsubseteq \text{ip} \ rt2"
using assms unfolding rt\_fresher\_def .

lemma rt\_fresher\_single [intro]:
  assumes "rt1 \sqsubseteq \text{ip} \ rt2"
  shows "the (rt1 ip) \sqsubseteq the (rt2 ip)"
using assms unfolding rt\_fresher\_def .

lemma rt\_fresher\_def2:
  assumes "dip \in kD(rt1)"
  and "dip \in kD(rt2)"
  shows "\((rt1 \sqsubseteq dip \ rt2) = (\text{nsqn} \ rt1 \ dip < \text{nsqn} \ rt2 \ dip
  \lor (\text{nsqn} \ rt1 \ dip = \text{nsqn} \ rt2 \ dip
  \land (\text{dhops} \ rt1 \ dip) \geq (\text{dhops} \ rt2 \ dip)))"
using assms unfolding rt\_fresher\_def by (simp add: kD_nsqn proj5_eq_dhops)

lemma rt\_fresher\_I1 [intro]:
  assumes "dip \in kD(rt1)"
  and "dip \in kD(rt2)"
  and "\text{nsqn} \ rt1 \ dip < \text{nsqn} \ rt2 \ dip"
  shows "rt1 \sqsubseteq dip \ rt2"
unfolding rt\_fresher\_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt\_fresher\_I2 [intro]:
  assumes "dip \in kD(rt1)"
  and "dip \in kD(rt2)"
  and "\text{nsqn} \ rt1 \ dip = \text{nsqn} \ rt2 \ dip"
  and "\text{the} \ (\text{dhops} \ rt1 \ dip) \geq \text{the} \ (\text{dhops} \ rt2 \ dip)"
  shows "rt1 \sqsubseteq dip \ rt2"
unfolding rt\_fresher\_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt\_fresher\_E [elim]:
  assumes "rt1 \sqsubseteq dip \ rt2"
  and "dip \in kD(rt1)"
  and "dip \in kD(rt2)"
  and "\[ \text{nsqn} \ rt1 \ dip < \text{nsqn} \ rt2 \ dip \] \Rightarrow P \ rt1 \ rt2 \ dip"
  and "\[ \text{nsqn} \ rt1 \ dip = \text{nsqn} \ rt2 \ dip;\]
  \text{the} \ (\text{dhops} \ rt1 \ dip) \geq \text{the} \ (\text{dhops} \ rt2 \ dip) \] \Rightarrow P \ rt1 \ rt2 \ dip"
  shows "P \ rt1 \ rt2 \ dip"
using assms(1) unfolding rt\_fresher\_def2 [OF assms(2-3)]
using assms(4-5) by auto

lemma rt\_fresher\_refl [simp]: "rt \sqsubseteq dip \ rt"
unfolding rt\_fresher\_def by simp

lemma rt\_fresher\_trans [elim, trans]:
  assumes "rt1 \sqsubseteq dip \ rt2"
  and "rt2 \sqsubseteq dip \ rt3"
  shows "rt1 \sqsubseteq dip \ rt3"
using assms unfolding rt\_fresher\_def by auto

lemma rt\_fresher\_if\_Some [intro!]:
assumes "the (rt dip) ⊆ r"
  shows "rt ⊆ dip (λip. if ip = dip then Some r else rt ip)"
using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
"rt_fresh_as ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ (rt2 ⊑ dip rt1)"

abbreviation
  rt_fresh_as_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/≈_/_)" [51, 999, 51] 50)
where
"rt1 ≈ i rt2 ≡ rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: "⋀rt dip. rt ≈ dip rt"
unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
"⋀rt1 rt2 rt3 dip. [ rt1 ≈ dip rt2; rt2 ≈ dip rt3 ] ⇒ rt1 ≈ dip rt3"
unfolding rt_fresh_as_def rt_fresher_def
by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt1"
shows "rt1 ≈ dip rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "the (rt1 dip) ⊆ the (rt2 dip)"
  and "the (rt2 dip) ⊆ the (rt1 dip)"
shows "rt1 ≈ dip rt2"
using assms unfolding rt_fresh_as_def
by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
assumes "dip ∈ kD(rt)"
  and "dip ∈ kD(rt')"
  and "nsqn rt dip = nsqn rt' dip"
  and "π5(the (rt dip)) = π5(the (rt' dip))"
shows "rt ≈ dip rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip) ≤ the (dhops rt dip)"
    by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt ⊆ dip rt'"
    by (rule rt_fresherI2)

next
  from assms(1-2,4) have dhops: "the (dhops rt dip) ≤ the (dhops rt' dip)"
    by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt' ⊆ dip rt"
    by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
assumes "rt1 ≈ dip rt2"
  and "[ rt1 ⊆ dip rt2; rt2 ⊆ dip rt1 ] ⇒ P rt1 rt2 dip"
shows "P rt1 rt2 dip"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
assumes "rt1 ≈ dip rt2"
shows "rt1 ⊆ dip rt2"
using assms unfolding rt_fresh_as_def by simp
lemma \textit{rt} \textit{fresh} \_as\textit{D2} [dest]:
    assumes "\textit{rt1} \approx_{\text{dip}} \textit{rt2}"
    shows "\textit{rt2} \sqsubseteq_{\text{dip}} \textit{rt1}"
    using assms unfolding \textit{rt} \textit{fresh} \_as\textit{def} by simp

lemma \textit{rt} \textit{fresh} \_as\textit{sym}:
    assumes "\textit{rt2} \approx_{\text{dip}} \textit{rt1}"
    shows "\textit{rt2} \sqsubseteq_{\text{dip}} \textit{rt1}"
    using assms unfolding \textit{rt} \textit{fresh} \_as\textit{def} by simp

lemma \textit{not}_\textit{rt} \textit{fresh} \_as\textit{I1} [intro]:
    assumes "\neg (\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2})"
    shows "\neg (\textit{rt1} \approx_{\text{dip}} \textit{rt2})"
    proof
        assume "\textit{rt1} \approx_{\text{dip}} \textit{rt2}"
        hence "\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2}" ..
        with \(\neg (\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2})\) show False ..
    qed

lemma \textit{not}_\textit{rt} \textit{fresh} \_as\textit{I2} [intro]:
    assumes "\neg (\textit{rt2} \sqsubseteq_{\text{ip}} \textit{rt1})"
    shows "\neg (\textit{rt1} \approx_{\text{dip}} \textit{rt2})"
    proof
        assume "\textit{rt1} \approx_{\text{dip}} \textit{rt2}"
        hence "\textit{rt2} \sqsubseteq_{\text{ip}} \textit{rt1}" ..
        with \(\neg (\textit{rt2} \sqsubseteq_{\text{ip}} \textit{rt1})\) show False ..
    qed

lemma \textit{not}_single_\textit{rt} \textit{fresher} [elim]:
    assumes "\neg (\textit{the (rt1 ip)} \sqsubseteq \textit{the (rt2 ip)})"
    shows "\neg (\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2})"
    proof
        assume "\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2}"
        hence "\textit{the (rt1 ip)} \sqsubseteq \textit{the (rt2 ip)}" ..
        with \(\neg (\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2})\) show False ..
    qed

lemmas \textit{not}_single_\textit{rt} \textit{fresh} \_as\textit{I1} [intro] = \textit{not}_\textit{rt} \textit{fresh} \_as\textit{I1} [OF \textit{not}_single_\textit{rt} \textit{fresher}]
lemmas \textit{not}_single_\textit{rt} \textit{fresh} \_as\textit{I2} [intro] = \textit{not}_\textit{rt} \textit{fresh} \_as\textit{I2} [OF \textit{not}_single_\textit{rt} \textit{fresher}]

lemma \textit{not}_\textit{rt} \textit{fresher} \_single [elim]:
    assumes "\neg (\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2})"
    shows "\neg (\textit{the (rt1 ip)} \sqsubseteq \textit{the (rt2 ip)})"
    proof
        assume "\textit{the (rt1 ip)} \sqsubseteq \textit{the (rt2 ip)}"
        hence "\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2}" ..
        with \(\neg (\textit{rt1} \sqsubseteq_{\text{ip}} \textit{rt2})\) show False ..
    qed

lemma \textit{rt} \textit{fresh} \_as\textit{nsqnr}:
    assumes "\textit{dip} \in kD(\textit{rt1})"
        and "\textit{dip} \in kD(\textit{rt2})"
        and "\textit{rt1} \approx_{\text{dip}} \textit{rt2}"
    shows "nsqn, (the (\textit{rt2 dip})) = nsqn, (the (\textit{rt1 dip}))"
    using assms(3) unfolding \textit{rt} \textit{fresh} \_as\textit{def}
    by (auto simp: \textit{rt} \textit{fresher} \_def2 [OF \textit{dip} \in kD(\textit{rt1)}] \textit{dip} \in kD(\textit{rt2})]
        kD_nsqn \[OF \textit{dip} \in kD(\textit{rt1})]
        kD_nsqn \[OF \textit{dip} \in kD(\textit{rt2})]

lemma \textit{rt} \textit{fresher} \_mapupd [intro!]:
    assumes "\textit{dip} \in kD(\textit{rt})"
        and "the (\textit{rt dip}) \sqsubseteq \textit{r}"
shows \( \text{rt} \subseteq_{\text{dip}} \text{rt}(\text{dip} \mapsto r) \)
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_map_update_other [intro!]:
assumes "\( \text{dip} \in \text{kD}(\text{rt}) \)"
and "\( \text{dip} \neq \text{ip} \)"
shows "\( \text{rt} \subseteq_{\text{dip}} \text{rt}(\text{ip} \mapsto r) \)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_update_other [simp]:
assumes \( \text{inkD}: \text{dip} \in \text{kD}(\text{rt}) \)\nand "\( \text{dip} \neq \text{ip} \)"
shows "\( \text{rt} \subseteq \text{dip} \cdot \text{update rt ip r} \)"
using assms unfolding update_def by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
assumes "\( \text{dip} \in \text{kD}(\text{rt}) \)"
and "\( \text{the}(\text{dhops rt dip}) \geq 1 \)"
and "\( \text{update_arg_wf r} \)"
shows "\( \text{rt} \subseteq \text{dip} \cdot \text{update rt ip r} \)"
proof (cases "\( \text{dip} = \text{ip} \)"
assume "\( \text{dip} \neq \text{ip} \)"
with \( \text{dip} \in \text{kD}(\text{rt}) \) show \( ?\text{thesis} \)
by (rule rt_fresher_update_other)
next
assume "\( \text{dip} = \text{ip} \)"
from \( \text{dip} \in \text{kD}(\text{rt}) \) \ obtain \( \text{dsn}, \text{dsk}, \text{f}, \text{hops}, \text{nhip}, \text{pre} \)
where \( \text{rtn} \) \[simp\]: "\( \text{the}(\text{rt dip}) = (\text{dsn}, \text{dsk}, \text{f}, \text{hops}, \text{nhip}, \text{pre}) \)"
by (metis prod_cases6)
with \( \text{the}(\text{dhops rt dip}) \geq 1 \)\nand \( \text{dip} \in \text{kD}(\text{rt}) \)
have "\( \text{hops} \geq 1 \)"
by (metis proj5_eq_dhops projs(4))
from \( \text{dip} \in \text{kD}(\text{rt}) \)\nhave \( \text{rtn} \) \[simp\]: "\( \text{sqn rt dip} = \text{dsn} \)"
and \( \text{rtn} \) \[simp\]: "\( \text{the}(\text{dhops rt dip}) = \text{hops} \)"
and \( \text{rtn} \) \[simp\]: "\( \text{the}(\text{flag rt dip}) = \text{f} \)"
by (simp add: sqn_def proj5_eq_dhops \[symmetric\] proj4_eq_flag \[symmetric\])+
from \( \text{update_arg_wf r} \)\ have "\( (\text{dsn}, \text{dsk}, \text{f}, \text{hops}, \text{nhip}, \text{pre}) \)\ ⊑ \( \text{the}(\text{update rt dip r} \cdot \text{dip}) \)"
proof (rule wf_r_cases)
fix nhip pre
from \( \text{hops} \geq 1 \)\ have "\( \forall \text{pre}'. (\text{dsn}, \text{dsk}, \text{f}, \text{hops}, \text{nhip}, \text{pre}) \)
\( \subseteq (\text{dsn}, \text{unk}, \text{val}, \text{Suc 0}, \text{nhip}, \text{pre}') \)"
unfolding fresher_def sqn_def by (cases f) auto
thus "\( (\text{dsn}, \text{dsk}, \text{f}, \text{hops}, \text{nhip}, \text{pre}) \)\ ⊑ \( \text{the}(\text{update rt dip r} \cdot \text{dip}) \)"
using \( \text{dip} \in \text{kD}(\text{rt}) \) by - (rule update_cases_kD, simp_all)
next
fix \( \text{dsn} :: \text{sqn} \) and \( \text{hops} \) \( \text{nhip} \) \( \text{pre} \)
assume "\( 0 < \text{dsn} \)"
show "\( (\text{dsn}, \text{dsk}, \text{f}, \text{hops}, \text{nhip}, \text{pre}) \)
\( \subseteq (\text{dsn}, \text{kno}, \text{val}, \text{hops}, \text{nhip}, \text{pre} \cup \text{pre}) \)"
proof (rule update_cases_kD [OF _ \( \text{dip} \in \text{kD}(\text{rt}) \)], simp_all add: \( 0 < \text{dsn} \))
assume "\( \text{dsn} < \text{dsn} \)"
thus "\( (\text{dsn}, \text{dsk}, \text{f}, \text{hops}, \text{nhip}, \text{pre}) \)
\( \subseteq (\text{dsn}, \text{kno}, \text{val}, \text{hops}, \text{nhip}, \text{pre} \cup \text{pre}) \)"
unfolding fresher_def by auto
next
assume "\( \text{dsn} = \text{dsn} \)"
and "\( \text{hops} < \text{hops} \)"
thus "\( (\text{dsn}, \text{dsk}, \text{f}, \text{hops}, \text{nhip}, \text{pre}) \)
\( \subseteq (\text{dsn}, \text{kno}, \text{val}, \text{hops}, \text{nhip}, \text{pre} \cup \text{pre}) \)"
unfolding fresher_def nsqn, def by simp
assume "dsnᵣ = dsn"
with ⟨0 < dsn⟩
show "⟨dsn, dskᵣ, inv, hopsᵣ, nhipᵣ, preᵣ⟩ ⊆ ⟨dsn, kno, val, hops, nhip, pre ∪ preᵣ⟩"
unfolding fresher_def by simp
qed

hence "rt ⊑ dip update rt dip r"
by (rule single_rt_fresher, simp)
with ⟨dip = ip⟩
show ?thesis
by simp
qed

theorem rt_fresher_invalidate [simp]:
assumes "dip ∈ kD(rt)"
and indests: "∀ rip ∈ dom(dests). rip ∈ vD(rt) ∧ sqn rt rip < the (dests rip)"
shows "rt ⊑ dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)"
assume "dip ∉ dom(dests)"
thus ?thesis using ⟨dip ∈ kD(rt)⟩
by (rule single_rt_fresher, simp)
next
assume "dip ∈ dom(dests)"
moreover with indests have "dip ∈ vD(rt)"
and "sqn rt dip < the (dests dip)"
by auto
ultimately show ?thesis
unfolding invalidate_def sqn_def
by (rule single_rt_fresher, auto simp: fresher_def)
qed

lemma nsqnᵣ.invalidate [simp]:
assumes "dip ∈ kD(rt)"
and "dip ∈ dom(dests)"
shows "nsqnᵣ (the (invalidate rt dests dip)) = the (dests dip) - 1"
using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
assumes "dip ∈ kD(rt)"
and "∀ rip ∈ dom(dests). rip ∈ vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
shows "rt ≈ dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)"
assume "dip ∈ dom(dests)"
with ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)"
by simp
with ⟨dip ∈ kD(rt)⟩
show ?thesis
by rule (simp_all add: ⟨dip ∈ dom(dests)⟩)
next
assume "dip ∈ dom(dests)"
with assms(2) have "dip ∈ vD(rt)"
and "the (dests dip) = inc (sqn rt dip)" by auto
from ⟨dip ∈ vD(rt)⟩ have "dip ∈ kD(rt)" by simp
moreover then have "dip ∈ kD(invalidate rt dests)" by simp
ultimately show ?thesis
proof (rule nsqnᵣ.invalidate)
from ⟨dip ∈ vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
also have "sqn rt dip = nsqnᵣ (the (invalidate rt dests dip))"
proof -
from ⟨dip ∈ kD(rt)⟩ have "nsqnᵣ (the (invalidate rt dests dip)) = the (dests dip) - 1"
using ⟨dip ∈ dom(dests)⟩ by (rule nsqnᵣ.invalidate)
with ⟨the (dests dip) = inc (sqn rt dip)⟩
show "sqn rt dip = inc (sqn rt dip)" by simp
qed
also from ⟨dip ∈ kD(invalidate rt dests)⟩
have "nsqn_r (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
by (simp add: kD_nsqn)
finally show "nsqn rt dip = nsqn (invalidate rt dests) dip".
qed simp
qed

lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]

lemma rt_fresh_as_addpreRT [simp]:
  assumes "ip\in\kD(rt)"
  shows "rt \approx dip \text{ the (addpreRT rt ip npre)}"
  using assms [THEN kD_Some] by (auto simp: addpreRT_def)
lemmas rt_fresher_addpreRT [simp] = rt_fresh_as_addpreRT [THEN rt_fresh_asD1]

1.5.4 Strictly comparing routing tables

definition rt_strictly_fresher :: "ip \Rightarrow rt \Rightarrow rt \Rightarrow bool"
where
  "rt_strictly_fresher \equiv \lambda dip rt1 rt2. (rt1 \sqsubseteq dip rt2) \land \neg(rt1 \approx dip rt2)"
abbreviation
  rt_strictly_fresher_syn :: "rt \Rightarrow ip \Rightarrow rt \Rightarrow bool" ("/_\_\_\_/" [51, 999, 51] 50)
where
  "rt1 \sqsubseteq dip rt2 \equiv rt_strictly_fresher dip rt1 rt2"

lemma rt_strictly_fresher_def'':
  "rt1 \sqsubseteq dip rt2 = ((rt1 \sqsubseteq dip rt2) \land \neg(rt2 \sqsubseteq dip rt1))"
unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresherI' [intro]:
  assumes "rt1 \sqsubseteq dip rt2"
  and "\neg(rt2 \sqsubseteq dip rt1)"
  shows "rt1 \sqsubseteq dip rt2"
  using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherE' [elim]:
  assumes "rt1 \sqsubseteq dip rt2"
  and "[ rt1 \sqsubseteq dip rt2; \neg(rt2 \sqsubseteq dip rt1) ] \Longrightarrow P rt1 rt2 i"
  shows "P rt1 rt2 i"
  using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherI [intro]:
  assumes "rt1 \approx dip rt2"
  and "\neg(rt1 \approx dip rt2)"
  shows "rt1 \sqsubseteq dip rt2"
  unfolding rt_strictly_fresher_def using assms ..
lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]

lemma rt_strictly_fresherE [elim]:
  assumes "rt1 \sqsubseteq dip rt2"
  and "[ rt1 \sqsubseteq dip rt2; \neg(rt1 \approx dip rt2) ] \Longrightarrow P rt1 rt2 i"
  shows "P rt1 rt2 i"
  using assms(1) unfolding rt_strictly_fresher_def
  by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
  "rt1 \sqsubseteq dip rt2 =
    (nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i))
    \lor (nsqn_r (the (rt1 i)) = nsqn_r (the (rt2 i)) \land \pi_5 (the (rt1 i)) > \pi_5 (the (rt2 i))))"
unfolding rt_strictly_fresher_def' rt_fresher_def freshness_def by auto

lemma rt_strictly_fresherD [dest]:
assumes "rt1 ⊑ dip rt2"
  shows "the (rt1 dip) ⊑ the (rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
  assumes "rt1 ⊑ dip rt2"
  shows "¬ rt1 ≈ dip rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt3"
  shows "rt1 ⊑ dip rt3"
using assms proof -
  from rt1 ⊑ dip rt2: obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
  also from rt2 ⊑ dip rt3: obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
  finally have "the (rt1 dip) ⊑ the (rt3 dip)".
moreover have "¬ (rt1 ≈ dip rt3)"
proof -
  from rt1 ⊑ dip rt2: obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
  also from rt2 ⊑ dip rt3: obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
  finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))".
  thus ?thesis ..
  qed
ultimately show "rt1 ⊑ dip rt3"..
  qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊑ dip rt)"
unfolding rt_strictly_fresher_def by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt3"
  shows "rt1 ⊑ dip rt3"
proof -
  from rt2 ⊑ dip rt3: have "rt2 ⊑ dip rt2"
    and "¬(rt2 ⊑ dip rt1)"
    unfolding rt_strictly_fresher_def'' by auto
  from this(1) and ⟨rt2 ⊑ dip rt3⟩ have "rt1 ⊑ dip rt3"..
moreover from "¬(rt2 ⊑ dip rt1)" have "¬(rt3 ⊑ dip rt1)"
proof (rule contrapos nn)
  assume "rt3 ⊑ dip rt1"
  with ⟨rt2 ⊑ dip rt3⟩ show "rt2 ⊑ dip rt1"..
  qed
ultimately show "rt1 ⊑ dip rt3"
unfolding rt_strictly_fresher_def'' by auto
  qed

lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt3"
  shows "rt1 ⊑ dip rt3"
proof -
  from rt2 ⊑ dip rt3: have "rt2 ⊑ dip rt3"
    and "¬(rt3 ⊑ dip rt2)"
    unfolding rt_strictly_fresher_def'' by auto
  from ⟨rt1 ⊑ dip rt2⟩ and this(1) have "rt1 ⊑ dip rt3"..
moreover from "¬(rt3 ⊑ dip rt2)" have "¬(rt3 ⊑ dip rt1)"
proof (rule contrapos nn)
assume "rt3 ⊆ dip rt1"
thus "rt3 ⊆ dip rt2" using ⟨rt1 ⊆ dip rt2⟩ ..
qed

ultimately show "rt1 ⊆ dip rt3"
unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_imp_nsqn_le:
assumes "rt1 ⊆ dip rt2"
and "ip ∈ kD rt1"
and "ip ∈ kD rt2"
shows "nsqn rt1 ip ≤ nsqn rt2 ip"
using assms(1)
by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

lemma rt_strictly_fresher_ltI [intro]:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "nsqn rt1 dip < nsqn rt2 dip"
shows "rt1 ⊏ dip rt2"
proof
  from assms show "rt1 ⊆ dip rt2" ..
next
  show "¬ (rt1 ≈ dip rt2)"
  proof
    assume "rt1 ≈ dip rt2"
hence "rt2 ⊆ dip rt1" ..
hence "nsqn rt2 dip ≤ nsqn rt1 dip"
    using ⟨dip ∈ kD(rt2)⟩ ⟨dip ∈ kD(rt1)⟩
    by (rule rt_fresher_imp_nsqn_le)
with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
      by simp
  qed
qed

lemma rt_strictly_fresher_eqI [intro]:
assumes "i ∈ kD(rt1)"
and "i ∈ kD(rt2)"
and "nsqn rt1 i = nsqn rt2 i"
and "π5(the (rt2 i)) < π5(the (rt1 i))"
shows "rt1 ⊏ i rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtssf_left [simp]:
".dests dip rt rt'. dests dip = None ⇒ ( inval rt dests ⊆ dip rt') = (rt ⊆ dip rt')"
unfolding invalidate_def rt_strictly_fresher_def'
by (rule iffI) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
assumes "dip ∈ vD(invalidate rt1 dests)"
shows "( inval rt1 dests ⊆ dip rt2) = (rt1 ⊆ dip rt2)"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∈ dom(dests)"
hence "dip ∉ vD(invalidate rt1 dests)"
  unfolding invalidate_def vD_def
  by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
with ⟨dip ∈ vD(invalidate rt1 dests)⟩ show ?thesis by simp
next
  assume "dip ∉ dom(dests)"
  hence "dests dip = None" by auto
moreover with ⟨dip ∈ vD(invalidate rt1 dests)⟩ have "dip ∈ vD(rt1)"
  unfolding invalidate_def vD_def
  by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
ultimately show \( ?\text{thesis} \)

unfolding invalidate_def rt_strictly_fresher_def' by auto

qed

lemma rt_strictly_fresher_update_other [elim!]:
"\( \forall \text{ dip ip r rt'. [ dip \neq ip; rt \sqsubseteq dip rt'] } \Rightarrow \) update ip r r' dip rt''\)

unfolding rt_strictly_fresher_def' by clarsimp

lemma addpreRT_strictly_fresher [simp]:
assumes "\( \text{ dip } \in \text{ kD(rt)} \)"
shows "\( \text{(the (addpreRT dip dip npre) } \sqsubseteq \text{ ip rt2)} = (\text{ rt dip rt2)} \)"
using assms unfolding rt_strictly_fresher_def' by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
assumes "\( \text{ dip } \in \text{ vD (rt2 nhip)} \)"
and *: "\( \text{ osn } < \text{ sqn (rt2 nhip) dip} \)"
and **: "\( \text{ rt } \neq \text{ update rt dip (osn, kno, val, hops, nhip, \{\})} \)"
shows "\( \text{ update rt dip (osn, kno, val, hops, nhip, \{\}) } \sqsubseteq \text{ dip rt2 nhip} \)"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
from ** have "\( \text{ nsqn (update rt dip (osn, kno, val, hops, nhip, \{\})}) dip = osn} \)"
  by (rule nsqn_update_changed_kno_val)
with \( \text{ dip } \in \text{ vD(rt2 nhip)} \)
  have "\( \text{ nsqn, (the (update rt dip (osn, kno, val, hops, nhip, \{\}) dip) = osn} \)"
  by (simp add: kD_nsqn)
also have "\( \text{ osn } < \text{ sqn (rt2 nhip) dip} \)" by (rule *)
also have "\( \text{ sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))} \)"
unfolding nsqn, def using \( \text{ dip } \in \text{ vD (rt2 nhip)} \)
by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "\( \text{ nsqn, (the (update rt dip (osn, kno, val, hops, nhip, \{\}) dip)) < nsqn, (the (rt2 nhip dip))} \)".
qed

lemma dhops_le_hops_imp_update_strictly_fresher:
assumes "\( \text{ dip } \in \text{ vD(rt2 nhip)} \)"
and sqn: "\( \text{ sqn (rt2 nhip) dip = osn} \)"
and hop: "\( \text{ the (dhops (rt2 nhip) dip) } \leq \text{ hops}\)"
and **: "\( \text{ rt } \neq \text{ update rt dip (osn, kno, val, Suc hops, nhip, \{\})} \)"
shows "\( \text{ update rt dip (osn, kno, val, Suc hops, nhip, \{\}) } \sqsubseteq \text{ dip rt2 nhip} \)"
unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
from ** have "\( \text{ nsqn (update rt dip (osn, kno, val, Suc hops, nhip, \{\})}) dip = osn} \)"
  by (rule nsqn_update_changed_kno_val)
with \( \text{ dip } \in \text{ vD (rt2 nhip)} \)
  have "\( \text{ nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, \{\}) dip) = osn} \)"
  by (simp add: kD_nsqn)
also have "\( \text{ osn } = \text{ sqn (rt2 nhip) dip} \)" by (rule sqn [symmetric])
also have "\( \text{ sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))} \)"
unfolding nsqn, def using \( \text{ dip } \in \text{ vD (rt2 nhip)} \)
by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "\( \text{ nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, \{\}) dip)) = nsqn, (the (rt2 nhip dip))} \)".
next
have "\( \text{ the (dhops (rt2 nhip) dip) } \leq \text{ hops}\)" by (rule hop)
also have "\( \text{ hops < hops + 1}\)" by simp
also have "\( \text{ hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, \{\}) dip)}\)"
  using ** by simp
finally have "\( \text{ the (dhops (rt2 nhip) dip) < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, \{\}) dip)}\)".
thus "\( \pi_5 (\text{ the (rt2 nhip dip)}) < \pi_5 (\text{ the (update rt dip (osn, kno, val, Suc hops, nhip, \{\}) dip)})\)"
using \( \text{ dip } \in \text{ vD (rt2 nhip)} \) by (simp add: proj5_eq_dhops)
qed

lemma nsqn_invalidate:
1.6 Invariant proofs on individual processes

theory A_Seq_Invariants
imports A2W.Invariants A_Aodv A_Aodv_Data A_Aodv_Predicates A_Fresher
begin

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

lemma sequence_number_increases:
  "paodv i |=A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
by inv_cterms

lemma sequence_number_one_or_bigger:
  "paodv i |= onl Γ_AODV (λ(ξ, _). 1 ≤ sn ξ)"
by (rule onll_step_to_invariantI [OF sequence_number_increases])
(auto simp: σAODV_def)

We can get rid of the onl/onll if desired...

lemma sequence_number_increases':
  "paodv i |= (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

lemma sequence_number_one_or_bigger':
  "paodv i |= (λ(ξ, _). 1 ≤ sn ξ)"
by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

lemma sip_in_kD:
  "paodv i |= onl Γ_AODV (λ(ξ, l). l ∈ {PAodv-:7} ∪ {PAodv-:5} ∪ {PRrep-:0..PRrep-:1}
                   ∪ {PRreq-:0..PRreq-:3})  −→ sip ξ ∈ kD (rt ξ)"
by inv_cterms

lemma rrep_1_update_changes:
  "paodv i |= onl Γ_AODV (λ(ξ, l). (l = PRrep-:1 −→ rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})))"
by inv_cterms

lemma addpreRT_partly_welldefined:
  "paodv i |= onl Γ_AODV (λ(ξ, l). (l ∈ {PRreq-:16..PRreq-:18} ∪ {PRrep-:2..PRrep-:6} −→ dip ξ ∈ kD (rt ξ))
                       ∧ (l ∈ {PRreq-:3..PRreq-:17} −→ oip ξ ∈ kD (rt ξ)))"
by inv_cterms

Proposition 7.38

lemma includes_nhip:
  "paodv i |= onl Γ_AODV (λ(ξ, l). ∀ dip ∈ kD(rt ξ). the (nhop (rt ξ) dip) ∈ kD(rt ξ))"
proof -
{ fix ip and $\xi', \xi :: \text{state} $
    assume "$\forall \text{dip} \in \text{kD (rt $\xi$)}. \text{the (nhop (rt $\xi$) dip) } \in \text{kD (rt $\xi$)}$
    and "$\xi' = (\text{rt := update (rt $\xi$) ip (0, unk, val, Suc 0, ip, {}}))$
    hence "$\forall \text{dip} \in \text{kD (rt $\xi$)}. \text{the (nhop (rt $\xi$) dip) } \in \text{kD (rt $\xi$)}$
    by clarsimp (metis nhop_update_unk_val update_another)
  }

note one_hop = this
{ fix ip sip sn hops and $\xi', \xi :: \text{state}$
    assume "$\forall \text{dip} \in \text{kD (rt $\xi$)}. \text{the (nhop (rt $\xi$) dip) } \in \text{kD (rt $\xi$)}$
    and "$\xi' = (\text{rt := update (rt $\xi$) ip (sn, kno, val, Suc hops, sip, {}}) |$
    and "$\text{sip} \in \text{kD (rt $\xi$)}$
    hence "$\forall \text{dip} \in \text{kD (rt $\xi$)}. \text{the (nhop (update (rt $\xi$) ip (sn, kno, val, Suc hops, sip, {})) dip) } = \text{ip} 
    \lor \text{the (nhop (update (rt $\xi$) ip (sn, kno, val, Suc hops, sip, {})) dip) } \in \text{kD (rt $\xi$)}$
    by (metis kD_update_unchanged nhop_update_changed update_another)
  }

note nhip_is_sip = this
show ?thesis by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf sip_in_kD] 
  onl_invariant_sterms [OF aodv_wf addpreRT_partly_welldefined] 
  solve: one_hop nhip_is_sip)
qed

Proposition 7.22: needed in Proposition 7.4

lemma addpreRT_welldefined:
  "paodv i ||= onl $\Gamma_{AODV}$ ($\lambda(\xi, p). (l \in \{PRreq-:16..PRreq-:18\} \rightarrow \text{dip } \xi \in \text{kD (rt $\xi$)} \land 
  (l = PRreq-:17 \rightarrow \text{oip } \xi \in \text{kD (rt $\xi$)} \land 
  (l = PRrep-:5 \rightarrow \text{dip } \xi \in \text{kD (rt $\xi$)} \land 
  (l = PRrep-:6 \rightarrow \text{(the (nhop (rt $\xi$) (dip $\xi$))) } \in \text{kD (rt $\xi$)})")

(is 
  "||= onl $\Gamma_{AODV}$ ?P")"
unfolding invariant_def
proof
  fix s
  assume "s \in \text{reachable (paodv i) TT}"
  then obtain $\xi$ p where "s = ($\xi$, p)"
  and 
  "($\xi$, p) \in \text{reachable (paodv i) TT}"
  by (metis prod.exhaust)
  have "onl $\Gamma_{AODV}$ ?P ($\xi$, p)"
  proof (rule onlII)
    fix l
    assume "1 \in \text{labels } \Gamma_{AODV} p"
    with ($\xi$, p) \in \text{reachable (paodv i) TT}
    have I1: "1 \in \{PRreq-:16..PRreq-:18\} \rightarrow \text{dip } \xi \in \text{kD (rt $\xi$)}"
    and I2: "1 = PRreq-:17 \rightarrow \text{oip } \xi \in \text{kD (rt $\xi$)}"
    and I3: "1 \in \{PRrep-:2..PRrep-:6\} \rightarrow \text{dip } \xi \in \text{kD (rt $\xi$)}"
    by (auto dest!: invariantD [OF addpreRT_partly_welldefined])
    moreover from ($\xi$, p) \in \text{reachable (paodv i) TT} 1 \in \text{labels } \Gamma_{AODV} p \and I1
    have "1 = PRrep-:6 \rightarrow \text{(the (nhop (rt $\xi$) (dip $\xi$))) } \in \text{kD (rt $\xi$)}"
    by (auto dest!: invariantD [OF includes_nhip])
    ultimately show "?P ($\xi$, 1)"
    by simp
    with s = ($\xi$, p) show "onl $\Gamma_{AODV}$ ?P s"
    by simp
  qed
qed

Proposition 7.4

lemma known_destinations_increase:
  "paodv i ||= onl $\Gamma_{AODV}$ ($\lambda((\xi, _), _, (\xi', _)). \text{kD (rt $\xi$) } \subseteq \text{kD (rt $\xi'$)}")"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined] 
  simp add: subset_insertI)
Proposition 7.5

```
lemma rreqs_increase:
  "paodv i |=_A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). rreqs ξ ⊆ rreqs ξ')"
by (inv_cterms simp add: subset_insertI)
```

```
lemma dests_bigger_than_sqn:
  "paodv i |= onl Γ_AODV (λ((ξ, _), _). ξ ⊆ rreqs ξ) → (∀ ip∈dom(dests ξ). ip∈kD(rt ξ) ∧ sqn (rt ξ) ip ≤ the (dests ξ ip)))"
proof -
  have sqninv:
    "∀ dests rt rsn ip.
      (∀ ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip)) → sqn (invalidate rt (dests ξ ι ip)) ip ≤ rsn"
    by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto

  have indests:
    "∀ dests rt rsn ip.
      (∀ ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip)); dests ip = Some rsn
      → ip∈kD(rt) ∧ sqn rt ip ≤ rsn"
    by (metis domI option.sel)

  show ?thesis
    by inv_cterms (clarsimp split: if_split_asm option.split_asm elim!: sqninv indests)+
qed
```

 Proposition 7.6

```
lemma sqns_increase:
  "paodv i |=_A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). ∀ ip. sqn (rt ξ) ip ≤ sqn (rt ξ') ip)"
proof -
  { fix ξ :: state
    assume *: "∀ ip∈dom(dests ξ). ip∈kD(rt ξ) ∧ sqn rt ξ ip ≤ the (dests ξ ip)"
    have "∀ ip. sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ ip))" by (clarsimp split: if_split_asm option.split_asm elim!: sqninv indests)+
    qed
  } note solve_invalidate = this

  show ?thesis
    by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]
      onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn]
      simp add: solve_invalidate)
qed
```

 Proposition 7.7

```
lemma ip_constant:
  "paodv i |=_A onl Γ_AODV (λ(ξ, _). ip ξ = i)"
by (inv_cterms simp add: σ_AODV_def)
```

 Proposition 7.8

```
lemma sender_ip_valid':
  "paodv i |=_A onll Γ_AODV (λ((ξ, _), a, _). anycast (λm. not_Pkt m −→ msg_sender m = ip ξ) a)"
by inv_cterms
```

```
lemma sender_ip_valid:
  "paodv i |=_A onll Γ_AODV (λ((ξ, _), a, _). anycast (λm. not_Pkt m −→ msg_sender m = i) a)"
by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
```
lemma received_msg_inv:
  \[\text{paodv } i \models (\text{recvmsg } P \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, 1). \ell \in \{\text{PAodv} \vdash \ell \mapsto P \text{ (msg } \xi)\})\]
  by \text{ inv terms}

Proposition 7.9

lemma sip_not_ip' :
  \[\text{paodv } i \models (\text{recvmsg } (\lambda m. \text{ not_Pkt } m \rightarrow \text{ msg_sender } m \neq i) \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, \_). \text{ sip } \xi \neq \text{ ip } \xi)\]
  by \text{(inv terms) inv add: onl_invariant_sterms \{OF aodv_wf received_msg_inv\}
  onl_invariant_sterms \{OF aodv_wf ip_constant \{THEN invariant_restrict_inD\}\]
  simp add: clear_locals_sip_not_ip' clarsimp+

lemma sip_not_ip:
  \[\text{paodv } i \models (\text{recvmsg } (\lambda m. \text{ not_Pkt } m \rightarrow \text{ msg_sender } m \neq i) \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, \_). \text{ sip } \xi \neq i)\]
  by \text{(inv terms) inv add: onl_invariant_sterms \{OF aodv_wf received_msg_inv\}
  onl_invariant_sterms \{OF aodv_wf ip_constant \{THEN invariant_restrict_inD\}\]
  simp add: clear_locals_sip_not_ip' clarsimp+

Neither sip_not_ip' nor sip_not_ip is needed to show loop freedom.

Proposition 7.10

lemma hop_count_positive:
  \[\text{paodv } i \models \text{ onl } \Gamma_{AODV} (\lambda(\xi, \_). \forall ip \in kD (rt \xi). \text{ the (dhops } (rt \xi) \text{ ip) } \geq 1)\]
  by \text{ (inv terms) inv add: onl_invariant_sterms \{OF aodv_wf addpreRT_welldefined\}\auto}

lemma rreq_dip_in_vD Dip_eq_ip:
  \[\text{paodv } i \models \text{ onl } \Gamma_{AODV} (\lambda(\xi, 1). (1 \in \{\text{PRreq-:16..PRreq-:18} \rightarrow) \text{ dip } \xi \in vD(rD \xi))\]
  \[\wedge (1 \in \{\text{PRreq-:5, PRreq-:6} \rightarrow) \text{ dip } \xi = \text{ ip } \xi)\]
  \[\wedge (1 \in \{\text{PRreq-:15..PRreq-:18} \rightarrow) \text{ dip } \xi \neq \text{ ip } \xi))\]

proof \text{(inv terms, elim conjE)\auto}
  fix \ell \in \pi p p'
  assume "(\ell, \pi) \in \text{ reachable (paodv } i) \text{ TT}"
  and "\{\text{PRreq-:17}\}[\lambda(\xi. \pi[rt := \text{ the (addpreRT } (rt \xi) \text{ (oip } \xi) \text{ the (nhop } (rt \xi) \text{ (dip } \xi)))])\} p' \in \text{ stterms } \Gamma_{AODV} \pi"
  and "1 = \text{PRreq-:17}"
  and "\text{dip } \xi \in vD (rt \xi)"
  from this(1-3) have "\text{oip } \xi \in kD (rt \xi)"
  by \text{(auto dest: onl_invariant_sterms \{OF aodv_wf addpreRT_welldefined, where l="PRreq-:17"\}}
  with \text{dip } \xi \in vD (rt \xi)\)
  show "\text{dip } \xi \in vD (the (addpreRT } (rt \xi) \text{ (oip } \xi) \text{ the (nhop } (rt \xi) \text{ (dip } \xi)))\)" by simp
dqd

Proposition 7.11

lemma anycast_msg_zhops:
  "\text{\backslash rreqid dip dsn dsqk oip osn sip.}
  \text{paodv } i \models \text{ onl } \Gamma_{AODV} (\lambda(\_, a, \_). \text{ anycast msg_zhops } a)"
proof \text{(inv terms) inv add:}
  \[\text{onl_invariant_sterms \{OF aodv_wf rreq_dip_in_vD Dip_eq_ip \{THEN invariant_restrict_inD\}\}
  \text{onl_invariant_sterms \{OF aodv_wf hop_count_positive \{THEN invariant_restrict_inD\}\}, \text{elim conjE)\auto}
  fix \ell \in \pi a p p' p'\}
  assume "(\ell, \pi) \in \text{ reachable (paodv } i) \text{ TT}"
  and "\{\text{PRreq-:18}\} \text{unicast}(\lambda(\xi. \text{ the (nhop } (rt \xi) \text{ (oip } \xi))),
  \lambda(\xi. \text{ Rrep } (\text{the (dhops } (rt \xi) \text{ (dip } \xi))) (\text{dip } \xi) (\text{sqn } (rt \xi) \text{ (dip } \xi)) (\text{oip } \xi) (\text{ip } \xi)).
  p' \mapsto pp' \in \text{ stterms } \Gamma_{AODV} \pi"
  and "1 = \text{PRreq-:18}"
  and "a = \text{unicast } (\text{the (nhop } (rt \xi) \text{ (oip } \xi)))
  (\text{Rrep } (\text{the (dhops } (rt \xi) \text{ (dip } \xi))) (\text{dip } \xi) (\text{sqn } (rt \xi) \text{ (dip } \xi)) (\text{oip } \xi) (\text{ip } \xi))"
  and "\forall ip \in kD (rt \xi). \text{ Suc } 0 \leq \text{ (the (dhops } (rt \xi) \text{ ip})"
  and "\text{dip } \xi \in vD (rt \xi)"
  from (\text{dip } \xi \in vD (rt \xi)) have "\text{dip } \xi \in kD (rt \xi)"
  by \text{(rule vD_id_gives_kD(1))}
  with * have "\text{Suc } 0 \leq \text{ (the (dhops } (rt \xi) \text{ (dip } \xi))" ..
thus "$0 < \text{the (dhops (rt } \xi ) \text{ (dip } \xi ))\)" by simp

qed

lemma hop_count_zero_oip_dip_sip:
"paodv i \models (\text{recvmsg msg_zhops } \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, 1).
\begin{align*}
(\lambda \in \{\text{PAodv-:4..PAodv-:5} \cup \{\text{PRreq-:n\|n. True}\} & \rightarrow \\
\text{hops } \xi = 0 \rightarrow \text{oip } \xi = \text{sip } \xi) \\
\land & \\
((\lambda \in \{\text{PAodv-:6..PAodv-:7} \cup \{\text{PRrep-:n\|n. True}\} & \rightarrow \\
\text{hops } \xi = 0 \rightarrow \text{dip } \xi = \text{sip } \xi))))
\end{align*}
\)

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:
"paodv i \models (\text{recvmsg rreq_rrep_sn } \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, 1).
\begin{align*}
\lambda (\xi, l). (l \in \{\text{PAodv-:4, PAodv-:5} \cup \{\text{PRreq-:n\|n. True}\} & \rightarrow \\
1 \leq \text{osn } \xi)
\end{align*}
\)

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq':
"paodv i \models (\text{recvmsg (\lambda m. rreq_rrep_sn m \& msg_zhops m) } \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, 1).
\begin{align*}
\lambda (\xi, l). (l \in \{\text{PAodv-:4, PAodv-:5} \cup \{\text{PRreq-:n\|n. True}\} & \rightarrow \\
1 \leq \text{osn } \xi)
\end{align*}
\)

proof (rule invariant_weakenE [OF osn_rreq])

fix a

assume "\text{recvmsg (\lambda m. rreq_rrep_sn m \& msg_zhops m) a}"

thus "\text{recvmsg rreq_rrep_sn a}"

by (cases a) simp_all

qed

lemma dsn_rrep:
"paodv i \models (\text{recvmsg rreq_rrep_sn } \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, 1).
\begin{align*}
\lambda (\xi, l). (l \in \{\text{PAodv-:6, PAodv-:7} \cup \{\text{PRreq-:n\|n. True}\} & \rightarrow \\
1 \leq \text{dsn } \xi)
\end{align*}
\)

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma dsn_rrep':
"paodv i \models (\text{recvmsg (\lambda m. rreq_rrep_sn m \& msg_zhops m) } \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, 1).
\begin{align*}
\lambda (\xi, l). (l \in \{\text{PAodv-:6, PAodv-:7} \cup \{\text{PRreq-:n\|n. True}\} & \rightarrow \\
1 \leq \text{dsn } \xi)
\end{align*}
\)

proof (rule invariant_weakenE [OF dsn_rrep])

fix a

assume "\text{recvmsg (\lambda m. rreq_rrep_sn m \& msg_zhops m) a}"

thus "\text{recvmsg rreq_rrep_sn a}"

by (cases a) simp_all

qed

lemma hop_count_zero_oip_dip_sip':
"paodv i \models (\text{recvmsg (\lambda m. rreq_rrep_sn m \& msg_zhops m) } \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, 1).
\begin{align*}
\lambda (\xi, l). (l \in \{\text{PAodv-:4..PAodv-:5} \cup \{\text{PRreq-:n\|n. True}\} & \rightarrow \\
\text{hops } \xi = 0 \rightarrow \text{oip } \xi = \text{sip } \xi) \\
\land & \\
((\lambda \in \{\text{PAodv-:6..PAodv-:7} \cup \{\text{PRrep-:n\|n. True}\} & \rightarrow \\
\text{hops } \xi = 0 \rightarrow \text{dip } \xi = \text{sip } \xi))))
\end{align*}
\)

proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])

fix a

assume "\text{recvmsg (\lambda m. rreq_rrep_sn m \& msg_zhops m) a}"

thus "\text{recvmsg msg_zhops a}"

by (cases a) simp_all

qed

Proposition 7.12

lemma zero_seq_unk_hops_one':
"paodv i \models (\text{recvmsg (\lambda m. rreq_rrep_sn m \& msg_zhops m) } \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, _).
\begin{align*}
\forall dip \in kD(rt \xi). (\text{sqn (rt } \xi ) \text{ dip } = 0 \rightarrow \text{sqnf (rt } \xi ) \text{ dip } = \text{unk) \\
\land & \\
\text{(sqnf (rt } \xi ) \text{ dip } = \text{unk} \rightarrow \text{the (dhops (rt } \xi ) \text{ dip } = 1) \\
\land & \\
\text{(the (dhops (rt } \xi ) \text{ dip } = 1 \rightarrow \text{the (nhop (rt } \xi ) \text{ dip } = \text{dip}))}
\end{align*}
\)

proof -

\{ fix dip and \xi :: state and P
assume "sqn (invalidate (rt ξ) (dests ξ)) dip = 0"
and all: "∀ ip. sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip"
and *: "sqn (rt ξ) dip = 0 =⇒ P ξ dip"

have "P ξ dip"

proof -
  from all have "sqn (rt ξ) dip ≤ sqn (invalidate (rt ξ) (dests ξ)) dip" ..
  with (sqn (invalidate (rt ξ) (dests ξ)) dip = 0) have "sqn (rt ξ) dip = 0" by simp
  thus "P ξ dip" by (rule *)
qed

} note sqn_invalidate_zero [elim!] = this

{ fix dsn hops :: nat and sip oip rt and ip dip :: ip
assume "∀ dip ∈ kD(rt).
  (sqn rt dip = 0 =⇒ π 3 (the (rt dip)) = unk) ∧
  (π 3 (the (rt dip)) = unk =⇒ the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 =⇒ the (nhop rt dip) = dip)"
and "hops = 0 =⇒ sip = dip"
and "Suc 0 ≤ dsn"
and "ip ≠ dip =⇒ ip ∈ kD(rt)"

hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0 =⇒
  the (nhop (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = ip"
by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
}

note prreq_ok1 [simp] = this

{ fix ip dsn hops sip oip rt dip
assume "∀ dip ∈ kD(rt).
  (sqn rt dip = 0 =⇒ π 3 (the (rt dip)) = unk) ∧
  (π 3 (the (rt dip)) = unk =⇒ the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 =⇒ the (nhop rt dip) = dip)"
and "Suc 0 ≤ dsn"
and "ip ≠ dip =⇒ ip ∈ kD(rt)"

hence "π 3 (the (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip)) = unk =⇒
  the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0"
by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
}

note prreq_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip
assume "∀ dip ∈ kD(rt).
  (sqn rt dip = 0 =⇒ π 3 (the (rt dip)) = unk) ∧
  (π 3 (the (rt dip)) = unk =⇒ the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 =⇒ the (nhop rt dip) = dip)"

and "Suc 0 ≤ dsn"
and "ip ≠ dip =⇒ ip ∈ kD(rt)"

hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip = 0 =⇒
  π 3 (the (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip)) = unk"
by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
}

note prreq_ok3 [simp] = this

{ fix rt sip
assume "∀ dip ∈ kD rt.
  (sqn rt dip = 0 =⇒ π 3 (the (rt dip)) = unk) ∧
  (π 3 (the (rt dip)) = unk =⇒ the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 =⇒ the (nhop rt dip) = dip)"

hence "∀ dip ∈ kD rt.
  (sqn (update rt sip (0, unk, val, Suc 0, sip, {})) dip = 0 =⇒
   π 3 (the (update rt sip (0, unk, val, Suc 0, sip, {})) dip)) = unk)
∧ (π 3 (the (update rt sip (0, unk, val, Suc 0, sip, {})) dip)) = unk =⇒
   the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0)
∧ (the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0 =⇒
   the (nhop (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = dip)"
by - (rule update_cases, simp_all add: sqnf_def sqn_def)
}

note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "∀ sip rt.
\[ \pi_3(\text{the (update rt sip (0, unk, val, Suc 0, sip, \{\}) sip)}) = \text{unk} \rightarrow \\
\text{the (dhops (update rt sip (0, unk, val, Suc 0, sip, \{\}) sip)}) = \text{Suc 0} \]
by (rule update_cases) simp_all

have prreq_ok6 [simp]: "\(\setminus\) sip rt. 
\[ \text{sqn (update rt sip (0, unk, val, Suc 0, sip, \{} sip) = 0} \rightarrow \\
\pi_3 (\text{the (update rt sip (0, unk, val, Suc 0, sip, \{} sip)}) = \text{unk} \]
by (rule update_cases) simp_all

show ?thesis 
by (inv_cterms inv: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined] 
  onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip'] 
  seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans] 
  onl_invariant_sterms [OF aodv_wf osn_rreq'] 
  onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+

qed

lemma zero_seq_unk_hops_one: 
"paodv i ||= (recvmsg (\lambda m. rreq_rrep_sn m & msg_zhops m) \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, _). 
\forall dip \in kD(rt \xi). (sqn (rt \xi) dip = 0 \rightarrow (sqnf (rt \xi) dip = unk \\
\wedge \text{the (dhops (rt \xi) dip)}) = 1 \\
\wedge \text{the (nhop (rt \xi) dip) = dip})))" 
by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one: 
"paodv i ||= (recvmsg rreq_rrep_sn \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, l). 
\forall dip \in kD(rt \xi). \pi_3(\text{the (rt \xi dip)}) = \text{unk} \lor 
\pi_2(\text{the (rt \xi dip)}) = \text{Suc 0} \leq \pi_2(\text{the (rt \xi dip)}))" 
proof - 
{ fix sip rtdsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhp1 nhp2 pre1 pre2 
  assume "dsn1 = unk \lor Suc 0 \leq dsn2" 
  hence "\pi_3(\text{the (update rt sip (dsn1, dsk1, flag1, hops1, nhp1, pre1) sip)}) = unk \\
\lor Suc 0 \leq \text{sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhp2, pre2)) sip}" 
    unfolding update_def by (cases "dsn1 = unk") (clarsimp split: option.split)+ 
  } note fromsip [simp] = this

{ fix dip sip rtdsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhp1 nhp2 pre1 pre2 
  assume allkd: "\forall dip \in kD(rt \xi). \pi_3(\text{the (rt dip)}) = unk \lor Suc 0 \leq \text{sqn rt dip}" 
  and **: "dsn1 = unk \lor Suc 0 \leq dsn2" 
  have "\forall dip \in kD(rt \xi). \pi_3(\text{the (update rt sip (dsn1, dsk1, flag1, hops1, nhp1, pre1) dip)}) = unk \\
\lor Suc 0 \leq \text{sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhp2, pre2)) dip}" 
    (is "\forall dip \in kD(rt \xi). \pi_2(\text{prop dip})") 
    proof 
      fix dip 
      assume "dip \in kD(rt \xi)" 
      thus "?prop dip" 
      proof (cases "dip = sip") 
        assume "dip = sip" 
        with ** show ?thesis 
        by simp 
      next 
      assume "dip \neq sip" 
      with \(dip \in kD(rt \xi)\) allkd show ?thesis 
      by simp 
    qed 
  } note solve_update [simp] = this

{ fix dip rt dests 
  assume *: "\forall ip \in \text{dom(dests)} \in kD(rt \xi) \land \text{sqn rt ip} \leq \text{the (dests ip)}" 
  and **: "\forall dip \in kD(rt \xi). \pi_3(\text{the (rt ip)}) = unk \lor Suc 0 \leq \text{sqn rt ip}" 
  have "\forall dip \in kD(rt \xi). \pi_3(\text{the (rt dip)}) = unk \lor Suc 0 \leq \text{sqn (invalidate rt dests dip)}" 
    proof 
      fix dip 
      assume "dip \in kD(rt \xi)"
with ** have "π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip" ..
thus "π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
proof
  assume "π₃ (the (rt dip)) = unk" thus ?thesis ..
next
  assume "Suc 0 ≤ sqn rt dip"
  have "Suc 0 ≤ sqn (invalidate rt dests) dip"
  proof (cases "dip ∈ dom(dests)")
    assume "dip ∈ dom(dests)"
    with * have "sqn rt dip ≤ the (dests dip)" by simp
    with ⟨Suc 0 ≤ sqn rt dip⟩ have "Suc 0 ≤ the (dests dip)" by simp
    with ⟨dip ∈ dom(dests)⟩ ⟨dip ∈ kD(rt)⟩ [THEN kD_Some] show ?thesis
  unfolding invalidate_def sqn_def by auto
next
  assume "dip /∈ dom(dests)"
  with ⟨Suc 0 ≤ sqn rt dip⟩ ⟨dip ∈ kD(rt)⟩ [THEN kD_Some]
  show ?thesis unfolding invalidate_def sqn_def by auto
qed
thus ?thesis by (rule disjI2)
qed
qed

Proposition 7.13

lemma rreq_rrep_sn_any_step_invariant:
  "paodv i |=ₐ (recvmsg rreq_rrep_sn → onll Γ AODV (λ(_, a, _). anycast rreq_rrep_sn a))"
proof -
  have sqnf_kno: "paodv i |= onl Γ AODV (λ(ξ, l).
    (l ∈ {PRreq-:16..PRreq-:18} → sqnf (rt ξ) (dip ξ) = kno))"
  proof
    inv_cterms fix l ξ pp p'
    assume "(ξ, pp) ∈ reachable (paodv i) TT"

  show ?thesis
    by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
      onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn
        [THEN invariant_restrict_inD]]
      onl_invariant_sterms [OF aodv_wf osn_rreq]
      onl_invariant_sterms [OF aodv_wf dsn_rrep]
      simp add: proj3_inv proj2_eq_sqn)
  qed

Proposition 7.14

lemma rreq_rrep_fresh_any_step_invariant:
  "paodv i |=ₐ onll Γ AODV (λ((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a)"
proof -
  have rreq_oip: "paodv i |= onl Γ AODV (λ(ξ, 1).
    (1 ∈ {PRreq-:3, PRreq-:4, PRreq-:15, PRreq-:27} → oip ξ ∈ kD(rt ξ)
     ∧ (sqn (rt ξ) (oip ξ) > (osn ξ)
      ∨ (sqn (rt ξ) (oip ξ) = (osn ξ)
       ∧ the (dhops (rt ξ) (oip ξ)) ≤ Suc (hops ξ)
       ∧ the (flag (rt ξ) (oip ξ)) = val)))"
  proof inv_cterms
    fix 1 ξ l' pp p'
    assume "(ξ, pp) ∈ reachable (paodv i) TT"
and "\{PRreq-:2\} λξ. ξ⟨rt :=
update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, \{\})\} p' ∈ st Terms Γ_{AODV} pp"
and "1' = PRreq-:3" show "osn ξ < sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, \{\})) (oip ξ)
∨ (sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, \{\})) (oip ξ) ≠ osn ξ
∧ the (dhops (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, \{\})) (oip ξ)
≤ Suc (hops ξ)
∧ the (flag (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, \{\})) (oip ξ))
= val)"
unfolding update_def by (clarsimp split: option.split)
(metis linorder_neqE_nat not_less)
qed

have rrep_prrep: "paodv i |= onl Γ_{AODV} (λ(ξ, 1).
(l ∈ \{PRrep-:2..PRrep-:7\} → (dip ξ ∈ kD(rt ξ)
∧ sqn (rt ξ) (dip ξ) = dsn ξ
∧ the (dhops (rt ξ) (dip ξ)) = Suc (hops ξ)
∧ the (flag (rt ξ) (dip ξ)) = val
∧ the (nhop (rt ξ) (dip ξ)) ∈ kD (rt ξ)))"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rrep_1_update_changes]
onl_invariant_sterms [OF aodv_wf sip_in_kD])

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip]
onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip]
onl_invariant_sterms [OF aodv_wf rrep_prrep])

qed

Proposition 7.15

lemma rerr_invalid_any_step_invariant:
"paodv i |=_A onl Γ_{AODV} (λ(ξ, _), a, _). ancast (rerr_invalid (rt ξ)) a)"
proof -
have dests_inv: "paodv i |=
onl Γ_{AODV} (λ(ξ, 1). (l ∈ \{PAodv-:15, PPkt-:7, PRreq-:9,
PRreq-:21, PRrep-:10, PRerr-:1\}
→ (∀ ip∈dom(dests ξ). ip∈iD(rt ξ)))
∧ (l ∈ \{PAodv-:16..PAodv-:19\}
∪ \{PPkt-:8..PPkt-:11\}
∪ \{PRreq-:10..PRreq-:13\}
∪ \{PRreq-:22..PRreq-:25\}
∪ \{PRrep-:11..PRrep-:14\}
∪ \{PRerr-:2..PRerr-:5\} → (∀ ip∈dom(dests ξ). ip∈iD(rt ξ)
∧ the (dests ξ ip) = sqn (rt ξ ip))
∧ (l = PPkt-:14 → dip ξ∈iD(rt ξ)))"
by inv_cterms (clarsimp split: if_split_asm option.split_asm simp: domIff)+
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_inv])

 qed

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs (rip, rsn) is assigned to the variable dests of type ip → sqn, or to the first argument of the function rerr, this set is a partial function, i.e., there is at most one entry (rip, rsn) for each destination rip: guaranteed by type.

lemma dests_vD_inc_sqn:
"paodv i |=
1.7 The quality increases predicate

theory A_Quality_Increases

imports A_Aodv_Predicates A_Fresher

begin

definition quality_increases :: ""("state ⇒ state ⇒ bool"" where "quality_increases ξ ξ' ≡ (∀ dip∈kD(rt ξ). dip∈kD(rt ξ') ∧ rt ξ ⊑ dip rt ξ')" ∧ (∀ dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip)"

lemma quality_increasesI [intro!]:

Proposition 7.27

lemma route_tables_fresher:

"paodv i |∼ A (recvmsg rreq rrep_sn → onll Γ AODV (λ ((ξ, _), _. (ξ', _))).
∀ dip∈kD(rt ξ). rt ξ ⊑ dip rt ξ')"

proof

fix ξ pp p'

assume "(ξ, pp) ∈ reachable (paodv i) (recvmsg rreq rrep_sn)"

and "(PReq-:2)]λξ. ξ[rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {}))]

"p' ∈ stems Γ AODV pp"

and "Suc 0 ≤ osn ξ"

and*: "∀ dip∈kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"

show "∀ dip∈kD (rt ξ). rt ξ ⊑ dip update (rt ξ) (dip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"

proof

fix ip

moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp

moreover from Suc 0 ≤ osn ξ

have "update_arg_wf (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"..

ultimately show "rt ξ ⊑ dip update (rt ξ) (dip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"

by (rule rt_fresher_update)

qed
assumes "⋀ dip. dip ∈ kD(rt ξ) ⇐⇒ dip ∈ kD(rt ξ')"
and "⋀ dip. [ dip ∈ kD(rt ξ); dip ∈ kD(rt ξ')] ⇒ rt ξ ⊑ dip rt ξ'"
and "⋀ dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip"
  shows "quality_increases ξ ξ'"
unfolding quality_increases_def using assms by clarsimp

lemma quality_increasesE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
  and "dip ∈ kD(rt ξ)"
  and "[ [ dip ∈ kD(rt ξ'); rt ξ ⊑ dip rt ξ'; sqn (rt ξ) dip ≤ sqn (rt ξ') dip ] ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rt_fresherD [dest]:
  fixes ip
  assumes "quality_increases ξ ξ'"
  and "ip ∈ kD(rt ξ)"
  shows "rt ξ ⊑ ip rt ξ'"
using assms by auto

lemma quality_increases_sqnE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
  and "sqn (rt ξ) dip ≤ sqn (rt ξ') dip ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases ξ ξ"
  by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
  fixes σ σ' dip
  assumes "rt (σ i) ⊑ dip rt (σ' nhip)"
  and qinc: "quality_increases (σ nhip) (σ' nhip)"
  and "dip ∈ kD(rt (σ' nhip))"
  shows "rt (σ i) ⊑ dip rt (σ' nhip)"
proof -
  from qinc have "rt (σ nhip) ⊑ dip rt (σ' nhip)" using ⟨dip ∈ kD(rt (σ' nhip))⟩
  by auto
  with ⟨rt (σ i) ⊑ dip rt (σ' nhip)⟩ show ?thesis ..
qed

lemma kD_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ')"
using assms by auto

lemma kD_nsqn_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ') ∧ nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
proof -
  from assms have "i ∈ kD(rt ξ')" ..
  moreover with assms have "rt ξ ⊑ i rt ξ'" by auto
  ultimately have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" using ⟨i ∈ kD(rt ξ)⟩ by (erule(2) rt_fresher_imp_nsqn_le)
  with ⟨i ∈ kD(rt ξ')⟩ show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
shows \( \text{nsqn (rt } \xi \text{)} i \leq \text{nsqn (rt } \xi' \text{)} i \)

using assms by (rule kD\_nsqn\_quality\_increases THEN conjunct2))

lemma kD\_nsqn\_quality\_increases\_trans [elim]:
assumes "\( i \in kD(\text{rt } \xi) \)"
and "\( s \leq \text{nsqn (rt } \xi \text{)} i \)"
and "\( \text{quality\_increases } \xi \xi' \)"
shows "\( i \in kD(\text{rt } \xi') \land s \leq \text{nsqn (rt } \xi' \text{)} i \)"

proof
from (\( i \in kD(\text{rt } \xi) \)) and (\( \text{quality\_increases } \xi \xi' \)) show "\( i \in kD(\text{rt } \xi') \)" ..

nextrom (\( i \in kD(\text{rt } \xi) \)) and (\( \text{quality\_increases } \xi \xi' \)) have "\( \text{nsqn (rt } \xi \text{)} i \leq \text{nsqn (rt } \xi' \text{)} i \)" ..
with (\( s \leq \text{nsqn (rt } \xi \text{)} i \)) show "\( s \leq \text{nsqn (rt } \xi' \text{)} i \)" by (rule le\_trans)
qed

lemma nsqn\_quality\_increases\_nsqn\_lt\_lt [elim]:
assumes "\( i \in kD(\text{rt } \xi) \)"
and "\( \text{quality\_increases } \xi \xi' \)"
and "\( s < \text{nsqn (rt } \xi \text{)} i \)"
shows "\( s < \text{nsqn (rt } \xi' \text{)} i \)"

proof -
from assms(1-2) have "\( \text{nsqn (rt } \xi \text{)} i \leq \text{nsqn (rt } \xi' \text{)} i \)" ..
with (\( s < \text{nsqn (rt } \xi \text{)} i \)) show "\( s < \text{nsqn (rt } \xi' \text{)} i \)" by simp
qed

lemma nsqn\_quality\_increases\_dhops [elim]:
assumes "\( i \in kD(\text{rt } \xi) \)"
and "\( \text{quality\_increases } \xi \xi' \)"
and "\( \text{nsqn (rt } \xi \text{)} i = \text{nsqn (rt } \xi' \text{)} i \)"
shows "\( \text{the (dhops (rt } \xi \text{)} i) \geq \text{the (dhops (rt } \xi' \text{)} i) \)"

using assms unfolding quality\_increases\_def
by (clarsimp) (drule(1) bspec, clarsimp simp: rt\_fresher\_def2)

lemma nsqn\_quality\_increases\_nsqn\_eq\_le [elim]:
assumes "\( i \in kD(\text{rt } \xi) \)"
and "\( \text{quality\_increases } \xi \xi' \)"
and "\( \text{nsqn (rt } \xi \text{)} i = \text{nsqn (rt } \xi' \text{)} i \)"
shows "\( \text{the (dhops (rt } \xi \text{)} i) \leq \text{the (dhops (rt } \xi' \text{)} i) \)"

using assms by (metis nat\_less\_le nsqn\_quality\_increases nsqn\_quality\_increases\_dhops)

lemma quality\_increases\_rreq\_rrep\_props [elim]:
fixes sn ip hops sip
assumes qinc: "\( \text{quality\_increases (} \sigma \text{ sip) (} \sigma' \text{ sip) }\)"
and "\( i \leq \text{sn} \)"
and *: "\( \text{ip} \in kD(\text{rt } (} \sigma \text{ sip)) \land \text{sn} \leq \text{nsqn (rt } (} \sigma \text{ sip)) \text{ ip} \)\n\land (\( \text{nsqn (rt } (} \sigma \text{ sip)) \text{ ip = sn}
\rightarrow (\text{the (dhops (rt } (} \sigma \text{ sip)) \text{ ip} \leq \text{hops}
\lor \text{the (flag (rt } (} \sigma \text{ sip)) \text{ ip) = inv}))\)"
shows "\( \text{ip} \in kD(\text{rt } (} \sigma' \text{ sip)) \land \text{sn} \leq \text{nsqn (rt } (} \sigma' \text{ sip)) \text{ ip} \)\n\land (\( \text{nsqn (rt } (} \sigma' \text{ sip)) \text{ ip = sn}
\rightarrow (\text{the (dhops (rt } (} \sigma' \text{ sip)) \text{ ip} \leq \text{hops}
\lor \text{the (flag (rt } (} \sigma' \text{ sip)) \text{ ip) = inv}))\)"

(is "\( i \leq \text{sn} \)"
\land ?nsqnafter")

proof -
from * obtain "\( \text{ip} \in kD(\text{rt } (} \sigma \text{ sip))\)" and "\( \text{sn} \leq \text{nsqn (rt } (} \sigma \text{ sip)) \text{ ip} \)" by auto

from quality\_increases (\( \sigma \text{ sip) (} \sigma' \text{ sip) }\)
have "\( \text{s q (rt } (} \sigma \text{ sip)) \text{ ip} \leq \text{s q (rt } (} \sigma' \text{ sip)) \text{ ip} \)" ..
from quality\_increases (\( \sigma \text{ sip) (} \sigma' \text{ sip) }\) and \( \text{ip} \in kD (\text{rt } (} \sigma \text{ sip))\)
have "\( \text{ip} \in kD (\text{rt } (} \sigma' \text{ sip))\)" ..
from (\( \text{sn} \leq \text{nsqn (rt } (} \sigma \text{ sip)) \text{ ip} \)) have ?nsqnafter
proof
assume "\( \text{sn} < \text{nsqn (rt } (} \sigma \text{ sip)) \text{ ip} \)"
also from \(\{ip \in kD(\text{rt } (\sigma \text{ sip}))\text{ and }\text{quality}\_\text{increases}(\sigma \text{ sip})(\sigma' \text{ sip})\}\) have "\(\ldots \leq \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\)" ..
finally have "sn < \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}" .
thus ?thesis by simp
next
assume "sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}"
with \(\{ip \in kD(\text{rt } (\sigma \text{ sip}))\text{ and }\text{quality}\_\text{increases}(\sigma \text{ sip})(\sigma' \text{ sip})\}\) have "sn < \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}"
\(\lor \text{ (sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}) \leq \text{ (dhops } (\text{rt } (\sigma' \text{ sip})) \text{ ip})\}"
.. hence "sn < \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}"
\(\lor \text{ (nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} = sn \land \text{ (dhops } (\text{rt } (\sigma' \text{ sip})) \text{ ip} \leq \text{ hops} \lor \text{ (flag } (\text{rt } (\sigma' \text{ sip})) \text{ ip} = \text{inv})\}"

proof
assume "sn < \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}" thus ?thesis ..
next
assume "sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}"
\(\land \text{ (dhops } (\text{rt } (\sigma \text{ sip})) \text{ ip}) \geq \text{ (dhops } (\text{rt } (\sigma' \text{ sip})) \text{ ip})\"

hence "sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}"
and "the (dhops (\text{rt } (\sigma' \text{ sip})) \text{ ip}) \leq \text{ (dhops } (\text{rt } (\sigma' \text{ sip})) \text{ ip})" by auto
from * and \(\{sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\}\) have "the (dhops (\text{rt } (\sigma' \text{ sip})) \text{ ip}) \leq \text{ hops} \lor \text{ (flag } (\text{rt } (\sigma' \text{ sip})) \text{ ip} = \text{inv})"

by simp
thus ?thesis

proof
assume "the (dhops (\text{rt } (\sigma' \text{ sip})) \text{ ip}) \leq \text{ hops}"
with \(\{\text{the } (\text{dhops } (\text{rt } (\sigma' \text{ sip})) \text{ ip}) \leq \text{ (dhops } (\text{rt } (\sigma' \text{ sip})) \text{ ip})\}\)

have "the (dhops (\text{rt } (\sigma' \text{ sip})) \text{ ip}) \leq \text{ hops}" by simp
with \(\{sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\}\) show ?thesis by simp
next
assume "the (\text{flag } (\text{rt } (\sigma \text{ sip})) \text{ ip}) = \text{inv}"
with \(\{\text{ip} \in kD(\text{rt } (\sigma \text{ sip}))\}\) have "\(\text{the } (\text{dhops } (\text{rt } (\sigma \text{ sip})) \text{ ip}) \leq \text{ hops} \lor \text{ (flag } (\text{rt } (\sigma \text{ sip})) \text{ ip}) = \text{inv})\"

by simp
thus ?thesis

from (rule vD_or_iD)
assume "ip \in iD(\text{rt } (\sigma' \text{ sip}))"

hence "the (\text{flag } (\text{rt } (\sigma' \text{ sip})) \text{ ip}) = \text{inv}" ..
with \(\{sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\}\) show ?thesis by simp
next
assume "ip \in vD(\text{rt } (\sigma' \text{ sip}))"

hence "\(\text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} = \text{sqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} - 1\)" ..

with \(\{sn \geq 1\land sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\}\)

have "\(\text{sqn } (\text{rt } (\sigma \text{ sip})) \text{ ip} > 1\)" by simp
from \(\{ip \in kD(\text{rt } (\sigma \text{ sip}))\}\) show ?thesis

proof (rule vD_or_iD)
assume "ip \in iD(\text{rt } (\sigma \text{ sip}))"

hence "the (\text{flag } (\text{rt } (\sigma \text{ sip})) \text{ ip}) = \text{inv}" ..
with \(\{sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\}\) show ?thesis by simp
next

assume "ip \in vD(\text{rt } (\sigma' \text{ sip}))"

hence "\(\text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} = \text{sqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\)" ..

with \(\{\text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} \leq \text{sqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\}\)

have "\(\text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} \geq \text{sqn } (\text{rt } (\sigma \text{ sip})) \text{ ip}\)" by simp
with \(\{\text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ ip} > 1\}\)

have "\(\text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} > \text{sqn } (\text{rt } (\sigma \text{ sip})) \text{ ip} - 1\)" by simp
with \(\{\text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} = \text{sqn } (\text{rt } (\sigma \text{ sip})) \text{ ip} - 1\}\)

have "\(\text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} > \text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ ip}\)" by simp
with \(\{sn = \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip}\}\) have "\(\text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ ip} > sn\)"

by simp
thus ?thesis ..
qed
qed
qed
thus ?thesis by (metis (mono_tags) le_cases not_le)
qed
with \(\{ip \in kD(\text{rt } (\sigma' \text{ sip}))\}\) show "ip \in kD(\text{rt } (\sigma' \text{ sip})) \land \text{?nsqnafter}" ..
qed

139
lemma quality_increases_rreq_rrep_props:
  fixes sn ip hops sip
  assumes "∀j. quality_increases (σ j) (σ' j)"
  and "1 ≤ sn"
  and *: "ip ∈ KD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip ∧ (nsqn (rt (σ sip)) ip = sn → (the (dhops (rt (σ sip)) ip)) ≤ hops ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip ∈ KD(rt (σ sip')) ∧ sn ≤ nsqn (rt (σ sip')) ip ∧ (nsqn (rt (σ sip')) ip = sn → (the (dhops (rt (σ sip')) ip)) ≤ hops ∨ the (flag (rt (σ sip')) ip) = inv))"
proof -
  from assms(1) have "quality_increases (σ sip) (σ sip')" ..
  thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props)
qed

lemma rteq_quality_increases:
  assumes "∀j. j ≠ i → quality_increases (σ j) (σ' j)"
  and "rt (σ' i) = rt (σ i)"
  shows "∀j. quality_increases (σ j) (σ' j)"
using assms by clarsimp (metis order_refl quality_increasesI rt_fresher_refl)
definition msg_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"
  where "msg_fresh σ m ≡
  case m of
    Rreq hopsc _ _ oipc osnc sipc ⇒ osnc ≥ 1 ∧ (sipc ≠ oipc → oipc ∈ KD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc ∧ (nsqn (rt (σ sipc)) oipc = osnc → (hopsc ≥ the (dhops (rt (σ sipc)) oipc) ∨ the (flag (rt (σ sipc)) oipc) = inv)))
  | Rrep hopsc dipc dsnc _ sipc ⇒ dsnc ≥ 1 ∧ (sipc ≠ dipc → dipc ∈ KD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc ∧ (nsqn (rt (σ sipc)) dipc = dsnc → (hopsc ≥ the (dhops (rt (σ sipc)) dipc) ∨ the (flag (rt (σ sipc)) dipc) = inv)))
  | Rerr destsc sipc ⇒ (∀ripc ∈ dom(destsc). (ripc ∈ KD(rt (σ sipc)) ∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))
  | _ ⇒ True"

lemma msg_fresh [simp]:
  "∀hops dip dsn dsk oip osn sip.
  msg_fresh σ (Rreq hops dip dsn dsk oip osn sip) =
  (osn ≥ 1 ∧ (sipc ≠ oipc → oipc ∈ KD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osn ∧ (nsqn (rt (σ sipc)) oipc = osn → (hopsc ≥ the (dhops (rt (σ sipc)) oipc) ∨ the (flag (rt (σ sipc)) oipc) = inv))))"
  "∀hops dip dsn oip sip. msg_fresh σ (Rreq hops dip dsn oip sip) =
  (dsn ≥ 1 ∧ (sipc ≠ dipc → dipc ∈ KD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsn ∧ (nsqn (rt (σ sipc)) dipc = dsn → (hopsc ≥ the (dhops (rt (σ sipc)) dipc) ∨ the (flag (rt (σ sipc)) dipc) = inv))))"
  "∀dests sipc. msg_fresh σ (Rerr dests sipc) =
  (∀ripc ∈ dom(dests). (ripc ∈ KD(rt (σ sipc)) ∧ the (dests ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))"
  "∀d dip. msg_fresh σ (Newpkt d dip) = True"
  "∀d dip sip. msg_fresh σ (Pkt d dip sip) = True"
unfolding msg_fresh_def by simp_all

lemma msg_fresh_inc_sn [simp, elim]:
  "msg_fresh σ m → rreq_rrep_sn m" by (cases m) simp_all
lemma recv_msg_fresh_inc_sn [simp, elim]:
"orecvmsg (msg_fresh) σ m \Rightarrow recvmsg rreq_rrep_sn m"
by (cases m) simp_all

lemma rreq_nsqn_is_fresh [simp]:
fixes σ msg hops dip dsn dip oip osn sip
assumes "rreq_rrep_fresh (rt (σ sip)) (Rreq hops dip dsn dip oip osn sip)"
and "rreq_rrep_sn (Rreq hops dip dsn dip oip osn sip)"
shows "msg_fresh σ (Rreq hops dip dsn dip oip osn sip)"
(is "msg_fresh σ ?msg")
proof -
let ?rt = "rt (σ sip)"
from assms(2) have "1 \leq osn" by simp
thus ?thesis
unfolding msg_fresh_def
proof (simp only: msg.case, intro conjI impI)
assume "sip \neq oip"
with assms(1) show "oip \in kD(?rt)" by simp
next
assume "sip \neq oip"
and "nsqn ?rt oip = osn"
show "the (dhops ?rt oip) \leq hops \vee the (flag ?rt oip) = inv"
proof (cases "oip \in vD(?rt)"
assume "oip \in vD(?rt)"
hence "nsqn ?rt oip = sqn ?rt oip" ..
with ⟨nsqn ?rt oip = osn⟩ have "sqn ?rt oip = osn" by simp
with assms(1) and ⟨sip \neq oip⟩ have "the (dhops ?rt oip) \leq hops"
by simp
thus ?thesis ..
next
assume "oip \notin vD(?rt)"
moreover from assms(1) and ⟨sip \neq oip⟩ have "oip \in kD(?rt)" by simp
ultimately have "oip \in iD(?rt)" by auto
hence "the (flag ?rt oip) = inv" ..
thus ?thesis ..
qed
next
assume "sip \neq oip"
with assms(1) have "osn \leq nsqn ?rt oip" by auto
thus "osn \leq nsqn (σ sip) oip" by auto
proof (rule nat_le_eq_or_lt)
assume "osn < nsqn ?rt oip"
hence "osn \leq nsqn ?rt oip - 1" by simp
also have "... \leq nsqn ?rt oip" by (rule sqn_nsqn)
finally show "osn \leq nsqn ?rt oip" .
next
assume "osn = nsqn ?rt oip"
with assms(1) and ⟨sip \neq oip⟩ have "oip \in kD(?rt)"
and "the (flag ?rt oip) = val"
by auto
hence "nsqn ?rt oip = sqn ?rt oip" ..
with ⟨osn = nsqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
thus "osn \leq nsqn ?rt oip" by simp
qed
qed simp

lemma rrep_nsqn_is_fresh [simp]:
fixes σ msg hops dip dsn dip oip osn sip
assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn dip oip osn sip)"
and "rreq_rrep_sn (Rrep hops dip dsn dip oip osn sip)"
shows "msg_fresh σ (Rrep hops dip dsn dip oip osn sip)"
(is "msg_fresh σ ?msg")
proof -
let \(?rt = \text{"rt (}\sigma\text{ sip)})"
from assms have "sip \(\neq\) dip \(\rightarrow\) dip \(\in\) kD(?rt) \(\wedge\) sqn ?rt dip = dsn \(\wedge\) the (flag ?rt dip) = val"
by simp
hence "sip \(\neq\) dip \(\rightarrow\) dip \(\in\) kD(?rt) \(\wedge\) nsqn ?rt dip \(\geq\) dsn"
by clarsimp
with assms show "msg_fresh \(\sigma\) ?msg"
by clarsimp
qed

lemma rerr_nsqn_is_fresh [simp]:
fixes \(\sigma\) \(\text{msg destinations sip}\)
assumes "rerr_invalid (rt (\(\sigma\) sip)) \(\text{(Rerr destinations sip)})"
shows "msg_fresh \(\sigma\) (\(\text{Rerr destinations sip})\)"
(is "msg_fresh \(\sigma\) ?msg")
proof
- let \(?rt = \text{"rt (}\sigma\text{ sip)})"
  from assms have 
  \[\forall \text{rip} \in \text{dom(destinations)}. \text{(rip} \in \text{iD(rt (}\sigma\text{ sip))})
  \wedge\) the (destinations rip) = sqn (rt (\(\sigma\) sip)) rip)"
  by clarsimp
  have 
  \[\forall \text{rip} \in \text{dom(destinations)}. \text{(rip} \in \text{kD(rt (}\sigma\text{ sip))})
  \wedge\) the (destinations rip) - 1 \(\leq\) nsqn (rt (\(\sigma\) sip)) rip)"
  proof
  fix rip
  assume "rip \in \text{dom destinations}"
  with \(*\) have "rip \(\in\) iD(rt (\(\sigma\) sip))" and "the (destinations rip) = sqn (rt (\(\sigma\) sip)) rip" by auto
  from this(2)
  show "rip \(\in\) kD(rt (\(\sigma\) sip))\) \(\wedge\) the (destinations rip) - 1 \(\leq\) nsqn (rt (\(\sigma\) sip)) rip)"
  by clarsimp
  qed
  thus "msg_fresh \(\sigma\) ?msg"
  by simp
  qed

lemma quality_increases_msg_fresh [elim]:
assumes qinc: "\(\forall j. \text{quality_increases (}\sigma j (\sigma' j))\)"
and "msg_fresh \(\sigma\) \(m\)"
shows "msg_fresh \(\sigma\) \(\text{m'})"
using assms(2)
proof (cases \(m\))
  fix hops rreqid dip dsn dsk oip osn sip
  assume [simp]: "\(m = \text{Rreq hops dip dsn dsk oip osn sip}\)"
  and "msg_fresh \(\sigma\) \(m\)"
  then have "osn \(\geq\) 1" and "sip = oip \(\lor\) (oip \(\in\) kD(rt (\(\sigma\) sip)) \(\wedge\) osn \(\leq\) nsqn (rt (\(\sigma\) sip)) oip
  \wedge\) nsqn (rt (\(\sigma\) sip)) oip = osn
  \(\rightarrow\) (the (dhops (rt (\(\sigma\) sip)) oip) \(\leq\) hops
  \lor\) the (flag (rt (\(\sigma\) sip)) oip) = inv)))"
  by auto
  from this(2) show \?thesis
  proof
    assume "sip = oip" with \(\text{osn} \geq 1\) show \?thesis by simp
    next
    assume "oip \(\in\) kD(rt (\(\sigma\) sip)) \(\wedge\) osn \(\leq\) nsqn (rt (\(\sigma\) sip)) oip
    \wedge\) nsqn (rt (\(\sigma\) sip)) oip = osn
    \(\rightarrow\) (the (dhops (rt (\(\sigma\) sip)) oip) \(\leq\) hops
    \lor\) the (flag (rt (\(\sigma\) sip)) oip) = inv))"
    moreover from qinc have "\(\text{quality_increases (}\sigma\) \(\sigma'\) sip)" ..
    ultimately have "oip \(\in\) kD(rt (\(\sigma'\) sip)) \(\wedge\) osn \(\leq\) nsqn (rt (\(\sigma'\) sip)) oip
ultimately
∧ (nsqn (rt (σ' sip)) oip = osn 
    → (the (dhops (rt (σ' sip)) oip) ≤ hops 
        ∨ the (flag (rt (σ' sip)) oip) = inv))" 
using ⟨osn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2]) 
with ⟨osn ≥ 1⟩ show "msg_fresh σ' m" 
by (clarsimp) 
qed 

next 
fix hops dip dsn oip sip 
assume [simp]: "m = Rrep hops dip dsn oip sip" 
and "msg_fresh σ m" 
then have "dsn ≥ 1" and "sip = dip ∨ (dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip 
∧ (nsqn (rt (σ sip)) dip = dsn 
    → (the (dhops (rt (σ sip)) dip) ≤ hops 
        ∨ the (flag (rt (σ sip)) dip) = inv))" 
by auto 
from this(2) show "?thesis" 
proof 
assume "sip = dip" with ⟨dsn ≥ 1⟩ show ?thesis by simp 
next 
assume "dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip 
∧ (nsqn (rt (σ sip)) dip = dsn 
    → (the (dhops (rt (σ sip)) dip) ≤ hops 
        ∨ the (flag (rt (σ sip)) dip) = inv))" 
moreover from qinc have "quality_increases (σ sip) (σ' sip)" .. 
ultimately have "dip∈kD(rt (σ' sip)) ∧ dsn ≤ nsqn (rt (σ' sip)) dip 
∧ (nsqn (rt (σ' sip)) dip = dsn 
    → (the (dhops (rt (σ' sip)) dip) ≤ hops 
        ∨ the (flag (rt (σ' sip)) dip) = inv))" 
using ⟨dsn ≥ 1⟩ by (rule quality_increases_rreq_rrep Props [rotated 2]) 
with ⟨dsn ≥ 1⟩ show "msg_fresh σ' m" 
by clarsimp 
qed 

next 
fix dests sip 
assume [simp]: "m = Rerr dests sip" 
and "msg_fresh σ m" 
then have "∀ rip∈dom(dests). rip∈kD(rt (σ sip)) 
∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip" 
by simp 
have "∀ rip∈dom(dests). rip∈kD(rt (σ' sip)) 
∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" 
proof 
fix rip 
assume "rip∈dom(dests)" 
with * have "rip∈kD(rt (σ sip))" and "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip" 
by (drule(1) bspec, clarsimp)+ 
moreover from qinc have "quality_increases (σ sip) (σ' sip)" by simp 
ultimately show "rip∈kD(rt (σ' sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" .. 
qed 
thus ?thesis by simp 
qed simp_all 
end 

1.8 The ‘open’ AODV model 

theory A_OAodv 
imports A_Aodv AWN.OAWN_SOS_Labels AWN.OAWN_Convert 
begins 
Definitions for stating and proving global network properties over individual processes. 
definition σAODV' :: "((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp)) set" 
where "σAODV' ≡ {λi. aodv_init i, ΓAODV PAodv}" 

end
abbreviation opaodv
  :: "ip ⇒ ((ip ⇒ state) × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where
  "opaodv i ≡ (init = σAODV', trans = oseqp_sos ΓAODV i)"

lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
  unfolding σAODV_def σAODV'_def by rule simp_all

lemma oadv_control_within [simp]: "control_within ΓAODV (init (opaodv i))"
  unfolding σAODV'_def by (rule control_withinI) (auto simp del: ΓAODV_sims)

lemma σAODV'_labels [simp]: "((σ, p) ∈ σAODV') ⇒ labels ΓAODV p = {PAodv-:0}"
  unfolding σAODV'_def by simp

lemma oadv_init_kD_empty [simp]: "((σ, p) ∈ σAODV') ⇒ kD (rt (σ i)) = {}"
  unfolding σAODV'_def kD_def by simp

lemma oadv_init_vD_empty [simp]: "((σ, p) ∈ σAODV') ⇒ vD (rt (σ i)) = {}"
  unfolding σAODV'_def vD_def by simp

lemma oadv_trans: "trans (opaodv i) = oseqp_sos ΓAODV i"
  by simp

declare oseq_invariant_ctermsI [OF aodv_wf oadv_control_within aodv_simple_labels oadv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf oadv_control_within aodv_simple_labels oadv_trans, cterms_intros]

end

1.9 Global invariant proofs over sequential processes

theory A_Global_Invariants
imports A_Seq_Invariants
  A_Aodv_Predicates
  A_Fresher
  A_Quality_Increases
  A_OAodv
begin

lemma other_quality_increases [elim]:
  assumes "other quality_increases I σ σ'"
  shows "∀ j. quality_increases (σ j) (σ' j)"
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q) σ σ' a"
    and weakenP: "∀ σ m. P σ m ⇒ P' σ m"
    and weakenQ: "∀ σ m. Q σ m ⇒ Q' σ m"
  shows "otherwith P' I (orecvmsg Q') σ σ' a"
  proof
    fix j
    assume "j /∈ I"
    with * have "P (σ j) (σ' j)" by auto
    thus "P' (σ j) (σ' j)" by (rule weakenP)
  next
    from * have "orecvmsg Q σ a" by auto
    thus "orecvmsg Q' σ a" by rule (erule weakenQ)
  qed
lemma orceived_msg_inv:
assumes other: \( \forall \sigma \sigma'. m. P \sigma m; \) other Q \( \{i\} \sigma \sigma' \imp \Rightarrow P \sigma' m \)
and local: \( \forall \sigma m. P \sigma m \Rightarrow P (\sigma(i := \sigma_i(msg := m))) m \)
shows \( \text{opaodv} i \imp (\text{otherwith} Q \{i\} (\text{orecvmsg} P), \) other Q \( \{i\} \imp \)
onl \( \Gamma_{AODV} (\lambda(\sigma, l). l \in \{\text{Paodv-}:1\} \imp P \sigma (msg \sigma i))) \)
proof (inv_cterms, intro impI)
fix \( \sigma, \sigma', l \)
assume \( l = \text{PAodv-}:1 \imp P \sigma (msg \sigma i) \)
and \( l = \text{PAodv-}:1 \)
and \( \text{other} Q \{i\} \sigma \sigma' \)
from this(1-2) have \( P \sigma (msg \sigma i) \)
.. hence \( P \sigma' (msg \sigma i) \)
using \( \langle \text{other} Q \{i\} \sigma \sigma' \rangle \)
by (rule other)
moreover from \( \langle \text{other} Q \{i\} \sigma \sigma' \rangle \) have \( \sigma' i = \sigma_i \)
.. ultimately show \( P \sigma' (msg \sigma' i) \)
by simp

next
fix \( \sigma, \sigma', \) msg
assume "otherwith Q \( \{i\} (\text{orecvmsg} P) \sigma \sigma' (\text{receive} \) msg)"
and \( \sigma' i = \sigma i[msg := msg] \)
from this(1) have \( P \sigma' (msg \sigma i) \)
by (rule local)
proof (rule other)
from \( \langle \sigma' i = \sigma i[msg := msg] \rangle \) and \( \langle \forall j. j \neq i \imp \text{Q} (\sigma j)(\sigma j') \rangle \)
show "other Q \( \{i\} (\sigma(i := \sigma_i[msg := msg])) \sigma' \)"
by - (rule otherI, auto)
qed

(Equivalent to) Proposition 7.27

lemma local_quality_increases:
"\text{paodv} i \ni (\text{recvmsg} \ rreq_rrep_sn) \imp \text{onll} \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{quality_increases} \xi \xi')"
proof (rule step_invariantI)
fix s a s'
assume sr: "s \in \text{reachable} (\text{paodv} i) (\text{recvmsg} \ rreq_rrep_sn)"
and tr: "(s, a, s') \in \text{trans} (\text{paodv} i)"
and rm: "\text{recvmsg} \ rreq_rrep_sn a"
from sr have srTT: "s \in \text{reachable} (\text{paodv} i) \text{TT} ..
from route_tables_fresher sr tr rm
have "\text{onll} \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{dip} \in \text{kD} (\text{rt} \xi). \text{rt} \xi \subseteq \text{dip} \text{rt} \xi') (s, a, s')"
by (rule step_invariantD)
moreover from known_destinations_increase srTT tr TT_True
have "\text{onll} \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{kD} (\text{rt} \xi) \subseteq \text{kD} (\text{rt} \xi')) (s, a, s')"
by (rule step_invariantD)
moreover from sqns_increase srTT tr TT_True
have "\text{onll} \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{ip} \in \text{sqn} (\text{rt} \xi) \text{ip} \leq \text{sqn} (\text{rt} \xi')) (s, a, s')"
by (rule step_invariantD)
ultimately show "\text{onll} \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{quality_increases} \xi \xi')(s, a, s')"
unfolding onll_def by auto
qed

lemmas olocal_quality_increases =
open_seq_step_invariant [OF local_quality_increases initiali_aodv oadodv_trans aodv_trans, simplified seqll_onll_swap]

lemma oquality_increases:
"\text{opaodv} i \imp (\text{otherwith} \text{quality_increases} \{i\} (\text{orecvmsg} (\lambda_. \text{rreq_rrep_sn})), \) other_quality_increases \{i\} \imp \)

145
lemma rreq_rrep_nsqn_fresh_any_step_invariant:
"opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} \to) \onll \Gamma_{AODV} (\lambda(\langle\sigma, _, \_\rangle, a, \_). anycast (msg_fresh \sigma) a)"
proof (rule ostep_invariantI, simp del: act_simp)
  fix \sigma \, p \, a \, \sigma' \, p'
  assume or: "\langle\sigma, p\rangle \in oreachable (opaodv i) \, QS (other quality_increases \{i\})"
  and l: "l \in labels \Gamma_{AODV} p"
  and \[\langle\sigma, \sigma', a\rangle\]
  and tr: "\langle(\langle\sigma, p\rangle, a, (\langle\sigma', p'\rangle)) \in oseqp_sos \, \Gamma_{AODV} i\"
  and l': "l' \in labels \Gamma_{AODV} p'"
  from this(1-3) have "orecvmsg (\lambda_. rreq_rrep_sn) \sigma a"
by (auto dest!: oreachable_weakenE, auto)
qed

lemma rreq_rrep_nsqn_fresh_new_step_invariant:
"opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} \to) \onll \Gamma_{AODV} (\lambda(\langle\xi, _, \_\rangle, a, \_). anycast (msg_fresh \xi) a)"
proof (rule ostep_invariantI, simp del: act_simp)
  fix \sigma \, p \, a \, \sigma' \, p'
  assume or: "\langle\sigma, p\rangle \in oreachable (opaodv i) (act (recvmsg rreq_rrep_sn)) (other A \{i\})"
  and l: "l \in labels \Gamma_{AODV} p"
  and \[\langle\sigma, \sigma', a\rangle\]
  and tr: "\langle(\langle\sigma, p\rangle, a, (\langle\sigma', p'\rangle)) \in oseqp_sos \, \Gamma_{AODV} i\"
  and recv: "act (recvmsg rreq_rrep_sn) \sigma \sigma' a"
  obtain l l' where "l \in labels \Gamma_{AODV} p" and "l' \in labels \Gamma_{AODV} p'"
by (metis aodv_ex_label)
  have tr: "\langle(\langle\sigma, p\rangle, a, (\langle\sigma', p'\rangle)) \in trans (opaodv i)\" by simp
have "anycast (rreq_rrep_fresh (rt (\sigma i))) a"
proof -
  have "opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} \to) \onll \Gamma_{AODV} (\lambda(\langle\xi, _, \_\rangle, a, \_). anycast (rreq_rrep_fresh (rt \xi) a))"
  by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto
  hence "\onll \Gamma_{AODV} (seqll i (\lambda(\langle\xi, _, \_\rangle, a, \_). anycast (rreq_rrep_fresh (rt \xi) a))) ((\langle\sigma, p\rangle, a, (\langle\sigma', p'\rangle))"
  using or tr recv by - (erule(4) ostep_invariantE)
  thus \?thesis
  using \(l \in labels \, \Gamma_{AODV} p\) and \(l' \in labels \, \Gamma_{AODV} p'\) by auto
qed

moreover have "anycast (rerr_invalid (rt (\sigma i))) a"
proof -
  have "opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} \to) \onll \Gamma_{AODV} (seqll i (\lambda(\langle\xi, _, \_\rangle, a, \_). anycast (rerr_invalid (rt \xi) a))"
  by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto
  hence "\onll \Gamma_{AODV} (seqll i (\lambda(\langle\xi, _, \_\rangle, a, \_). anycast (rerr_invalid (rt \xi) a))) ((\langle\sigma, p\rangle, a, (\langle\sigma', p'\rangle))"
  using or tr recv by - (erule(4) ostep_invariantE)
  thus \?thesis
  using \(l \in labels \, \Gamma_{AODV} p\) and \(l' \in labels \, \Gamma_{AODV} p'\) by auto
qed

moreover have "anycast rreq_rrep_sn a"
proof -  
  from or tr recv  
  have "onll Γ_AODV (seqll i (\(_\), a, \(_\). ancast rreq_rrep_sn a)) ((σ, p), a, (σ', p'))"  
    by (rule ostep_invariantE [OF  
      open_seq_step_invariant [OF rreq_rrep_sn_any_step_invariant initiali_aodv  
        oadv_trans adv_trans,  
        simplified seqll_onll_swap]])  
  thus ?thesis  
    using ⟨l ∈ labels Γ_AODV p⟩ and ⟨l' ∈ labels Γ_AODV p'⟩ by auto  
qed  
  moreover have "anycast (\(\lambda m. \not\text{Pkt} m \rightarrow \text{msg_sender} m = i\)) a"  
    proof -  
      have "opaodv i \|=A (act (recvmsg rreq_rrep_sn), other A \{i\} \rightarrow)  
        onll Γ_AODV (seqll i (\(\lambda(\xi, \_), a, \_\). ancast (\(\lambda m. \not\text{Pkt} m \rightarrow \text{msg_sender} m = i\)) a))"  
        by (rule ostep_invariant_weakenE [OF  
          open_seq_step_invariant [OF sender_ip_valid initiali_aodv,  
            simplified seqll_onll_swap]]) auto  
      thus ?thesis  
        using or tr recv (l ∈ labels Γ_AODV p) and (l' ∈ labels Γ_AODV p') by - (drule(3) onll_ostep_invariantD, auto)  
    qed  
  ultimately have "anycast (msg_fresh σ) a"  
    by (simp_all add: anycast_def  
      del: msg_fresh  
      split: seq_action.split_asm msg.split_asm) simp_all  
  thus "onll Γ_AODV (\(\lambda(\sigma, \_), a, \_\). ancast (msg_fresh σ) a) ((σ, p), a, (σ', p'))"  
    by auto  
qed  
lemma oreceived_rreq_rrep_nsqn_fresh_inv:  
  "opaodv i \|= (otherwith quality_increases \{i\} (orecvmsg msg_fresh),  
    other quality_increases \{i\} \rightarrow)  
    onl Γ_AODV (\(\lambda(\sigma, l). l \in \{PAodv-:1\} \rightarrow msg_fresh σ (msg (\sigma i))\))"  
proof (rule oreceived_msg_inv)  
  fix σ σ' m  
  assume *: "msg_fresh σ m"  
  and "other quality_increases \{i\} σ σ'"  
  from this(2) have "\(\forall j. quality_increases (σ j) (σ' j)\) .."  
    thus "msg_fresh σ' m" using * ..  
  next  
  fix σ m  
  assume "msg_fresh σ m"  
  thus "msg_fresh (σ i := σ i\(\|msg := m\)) m"  
proof (cases m)  
  fix dests sip  
  assume "m = Rerr dests sip"  
    with (msg_fresh σ m) show ?thesis by auto  
  qed auto  
qed  
lemma oquality_increases_nsqn_fresh:  
  "opaodv i \|= (otherwith quality_increases \{i\} (orecvmsg msg_fresh),  
    other quality_increases \{i\} \rightarrow)  
    onll Γ_AODV (\(\lambda((\sigma, \_), \sigma', \_). \forall j. quality_increases (σ j) (σ' j))\)"  
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto  
lemma oosn_rreq:  
  "opaodv i \|= (otherwith quality_increases \{i\} (orecvmsg msg_fresh),  
    other quality_increases \{i\} \rightarrow)  
    onll Γ_AODV (seqll i (\(\lambda(\xi, 1). 1 \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq:n \ln True\} \rightarrow 1 \leq osn ξ\))"  
by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])  
  (auto simp: seqll_onl_swap)
lemma rreq_sip:
"opaodv i \models \text{other quality increases \{i\} (orecvmsg msg\_fresh),
other quality increases \{i\} \rightarrow}
onl \Gamma_{AODV} (\lambda (s, i).
(1 \in \{PAodv\=4, PAodv\=5, PRreq\=0, PRreq\=-2\} \land sip (\sigma i) \neq oip (\sigma i))
\rightarrow oip (\sigma i) \in kd (rt (\sigma (sip (\sigma i))))
\land nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)) \geq osn (\sigma i)
\land (nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = osn (\sigma i)
\rightarrow (hops (\sigma i) \geq \text{the (dhops (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)))}
\lor \text{the (flag (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)))) = inv)))"
(is \"=" \models (?S, ?U \rightarrow \_\"")

proof (inv\_cterm inv add: oseq\_step\_invariant\_sterns [OF equality\_increases\_nsqn\_fresh
aodv\_wf oaoqtrans
onl\_oinvariant\_sterns [OF aodv\_wf oreceived\_rreq\_rrep\_nsqn\_fresh\_inv
onl\_oinvariant\_sterns [OF aodv\_wf oosn\_rreq]
simp add: seqlsimp
simp del: \text{One\_nat\_df, rule impl1})

fix \sigma \sigma' \_ l
assume "(\sigma, p) \in \text{oreachable (opaodv i) ?S ?U}"
and "1 \in labels \Gamma_{AODV} p"

and pre:
"(1 = PAodv\=4 \lor 1 = PAodv\=5 \lor 1 = PRreq\=0 \lor 1 = PRreq\=-2) \land sip (\sigma i) \neq oip (\sigma i)
\rightarrow oip (\sigma i) \in kd (rt (\sigma (sip (\sigma i))))
\land osn (\sigma' i) \leq nsqn (rt (\sigma' (sip (\sigma' i)))) (oip (\sigma' i))
\land (nsqn (rt (\sigma' (sip (\sigma' i)))) (oip (\sigma' i))) = osn (\sigma' i)
\rightarrow (hops (\sigma' i) \leq \text{hops (\sigma' i)})
\lor \text{the (flag (rt (\sigma' (sip (\sigma' i)))) (oip (\sigma' i)))) = inv)"

and "other quality increases \{i\} \sigma \sigma'"

and hyp: "(1=PAodv\=4 \lor 1=PAodv\=5 \lor 1=PRreq\=0 \lor 1=PRreq\=-2) \land sip (\sigma' i) \neq oip (\sigma' i)"
(is \"?labels \land sip (\sigma' i) \neq oip (\sigma' i)\"
from \text{this(4)} have \"\sigma' i = \sigma i \_
\)
with hyp have hyp': "?labels \land sip (\sigma i) \neq oip (\sigma i)" by simp

show "oip (\sigma' i) \in kd (rt (\sigma' (sip (\sigma' i))))
\land osn (\sigma' i) \leq nsqn (rt (\sigma' (sip (\sigma' i)))) (oip (\sigma' i))
\land (nsqn (rt (\sigma' (sip (\sigma' i)))) (oip (\sigma' i))) = osn (\sigma' i)
\rightarrow (hops (rt (\sigma' (sip (\sigma' i)))) (oip (\sigma' i))) \leq hops (\sigma' i)
\lor \text{the (flag (rt (\sigma' (sip (\sigma' i)))) (oip (\sigma' i)))) = inv)"

proof cases "sip (\sigma i) = i"
assume "sip (\sigma i) \neq i"

from \text{other quality increases \{i\} \sigma \sigma' Recall}
\begin{itemize}
\item have "quality\_increases (\sigma (sip (\sigma i))) (\sigma' (sip (\sigma' i)))"
by (rule otherE) (clarsimp simp:
\item hyp' and pre
have "oip (\sigma' i) \in kd (rt (\sigma (sip (\sigma' i))))
\land osn (\sigma' i) \leq nsqn (rt (\sigma (sip (\sigma' i)))) (oip (\sigma' i))
\land (nsqn (rt (\sigma (sip (\sigma' i)))) (oip (\sigma' i))) = osn (\sigma' i)
\rightarrow (hops (rt (\sigma (sip (\sigma' i)))) (oip (\sigma' i))) \leq hops (\sigma' i)
\lor \text{the (flag (rt (\sigma (sip (\sigma' i)))) (oip (\sigma' i)))) = inv)"
\end{itemize}

by (auto simp: "$\sigma' i = \sigma i\)"
ultimately show \text{thesis}
by (rule quality\_increases\_rreq\_rrep\_props)
next
assume "sip (\sigma i) = i" thus \text{thesis}
using \sigma' i = \sigma i hyp and pre by auto

\begin{itemize}
\item qed
\item \text{auto elim!: quality\_increases\_rreq\_rrep\_props'}
\end{itemize}

lemma odsn\_rreq:
"opaodv i \models \text{other quality increases \{i\} (orecvmsg msg\_fresh),
other quality increases \{i\} \rightarrow}\)
lemma rrep_sip:  
"(oapadv i) ⇒ (otherwise quality_increases {i} (orecvmsg msg_fresh),  
other quality_increases {i} →)

proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf  
aodv_trans]  
onl_onvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]  
onl_onvariant_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impl)

fix σ σ' p l
assume "σ ᵇ σ' ∈ oreachable (oapadv i) ?S ?U"
and "1 ∈ labels Γ_AODV p"
and pre:  
"(l = PAodv−:6 ∨ l = PAodv−:7 ∨ l = PRrep−:0 ∨ l = PRrep−:1) ∧ sip (σ i) ≠ dip (σ i)  
→ dip (σ i) ∈ kd (rt (σ (sip (σ i))))  
∧ nsqn (rt (σ (sip (σ i)))) (sip (σ i)) ≥ dsn (σ i)  
∧ (nsqn (rt (σ (sip (σ i)))) (sip (σ i))) = dsn (σ i)  
→ (hops (σ i) ≥ the (dhops (σ i)))) (sip (σ i)) ≤ hops (σ i)  
∧ the (flag (σ i))) (sip (σ i))) = inv)"
  (is "=" (?S, ?U → " _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf  
aodv_trans]  
onl_onvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]  
onl_onvariant_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impl)

fix σ σ' p l
assume "σ ᵇ σ' ∈ oreachable (oapadv i) ?S ?U"
and "1 ∈ labels Γ_AODV p"
and pre:  
"(l = PAodv−:6 ∨ l = PAodv−:7 ∨ l = PRrep−:0 ∨ l = PRrep−:1) ∧ sip (σ i) ≠ dip (σ i)  
→ dip (σ i) ∈ kd (rt (σ (sip (σ i))))  
∧ nsqn (rt (σ (sip (σ i)))) (sip (σ i)) ≥ dsn (σ i)  
∧ (nsqn (rt (σ (sip (σ i)))) (sip (σ i))) = dsn (σ i)  
→ (hops (σ i) ≥ the (dhops (σ i)))) (sip (σ i)) ≤ hops (σ i)  
∧ the (flag (σ i))) (sip (σ i))) = inv)"
  (is "=" (?S, ?U → " _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf  
aodv_trans]  
onl_onvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]  
onl_onvariant_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impl)

fix σ σ' p l
assume "σ ᵇ σ' ∈ oreachable (oapadv i) ?S ?U"
and "1 ∈ labels Γ_AODV p"
and pre:  
"(l = PAodv−:6 ∨ l = PAodv−:7 ∨ l = PRrep−:0 ∨ l = PRrep−:1) ∧ sip (σ i) ≠ dip (σ i)  
→ dip (σ i) ∈ kd (rt (σ (sip (σ i))))  
∧ nsqn (rt (σ (sip (σ i)))) (sip (σ i)) ≥ dsn (σ i)  
∧ (nsqn (rt (σ (sip (σ i)))) (sip (σ i))) = dsn (σ i)  
→ (hops (σ i) ≥ the (dhops (σ i)))) (sip (σ i)) ≤ hops (σ i)  
∧ the (flag (σ i))) (sip (σ i))) = inv)"
  (is "=" (?S, ?U → " _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf  
aodv_trans]  
onl_onvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]  
onl_onvariant_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impl)

fix σ σ' p l
assume "σ ᵇ σ' ∈ oreachable (oapadv i) ?S ?U"
and "1 ∈ labels Γ_AODV p"
and pre:  
"(l = PAodv−:6 ∨ l = PAodv−:7 ∨ l = PRrep−:0 ∨ l = PRrep−:1) ∧ sip (σ i) ≠ dip (σ i)  
→ dip (σ i) ∈ kd (rt (σ (sip (σ i))))  
∧ nsqn (rt (σ (sip (σ i)))) (sip (σ i)) ≥ dsn (σ i)  
∧ (nsqn (rt (σ (sip (σ i)))) (sip (σ i))) = dsn (σ i)  
→ (hops (σ i) ≥ the (dhops (σ i)))) (sip (σ i)) ≤ hops (σ i)  
∧ the (flag (σ i))) (sip (σ i))) = inv)"
  (is "=" (?S, ?U → " _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf  
aodv_trans]  
onl_onvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]  
onl_onvariant_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impl)

fix σ σ' p l
assume "σ ᵇ σ' ∈ oreachable (oapadv i) ?S ?U"
and "1 ∈ labels Γ_AODV p"
and pre:  
"(l = PAodv−:6 ∨ l = PAodv−:7 ∨ l = PRrep−:0 ∨ l = PRrep−:1) ∧ sip (σ i) ≠ dip (σ i)  
→ dip (σ i) ∈ kd (rt (σ (sip (σ i))))  
∧ nsqn (rt (σ (sip (σ i)))) (sip (σ i)) ≥ dsn (σ i)  
∧ (nsqn (rt (σ (sip (σ i)))) (sip (σ i))) = dsn (σ i)  
→ (hops (σ i) ≥ the (dhops (σ i)))) (sip (σ i)) ≤ hops (σ i)  
∧ the (flag (σ i))) (sip (σ i))) = inv)"
  (is "=" (?S, ?U → " _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf  
aodv_trans]  
onl_onvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]  
onl_onvariant_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impl)

fix σ σ' p l
assume "σ ᵇ σ' ∈ oreachable (oapadv i) ?S ?U"
and "1 ∈ labels Γ_AODV p"
and pre:  
"(l = PAodv−:6 ∨ l = PAodv−:7 ∨ l = PRrep−:0 ∨ l = PRrep−:1) ∧ sip (σ i) ≠ dip (σ i)  
→ dip (σ i) ∈ kd (rt (σ (sip (σ i))))  
∧ nsqn (rt (σ (sip (σ i)))) (sip (σ i)) ≥ dsn (σ i)  
∧ (nsqn (rt (σ (sip (σ i)))) (sip (σ i))) = dsn (σ i)  
→ (hops (σ i) ≥ the (dhops (σ i)))) (sip (σ i)) ≤ hops (σ i)  
∧ the (flag (σ i))) (sip (σ i))) = inv)"
  (is "=" (?S, ?U → " _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf  
aodv_trans]  
onl_onvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]  
onl_onvariant_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impl)
lemma rerr_sip:
"{opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} ->)

  onl \( \Gamma_{AODV} (\lambda (\sigma, 1).
    l \in \{PAodv:-8, PAodv:-9, PRerr:-0, PRerr:-1\} 
    \rightarrow (\forall \text{rip} \in \text{dom(dests (\sigma i))). \text{rip} \in kD(rt (\sigma (sip (\sigma i)))) \land 
    the (\text{dests (\sigma i) ripc}) - 1 \leq nsqn (rt (\sigma (sip (\sigma i)))) ripc))"

(is "_ |= (?S, ?U -> _)")

proof -
{ fix dests rip sip rsn and \( \sigma' :: \text{ip} \Rightarrow \text{state} \) 
  assume qinc: "\( \forall j. \text{quality_increases (\sigma j) (\sigma' j)} \)"
  and *: "\( \forall \text{rip} \in \text{dom dests. rip} \in kD(rt (\sigma sip)) \land 
    the (\text{dests rip}) - 1 \leq nsqn (rt (\sigma sip)) rip \)"

  from this(3) have "\( \text{rip} \in \text{dom dests} \)" by auto
  with * and \( \text{dests rip = Some rsn} \) have "\( \text{rip} \in kD(rt (\sigma sip)) \)"
    and "\( \text{rsn - 1 \leq nsqn (rt (\sigma sip)) rip} \)"
    by (auto dest!: bspec)
  from qinc have "\( \text{quality_increases (\sigma sip) (\sigma' sip)} \) ..
  have "\( \text{rip} \in kD(rt (\sigma sip)) \land \text{rsn - 1 \leq nsqn (rt (\sigma sip)) rip} \)"
  proof from \( \text{rip} \in kD(rt (\sigma sip)) \) and \( \text{quality_increases (\sigma sip) (\sigma' sip)} \)
    show "\( \text{rip} \in kD(rt (\sigma' sip)) \) ..
  next from \( \text{rip} \in kD(rt (\sigma sip)) \) and \( \text{quality_increases (\sigma sip) (\sigma' sip)} \)
    have "\( \text{nsqn (rt (\sigma sip)) rip \leq nsqn (rt (\sigma' sip)) rip} \) ..
    with \( \text{rsn - 1 \leq nsqn (rt (\sigma sip)) rip} \) show "\( \text{rsn - 1 \leq nsqn (rt (\sigma' sip)) rip} \)"
    by (rule le_trans)
  qed
}

note partial = this

show ?thesis by (inv_cterms inv add): oseq_step_invariant_sterms [OF equality_increases_nsqn_fresh aodv_wf
oaodv_trans]
  onloinvariant_sterms [OF aodv_wf orecvmsg msg_fresh
aodv_trans, simplified seql_onl_swap, THEN oinvariant_anyact]

  one_nat_def, intro conjl)
  { clarsimp simp del: One_nat_def split: option.split_asm erule(2) partial} +

qed

lemma prerr_guard: "{opaodv i |=

  onl \( \Gamma_{AODV} (\lambda (\xi, 1). (l = PRerr:-1 
    \rightarrow (\forall \text{rip} \in \text{dom(dests \xi)). \text{rip} \in \text{vD(rt \xi)}) 
    \land \text{the (nhop (rt \xi) rip) = sip \xi }
    \land \text{sqn (rt \xi) rip < the (dests \xi ripc)})")"

by (inv_cterms) (clarsimp split: option.split_asm if_split_asm)

lemmas oaddpreRT_welldefined = 
  open_seq_invariant [OF addpreRT_welldefined initiali_aodv oaddpreRT oaddpreRT_trans, simplified seql_onl_swap, THEN oinvariant_anyact]

lemmas odests_vD_inc_nsqn =
  open_seq_invariant [OF odests_vD_inc_nsqn initiali_aodv oaddpreRT oaddpreRT_trans, simplified seql_onl_swap, THEN oinvariant_anyact]

lemmas oprerr_guard =
  open_seq_invariant [OF prerr_guard initiali_aodv oaddpreRT oaddpreRT_trans, simplified seql_onl_swap, THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':
\texttt{\texttt{opaodv\ i = (otherwith quality\_increases\ {i} (orecmsg\ msg\_fresh), other quality\_increases\ {i} \Rightarrow onl \Gamma_{AODV} (\lambda(\sigma, _). \\
\forall dip. \text{let nhip = the (nhop (rt (\sigma i)) dip) \\
in dip \in kD(rt (\sigma i)) \land nhip \neq dip \rightarrow \\
dip \in kD(rt (\sigma nhip)) \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma nhop)) dip)\"}}

(is "\_ \Rightarrow (?S, ?U \rightarrow \_")

\textbf{proof -}

\{ fix nhop and \sigma \sigma': "ip \Rightarrow state"
assume pre: \texttt{\forall dip\in\textit{kD}(rt (\sigma i)). nhop dip \neq dip \rightarrow \textit{dip}\in\textit{kD}(rt (\sigma (nhop dip))) \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip"
and qinc: \texttt{\forall j. quality\_increases (\sigma j) (\sigma' j)"}
have \texttt{\forall dip\in\textit{kD}(rt (\sigma i)). nhop dip \neq dip \rightarrow \textit{dip}\in\textit{kD}(rt (\sigma' (nhop dip))) \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"} ..

\textbf{proof (intro ballI impI)}
fix dip
assume \texttt{"dip\in\textit{kD}(rt (\sigma i))"}
and \texttt{"nhop dip \neq dip"}
with pre have \texttt{"dip\in\textit{kD}(rt (\sigma (nhop dip)))"}
and \texttt{"nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip"} ..
by auto
from qinc have qinc_nhop: \texttt{"quality\_increases (\sigma (nhop dip)) (\sigma' (nhop dip))"} ..
with \texttt{\textit{dip}\in\textit{kD}(rt (\sigma (nhop dip)))} have \texttt{"\textit{dip}\in\textit{kD} (rt (\sigma' (nhop dip)))"} ..
moreover have \texttt{"nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"} ..

\textbf{proof -}
from \texttt{\textit{dip}\in\textit{kD}(rt (\sigma (nhop dip)))} qinc_nhop
have \texttt{"nsqn (rt (\sigma (nhop dip))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"} ..
with \texttt{nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip} show ?thesis
by simp
qed
ultimately show \texttt{"\textit{dip}\in\textit{kD}(rt (\sigma' (nhop dip))) \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"} ..

\textbf{qed}
\}
\textbf{note basic = this}

\{ fix nhop and \sigma \sigma': "ip \Rightarrow state"
assume pre: \texttt{\forall dip\in\textit{kD}(rt (\sigma i)). nhop dip \neq dip \rightarrow \textit{dip}\in\textit{kD}(rt (\sigma (nhop dip)))}
\land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip"
and ndest: \texttt{\forall ripc\in\textit{dedom}(dests (\sigma i)). ripc \in kD (rt (\sigma (sip (\sigma i))))}
\land \texttt{the (dests (\sigma i) ripc) - 1 \leq nsqn (rt (\sigma (sip (\sigma i)))) ripc"}
and issip: \texttt{\forall ip\in\textit{dedom}(dests (\sigma i)). nhop ip = sip (\sigma i)"}
and qinc: \texttt{\forall j. quality\_increases (\sigma j) (\sigma' j)"}
have \texttt{\forall dip\in\textit{kD}(rt (\sigma i)). nhop dip \neq dip \rightarrow \textit{dip}\in\textit{kD} (rt (\sigma' (nhop dip)))}
\land \texttt{nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"} ..

\textbf{proof (intro ballI impI)}
fix dip
assume \texttt{"\textit{dip}\in\textit{kD}(rt (\sigma i))"}
and \texttt{"nhop dip \neq dip"}
with pre and qinc have \texttt{"\textit{dip}\in\textit{kD}(rt (\sigma' (nhop dip)))"}
and \texttt{"nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"} ..
by (auto dest!: basic)

have \texttt{"nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"} ..

\textbf{proof (cases "\textit{dip}\in\textit{dedom}(dests (\sigma i))")}
assume \texttt{"\textit{dip}\in\textit{dedom}(dests (\sigma i))"}
with \texttt{\textit{dip}\in\textit{kD}(rt (\sigma i))} obtain dsn where \texttt{"dests (\sigma i) dip = Some dsn"}
by auto
with \texttt{\textit{dip}\in\textit{kD}(rt (\sigma i))} have \texttt{"nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip = dsn - 1"}
by (rule nsqn_invalidate_eq)
moreover have \texttt{"dsn - 1 \leq nsqn (rt (\sigma' (nhop dip))) dip"} ..

\textbf{proof -}
from \texttt{\textit{dests (\sigma i) dip = Some dsn}} have \texttt{"the (dests (\sigma i) dip) = dsn"} by simp
proof
fix dip
assume "dip \notin \text{dom} (\text{dests} (\sigma i))"
with dip\in\text{kd}(\text{rt} (\sigma i))
\text{have } "\text{nsqn} (\text{rt} (\sigma i)) (\text{dests} (\sigma i)) \text{ dip} \neq \text{dip}"
\text{and } "\text{nsqn} (\text{rt} (\sigma i)) (\text{dests} (\sigma i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma i)) (\text{dests} (\sigma i)) \text{ dip}"
\text{by } (\text{drule(1)}, \text{basic auto})
thus "(?P dip)" by (cases "\text{dip} = \text{sip} (\sigma i)") auto
qed

} note nhop_update_sip = this

{ fix \sigma \sigma' :: "\text{ip} \Rightarrow \text{state}"
assume a1: "\forall dip\in\text{kd}(\text{rt} (\sigma i)). \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip}"
\text{and } a2: "\forall j. \text{quality\_increases} (\sigma j) (\sigma' j)"
\text{have } "\forall dip\in\text{kd}(\text{rt} (\sigma i)).
\text{the} (\text{nhop} (\text{rt} (\sigma i)) (\text{sip} (\sigma i)) (0, \text{unk}, \text{val}, \text{Suc} 0, \text{sip} (\sigma i), \text{\{}\text{\}})) \text{ dip} \neq \text{dip}"
\text{and } \text{pre'}: "\forall dip\in\text{kd}(\text{rt} (\sigma i)). \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip}"
\text{and } \text{qinc: } "\forall j. \text{quality\_increases} (\sigma j) (\sigma' j)"
\text{and } *: "\text{dip} \neq \text{oip} \Rightarrow \text{oip}\in\text{kd}(\text{rt} (\sigma \text{sip}))"
\text{and } \text{nsqn} \leq \text{nsqn} (\text{rt} (\sigma \text{sip})) \text{ oip}
\text{and } \text{osn} = \text{osn}
\text{and } \text{osn} \Rightarrow \text{the} (\text{dip}) (\text{rt} (\sigma \text{sip})) \text{ oip} \leq \text{hops}
\text{and } \text{osn} \Rightarrow \text{the} (\text{flag}) (\text{rt} (\sigma \text{sip})) \text{ oip} = \text{inv}"
\text{from } \text{pre} \text{ and } \text{qinc}
\text{have } \text{pre'}: "\forall dip\in\text{kd}(\text{rt} (\sigma i)). \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip}"
\text{and } \text{pre': } "\forall dip\in\text{kd}(\text{rt} (\sigma i)). \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip}"
\text{by } (\text{rule basic})

} note basic_prerr = this
have "(the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) oip) ≠ oip
→ oip∈kD(rt (σ' (the (nhop (update (rt (σ i)) oip
(osn, kno, val, Suc hops, sip, {})) oip)))))
∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) oip
≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
(osn, kno, val, Suc hops, sip, {})) oip)))) oip)"
(is "?nhop_not_oip → ?oip_in_kD ∧ ?nsqn_le_nsqn")
proof (rule, split update_rt_split_asm)
assume "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
and "the (nhop (rt (σ i)) oip) ≠ oip"
with pre' show "?oip_in_kD ∧ ?nsqn_le_nsqn" by auto
next
assume rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})
and notoip: ?nhop_not_oip
with * qinc
have ?oip_in_kD by (clarsimp elim!: kD_quality_increases)
moreover with * pre qinc rtnot notoip
have ?nsqn_le_nsqn by simp (metis kD_nsqn_quality_increases_trans)
ultimately show "?oip_in_kD ∧ ?nsqn_le_nsqn" ..
qed
} note update1 = this
{ fix σ σ' oip sip osn hops
assume pre: "∀ dip∈kD (rt (σ i)). the (nhop (update (rt (σ i)) dip) ≠ dip
→ dip∈kD(rt (σ (the (nhop (rt (σ i)) dip))))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip))))) dip"
and qinc: "∀ j. quality_increases (σ j) (σ' j)"
and *: "sip ≠ oip
→ oip∈kD(rt (σ sip))
∧ (nsqn (rt (σ sip)) oip = osn
→ the (dhops (rt (σ sip)) oip) ≤ hops
∨ the (flag (rt (σ sip)) oip) = inv)"
from pre and qinc
have pre': "∀ dip∈kD (rt (σ i)). the (nhop (update (rt (σ i)) dip) ≠ dip
→ dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip))))) dip"
by (rule basic)
have "∀ dip∈kD(rt (σ i)).
the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) oip) ≠ dip
→ dip∈kD(rt (σ' (the (nhop (update (rt (σ i)) oip
(osn, kno, val, Suc hops, sip, {})) dip))))
∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) oip
≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
(osn, kno, val, Suc hops, sip, {})) dip)))) oip)"
(is "∀ dip∈kD(rt (σ i)). _ → ?dip_in_kD dip ∧ ?nsqn_le_nsqn dip")
proof (intro ballI impI, split update_rt_split_asm)
fix dip
assume "dip∈kD(rt (σ i))"
and "the (nhop (rt (σ i)) dip) ≠ dip"
and "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
with pre' show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip" by simp
next
fix dip
assume "dip∈kD(rt (σ i))"
and notdip: "the (nhop (update (rt (σ i)) dip) (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"
and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip"
proof (cases "dip = oip")
assume "dip ≠ oip"
with pre' (dip∈kD(rt (σ i))) notdip
show ?thesis by clarsimp
next
assume "dip = oip"
with \( \text{rtnot qinc} \ (\text{dip} \in \text{kD}(\text{rt} (\sigma i))) \) \text{notdip} *

have "?dip_in_kD dip" by simp (metis \text{kD._quality_increases})

moreover from \( \langle \text{dip} = \text{oip} \rangle \) \( \text{rtnot qinc} \ (\text{dip} \in \text{kD}(\text{rt} (\sigma i))) \) \text{notdip} *

have "?nsqn_le_nsqn dip" by simp (metis \text{kD._nsqn_quality_increases_trans})

ultimately show ?thesis ..

qed

qed

Note update2 = this

have "opaodv i \models (?S, ?U \rightarrow) \text{onl} \ AODV \ (\lambda (\sigma, -)).
\forall \text{dip} \in \text{kD}(\text{rt} (\sigma i)). \ (\text{nhop} (\text{rt} (\sigma i)) \text{dip} \neq \text{dip}
\rightarrow \text{dip} \in \text{kD}(\text{rt} (\sigma (\text{nhop} (\text{rt} (\sigma i)) \text{dip}))))
\wedge \text{nsqn} (\text{rt} (\sigma i)) \text{dip} \leq \text{nsqn} (\text{rt} (\sigma (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \) dip"

by (inv_c terms inv add: oseq_stepInvariant Sterms [OF oquality_increases_nsqn_fresh aodv_wf
\text{oaodv} trans]

\text{onl} invTrait Sterms [OF aodv_wf oaddPreRT_welldefined]
\text{onl} invTrait Sterms [OF aodv_wf odests_vD_inc_sqn]
\text{onl} invTrait Sterms [OF aodv_wf oprerr_guard]
\text{onl} invTrait Sterms [OF aodv_wf rreq_sip]
\text{onl} invTrait Sterms [OF aodv_wf rrep_sip]
\text{other} quality_increases
\text{other} localD

solve: basic basic_prerr
simp add: seqlsimp nsqn_invalidate nhop_update_sip
simp del: One_nat_def)

(thus ?thesis unfolding \text{Let_def} by auto

qed

Proposition 7.30

lemmas \text{okD.unk_or.atleast_one} =
open_seqInvariant [OF \text{kD.unk_or.atleast_one initiali_aodv},
simplified seql_onl_swap]

lemmas \text{ozero_seq.unk.hops_one} =
open_seqInvariant [OF \text{zero_seq.unk.hops_one initiali_aodv},
simplified seql_onl_swap]

lemma \text{oreachable.fresh.okD.unk.or.atleast_one}:
fixes dip
assumes "\((\sigma, p) \in \text{oreachable} \ (\text{opaodv} i)"
\((\text{otherwith} ((\_)) \ i) \ (\text{orecvmsg} (\lambda \sigma \ m. \ \text{msg.fresh} \ \sigma \ m
\wedge \text{msg.zhops} \ m)))"

\((\text{other} quality_increases \ i))"

and "\text{dip} \in \text{kD}(\text{rt} (\sigma i))"

shows "\(\pi_3(\text{the} (\text{rt} (\sigma i) \text{dip})) = \text{unk} \vee 1 \leq \pi_2(\text{the} (\text{rt} (\sigma i) \text{dip}))"

(is "?P dip")

proof -
have "\exists l \in \text{labels} AODV \ p" by (metis aodv.ex_label)
with \text{assms(1)} have "\forall \text{dip} \in \text{kD}(\text{rt} (\sigma i)). \ ?P dip"
by - (drule oinvariant_weakenD [OF \text{okD.unk_or.atleast_one} [OF aodv_trans aodv_trans]],
auto dest!: otherwith.actionD onlD simp: seqlsimp)

with \(\text{dip} \in \text{kD}(\text{rt} (\sigma i))\) show ?thesis by simp

qed

lemma \text{oreachable.fresh.ozero_seq.unk.hops_one}:
fixes dip
assumes "\((\sigma, p) \in \text{oreachable} \ (\text{opaodv} i)"
\((\text{otherwith} ((\_)) \ i) \ (\text{orecvmsg} (\lambda \sigma \ m. \ \text{msg.fresh} \ \sigma \ m
\wedge \text{msg.zhops} \ m)))"

\((\text{other} quality_increases \ i))"
and "dip∈KD(rt (σ i))"
shows "sqn (rt (σ i)) dip = 0 →
sqn (rt (σ i)) dip = unk
∧ the (dhops (rt (σ i)) dip) = 1
∧ the (nhop (rt (σ i)) dip) = dip"
(is "?P dip")

proof -
  have \("∃i. l∈labels Γ_AODV p" by (metis aodv_ex_label)
with assms(iI) have "∀dip∈KD (rt (σ i)). ?P dip"
    by -(drule invariant_weakenD [OF ozero_seq_unk_hops_one [OF oadv_trans aodv_trans]],
      auto dest!: onlD otherwith_actionD simp: seqsimp)

  with \("dip∈KD(rt (σ i))\) show \(?\)thesis by simp

qed

lemma seq_nhdp_quality_increases':
shows "oahd i \(\models\) (otherwith ((=)) \{i\})
   (orecmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
other quality_increases (i) \(\rightarrow\)
onl Γ_AODV (λ(σ, _). ∀dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) \(\cap\) vD (rt (σ nhip))
∧ nhip \(\ne\) dip
→ (rt (σ i)) \(\sqcap\) dip (rt (σ (nhop (rt (σ i)))))"
(is \("\_ \(\models\) (?S i, \_ \(\rightarrow\) \_)\))

proof -
  have weaken:
"∀P I Q R P. p \(\models\) (otherwith quality_increases I (orecmsg Q), other quality_increases I \(\rightarrow\) P)
⇒ p \(\models\) (otherwith ((=)) I (orecmsg (λσ m. Q σ m \& R σ m)), other quality_increases I \(\rightarrow\) P"
by auto

{ fix i a and σ a' :: "ip ⇒ state"
  assume aI: "∀dip. dip∈vD(rt (σ i))
∧ dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))
∧ (the (nhop (rt (σ i)) dip)) \(\ne\) dip
→ rt (σ i) \(\sqcap\) dip rt (σ (the (nhop (rt (σ i)) dip)))"
  and ow: "?S i σ σ' a"
  have "∀dip. dip∈vD(rt (σ i))
∧ dip∈vD (rt (σ' (the (nhop (rt (σ i)) dip))))
∧ (the (nhop (rt (σ i)) dip)) \(\ne\) dip
→ rt (σ i) \(\sqcap\) dip rt (σ' (the (nhop (rt (σ i)) dip))))"

proof clarify
  fix dip
assume a2: "dip∈vD(rt (σ i))"
  and a3: "dip∈vD (rt (σ' (the (nhop (rt (σ i)) dip))))"
  and a4: "(the (nhop (rt (σ i)) dip)) \(\ne\) dip"
  from ow have "∀j. j \(\ne\) i \(\rightarrow\) σ j = σ' j" by auto

show "rt (σ i) \(\sqcap\) dip rt (σ' (the (nhop (rt (σ i)) dip))"

proof (cases "(the (nhop (rt (σ i)) dip)) = i")
  assume "(the (nhop (rt (σ i)) dip)) = i"
  with \("dip∈vD(rt (σ i))\) have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))" by simp
  with aI a2 a4 have "rt (σ (the (nhop (rt (σ i)) dip))) = i" by simp
  with \("(the (nhop (rt (σ i)) dip)) = i\) have "rt (σ i) \(\sqcap\) dip rt (σ i)" by simp
  hence False by simp
thus ?thesis ..
next
assume "(the (nhop (rt (σ i)) dip)) \(\ne\) i"
  with \("∀j. j \(\ne\) i \(\rightarrow\) σ j = σ' j\)
    have "σ (the (nhop (rt (σ i)) dip)) = σ' (the (nhop (rt (σ i)) dip))" by simp
  with \("dip∈vD(rt (σ' (the (nhop (rt (σ i)) dip))))\)
    have "dip∈vD (rt (σ (the (nhop (rt (σ i)) dip))))" by simp
  with aI a2 a4 have "rt (σ (the (nhop (rt (σ i)) dip))) = i" by simp
  with * show ?thesis by simp

qed

qed

} note basic = this
{ fix σ σ' a dip sip i
  assume a1: "∀ dip. dip ∈ vD(rt (σ i)) ∧ dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip)))) ∧ the (nhop (rt (σ i)) dip) ≠ dip ∨ rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip)))"
  and ov: "∀ S i σ σ' a"

  have "∀ dip. dip ∈ vD(update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) ∧ dip ∈ vD(rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip))) ∧ the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip) ≠ dip ∨ update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) ⊏ dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip)))"

proof clarify

fix dip

assume a2: "dip ∈ vD(update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {}))"
  and a3: "dip ∈ vD(rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip)))))"
  and a4: "the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip) ≠ dip"

show "update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) ⊏ dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip)))"

proof (cases "dip = sip")

assume "dip = sip"

with "the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip) ≠ dip" have False by simp

thus ?thesis ..

next

assume [simp]: "dip ≠ sip"

from a2 have "dip ∈ vD(rt (σ i)) ∨ dip = sip"
  by (rule vD_update_val)

with "dip ≠ sip" have "dip ∈ vD(rt (σ i))" by simp

moreover from a3 have "dip ∈ vD(rt (σ' (the (nhop (rt (σ i)) dip))))" by simp

moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp

ultimately have "rt (σ i) ⊏ dip rt (σ' (the (nhop (rt (σ i)) dip)))"

using a1 ov by - (drule(1) basic, simp)

with "dip ≠ sip" show ?thesis
  by - (erule rt_strictly_fresher_update_other, simp)

qed

qed

} note update_0_unk = this

{ fix σ a σ' nhop
  assume pre: "∀ dip. dip ∈ vD(rt (σ i)) ∧ dip ∈ vD(rt (σ (nhop dip))) ∧ nhop dip ≠ dip → rt (σ i) ⊏ dip rt (σ (nhop dip))"
  and ov: "∀ S i σ σ' a"

  have "∀ dip. dip ∈ vD(invalidate (rt (σ i)) (dests (σ i))) ∧ dip ∈ vD (rt (σ' (nhop dip))) ∧ nhop dip ≠ dip → rt (σ i) ⊏ dip rt (σ' (nhop dip))"

proof clarify

fix dip

assume "dip ∈ vD(invalidate (rt (σ i)) (dests (σ i)))"
  and "dip ∈ vD(rt (σ' (nhop dip)))"
  and "nhop dip ≠ dip"

from this(1) have "dip ∈ vD (rt (σ i))"
  by (clarsimp dest!: vD_invalidate_vD_not_dests)

moreover from ov have "∀ j. j ≠ i → σ j = σ' j" by auto

ultimately have "rt (σ i) ⊏ dip rt (σ (nhop dip))"
  using pre "dip ∈ vD (rt (σ' (nhop dip))) ∧ nhop dip ≠ dip" by metis

with "∀ j. j ≠ i → σ j = σ' j" show "rt (σ i) ⊏ dip rt (σ' (nhop dip))"
  by (metis rt_strictly_fresher_irefl)

qed

} note invalidate = this

{ fix σ a σ' dip oip osn sip hops i
assume pre: "∀dip. dip ∈ vD (rt (σ i))
  ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
  ∧ the (nhop (rt (σ i)) dip) ≠ dip
  → rt (σ i) □ dip rt (σ (the (nhop (rt (σ i)) dip)))"
and ow: "?S i σ ' a"
and "Suc 0 ≤ osn"
and a6: "sip ≠ oip → oip ∈ kD (rt (σ sip))
∧ osn ≤ nsqn (rt (σ sip)) oip
∧ (nsqn (rt (σ sip)) oip = osn
  → the (dhops (rt (σ sip)) oip) ≤ hops
  ∨ the (flag (rt (σ sip)) oip) = inv)"
and after: "σ' i = σ i(∀rt := update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {}))"
have "∀dip. dip ∈ vD (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {}))
∧ dip ∈ vD (rt (σ' (the (nhop (update (rt (σ i)) oip
  (osn, kno, val, Suc hops, sip, {})) dip)))
∧ the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip
  → update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})
□ dip
rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))"
proof clarify
fix dip
assume a2: "dip∈vD(update (rt (σ i)) oip (osn, kno, val, Suc (hops), sip, {}))"
  and a3: "dip∈vD(rt (σ' (the (nhop (update (rt (σ i)) oip
  (osn, kno, val, Suc hops, sip, {})) dip))))"
  and a4: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"
from ow have a5: "∀j. j ≠ i → σ j = σ' j" by auto
show "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})
□ dip
rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))"
(is "?rt1 □ dip ?rt2 dip")
proof (cases "?rt1 = rt (σ i)"

assume nochange [simp]:
"update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {}) = rt (σ i)"
from after have "σ' i = σ i" by simp
with a5 have "∀j. j = σ' j" by metis
from a2 have "dip∈vD (rt (σ i))" by simp
moreover from a3 have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
  using nochange and ∀j. σ j = σ' j by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) □ dip rt (σ (the (nhop (rt (σ i)) dip)))"
  using pre by simp
hence "rt (σ i) □ dip rt (σ' (the (nhop (rt (σ i)) dip)))"
  using (∀j. σ j = σ' j) by simp
thus "?thesis" by simp
next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip∈kD(rt (σ' i))" by auto
show ?thesis
proof (cases "dip = oip")
  assume "dip ≠ oip"
  with a2 have "dip∈vD (rt (σ i))" by auto
  moreover with a3 a5 after and ⟨dip ≠ oip⟩
    have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
    by simp metis
  moreover from a4 and ⟨dip ≠ oip⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
  ultimately have "rt (σ i) □ dip rt (σ (the (nhop (rt (σ i)) dip)))"
    using pre by simp
  with after and a5 and ⟨dip ≠ oip⟩ show ?thesis
    by simp (metis rt_strictly_fresher_update_other
next
assume "dip = oip"

with a4 and change have "sip ≠ oip" by simp
with a6 have "oip∈kD(rt (σ sip))"
  and "osn ≤ nsqn (rt (σ sip)) oip" by auto

from a3 change ⟨dip = oip⟩ have "oip∈vD(rt (σ ' sip))" by simp
hence "the (flag (rt (σ ' sip)) oip) = val" by simp

from ⟨oip∈kD(rt (σ sip))⟩
have "osn < nsqn (rt (σ ' sip)) oip ∨ (osn = nsqn (rt (σ ' sip)) oip
  ∧ the (dhops (rt (σ ' sip)) oip) ≤ hops)"

proof
assume "oip∈vD(rt (σ sip))"
hence "the (flag (rt (σ sip)) oip) = val" by simp
with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn →
  the (dhops (rt (σ sip)) oip) ≤ hops"
  by simp
show ?thesis
proof (cases "sip = i")
  assume "sip ≠ i"
  with a5 have "σ sip = σ ' sip" by simp
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  and ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  show ?thesis by auto
next
  — alternative to using sip_not_ip
  assume [simp]: "sip = i"
  have "?rt1 = rt (σ i)"
  proof (rule update_cases_kD, simp_all)
    from ⟨Suc 0 ≤ osn⟩ show "0 < osn" by simp
  next
    from ⟨oip∈kD(rt (σ sip))⟩ and ⟨sip = i⟩
    show "oip∈kD(rt (σ i))" by simp
  next
    assume "sqn (rt (σ i)) oip < osn"
    also from ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
    have "... ≤ nsqn (rt (σ i)) oip" by simp
    also have "... ≤ sqn (rt (σ i)) oip" by (rule nsqn_sqn)
    finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip".
    hence False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
    else rt (σ i) a) = rt (σ i)" ..
next
assume "sqn (rt (σ i)) oip = osn"
and "Suc hops < the (dhops (rt (σ i)) oip)"
from this(1) and ⟨oip ∈ vD (rt (σ sip))⟩ have "nsqn (rt (σ i)) oip = osn"
  by simp
with ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  have "the (dhops (rt (σ i)) oip) ≤ hops" by simp
with ⟨Suc hops < the (dhops (rt (σ i)) oip)⟩ have False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
  else rt (σ i) a) = rt (σ i)" ..
next
assume "the (flag (rt (σ i)) oip) = inv"
with ⟨the (flag (rt (σ sip)) oip) = val⟩ have False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
  else rt (σ i) a) = rt (σ i)" ..
next
  from \(\langle oip \in kD(rt (\sigma \ sip)) \rangle\)
  show "(\(\lambda a.\) if \(a = oip\) then Some (the (rt (\(\sigma \ i\) oip)) else rt (\(\sigma \ i\) a)) = rt (\(\sigma \ i\))\)"
  by (auto dest!: kD_Some)
qed

with change have False ..
thus ?thesis ..
qed

next
assume "\(oip \in iD(rt (\sigma \ sip))\)"
with \(\langle\) the (flag (rt (\(\sigma \ ' i\)) oip) = val \rangle\) and a5 have "\(\sigma \ sip = i\)"
  by (metis f.distinct(1) iD_flag_is_inv)
from \(\langle oip \in iD(rt (\sigma \ sip)) \rangle\) have "\(\sigma \ sip = i\)" by simp

thus ?thesis ..
qed

next
assume osnlt: "osn < nsqn (rt (\(\sigma \ ' i\)) oip)" from \(\langle dip \in kD(rt (\(\sigma \ ' i\))) \rangle\) and \(\langle dip = oip\rangle\) have "\(dip \in kD(?rt1)\)" by simp
moreover from a3 have "\(dip \in kD(?rt2 dip)\)" by simp
moreover have "\(nsqn ?rt1 dip < nsqn (?rt2 dip) dip\)"
proof -
  have "\(nsqn ?rt1 oip = osn\)"
    by (simp add: \(\langle dip = oip\rangle\) nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "\(\ldots < nsqn (rt (\(\sigma \ sip\)) oip)\) using osnlt .
  also have "\(\ldots = nsqn (?rt2 oip) oip\)" by (simp add: change)
  finally show ?thesis
    using \(\langle dip = oip\rangle\) by simp
qed
ultimately show ?thesis
  by (rule rt_strictly_fresher_ltI)
next
assume osnleq: "osn = nsqn (rt (\(\sigma \ ' i\)) oip) \& the (dhops (rt (\(\sigma \ ' i\)) oip)) \leq hops"
moreover from \(\langle dip \in kD(?rt1)\rangle\) by simp
moreover have "\(\langle dip = oip\rangle\) have "\(oip \in kD(?rt2 oip)\)" by simp
moreover have "\(nsqn ?rt1 oip = nsqn (?rt2 oip)\) oip"
proof -
  from osnleq have "osn = nsqn (rt (\(\sigma \ sip\)) oip) ..
  also have "osn = nsqn ?rt1 oip"
    by (simp add: \(\langle dip = oip\rangle\) nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "\(nsqn (rt (\(\sigma \ sip\)) oip) = nsqn (?rt2 oip) oip\)"
    by (simp add: change)
  finally show ?thesis .
qed

moreover have "\(\pi_5(\langle\) (\(\sigma \ ' i\) oip)\) < \(\pi_5(\langle\) (\(\sigma \ ' i\) oip)\)\)"
proof -
  from osnleq have "the (dhops (rt (\(\sigma \ sip\)) oip)) \leq hops" ..
  moreover from \(\langle oip \in vD (rt (\(\sigma \ sip\))\rangle\) have "\(oip \in kD(rt (\sigma \ sip))\)" by auto
  ultimately have "\(\pi_5(\langle\) (\(\sigma \ sip\)) oip)\) \leq hops"
    by (auto simp add: proj5_eq_dhops)
  also from change after have "hops < \(\pi_5(\langle\) (\(\sigma \ ' i\) oip)\)\)"
    by (simp add: proj5_eq_dhops) (metis dhops_update_changed_lessI)
  finally have "\(\pi_5(\langle\) (\(\sigma \ sip\)) oip)\) < \(\pi_5(\langle\) (\(\sigma \ ' i\) oip)\)\)" .
  with change after show ?thesis by simp
ultimately have "?rt1 ⊏ oip ?rt2 oip" 
by (rule rt_strictly_fresher_eqI)
with (dip = oip) show ?thesis by simp 
qed
qed
qed
qed

have "opad v i |= (otherwith (((=)) {i} (orecvmsg (λσ m. msg_fresh σ m 
  ∧ msg_zhops m)));
other quality_increases {i} →)
o nl Γ_{AODV}
(λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip))))
  ∧ the (nhop (rt (σ i)) dip) ≠ dip
  ⟹ rt (σ i) □ dip rt (σ (the (nhop (rt (σ i)) dip)))))" 
proof (inv_cterms inv add: onl_oinvariant_sterms [OF aodv_wf rreq_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rerr_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf odsn_rrep [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf oaddpreRT_welldefined]
solve: basic update_0_unk invalidate rreq_rrep_update 
simp add: seqlsimp)

fix σ σ' p l
assume or: "((σ, p) ∈ oreachable (opad v i) (?S i) (other quality_increases {i}))" 
and "other quality_increases {i} σ σ'" 
and ll: "l ∈ labels Γ_{AODV} p" 
and pre: "∀dip. dip ∈ vD (rt (σ i))
  ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
  ∧ the (nhop (rt (σ i)) dip) ≠ dip
  ⟹ rt (σ i) □ dip rt (σ (the (nhop (rt (σ i)) dip))))"

from this(1-2) have or': "(σ', p) ∈ oreachable (opad v i) (?S i) (other quality_increases {i})" 
by - (rule oreachable_other')

from or and ll have next_hop: "∀dip. let nhip = the (nhop (rt (σ i)) dip)
  in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip
  ⟹ dip ∈ kD(rt (σ nhip))
  ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip" 
by (auto dest!: onl_oinvariant_weakenD [OF seq_compare_next_hop'])

from or and ll have unk_hops_one: "∀dip. dip ∈ kD (rt (σ i)). sqn (rt (σ i)) dip = 0
  ⟹ sqnf (rt (σ i)) dip = unk
  ∧ the (dhops (rt (σ i)) dip) = 1
  ∧ the (nhop (rt (σ i)) dip) = dip" 
by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one 
[OF oaddpreRT_welldefined]]
otherwith_actionD 
simp: seqlsimp)

from (other quality_increases {i} σ σ') have "σ' i = σ i" by auto
hence "quality_increases (σ i) (σ' i)" by auto
with (other quality_increases {i} σ σ') have "∀j. quality_increases (σ j) (σ' j)" 
by - (erule otherE, metis singleton_iff)

show "∀dip. dip ∈ vD (rt (σ i))
  ∧ dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))
  ∧ the (nhop (rt (σ i)) dip) ≠ dip
  ⟹ rt (σ i) □ dip rt (σ' (the (nhop (rt (σ i)) dip)))" 
proof clarify
fix dip
assume \( \text{"dip} \in \text{vD}(\text{rt} (\sigma' \ i))\)
and \( \text{"dip} \in \text{vD}(\text{rt} (\sigma' \ (\text{nhop} (\text{rt} (\sigma' \ i)) \ \text{dip})))\)
and \( \text{"the} \ (\text{nhop} (\text{rt} (\sigma' \ i)) \ \text{dip}) \neq \text{dip}" \)

from this(1) and \( \sigma' \ i = \sigma \ i \) have \( \text{"dip} \in \text{vD}(\text{rt} (\sigma \ i))\)
and \( \text{"dip} \in \text{kD}(\text{rt} (\sigma \ i))\)

by auto

from \( \langle \text{the} \ (\text{nhop} (\text{rt} (\sigma' \ i)) \ \text{dip}) \neq \text{dip} \rangle \) and \( \sigma' \ i = \sigma \ i \)
have \( \text{"the} \ (\text{nhop} (\text{rt} (\sigma \ i)) \ \text{dip}) \neq \text{dip}" \) (is \( ?\text{nhip} \neq \_\) ) by simp

with \( \text{dip} \in \text{kD}(\text{rt} (\sigma \ i))\) and next_hop
have \( \text{"dip} \in \text{kD}(\text{rt} (\sigma \ (\text{nhip})))" \)
and nsqns: \( \text{"nsqn} \ (\text{rt} (\sigma \ i)) \ \text{dip} \leq \text{nsqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip}" \)
by \( \langle \text{auto} \ \text{simp: Let_def} \rangle \)

have \( 0 < \text{sqn} \ (\text{rt} (\sigma \ i)) \ \text{dip}" \)
proof \( \langle \text{rule neq0_conv [THEN iffD1, OF notI]} \rangle \)
assume \( \text{"sqn} \ (\text{rt} (\sigma \ i)) \ \text{dip} = 0" \)
with \( \text{dip} \in \text{kD}(\text{rt} (\sigma \ i))\) and unk_hops_one
have \( \text{"?nhip} = \text{dip}" \) by simp
with \( ?\text{nhip} \neq \text{dip} \) show False ..
qed

also have \( \ldots = \text{nsqn} \ (\text{rt} (\sigma \ i)) \ \text{dip}" \)
by \( \langle \text{rule vD_nsqn_sqn [OF dip\in\text{vD}(\text{rt} (\sigma \ i)), THEN sym]} \rangle \)
also have \( \ldots \leq \text{nsqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip}" \)
by \( \langle \text{rule nsqns} \rangle \)
also have \( \ldots \leq \text{sqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip}" \)
by \( \langle \text{rule nsqn} \rangle \)
finally have \( 0 < \text{sqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip}" \).

have \( \text{"rt} (\sigma \ i) \sqsubseteq \text{dip} \ \text{rt} (\sigma' \ ?\text{nhip})" \)
proof \( \langle \text{cases "dip} \in \text{vD}(\text{rt} (\sigma \ ?\text{nhip}))\rangle \)
assume \( \text{"dip} \in \text{vD}(\text{rt} (\sigma \ ?\text{nhip}))" \)
with pre \( \text{dip} \in \text{vD}(\text{rt} (\sigma \ i))\) and \( ?\text{nhip} \neq \text{dip} \)
have \( \text{"rt} (\sigma \ i) \sqsubseteq \text{dip} \ (\sigma \ ?\text{nhip})" \) by auto
moreover from \( \forall j. \ \text{quality_increases} (\sigma \ j) \ (\sigma' \ j) \)
have \( \text{"quality_increases} (\sigma \ ?\text{nhip}) \ (\sigma' \ ?\text{nhip})" \) ..
ultimately show \( \text{?thesis} \)
using \( \text{dip} \in \text{kD}(\text{rt} (\sigma \ ?\text{nhip})) \)
by \( \langle \text{rule strictly_fresher_quality_increases_right} \rangle \)

next
assume \( \text{"dip} \in \text{vD}(\text{rt} (\sigma \ ?\text{nhip}))" \)
with \( \text{dip} \in \text{kD}(\text{rt} (\sigma \ ?\text{nhip})) \) have \( \text{"dip} \in \text{iD}(\text{rt} (\sigma \ ?\text{nhip}))" \) ..
hence \( \text{"the} \ (\text{flag} (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip}) = \text{inv}" \)
by auto
have \( \text{"nsqn} \ (\text{rt} (\sigma \ i)) \ \text{dip} \leq \text{nsqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip}" \)
by \( \langle \text{rule nsqns} \rangle \)
also from \( \text{dip} \in \text{iD}(\text{rt} (\sigma \ ?\text{nhip})) \)
have \( \ldots = \text{sqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip} - 1" \) ..
also have \( \ldots < \text{sqn} \ (\text{rt} (\sigma' \ ?\text{nhip})) \ \text{dip}" \)
proof -
from \( \forall j. \ \text{quality_increases} (\sigma \ j) \ (\sigma' \ j) \)
have \( \text{"quality_increases} (\sigma \ ?\text{nhip}) \ (\sigma' \ ?\text{nhip})" \) ..
hence \( \forall \text{ip}. \ \text{sqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{ip} \leq \text{sqn} \ (\text{rt} (\sigma' \ ?\text{nhip})) \ \text{ip}" \) by auto
hence \( \text{"sqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip} \leq \text{sqn} \ (\text{rt} (\sigma' \ ?\text{nhip})) \ \text{dip}" \) ..
with \( 0 < \text{sqn} \ (\text{rt} (\sigma \ ?\text{nhip})) \ \text{dip} \) show \( \text{?thesis} \) by auto
qed
also have \( \ldots = \text{nsqn} \ (\text{rt} (\sigma' \ ?\text{nhip})) \ \text{dip}" \)
proof \( \langle \text{rule vD_nsqn_sqn [THEN sym]} \rangle \)
from \( \text{dip} \in \text{vD}(\sigma' \ (\text{the} \ (\text{nhop} (\text{rt} (\sigma' \ i)) \ \text{dip})))) \) and \( \sigma' \ i = \sigma \ i \)
show \( \text{"dip} \in \text{vD}(\text{rt} (\sigma' \ ?\text{nhip}))" \) by simp
qed
finally have \( \text{"nsqn} \ (\text{rt} (\sigma \ i)) \ \text{dip} < \text{nsqn} \ (\text{rt} (\sigma' \ ?\text{nhip})) \ \text{dip}" \).

moreover from \( \text{dip} \in \text{vD}(\text{rt} (\sigma' \ (\text{the} \ (\text{nhop} (\text{rt} (\sigma' \ i)) \ \text{dip})))) \) and \( \sigma' \ i = \sigma \ i \)
have "dip ∈ kD(rt (σ’ nhhip))" by auto
ultimately show "rt (σ i) ⊏ dip rt (σ’ nhhip)"
using (dip ∈ kD(rt (σ i))) by - (rule rt_strictly_fresher_ltI)
qed
with (σ’ i = σ i) show "rt (σ’ i) ⊏ dip rt (σ’ (the (nhop (rt (σ’ i)) dip)))"
by simp
qed
qed
thus ?thesis unfolding Let_def .
qed

lemma seq_compare_next_hop:
  fixes w
  shows "opaodv i | (otherwith ((=)) {i} (orecvmsg msg_fresh),
  other quality_increases {i} →)
  global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →
dip ∈ kD(rt (σ nhip)) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"
by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onlD)

lemma seq_nhop_quality_increases:
  shows "opaodv i | (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other quality_increases {i} →)
  global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip →
(rt (σ i)) ⊏ dip (rt (σ nhip)))"
by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onlD)
end

1.10 Routing graphs and loop freedom

theory A_Loop_Freedom
imports A_Aodv_Predicates A_Fresher
begin

Define the central theorem that relates an invariant over network states to the absence of loops in the associate routing graph.

definition
  rt_graph :: "(ip ⇒ state) ⇒ ip ⇒ ip rel"
where
  "rt_graph σ = (λdip. 
  {(ip, ip') | ip ip' dsn dsk hops pre.
  ip ≠ dip ∧ rt (σ ip) dip = Some (dsn, dsk, val, hops, ip', pre))"

Given the state of a network σ, a routing graph for a given destination ip address dip abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

lemma rt_graphE [elim]:
  fixes n dip ip ip'
  assumes "(ip, ip') ∈ rt_graph σ dip"
  shows "ip ≠ dip ∧ (∃r. rt (σ ip) = r ∧ (∃dsn dsk hops pre. r dip = Some (dsn, dsk, val, hops, ip', pre)))"
  using assms unfolding rt_graph_def by auto

lemma rt_graph_vD [dest]:
  "\ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip \implies dip ∈ vD(rt (σ ip))"
  unfolding rt_graph_def vD_def by auto

lemma rt_graph_vD_trans [dest]:
  "\ip ip' σ dip. (ip, ip') ∈ (rt_graph σ dip)+ \implies dip ∈ vD(rt (σ ip))"
  by (erule converse_tranclE) auto
lemma rt_graph_not_dip [dest]:
"\(\forall ip \ ip' \ \sigma \ dip. (ip, ip') \in rt_graph \ \sigma \ dip \implies ip \neq dip\)"
unfolding rt_graph_def by auto

lemma rt_graph_not_dip_trans [dest]:
"\(\forall ip \ ip' \ \sigma \ dip. (ip, ip') \in (rt_graph \ \sigma \ dip)^+ \implies ip \neq dip\)"
by (erule converse_tranclE) auto

NB: the property below cannot be lifted to the transitive closure

lemma rt_graph_nhip_is_nhop [dest]:
"\(\forall ip \ ip' \ \sigma \ dip. (ip, ip') \in rt_graph \ \sigma \ dip \implies ip' = the (nhop (rt (\sigma \ ip)) dip)\)"
unfolding rt_graph_def by auto

theorem inv_to_loop_freedom:
assumes "\(\forall i \ dip. let nhip = the (nhop (rt (\sigma \ i)) dip)\)
\(\in dip \in vD (rt (\sigma \ i)) \cap vD (rt (\sigma \ nhip)) \wedge nhip \neq dip \implies rt (\sigma \ i) \sqsubseteq dip \ (rt (\sigma \ nhip))\)"
shows "\(\forall dip. irrefl ((rt_graph \ \sigma \ dip)^+)\)"
using assms proof (intro allI)
fix \sigma :: "ip \Rightarrow state" and dip
assume inv:
"\(\forall ip \ dip. let nhip = the (nhop (rt (\sigma \ ip)) dip)\)
\(\in dip \in vD (rt (\sigma \ i)) \cap vD (rt (\sigma \ nhip)) \wedge nhip \neq dip \implies rt (\sigma \ i) \sqsubseteq dip \ (rt (\sigma \ nhip))\)"

{ fix ip ip'
  assume "\((ip, ip') \in (rt_graph \ \sigma \ dip)^+\)"
  and "\(\text{dip} \in vD (rt (\sigma \ ip'))\)"
  and "\(ip' \neq dip\)"
  hence "\(rt (\sigma \ ip) \sqsubseteq dip \ (rt (\sigma \ ip'))\)"
  proof induction
  fix nhip
  assume "\((ip, nhip) \in rt_graph \ \sigma \ dip\)" have "\(\text{dip} \in vD (rt (\sigma \ ip))\)"
  and "\(nhip \neq dip\)"
  from \((ip, nhip) \in rt_graph \ \sigma \ dip\) have "\(\text{dip} \in vD (rt (\sigma \ i)) \cap vD (rt (\sigma \ nhip)) \wedge nhip \neq dip \implies rt (\sigma \ i) \sqsubseteq dip \ (rt (\sigma \ nhip))\)"
  by auto
  from \(\text{dip} \in vD (rt (\sigma \ i)) \cap vD (rt (\sigma \ nhip))\) have "\(\text{dip} \in vD (rt (\sigma \ i)) \cap vD (rt (\sigma \ nhip))\)" ..
  with \(\text{nhip} = the (nhop (rt (\sigma \ ip)) dip)\)
  and \(\text{nhip} \neq dip\)
  and inv
  show "\(rt (\sigma \ ip) \sqsubseteq dip \ (rt (\sigma \ nhip))\)"
  by (clarsimp simp: Let_def)
  next
  fix nhip nhip'
  assume "\((ip, nhip) \in rt_graph \ \sigma \ dip\)" and "\(nhip \neq dip\)"
  and IH: "\(\text{dip} \in vD (rt (\sigma \ i)) \cap vD (rt (\sigma \ nhip))\)" ..
  from \((nhip, nhip') \in rt_graph \ \sigma \ dip\) have 1: "\(\text{dip} \in vD (rt (\sigma \ i))\)"
  and 2: "\(nhip \neq dip\)"
  and 3: "\(nhip' = the (nhop (rt (\sigma \ ip)) dip)\)
  by auto
  from 1 2 have "\(rt (\sigma \ i) \sqsubseteq dip \ (rt (\sigma \ nhip))\)" by (rule IH)
  also have "\(rt (\sigma \ nhip) \sqsubseteq dip \ (rt (\sigma \ nhip'))\)"
  proof
  from \(\text{dip} \in vD (rt (\sigma \ nhip)) \cap vD (rt (\sigma \ nhip'))\) have "\(\text{dip} \in vD (rt (\sigma \ nhip)) \cap vD (rt (\sigma \ nhip'))\)" ..
  with \(\text{nhip'} \neq dip\)
  and \(\text{nhip'} = the (nhop (rt (\sigma \ nhip)) dip)\)
  and inv
show "rt (σ nhip) ⊏ rt (σ nhip)"
  by (clarsimp simp: Let_def)
qed

finally show "rt (σ ip) ⊏ rt (σ nhip)".
qed

} note fresher = this

show "irrefl ((rt_graph σ dip)⁺)"
unfolding irrefl_def proof (intro allI notI)
  fix ip
  assume "(ip, ip) ∈ (rt_graph σ dip)⁺"
  moreover then have "dip ∈ vD (rt (σ ip))"
    and "ip ≠ dip"
  by auto
  ultimately have "rt (σ ip) ⊏ dip rt (σ nhip)" by (rule fresher)
  thus False by simp
qed
qed

1.11 Lift and transfer invariants to show loop freedom

theory A_Aodv_Loop_Freedom
imports AWN.OClosed_Transfer AWN.Qmsg_Lifting A_Global_Invariants A_Loop_Freedom
begin

lift to parallel processes with queues

lemma par_step_no_change_on_send_or_receive:
  fixes σ s a σ' s'
  assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos Γ AODV i) (seqp_sos Γ QMSG)"
    and "a ≠ τ"
  shows "σ' i = σ i"
using assms by (rule qmsg_no_change_on_send_or_receive)

lemma par_nhop_quality_increases:
  shows "opaodv i ((i qmsg |= (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other_quality_increases {i} →)
    global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊏ dip (rt (σ nhip)))""
proof (rule lift_into_qmsg [OF seq_nhop_quality_increases])
  show "opaodv i |A (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other_quality_increases {i} →)
    globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
proof (rule ostep_invariant_weakenE [OF oquality_increases], simp_all)
  fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
  assume "call Γ AODV (λ((σ, _, _), (σ', _)). ∀j. quality_increases (σ j) (σ' j)) t"
  thus "quality_increases (fst (fst t) i) (fst (snd (snd t)) i)"
  by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)
next
  fix σ σ' a
  assume "otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other_quality_increases {i} →)
  show "otherwith quality_increases {i} (orecvmsg (λ_. rreq_rrep_sn)) σ σ' a"
  by - (erule weaken_otherwith, auto)
qed
qed auto

lemma par_rreq_rrep_sn_quality_increases:
  "opaodv i (i qmsg |=A (λσ _. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_. _. True) {i} →)
   globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
proof -
have "opaodv i \(\vdash\) _ (\(\lambda\)_ .. orecvmsg (\(\lambda\)_ .. rreq_rrep_sn) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), _, \(\sigma\)'._). quality_increases (\(\sigma\) i) (\(\sigma\)' i))"
by (rule ostep_invariant_weakenE [OF olocal_quality_increases])
(auto dest!: oadv_ex_label)
hence "opaodv i \(\vdash\) _ (\(\lambda\)_ .. orecvmsg (\(\lambda\)_ .. rreq_rrep_sn) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), a, \(\sigma\)'._). anycast (msg_fresh \(\sigma\) a) a)"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thus \(\text{thesis}\) by rule auto
qed

lemma par_rreq_rrep_nsqn_fresh_any_step:
shows "opaodv i \(\vdash\) _ (\(\lambda\)_ .. orecvmsg (\(\lambda\)_ .. rreq_rrep_sn) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), a, \(\sigma\)'._). anycast (msg_fresh \(\sigma\) a) a)"
proof -
have "opaodv i \(\vdash\) _ (\(\lambda\)_ .. (orecvmsg (\(\lambda\)_ .. rreq_rrep_sn))) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), a, \(\sigma\)'._). anycast (msg_fresh \(\sigma\) a) a)"
proof (rule ostep_invariant_weakenE [OF olocal_quality_increases])
fix t 
assume "onll \(\Gamma\)\(\lambda\)_ (\(\lambda(\sigma\), _, a, _)._ . anycast (msg_fresh \(\sigma\) a) t)"
thus "globala (\(\lambda(\sigma\), a, \(\sigma\)'._). anycast (msg_fresh \(\sigma\) a) t)"
by (cases t) (clarsimp dest!: onllD onllD, metis aodv_ex_label)
qed auto
hence "opaodv i \(\vdash\) _ (\(\lambda\)_ .. (orecvmsg (\(\lambda\)_ .. rreq_rrep_sn))) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), a, \(\sigma\)'._). anycast (msg_fresh \(\sigma\) a) a)"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thus \(\text{thesis}\) by rule auto
qed

lemma par_anycast_msg_zhops:
shows "opaodv i \(\vdash\) _ (\(\lambda\)_ .. orecvmsg (\(\lambda\)_ .. rreq_rrep_sn) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), a, _)._ . anycast msg_zhops a)"
proof -
from anycast_msg_zhops_initiali_aodv oadv_trans aodv_trans
have "opaodv i \(\vdash\) _ (act TT, other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
seqll i (onll \(\Gamma\)_ \(\lambda(\sigma\), a, _)._ . anycast msg_zhops a))"
by (rule open_seq_step_invariant)
hence "opaodv i \(\vdash\) _ (\(\lambda\)_ .. orecvmsg (\(\lambda\)_ .. rreq_rrep_sn) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), a, _)._ . anycast msg_zhops a)"
proof (rule ostep_invariant_weakenE)
fix t :: "(((nat \Rightarrow state) \times (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
assume "seqll i (onll \(\Gamma\)_ \(\lambda(\sigma\), a, _)._ . anycast msg_zhops a)) t)"
thus "globala (\(\lambda(\sigma\), a, _)._ . anycast msg_zhops a) t)"
by (cases t) (clarsimp dest!: seqllD onllD, metis aodv_ex_label)
qed simp_all
hence "opaodv i \(\vdash\) _ (\(\lambda\)_ .. orecvmsg (\(\lambda\)_ .. rreq_rrep_sn) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), a, _)._ . anycast msg_zhops a)"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thus \(\text{thesis}\) by rule auto
qed

1.11.1 Lift to nodes

lemma node_step_no_change_on_send_or_receive:
assumes "((\(\sigma\), NodeS i P R), a, (\(\sigma\)'., NodeS i' P' R')) \in onode_sos 
(oparp_sos i (oseqp_sos \(\Gamma\)_ aodv_trans i) (seqp_sos \(\Gamma\)_ QMSG))"
and "a \neq \tau"
shows "\(\sigma\)' i = \sigma i"
using assms
by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

lemma node_nhop_quality_increases:
shows "% i : opaodv i \(\vdash\) _ (\(\lambda\)_ .. orecvmsg (\(\lambda\)_ .. rreq_rrep_sn) \(\sigma\), other (\(\lambda\)_ .. True) \{i\} \rightarrow\) 
globala (\(\lambda(\sigma\), a, _)._ . anycast msg_zhops a)"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thus \(\text{thesis}\) by rule auto
qed
\[ \text{oarrivemsg } (\lambda M. \text{msg\_fresh } M), \]
\[ \text{other quality\_increases } (i) \]
\[ \rightarrow \text{global } (\lambda \sigma. \forall \lambda \text{ dip} \cdot \text{let nhip } = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip}) \]
\[ \text{in dip } \in \mathcal{V} (\text{rt } (\sigma i)) \cap \mathcal{V} (\text{rt } (\sigma \text{ nhip})) \land \text{nhip } \neq \text{ dip} \]
\[ \rightarrow (\text{rt } (\sigma i)) \not\subseteq \text{ dip } (\text{rt } (\sigma \text{ nhip})))" \]
by (rule node\_lift [OF par\_nhop\_quality\_increases]) auto

**Lemma node\_quality\_increases:**
\[ "(i : \text{opaodv } i \langle \langle i \text{ qmsg : } R \rangle_o \rangle =_{\text{A}} (\lambda \sigma \cdot \text{ oarrivemsg } (\lambda. \text{ req\_rrep\_sn }) \sigma), \]
\[ \text{other } (\lambda \cdot \text{ True}) \{i\} \rightarrow \]
\[ \text{globala } (\lambda (\sigma, a, \sigma'). \text{ quality\_increases } (\sigma i) (\sigma' i))" \]
by (rule node\_lift\_step\_statelessassm [OF par\_req\_rrep\_sn\_quality\_increases]) simp

**Lemma node\_rreq\_rrep\_nsqn\_fresh\_any\_step:**
\[ "(i : \text{opaodv } i \langle \langle i \text{ qmsg : } R \rangle_o \rangle =_{\text{A}} (\lambda \sigma \cdot \text{ oarrivemsg } (\lambda. \text{ req\_rrep\_sn }) \sigma), \]
\[ \text{other } (\lambda \cdot \text{ True}) \{i\} \rightarrow \]
\[ \text{globala } (\lambda (\lambda, a, \sigma'). \text{ castmsg } (\text{msg\_fresh } a))" \]
by (rule node\_lift\_anycast\_statelessassm [OF par\_rreq\_rrep\_nsqn\_fresh\_any\_step])

**Lemma node\_anycast\_msg\_zhops:**
\[ "(i : \text{opaodv } i \langle \langle i \text{ qmsg : } R \rangle_o \rangle =_{\text{A}} (\lambda \sigma \cdot \text{ oarrivemsg } (\lambda. \text{ rreq\_rrep\_sn }) \sigma), \]
\[ \text{other } (\lambda \cdot \text{ True}) \{i\} \rightarrow \]
\[ \text{globala } (\lambda (\lambda, a). \text{ castmsg } \text{msg\_zhops } a)" \]
by (rule node\_lift\_anycast\_statelessassm [OF par\_anycast\_msg\_zhops])

**Lemma node\_silent\_change\_only:**
\[ "(i : \text{opaodv } i \langle \langle i \text{ qmsg : } R \rangle_o \rangle =_{\text{A}} (\lambda \sigma \cdot \text{ oarrivemsg } (\lambda. \text{ True}) \sigma), \]
\[ \text{other } (\lambda \cdot \text{ True}) \{i\} \rightarrow \]
\[ \text{globala } (\lambda (\sigma, a, \sigma'). \text{ a } \neq \text{ \tau } \rightarrow \sigma' i = \sigma i)" \]
proof (rule ostep\_invariantI, simp (no_asm), rule impI)
\[ \text{fix } \sigma \zeta a \sigma' \zeta' \]
\[ \text{assume or: } "(\sigma, \zeta) \in \text{reachable } (\langle i : \text{opaodv } i \langle \langle i \text{ qmsg : } R_i \rangle_o \rangle =_{\text{A}} (\lambda \sigma \cdot \text{ oarrivemsg } (\lambda. \text{ True}) \sigma), \]
\[ \text{other } (\lambda \cdot \text{ True}) \{i\} \rightarrow \]
\[ \text{globala } (\lambda (\lambda, a, \sigma'). \text{ a } \neq \text{ \tau } \rightarrow \sigma' i = \sigma i)" \]
and tr: "((\sigma, \zeta), a, (\sigma', \zeta')) \in \text{trans } (\langle i : \text{opaodv } i \langle \langle i \text{ qmsg : } R_i \rangle_o \rangle =_{\text{A}} (\lambda \sigma \cdot \text{ oarrivemsg } (\lambda. \text{ True}) \sigma), \]
\[ \text{other } (\lambda \cdot \text{ True}) \{i\} \rightarrow \]
\[ \text{globala } (\lambda (\lambda, a, \sigma'). \text{ a } \neq \text{ \tau } \rightarrow \sigma' i = \sigma i)" \]
from or obtain p R where "(\zeta = \text{NodeS } i p R)"
by (drule node\_net\_state, metis)
with tr have "((\sigma, \text{NodeS } i p R), a, (\sigma', \zeta'))
\[ \in \text{onode\_sos } \text{(oparp\_sos } i \langle \text{trans } (\text{opaodv } i) \rangle \langle \text{trans qmsg} \rangle)" \]
by simp
thus "\sigma' i = \sigma i" using (a \neq \tau_n)\]
by (cases rule: onode\_sos\_cases)
(auto elim: qmsg\_no\_change\_on\_send\_or\_receive)
qed

### 1.11.2 Lift to partial networks

**Lemma arrive\_rreq\_rrep\_nsqn\_fresh\_inc\_sn [simp]:**
\[ \text{assumes } "\text{oarrivemsg } (\lambda M. \text{msg\_fresh } M) (\lambda M. \text{msg\_fresh } M) (\lambda \sigma. \text{ \tau } M) \sigma M" \]
\[ \text{shows } "\text{oarrivemsg } (\lambda. \text{ req\_rrep\_sn }) \sigma M" \]
using assms by (cases m) auto

**Lemma opnet\_nhop\_quality\_increases:**
\[ "\text{opnet } (\lambda i. \text{opaodv } i \langle \langle i \text{ qmsg : } R \rangle_o \rangle =_{\text{A}} (\lambda \sigma. \text{msg\_fresh } M) \sigma M) \sigma M, \]
\[ \text{other quality\_increases } (\text{net\_tree\_ips } p) \rightarrow \]
\[ \text{globala } (\lambda \sigma. \forall i \in \text{net\_tree\_ips } p. \forall \text{ dip. let nhip } = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip}) \]
\[ \text{in dip } \in \mathcal{V} (\text{rt } (\sigma i)) \cap \mathcal{V} (\text{rt } (\sigma \text{ nhip})) \land \text{nhip } \neq \text{ dip} \]
\[ \rightarrow (\text{rt } (\sigma i)) \not\subseteq \text{ dip } (\text{rt } (\sigma \text{ nhip})))" \]
proof (rule pnet\_lift [OF node\_nhop\_quality\_increases])
\[ \text{fix } i R \]

166
have "(i : opaodv i ⟨⟨i ⟩⟩ qmsg : R) ⊢ A (λσ_. oarrivemsg (λ_. rreq_rrep_sn) σ, 
other (λ_. True) {i} →) globala (λ(σ, a, σ'). 
castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
proof (rule inclosed_closed)
fix σ s a σ’ s’
assume or: "(σ, s) ∈ ooreachable ((i : opaodv i ⟨⟨i ⟩⟩ qmsg : R) o) 
(λσ_. oarrivemsg (λ_. rreq_rrep_sn) σ) 
(other (λ_. True) {i})"
and tr: "((σ, s), a, (σ’, s’)) ∈ otrans ((i : opaodv i ⟨⟨i ⟩⟩ qmsg : R) o)"
and am: "oarrivemsg (λ_. rreq_rrep_sn) σ a"
from or tr am have "castmsg (msg_fresh σ) a"
  by (auto dest!: ostep_invariantD [OF node_rreq_rrep_msnq_fresh_any_step])
moreover from or tr am have "castmsg (msg_zhops m) a"
  by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
ultimately show "castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a"
  by (case_tac a) auto
qed
thus "(i : opaodv i ⟨⟨i ⟩⟩ qmsg : R) ⊢ A 
(λσ_. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ, 
other quality_increases {i} →) globala (λ(σ, a, σ’). 
castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
  by rule auto
next
fix i R
show "(i : opaodv i ⟨⟨i ⟩⟩ qmsg : R) ⊢ A 
(λσ_. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ, 
other quality_increases {i} →) globala (λ(σ, a, σ’). 
a ≠ τ ∧ (∀d. a ≠ i:deliver(d)) → σ i = σ’ i)"
  by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto
next
fix i R
show "(i : opaodv i ⟨⟨i ⟩⟩ qmsg : R) ⊢ A 
(λσ_. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ, 
other quality_increases {i} →) globala (λ(σ, a, σ’). 
a = τ ∨ (∃d. a = i:deliver(d)) → quality_increases (σ i) (σ’ i))"
  by (rule ostep_invariant_weakenE [OF node_quality_increases]) auto
qed simp_all

1.1.3 Lift to closed networks

lemma onet_nhops_quality_increases:
sows "oclosed (opnet (λi. opaodv i ⟨⟨i ⟩⟩ qmsg) p) 
  ⊢ (λ_. True, other quality_increases (net_tree_ips p) →) 
global (λσ. ∀i:net_tree_ips p. ∀dip. 
  let nhhip = the (nhop (rt (σ i)) dip) 
  in dip ∈ νD (rt (σ i)) ∩ νD (rt (σ nhhip)) ∧ nhhip ≠ dip 
  → (rt (σ i) ⊆ dip (rt (σ nhhip))))"
(is "_= ⊢ (., ?U →) ?inv")
proof (rule inclosed_closed)
from onet_nhops_quality_increases
show "opnet (λi. opaodv i ⟨⟨i ⟩⟩ qmsg) p 
  ⊢ (otherwihh ((=)) (net_tree_ips p) inclosed, ?U →) ?inv"
proof (rule inclosed_closed)
fir σ σ’ : "ip ⇒ state" and a :: "msg node_action"
assume "otherwihh ((=)) (net_tree_ips p) inclosed σ σ’ a"
thus "otherwihh ((=)) (net_tree_ips p) 
  (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)) σ σ’ a"
proof (rule otherwihhE)
fir σ : "ip ⇒ state" and a :: "msg node_action"
assume "inclosed σ a"
thus "oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ a"
proof (cases a)
  fix ii ni ms
  assume "a = ii ni.arrive(ms)"
moreover with \( \text{inclosed} \ \sigma \ a \) obtain \( d \) \( di \) where \( \text{ms} = \text{newpkt}(d, di) \)
by \((\text{cases } \text{ms})\) \(\text{auto}\)
ultimately show \(?\text{thesis}\) by \(\text{simp}\)
\(\text{qed}\) \(\text{simp\_all}\)
\(\text{qed}\)
\(\text{qed}\)

1.11.4 Transfer into the standard model

interpretation \(\text{aodv}\_\text{openproc}: \text{openproc paodv opaodv id}\)
rewrites \"\(\text{aodv}\_\text{openproc}.\text{initmissing} = \text{initmissing}\)"
proof -
  show \"\(\text{openproc paodv opaodv id}\)"
  proof unfold_locales
    fix \(i::i\p\)
    have \"\((\sigma, \zeta). (\sigma i, \zeta) \in \sigma_{AODV} \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst } \sigma_{AODV} ' j)) \subseteq \sigma_{AODV}'."\)
    unfolding \(\sigma_{AODV}\_\text{def}\) \(\sigma_{AODV}'.\_\text{def}\)
    proof \((\text{rule equalityD1})\)
      show \"\((\forall f p. ((\sigma, \zeta). (\sigma i, \zeta) \in \{(f i, p)\}) \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst } \{(f j, p)\}) = \{(f, p)\})\)"\)
      by \((\text{rule set_eqI})\) \(\text{auto}\)
    qed
    thus \"\(\{ ((\sigma, \zeta) | \sigma \zeta s. s \in \text{init } (\text{paodv } i) \land (\sigma i, \zeta) = \text{id } s \land (\forall j. j \neq i \rightarrow \sigma j \in \text{(fst o id) } ' \text{init } (\text{paodv } j)) \} \subseteq \text{init } (\text{opaodv } i)\)"\)
    by \(\text{simp}\)
  next
    show \"\(\forall j. \text{init } (\text{paodv } j) \neq \{}\)\"
    unfolding \(\sigma_{AODV}\_\text{def}\) by \(\text{simp}\)
  next
    fix \(i s a s' \sigma \sigma'\)
    assume \"\(\sigma i = \text{fst } (\text{id } s)\)"\)
    and \"\(\sigma' i = \text{fst } (\text{id } s')\)"\)
    and \"\((s, a, s') \in \text{trans } (\text{paodv } i)\)"\)
    then obtain \(q q'\) where \"\(s = (\sigma i, q)\)"\)
    and \"\(s' = (\sigma' i, q')\)"\)
    and \"\(((\sigma i, q), a, (\sigma' i, q')) \in \text{trans } (\text{paodv } i)\)"\)
    by \((\text{cases } s, \text{cases } s')\) \(\text{auto}\)
    from \(\text{this}(3)\) have \"\(((\sigma, q), a, (\sigma', q')) \in \text{trans } (\text{opaodv } i)\)"\)
    by \(\text{simp (rule open_seqp_action [OF aodv_wf])}\)
    with \(s = (\sigma i, q)\) and \(s' = (\sigma' i, q')\),
    show \"\(((\sigma, \text{snd } (\text{id } s)), a, (\sigma', \text{snd } (\text{id } s'))) \in \text{trans } (\text{opaodv } i)\)"\)
    by \(\text{simp}\)
  qed
  then interpret \(\text{opn: openproc paodv opaodv id} .\)
  have \[simp\]: \"\(\forall i. (\text{SOME x. } x \in (\text{fst o id) } ' \text{init } (\text{paodv } i)) = \text{aodv}\_\text{init } i\)"\)
  unfolding \(\sigma_{AODV}\_\text{def}\) by \(\text{simp}\)
  hence \"\(\forall i. \text{openproc}\_\text{initmissing } \text{paodv id } i = \text{initmissing } i\)"\)
  unfolding \(\text{opn}\_\text{initmissing}\_\text{def}\) \(\text{opn}\_\text{someinit}\_\text{def}\) \(\text{initmissing}\_\text{def}\)
  by \((\text{auto split: option.split})\)
  thus \"\(\text{openproc}\_\text{initmissing } \text{paodv id } = \text{initmissing}\)" ..
  \(\text{qed}\)

interpretation \(\text{aodv}\_\text{openproc}\_\text{par}\_\text{qmsg}: \text{openproc_parq paodv opaodv id } qmsg\)
rewrites \"\(\text{aodv}\_\text{openproc}\_\text{par}\_\text{qmsg}.\text{netglobal} = \text{netglobal}\)"
and \"\(\text{aodv}\_\text{openproc}\_\text{par}\_\text{qmsg}.\text{initmissing} = \text{initmissing}\)"
proof -
  show \"\(\text{openproc_parq paodv opaodv id } qmsg\)"
  by \((\text{unfold_locales})\) \(\text{simp}\)
  then interpret \(\text{opq: openproc_parq paodv opaodv id } qmsg .\)
  have \(\text{im}: \"\(\forall \sigma. \text{openproc}\_\text{initmissing } (\lambda i. \text{paodv } i \langle qmsg \rangle) (\lambda(p, q). (\text{fst } (\text{id } p), \text{snd } (\text{id } p), q)) \sigma\)"
unfolding opq.initmissing_def opq.someinit_def initmissing_def
unfolding σ_AODV_def σ_QMSG_def by (clarsimp cong: option.case_cong)
thus "openproc.initmissing (λi. paodv i (qmsg) (λ(p, q). (fst (id p), snd (id p), q))) = initmissing"
by (rule ext)
have "∀P σ. openproc.netglobal (λi. paodv i (qmsg) (λ(p, q). (fst (id p), snd (id p), q))) P σ = netglobal P σ"
unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
unfolding σ_AODV_def σ_QMSG_def by (clarsimp cong: option.case_cong)
thus "openproc.netglobal (λi. paodv i (qmsg) (λ(p, q). (fst (id p), snd (id p), q))) = netglobal"
by (rule refl)
qed

lemma net_nhop_quality_increases:
assumes "wf_net_tree n"
shows "closed (pnet (λi. paodv i (qmsg) n) ||= netglobal (λσ. ∀ i dip. let nhip = the (nhop (rt (σ i)) dip) in dip ∈ vD (rt (σ i)) \cap vD (rt (σ nhip)) ∧ nhip ≠ dip → (rt (σ i)) ⊑ dip (rt (σ nhip)))"
(is "_ ||= netglobal (λσ. ∀ i. ?inv σ i)")
proof -
from ⟨wf_net_tree n⟩
have proto: "closed (pnet (λi. paodv i (qmsg) n) ||= netglobal (λσ. ∀ i∈net_tree_ips n. ∀ dip. let nhip = the (nhop (rt (σ i)) dip) in dip ∈ vD (rt (σ i)) \cap vD (rt (σ nhip)) ∧ nhip ≠ dip → (rt (σ i)) ⊑ dip (rt (σ nhip)))"
by (rule aodv_openproc_par_qmsg.close_opnet [OF _ onet_nhop_quality_increases])
show "?thesis"
unfolding invariant_def opnet_sos.opnet_tau1
proof (rule, simp only: aodv_openproc_par_qmsg.netglobalsimp
fst_initmissing_netgmap_pair_fst, rule allI)
fix σ i
assume sr: "σ ∈ reachable (closed (pnet (λi. paodv i (qmsg) n)) TT"
hence " ∀ i∈net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"
by - (drule invariantD [OF proto],
simp only: aodv_openproc_par_qmsg.netglobalsimp
fst_initmissing_netgmap_pair_fst)
thus "?inv (fst (initmissing (netgmap fst σ))) i"
proof (cases "i∈net_tree_ips n")
assume "i∉net_tree_ips n"
from sr have "σ ∈ reachable (pnet (λi. paodv i (qmsg) n)) TT" ..
hence "net_ips σ = net_tree_ips n" ..
with "i∉net_tree_ips n" have "i∉net_ips σ" by simp
hence "(fst (initmissing (netgmap fst σ))) i = aodv_init i"
by simp
thus ?thesis by simp
qed metis
qed

1.11.5 Loop freedom of AODV

theorem aodv_loop_freedom:
assumes "wf_net_tree n"
shows "closed (pnet (λi. paodv i (qmsg) n) ||= netglobal (λσ. ∀ dip. irrefl ((rt_graph σ dip)⁺))"
using assms by (rule aodv_openproc_par_qmsg.netglobal_weakenE
[OF net_nhop_quality_increases inv_to_loop_freedom])
end
Chapter 2

Variant B: Forwarding the Route Reply

Explanation [4, §10.2]: In AODV’s route discovery process, a RREP message from the destination node is unicast back along a route towards the originator of the RREQ message. Every intermediate node on the selected route will process the RREP message and, in most cases, forward it towards the originator node. However, there is a possibility that the RREP message is discarded at an intermediate node, which results in the originator node not receiving a reply. The discarding of the RREP message is due to the RFC specification of AODV [6] stating that an intermediate node only forwards the RREP message if it is not the originator node and it has created or updated a routing table entry to the destination node described in the RREP message. The latter requirement means that if a valid routing table entry to the destination node already exists, and is not updated when processing the RREP message, then the intermediate node will not forward the message. A solution to this problem is to require intermediate nodes to forward all RREP messages that they receive.

2.1 Predicates and functions used in the AODV model

theory B_Aodv_Data
imports B_Fwdrreps
begin

2.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

definition inc :: "sqn ⇒ sqn"
  where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ inc x"
  unfolding inc_def by simp

lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp

lemma inc_never_one' [simp, intro]: "inc x ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
  by simp

2.1.2 Modelling Routes

A route is a 6-tuple, (dsn, dsk, flag, hops, nhip, pre) where dsn is the ‘destination sequence number’, dsk is the ‘destination-sequence-number status’, flag is the route status, hops is the number of hops to the destination, nhip is the next hop toward the destination, and pre is the set of ‘precursor nodes’—those interested in hearing about changes to the route.

type_synonym r = "sqn × k × f × nat × ip × ip set"

definition proj2 :: "r ⇒ sqn" ("π₂")
where \( \pi_2 \equiv \lambda(dsn, _, _, _, _, _). dsn \)

definition proj3 :: "r ⇒ k" ("\(\pi_3\)"
where \( \pi_3 \equiv \lambda(_, dsk, _, _, _, _). dsk \)

definition proj4 :: "r ⇒ f" ("\(\pi_4\)"
where \( \pi_4 \equiv \lambda(_, _, flag, _, _, _). flag \)

definition proj5 :: "r ⇒ nat" ("\(\pi_5\)"
where \( \pi_5 \equiv \lambda(_, _, _, hops, _, _). hops \)

definition proj6 :: "r ⇒ ip" ("\(\pi_6\)"
where \( \pi_6 \equiv \lambda(_, _, _, _, nhip, _). nhip \)

definition proj7 :: "r ⇒ ip set" ("\(\pi_7\)"
where \( \pi_7 \equiv \lambda(_, _, _, _, _, pre). pre \)

lemma projs [simp]:
\[
\pi_2 (dsn, dsk, flag, hops, nhip, pre) = dsn
\]
\[
\pi_3 (dsn, dsk, flag, hops, nhip, pre) = dsk
\]
\[
\pi_4 (dsn, dsk, flag, hops, nhip, pre) = flag
\]
\[
\pi_5 (dsn, dsk, flag, hops, nhip, pre) = hops
\]
\[
\pi_6 (dsn, dsk, flag, hops, nhip, pre) = nhip
\]
\[
\pi_7 (dsn, dsk, flag, hops, nhip, pre) = pre
\]
by (clarsimp simp: proj2_def proj3_def proj4_def proj5_def proj6_def proj7_def)+

lemma proj3_pred [intro]: \("[ [ P kno; P unk ] ⇒ P (\pi_3 x) ]\"
by (rule k.induct)

lemma proj4_pred [intro]: \("[ [ P val; P inv ] ⇒ P (\pi_4 x) ]\"
by (rule f.induct)

lemma proj6_pair_snd [simp]:
fixes dsn' r
shows \("\pi_6 (dsn', snd (r)) = \pi_6 (r)\"
by (cases r) simp

2.1.3 Routing Tables

Routing tables map ip addresses to route entries.

type_synonym rt = "ip ⇒ r"

syntax
"_Sigma_route" :: "rt ⇒ ip ⇒ r" ("\(\sigma_{route}(_, _)\)"
translations
"\sigma_{route}(rt, dip)" ⇒ "rt dip"

definition sqn :: "rt ⇒ ip ⇒ sqn"
where \( "sqn rt dip \equiv case \sigma_{route}(rt, dip) of Some r ⇒ \pi_2 (r) \mid \text{None} ⇒ 0" \)

definition sqnf :: "rt ⇒ ip ⇒ k"
where \( "sqnf rt dip \equiv case \sigma_{route}(rt, dip) of Some r ⇒ \pi_3 (r) \mid \text{None} ⇒ \text{unk}" \)

abbreviation flag :: "rt ⇒ ip ⇒ f"
where \( "flag rt dip \equiv map_option \pi_4 (\sigma_{route}(rt, dip))" \)

abbreviation dhops :: "rt ⇒ ip ⇒ nat"
where \( "dhops rt dip \equiv map_option \pi_5 (\sigma_{route}(rt, dip))" \)

abbreviation nhop :: "rt ⇒ ip ⇒ ip"
where \( "nhop rt dip \equiv map_option \pi_6 (\sigma_{route}(rt, dip))" \)
abbreviation \texttt{precs} :: "rt \Rightarrow \text{ip \to ip \set}"
\hspace{1em}where \texttt{precs rt dip = map\_option \tau (\sigma\_route(rt, dip))}"

definition \texttt{vD} :: "rt \Rightarrow \text{ip \set}"
\hspace{1em}where \texttt{vD rt = \{dip. flag rt dip = Some val\}}"

definition \texttt{iD} :: "rt \Rightarrow \text{ip \set}"
\hspace{1em}where \texttt{iD rt = \{dip. flag rt dip = Some inv\}}"

definition \texttt{kD} :: "rt \Rightarrow \text{ip \set}"
\hspace{1em}where \texttt{kD rt = \{dip. rt dip \neq None\}}"

lemma \texttt{kD\_is\_vD\_and\_iD}: "kD rt = vD rt \cup iD rt"
\hspace{1em}unfolding kD_def vD_def iD_def by auto

lemma \texttt{vD\_iD\_gives\_kD \[simp\]}:
\hspace{1em}"\forall ip rt. ip \in vD rt \Rightarrow ip \in kD rt"
\hspace{1em}"\forall ip rt. ip \in iD rt \Rightarrow ip \in kD rt"
\hspace{1em}unfolding kD_is_vD_and_iD by simp_all

lemma \texttt{kD\_Some \[dest\]}:
\hspace{1em}fixes dip rt
\hspace{1em}assumes "dip \in kD rt"
\hspace{1em}shows "\exists dsn dsk hops nhip pre.
\hspace{1em}\sigma\_route(rt, dip) = Some (dsn, dsk, flag, hops, nhip, pre)"
\hspace{1em}using assms unfolding kD_def by simp

lemma \texttt{kD\_None \[dest\]}:
\hspace{1em}fixes dip rt
\hspace{1em}assumes "dip \notin kD rt"
\hspace{1em}shows "\sigma\_route(rt, dip) = None"
\hspace{1em}using assms unfolding kD_def
\hspace{1em}by (metis (mono_tags) mem_Collect_eq)

lemma \texttt{vD\_Some \[dest\]}:
\hspace{1em}fixes dip rt
\hspace{1em}assumes "dip \in vD rt"
\hspace{1em}shows "\exists dsn dsk hops nhip pre.
\hspace{1em}\sigma\_route(rt, dip) = Some (dsn, dsk, val, hops, nhip, pre)"
\hspace{1em}using assms unfolding vD_def by simp

lemma \texttt{vD\_empty \[simp\]}: "vD Map.empty = {}"
\hspace{1em}unfolding vD_def by simp

lemma \texttt{iD\_Some \[dest\]}:
\hspace{1em}fixes dip rt
\hspace{1em}assumes "dip \in iD rt"
\hspace{1em}shows "\exists dsn dsk hops nhip pre.
\hspace{1em}\sigma\_route(rt, dip) = Some (dsn, dsk, inv, hops, nhip, pre)"
\hspace{1em}using assms unfolding iD_def by simp

lemma \texttt{val\_is\_vD \[elim\]}:
\hspace{1em}fixes ip rt
\hspace{1em}assumes "ip \in kD(rt)"
\hspace{1em}and "the (flag rt ip) = val"
\hspace{1em}shows "ip \in vD(rt)"
\hspace{1em}using assms unfolding vD_def by auto

lemma \texttt{inv\_is\_iD \[elim\]}:
\hspace{1em}fixes ip rt
\hspace{1em}assumes "ip \in kD(rt)"
\hspace{1em}and "the (flag rt ip) = inv"
\hspace{1em}shows "ip \in iD(rt)"
\hspace{1em}using assms unfolding iD_def by auto
lemma iD_flag_is_inv [elim, simp]:
  fixes ip rt
  assumes "ip ∈ iD(rt)"
  shows "the (flag rt ip) = inv"
proof -
  from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)" by auto
  with assms show ?thesis unfolding iD_def by auto
qed

lemma kD_but_not_vD_is_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)" and "ip /∈ vD(rt)"
  shows "ip ∈ iD(rt)"
proof -
  from ⟨ip ∈ kD(rt)⟩ obtain dsn dsk f hops nhop pre
  where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop, pre)"
    by (metis kD_Some)
  from ⟨ip /∈ vD(rt)⟩ have "f ≠ val"
    proof (rule contrapos_nn)
      assume "f = val"
      with rtip have "the (flag rt ip) = val" by simp
      with ⟨ip ∈ kD(rt)⟩ show "ip ∈ vD(rt)" ..
    qed
  with rtip have "the (flag rt ip) = inv" by simp
  with ⟨ip ∈ kD(rt)⟩ show "ip ∈ iD(rt)" ..
qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "ip ∈ vD(rt) ⟷ P rt ip"
  and "ip ∈ iD(rt) ⟷ P rt ip"
  shows "P rt ip"
proof -
  from ⟨ip ∈ kD(rt)⟩ have "ip ∈ vD(rt) ∪ iD(rt)"
    by (simp add: kD_is_vD_and_id)
  thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "∀ dip rt. dip ∈ kD(rt) ⟷ π₅(the (rt dip)) = the (dhops rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "∀ dip rt. dip ∈ kD(rt) ⟷ π₄(the (rt dip)) = the (flag rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "∀ dip rt. dip ∈ kD(rt) ⟷ π₂(the (rt dip)) = sqn rt dip"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "∀ ip rt. ip ∈ kD(rt) ⟷ sqnf rt ip = π₃(the (rt ip))"
  unfolding sqnf_def by auto

lemma vD_flag_val [simp]:
  "∀ dip rt. dip ∈ vD (rt) ⟷ the (flag rt dip) = val"
  unfolding vD_def by clarsimp

lemma kD_update [simp]:
  "∀ rt nip v. kD (rt(nip ↦ v)) = insert nip (kD rt)"
  unfolding kD_def by auto

lemma kD_empty [simp]: "kD Map.empty = {}"
  unfolding kD_def by simp
lemma ip_equal_or_known [elim]:
  fixes rt ip ip'
  assumes "ip = ip' ∨ ip ∈ kD(rt)"
  and "ip = ip' ⟹ P rt ip ip'"
  and "[ ip ≠ ip'; ip ∈ kD(rt)] ⟹ P rt ip ip'"
  shows "P rt ip ip'"
  using assms by auto

2.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in
invariant proofs.

Updating Precursor Lists

definition addpre :: "r ⇒ ip set ⇒ r"
  where "addpre r npre ≡ let (dsn, dsk, flag, hops, nhip, pre) = r in
  (dsn, dsk, flag, hops, nhip, pre ∪ npre)"

lemma proj2_addpre:
  fixes v pre
  shows "π2(addpre v pre) = π2(v)"
  unfolding addpre_def by (cases v) simp

lemma proj3_addpre:
  fixes v pre
  shows "π3(addpre v pre) = π3(v)"
  unfolding addpre_def by (cases v) simp

lemma proj4_addpre:
  fixes v pre
  shows "π4(addpre v pre) = π4(v)"
  unfolding addpre_def by (cases v) simp

lemma proj5_addpre:
  fixes v pre
  shows "π5(addpre v pre) = π5(v)"
  unfolding addpre_def by (cases v) simp

lemma proj6_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "π6(addpre v npre) = π6(v)"
  unfolding addpre_def by (cases v) simp

lemma proj7_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "π7(addpre v npre) = π7(v) ∪ npre"
  unfolding addpre_def by (cases v) simp

lemma addpre_empty: "addpre r {} = r"
  unfolding addpre_def by simp

lemma addpre_r: "addpre (dsn, dsk, fl, hops, nhip, pre) npre = (dsn, dsk, fl, hops, nhip, pre ∪ npre)"
  unfolding addpre_def by simp

lemmas addpre_simps [simp] = proj2_addpre proj3_addpre proj4_addpre proj5_addpre proj6_addpre proj7_addpre addpre_empty addpre_r

definition addpreRT :: "rt ⇒ ip ⇒ ip set ⇒ rt"
  where "addpreRT rt dip npre ≡ map_option (λs. rt (dip ↦ addpre s npre)) (σroute(rt, dip))"
lemma snd_addpre [simp]:
"dsn dsn' v pre. (dsn, and(addpre (dsn', v) pre)) = addpre (dsn, v) pre"
unfolding addpre_def by clarsimp

lemma proj2_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip\in kD rt"
  and "ip'\in kD rt"
  shows "\pi_2\(the\(the\(addpreRT\ rt\ ip'\ npre\)\ ip))\) = \pi_2\(the\(rt\ ip))"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj3_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip\in kD rt"
  and "ip'\in kD rt"
  shows "\pi_3\(the\(the\(addpreRT\ rt\ ip'\ npre\)\ ip))\) = \pi_3\(the\(rt\ ip))"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj5_addpreRT [simp]:
"\rt dip ip npre. dip\in kD(rt)\Rightarrow \pi_5\(the\(the\(addpreRT\ rt\ dip\ npre\)\ ip))\) = \pi_5\(the\(rt\ ip))"
unfolding addpreRT_def by auto

lemma flag_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip\in kD rt"
  shows "flag \(the\(addpreRT\ rt\ dip\ pre\)\ ip)\) = flag \(rt\ ip)"
unfolding addpreRT_def using assms [THEN kD_Some] by (clarsimp)

lemma kD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip\in kD rt"
  shows "kD \(the\(addpreRT\ rt\ dip\ npre\)\) = kD \(rt)"
unfolding kD_def addpreRT_def using assms [THEN kD_Some] by clarsimp blast

lemma vD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip\in kD rt"
  shows "vD \(the\(addpreRT\ rt\ dip\ npre\)\) = vD \(rt)"
unfolding vD_def addpreRT_def using assms [THEN kD_Some] by clarsimp auto

lemma iD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip\in kD rt"
  shows "iD \(the\(addpreRT\ rt\ dip\ npre\)\) = iD \(rt)"
unfolding iD_def addpreRT_def using assms [THEN kD_Some] by clarsimp auto

lemma nhop_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip\in kD rt"
  shows "nhop \(the\(addpreRT\ rt\ dip\ pre\)\ ip)\) = nhop \(rt\ ip)"
unfolding sqn_def addpreRT_def using assms [THEN kD_Some] by (clarsimp)

lemma sqn_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip\in kD rt"
  shows "sqn \(the\(addpreRT\ rt\ dip\ pre\)\ ip)\) = sqn \(rt\ ip)"
unfolding sqn_def addpreRT_def using assms [THEN kD_Some] by (clarsimp)
lemma dhops_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "dhops (the (addpreRT rt dip pre)) ip = dhops rt ip"
  unfolding addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma sqnf_addpreRT [simp]:
  "\ip dip. ip ∈ kD (rt ξ) \implies sqnf (the (addpreRT (rt ξ) ip npre)) dip = sqnf (rt ξ) dip"
  unfolding sqnf_def addpreRT_def by auto

Updating route entries

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD rt"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = es (the (rt dip))"
  using assms [THEN kD_Some] by auto

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip /∈ kD rt"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = en"
  using assms [THEN kD_None] by auto

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip pre
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip /∈ kD rt"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def by simp

definition update_arg_wf :: "r ⇒ bool"
where "update_arg_wf r ≡ π_4(r) = val ∧
        (π_2(r) = 0) = (π_3(r) = unk) ∧
        (π_3(r) = unk → π_5(r) = 1)"

lemma update_arg_wf_gives_cases:
  "\r. update_arg_wf r \implies (π_2(r) = 0) = (π_3(r) = unk)"
  unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples [simp]:
  "\nhip pre. update_arg_wf (0, unk, val, Suc 0, nhip, pre)"
  "\n hops nhip pre. update_arg_wf (Suc n, kno, val, hops, nhip, pre)"
  unfolding update_arg_wf_def by auto

lemma update_arg_wf_tuples' [elim]:
  "\n hops nhip pre. Suc 0 ≤ n \implies update_arg_wf (n, kno, val, hops, nhip, pre)"
  unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: "\nhip pre. P (0, unk, val, Suc 0, nhip, pre)"
  and c2: "\dsn hops nhip pre. dsn > 0 \implies P (dsn, kno, val, hops, nhip, pre)"
  shows "P r"
  proof -
obtain \( dsn \) \( dsk \) flag hops nhip pre
where \( *: "r = (dsn, dsk, flag, hops, nhip, pre)" \) by (cases \( r \))
with \( \text{update\_arg\_wf \( r \)} \) have \( \text{wf1}: "\text{flag = val}" \)
and \( \text{wf2}: "(dsn = 0) = (dsk = unk)" \)
and \( \text{wf3}: "dsk = unk \rightarrow (hops = 1)" \)

unfolding \( \text{update\_arg\_wf\_def} \) by auto
have "\( P (dsn, dsk, flag, hops, nhip, pre) \)"
proof (cases \( dsk \))
assume "dsk = unk"
moreover with \( \text{wf2 \  \text{wf3} \} \) have "dsn = 0" and "hops = Suc 0" by auto
ultimately show \( \text{thesis} \) using \( \langle \text{flag = val} \rangle \) by simp (rule c1)
next
assume "dsk = kno"
moreover with \( \text{wf2} \) have "dsn > 0" by simp
ultimately show \( \text{thesis} \) using \( \langle \text{flag = val} \rangle \) by simp (rule c2)
qed

with \( * \) show "\( P \ r \)" by simp

qed

definition update :: "rt \⇒ \ ip \⇒ \ r \⇒ \ rt"
where
"\text{update \( rt \  \text{ip} \  \text{r} \) }\equiv \text{case } \sigma_{\text{route}}(rt, \text{ip}) \text{ of} \)
None \⇒ rt (\text{ip} \mapsto \text{r})
| Some s \⇒
if \( \pi_2(s) < \pi_2(r) \) then rt (\text{ip} \mapsto \text{addpre r (} \pi_7(s)\text{)})
else if \( \pi_2(s) = \pi_2(r) \land (\pi_5(s) > \pi_5(r) \lor \pi_4(s) = \text{inv}) \)
then rt (\text{ip} \mapsto \text{addpre r (} \pi_7(s)\text{)})
else if \( \pi_3(r) = \text{unk} \)
then rt (\text{ip} \mapsto (\pi_3(s), \text{snd (addpre r (} \pi_7(s)\text{)))})
else rt (\text{ip} \mapsto \text{addpre s (} \pi_7(r)\text{)})"

lemma update_simps [simp]:
fixes \( r \) \( s \) \( nr \) \( nr' \) \( ns \) \( rt \) \( ip \)
defines "\( s \equiv \text{the } \sigma_{\text{route}}(rt, \text{ip}) \)"
and "\( nr \equiv \text{addpre r (} \pi_7(s)\text{)} \)"
and "\( nr' \equiv (\pi_2(s), \pi_3(nr), \pi_4(nr), \pi_5(nr), \pi_6(nr), \pi_7(nr)) \)"
and "\( ns \equiv \text{addpre s (} \pi_7(r)\text{)} \)"
sows
"[\( \text{ip} \not\in kD(rt) \)] \Rightarrow \text{update \( rt \  \text{ip} \  \text{r} \) } = rt (\text{ip} \mapsto \text{r})"
"[\( \text{ip} \in kD(rt); \text{sqn rt ip} < \pi_2(r) \)] \Rightarrow \text{update \( rt \  \text{ip} \  \text{r} \) } = rt (\text{ip} \mapsto \text{nr})"
"[\( \text{ip} \in kD(rt); \text{sqn rt ip} = \pi_2(r); \text{flag rt ip} = \text{Some inv} \)] \Rightarrow \text{update \( rt \  \text{ip} \  \text{r} \) } = rt (\text{ip} \mapsto \text{nr})"
"[\( \text{ip} \in kD(rt); \pi_3(r) = \text{unk}; (\pi_2(r) = 0) = (\pi_3(r) = \text{unk}) \)] \Rightarrow \text{update \( rt \  \text{ip} \  \text{r} \) } = rt (\text{ip} \mapsto \text{nr'})"
"[\( \text{ip} \in kD(rt); \text{sqn rt ip} \geq \pi_2(r); \pi_3(r) = \text{kno}; \text{sqn rt ip} = \pi_2(r) \Rightarrow \text{the (dhops rt ip) \leq \pi_5(r) } \land \text{the (flag rt ip) = val } \)] \Rightarrow \text{update \( rt \  \text{ip} \  \text{r} \) } = rt (\text{ip} \mapsto \text{ns})"

proof -
assume "\( \text{ip} \not\in kD(rt) \)"
hence "\( \sigma_{\text{route}}(rt, \text{ip}) = \text{None} \)" ..
thus "\( \text{update \( rt \  \text{ip} \  \text{r} \) } = rt (\text{ip} \mapsto \text{r}) \)"

unfolding update_def by simp

next
assume "\( \text{ip} \in kD(rt) \)"
and "\( \text{sqn rt ip} < \pi_2(r) \)"
from this(1) obtain \( dsn \) \( dsk \) fl hops nhip pre
where "\( \text{rt ip} = \text{Some } (dsn, dsk, fl, hops, nhip, pre) \)"
by (metis kD_Some)
with \( \text{sqn rt ip} < \pi_2(r); \) show "\( \text{update \( rt \  \text{ip} \  \text{r} \) } = rt (\text{ip} \mapsto \text{nr}) \)"

unfolding update_def nr_def s_def by auto

next
assume "\( \text{ip} \in kD(rt) \)"
and "sqn rt ip = π₂(r)"
and "the (dhops rt ip) > π₅(r)"
from this(1) obtain dsn dsk fl hops nhip pre
where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
with ⟨sqn rt ip = π₂(r)⟩ and ⟨the (dhops rt ip) > π₅(r)⟩
show "update rt ip r = rt (ip ↦→ nr)"
unfolding update_def nr_def s_def by auto
next
assume "ip ∈ kD(rt)"
and "sqn rt ip = π₃(r)"
and "flag rt ip = Some inv"
from this(1) obtain dsn dsk fl hops nhip pre
where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
with ⟨sqn rt ip = π₃(r)⟩ and ⟨flag rt ip = Some inv⟩
show "update rt ip r = rt (ip ↦→ nr)"
unfolding update_def nr_def s_def by auto
next
assume "ip ∈ kD(rt)"
and "π₃(r) = unk"
and "(π₂(r) = 0) = (π₃(r) = unk)"
from this(1) obtain dsn dsk fl hops nhip pre
where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
with ⟨π₂(r) = 0⟩ = ⟨π₃(r) = unk⟩ and ⟨π₃(r) = unk⟩
show "update rt ip r = rt (ip ↦→ nr')"
unfolding update_def nr'_def nr_def s_def by (cases r) simp
next
assume "ip ∈ kD(rt)"
and otherassms: "sqn rt ip ≥ π₂(r)"
"π₃(r) = kno"
"sqn rt ip = π₂(r) =⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val"
from this(1) obtain dsn dsk fl hops nhip pre
where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
with otherassms show "update rt ip r = rt (ip ↦→ ns)"
unfolding update_def ns_def s_def by auto
qed

lemma update_cases [elim]:
assumes "(π₂(r) = 0) = (π₃(r) = unk)"
and c1: "[\[ ip \notin kD(rt) \] \implies P (rt (ip ↦→ r))]"
and c2: "[\[ ip \in kD(rt); sqn rt ip < π₂(r) \] \implies P (rt (ip ↦→ addpre r (π₇(the σroute(rt, ip)))))]"
and c3: "[\[ ip \in kD(rt); sqn rt ip = π₂(r); the (dhops rt ip) > π₅(r) \] \implies P (rt (ip ↦→ addpre r (π₇(the σroute(rt, ip)))))]"
and c4: "[\[ ip \in kD(rt); sqn rt ip = π₃(r); the (flag rt ip) = inv \] \implies P (rt (ip ↦→ addpre r (π₇(the σroute(rt, ip)))))]"
and c5: "[\[ ip \in kD(rt); π₃(r) = unk \] \implies P (rt (ip ↦→ (π₇(the σroute(rt, ip)), π₃(r),
π₄(r), π₅(r), π₆(r), π₇(addpre r (π₇(the σroute(rt, ip)))))))")"
and c6: "[\[ ip \in kD(rt); sqn rt ip ≥ π₂(r); π₃(r) = kno;
sqn rt ip = π₂(r) =⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val \] \implies P (rt (ip ↦→ addpre (the σroute(rt, ip)) (π₇(r))))")"
shows "(P (update rt ip r))"
proof (cases "ip ∈ kD(rt)")
assume "ip \notin kD(rt)"
with c1 show ?thesis
by simp
next
assume "ip ∈ kD(rt)"
moreover then obtain dsn dsk fl hops nhip pre
where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
moreover obtain dsn' dsk' fl' hops' nhip' pre'
where req: "r = (dsn', dsk', fl', hops', nhip', pre')"
by (cases r) metis
ultimately show "thesis"
using ⟨(π2(r) = 0) = (π3(r) = unk)⟩
shows "P (update rt ip r)"
using assms(1) proof (rule update_cases)
assume "sqn rt ip < π2(r)"
thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))
by (rule c2)
next
assume "sqn rt ip = π2(r)"
and "the (dhops rt ip) > π5(r)"
thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))
by (rule c3)
next
assume "sqn rt ip = π2(r)"
and "the (flag rt ip) = inv"
thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))
by (rule c4)
next
assume "π3(r) = unk"
thus "P (rt (ip ↦ (π7(addpre r (π7(the (rt ip)))))))" 
by (rule c5)
next
assume "sqn rt ip ≥ π2(r)"
and "π3(r) = kno"
and "sqn rt ip = π3(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
thus "P (rt (ip ↦ addpre (the (rt ip))) (π7(r)))))" 
by (rule c6)
qed (simp add: ⟨ip ∈ kD(rt)⟩)
lemma in_kD_after_update [simp]:
fixes rt nip dsn dsk flag hops nhip pre
shows "kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) = insert nip (kD rt)"
unfolding update_def
by (cases "rt nip") auto
lemma nhop_of_update [simp]:
fixes rt dip dsn dsk flag hops nhip
assumes "rt ≠ update rt dip (dsn, dsk, flag, hops, nhip, {})")"
says "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip, {})) dip) = nhip"
proof -
from assms
have update_neq: "∀ v. rt dip = Some v ⇒
  update rt dip (dsn, dsk, flag, hops, nhip, {})
  ≠ rt(dip ↦ addpre (the (rt dip)) (πτ (dsn, dsk, flag, hops, nhip, {})))"
  by auto
show ?thesis
proof (cases "rt dip = None")
  assume "rt dip = None"
  thus ?thesis
  unfolding update_def
  by clarsimp
next
  assume "rt dip ≠ None"
  then obtain v where "rt dip = Some v" by (metis not_None_eq)
  with update_neq [OF this]
  show ?thesis
  unfolding update_def
  by auto
qed

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn (λx. if x = rip then Some v else rt x) ip
    = (if ip = rip then π2(v) else sqn rt ip)"
  unfolding sqn_def
  by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip pre
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip, pre)) dip"
  proof (rule update_cases)
    show "("π2 (dsn, dsk, val, hops, nhip, pre) = 0) = (π3 (dsn, dsk, val, hops, nhip, pre) = unk)"
      by simp (rule assms)
  qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip pre
  assumes "1 ≤ hops"
  shows "sqn rt ip ≤ sqn (update rt ip' (dsn, dsk, flag, hops, nhip, pre)) ip"
  using assms unfolding update_def sqn_def
  by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops ip rip nhip pre
  assumes ex: "∀ ip∈kD rt. the (dhops rt ip) ≥ 1"
  and ip: "(ip = rip ∧ Suc 0 ≤ hops) ∨ (ip ≠ rip ∧ ip∈kD rt)"
  shows "Suc 0 ≤ the (dhops (update rt rip (dsn, dsk, flag, hops, nhip, pre)) ip)"
  using ip proof
    assume "ip = rip ∧ Suc 0 ≤ hops" thus ?thesis
    unfolding update_def using ex
    by (cases "rip∈kD rt") (drule(1) bspec, auto)
  next
    assume "ip ≠ rip ∧ ip∈kD rt" thus ?thesis
    using ex unfolding update_def
    by (cases "rip∈kD rt") auto
  qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)
lemma nhop_update_another [simp]:
  fixes dip ip rt dan dsk flag hops nhip pre
assumes "ip ≠ dip"
shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = nhop rt ip"
using assms unfolding update_def
by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dan dsk flag hops nhip pre
assumes "ip ≠ dip"
shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = dhops rt ip"
using assms unfolding update_def
by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
"∀ rt ip dsn dsk flag hops nhip pre. sqn (rt(ip ↦ v)) ip = π₂(v)"
unfolding sqn_def
by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
shows "the (dhops (update rt dip (osn, kno, val, hops, nhip, {})) dip) = hops"
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma nhop_update_unk_val [simp]:
"∀ rt dip dsn dsk flag hops sip.
  the (nhop (update rt dip (dsn, unk, val, hops, ip, npre)) dip) = ip"
unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
assumes "update rt dip (dsn, dsk, flg, hops, sip, {}) ≠ rt"
shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
using assms unfolding update_def
by (clarsimp split: option.splits if_split_asm) auto

lemma update_rt_split_asm:
"∀ rt ip dsn dsk flag hops sip.
P (update rt ip (dsn, dsk, flag, hops, sip, {})) =
(¬ (rt = update rt ip (dsn, dsk, flag, hops, sip, {}) ∧ ¬ P rt
  ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip, {})
  ∧ ¬ P (update rt ip (dsn, dsk, flag, hops, sip, {}))))"
by auto

lemma sqn_update [simp]: "∀ rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
  ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsn"
unfolding update_def by (clarsimp split: option.split if_split_asm) auto

lemma sqnf_update [simp]: "∀ rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
  ⇒ sqnf (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma update_kno_dsn_greater_zero:
"∀ rt dip dsn hops npre.
  1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip, npre)) dip)"
unfolding update_def
by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∀ rt dip dsn dsk flg hops sip.
\[
rt \neq \text{update } rt \ dip (dsn, dsk, flg, hops, sip, \{\}) \\
\Rightarrow \pi_3(\text{the } (\text{update } rt \ dip (dsn, dsk, flg, hops, sip, \{\}) \ d dip)) = dsk"
\]
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma nhop_update_changed_kno_val [simp]: "\forall rt ip dsn dsk hops nhip.
\rt \neq \text{update } rt \ ip (dsn, kno, val, hops, nhip, \{\}) \\
\Rightarrow \text{the } (\text{nhop } (\text{update } rt \ ip (dsn, kno, val, hops, nhip, \{\}) \ ip)) = nhip"
unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma flag_update [simp]: "\forall rt dip dsn flg hops sip.
\rt \neq \text{update } rt \ dip (dsn, kno, flg, hops, sip, \{\}) \\
\Rightarrow \text{the } (\text{flag } (\text{update } rt \ dip (dsn, kno, flg, hops, sip, \{\}) \ dip)) = flg"
unfolding update_def
by (clarsimp split: option.split if_split_asm) auto

lemma the_flag_Some [dest!]:
fixes ip rt
assumes "\text{the } (\text{flag } rt \ ip) = x"
and "ip \in kD rt"
shows "\text{flag } rt \ ip = \text{Some } x"
using assms by auto

lemma kD_update_unchanged [dest]:
fixes rt dip dsn dsk flag hops nhip pre
assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip, pre)"
shows "dip \in kD(rt)"
proof -
  have "dip \in kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp
  with assms show ?thesis by simp
qed

lemma nhop_update [simp]: "\forall rt dip dsn dsk flg hops sip.
\rt \neq \text{update } rt \ dip (dsn, dsk, flg, hops, sip, \{\}) \\
\Rightarrow \text{the } (\text{nhop } (\text{update } rt \ dip (dsn, dsk, flg, hops, sip, \{\}) \ dip)) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma sqn_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip \neq dip"
shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip \neq dip"
shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:
"\forall dip rt dip' dsn dsk hops nhip pre.
\ dip \in vD(update rt dip' (dsn, dsk, val, hops, nhip, pre)) \Rightarrow (dip \in vD(rt) \lor dip=dip')"
unfolding update_def vD_def by (clarsimp split: option.splits_asm if_split_asm)

Invalidating route entries

definition invalidate :: "rt \Rightarrow (ip \Rightarrow sqn) \Rightarrow rt"
where "invalidate rt dests \equiv 
\lambda ip. \text{case } (rt \ ip, dests \ ip) \ of 
(\text{None}, _) \Rightarrow \text{None}"
\]
lemma proj3_invalidate [simp]:
"\( \forall dip. \pi_3(\text{the } (\text{invalidate } rt \text{ dests} \text{ dip})) = \pi_3(\text{the } (rt \text{ dip}))\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj5_invalidate [simp]:
"\( \forall dip. \pi_5(\text{the } (\text{invalidate } rt \text{ dests} \text{ dip})) = \pi_5(\text{the } (rt \text{ dip}))\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
"\( \forall dip. \pi_6(\text{the } (\text{invalidate } rt \text{ dests} \text{ dip})) = \pi_6(\text{the } (rt \text{ dip}))\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj7_invalidate [simp]:
"\( \forall dip. \pi_7(\text{the } (\text{invalidate } rt \text{ dests} \text{ dip})) = \pi_7(\text{the } (rt \text{ dip}))\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_kD_inv [simp]:
"\( \forall rt \text{ dests}. kD (\text{invalidate } rt \text{ dests}) = kD rt\)"
unfolding invalidate_def kD_def
by (simp split: option.split)

lemma invalidate_sqn:
fixes rt dip dests
assumes "\( \forall rsn. \text{ dests dip = Some rsn } \rightarrow \text{ sqn rt dip } \leq rsn\)"
shows "\( \text{ sqn rt dip } \leq \text{ sqn } (\text{invalidate } rt \text{ dests} \text{ dip})\)"
proof (cases "dip \notin kD(rt)")
  assume "\( \neg dip \notin kD(rt)\)"
  hence "\( dip \in kD(rt)\)" by simp
  then obtain dsn dsk flag hops nhip pre where "\( rt \text{ dip } = \text{ Some } (dsn, dsk, flag, hops, nhip, pre)\)"
  by (metis kD_Some)
  with asms show "\( \text{ sqn rt dip } \leq \text{ sqn } (\text{invalidate } rt \text{ dests} \text{ dip})\)"
  by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
qed simp

lemma sqn_invalidate_in_dests [simp]:
fixes dests ipa rsn rt
assumes "\( \text{ dests ipa = Some rsn} \)"
and "\( \text{ ipa} \in kD(rt)\)"
shows "\( \text{ sqn } (\text{invalidate } rt \text{ dests}) \text{ ipa } = rsn\)"
unfolding invalidate_def sqn_def
using asms(1) asms(2) [THEN kD_Some]
by clarsimp

lemma dhops_invalidate [simp]:
"\( \forall dip. \text{ the } (\text{dhops } (\text{invalidate } rt \text{ dests} \text{ dip})) = \text{ the } (\text{dhops } rt \text{ dip})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
"\( \forall dip. \text{ sqnf } (\text{invalidate } (rt \xi) (\text{dests } \xi)) \text{ dip } = \text{ sqnf } (rt \xi) \text{ dip}\)"
unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
"\( \forall dip. \text{ the } (\text{nhop } (\text{invalidate } (rt \xi) (\text{dests } \xi)) \text{ dip}) = \text{ the } (\text{nhop } (rt \xi) \text{ dip})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
fixes rt dests dip
assumes "\( \text{ dip } \notin \text{ dom } (\text{ dests})\)"
shows "\( \text{ invalidate rt dests dip } = rt \text{ dip}\)"
using asms unfolding invalidate_def

| (Some s, None) ⇒ Some s
| (Some (_, dsk, _, hops, nhip, pre), Some rsn) ⇒ Some (rsn, dsk, inv, hops, nhip, pre)"
by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes rt dests dip
  assumes "dip ∈ kD(rt)"
  shows "invalidate rt dests dip = None"
  using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
  "\dip rt dests. dip ∈ vD(invalidate rt dests) ⇒ dip ∈ vD(rt) ∧ dests dip = None"
  unfolding invalidate_def vD_def
  by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
  fixes dests dip rt
  assumes "dip /∈ dom(dests)"
  shows "sqn (invalidate rt dests) dip = sqn rt dip"
  using assms unfolding sqn_def by simp

lemma invalidate_changes:
  fixes rt dests dip dsn dsk flag hops nhip pre
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "dsn = (case dests dip of None ⇒ π2(the (rt dip)) | Some rsn ⇒ rsn)
  ∧ dsk = π3(the (rt dip))
  ∧ flag = (if dests dip = None then π4(the (rt dip)) else inv)
  ∧ hops = π5(the (rt dip))
  ∧ nhip = π6(the (rt dip))
  ∧ pre = π7(the (rt dip))"
  using assms unfolding invalidate_def
  by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "\dip rt dests. dip ∈ kD (rt)⇒ π3(the (invalidate rt dests dip)) = π3(the (rt dip))"
  by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
  assumes "dests ip = Some rsn" and "ip ∈ kD(rt)"
  shows "ip ∈ iD(invalidate rt dests)"
  using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
  by (clarsimp split: option.split)

2.1.5 Route Requests

Generate a fresh route request identifier.

definition nrreqid :: "(ip × rreqid) set ⇒ ip ⇒ rreqid"
  where "nrreqid rreqs ip ≡ Max ({n . (ip, n) ∈ rreqs} ∪ {0}) + 1"

2.1.6 Queued Packets

Functions for sending data packets.

type_synonym store = "ip ⇒ (p × data list)"

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("\sigma_queue(_, _)"
  where "\sigma_queue(store, dip) ≡ case store dip of None ⇒ [] | Some (p, q) ⇒ q"

definition qD :: "store ⇒ ip set"
  where "qD ≡ dom"

definition add :: "data ⇒ ip ⇒ store ⇒ store"
  where "add d dip store ≡ case store dip of
  None ⇒ store (dip ⇒ (req, [d]))"
Some \((p, q) \Rightarrow \text{store} (\text{dip} \mapsto (p, q @ [d])))

```
lemma qD_add [simp]:
  fixes d dip store
  shows "qD(add d dip store) = insert dip (qD store)"
  unfolding add_def Let_def qD_def
  by (clarsimp split: option.split)
```

definition drop :: "ip \Rightarrow \text{store} \Rightarrow \text{store}"
  where "drop dip store ≡
    map_option (\l(p, q). if tl q = [] then store (dip := None)
      else store (dip \mapsto (p, tl q))) (store dip)"

definition sigma_p_flag :: "\text{store} \Rightarrow \text{ip} \Rightarrow \text{p}"
  where "\sigma_p-flag(store, dip) ≡ map_option fst (store dip)"

definition unsetRRF :: "\text{store} \Rightarrow \text{ip} \Rightarrow \text{store}"
  where "unsetRRF store dip ≡ case store dip of
   None ⇒ store
   | Some (p, q) ⇒ store (dip \mapsto (noreq, q))"

definition setRRF :: "\text{store} \Rightarrow \text{ip} \Rightarrow \text{store}"
  where "setRRF store dests ≡ \lambda dip. if dests dip = None then store dip
   else map_option (\l(_, q). (req, q)) (store dip)"

2.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. \(\text{nhop}\) is partial, thus a ‘the’ is needed, similarly for \(\text{dhops}\) and \(\text{addpreRT}\).
2. \(\text{prec}\) is partial.
3. \(\sigma_p\text{-flag}(store, dip)\) is partial.
4. The routing table \((\text{rt})\) is modelled as a map \((\text{ip} \Rightarrow r \text{ option})\) rather than a set of 7-tuples, likewise, the \(r\) is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address \((\text{dip})\) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the dests mapping passed to \(\text{invalidate}\), and \(\text{store}\).

end

2.2 AODV protocol messages

theory B_Aodv_Message
imports B_Fwdrreps
begin

datatype msg =
  Rreq nat rreqid ip sqn k ip sqn ip
  | Rrep nat ip sqn ip ip
  | Rerr "ip \Rightarrow sqn" ip
  | Newpkt data ip
  | Pkt data ip ip

instantiation msg :: msg
begin
  definition newpkt_def [simp]: "newpkt ≡ \lambda (d, dip). Newpkt d dip"
  definition eq_newpkt_def: "eq_newpkt m ≡ case m of Newpkt d dip ⇒ True | _ ⇒ False"

  instance by intro_classes (simp add: eq_newpkt_def)
```

The msg type models the different messages used within AODV. The instantiation as a msg is a technicality due to the special treatment of newpkt messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

definition rreq :: "nat × rreqid × ip × sqn × k × ip × sqn × ip ⇒ msg"
  where "rreq ≡ λ(hops, rreqid, dip, dsn, dsk, oip, osn, sip).
         Rreq hops rreqid dip dsn dsk oip osn sip"

lemma rreq_simp [simp]:
  "rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip) = Rreq hops rreqid dip dsn dsk oip osn sip"
  unfolding rreq_def by simp

definition rrep :: "nat × ip × sqn × ip × ip ⇒ msg"
  where "rrep ≡ λ(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"

lemma rrep_simp [simp]:
  "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
  unfolding rrep_def by simp

definition rerr :: "(ip ↦ sqn) × ip ⇒ msg"
  where "rerr ≡ λ(dests, sip). Rerr dests sip"

lemma rerr_simp [simp]:
  "rerr(dests, sip) = Rerr dests sip"
  unfolding rerr_def by simp

lemma not_eq_newpkt_rreq [simp]: "¬eq_newpkt (Rreq hops rreqid dip dsn dsk oip osn sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rrep [simp]: "¬eq_newpkt (Rrep hops dip dsn oip sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rerr [simp]: "¬eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_pkt [simp]: "¬eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp

definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"

lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp

end

2.3 The AODV protocol

theory B_Aodv
import B_Aodv_Data B_Aodv_Message
  AWN.AWN_SOS_Labels AWN.AWN_Invariants
begin

2.3.1 Data state

record state =
  ip :: "ip"
  sn :: "sqn"
  rt :: "rt"
  rreqs :: "(ip × rreqid) set"
  store :: "store"

abbreviation aodv_init :: "ip ⇒ state"
where "aodv_init i ≡ (|
  ip = i,
  sn = 1,
  rt = Map.empty,
  rreqs = {},
  store = Map.empty,
  msg = (SOME x. True),
  data = (SOME x. True),
  dests = (SOME x. True),
  pre = (SOME x. True),
  rreqid = (SOME x. True),
  dip = (SOME x. True),
  oip = (SOME x. True),
  hops = (SOME x. True),
  dsn = (SOME x. True),
  dsk = (SOME x. True),
  osn = (SOME x. True),
  sip = (SOME x. x ≠ ip i)
|)"

lemma some_neq_not_eq [simp]: "¬((SOME x :: nat. x ≠ i) = i)"
  by (subst some_eq_ex) (metis zero_neq_numeral)

definition clear_locals :: "state ⇒ state"
where "clear_locals ξ ≡ (|
  msg := (SOME x. True),
  data := (SOME x. True),
  dests := (SOME x. True),
  pre := (SOME x. True),
  rreqid := (SOME x. True),
  dip := (SOME x. True),
  oip := (SOME x. True),
  hops := (SOME x. True),
  dsn := (SOME x. True),
  dsk := (SOME x. True),
  osn := (SOME x. True),
  sip := (SOME x. x ≠ ip ξ)
|)"

lemma clear_locals_sip_not_ip [simp]: "¬(sip (clear_locals ξ) = ip ξ)"
  unfolding clear_locals_def by simp

lemma clear_locals_but_not_globals [simp]:
  "ip (clear_locals ξ) = ip ξ"
  "sn (clear_locals ξ) = sn ξ"
  "rt (clear_locals ξ) = rt ξ"
  "rreqs (clear_locals ξ) = rreqs ξ"
  "store (clear_locals ξ) = store ξ"

2.3.2 Auxiliary message handling definitions

definition is_newpkt
where "is_newpkt ξ ≡ case msg ξ of
    Newpkt data' dip' ⇒ { ξ(data := data', dip := dip') }
  | _ ⇒ {}"

definition is_pkt
where "is_pkt ξ ≡ case msg ξ of
    Pkt data' dip' oip' ⇒ { ξ(data := data', dip := dip', oip := oip') }
  | _ ⇒ {}"

definition is_rreq
where "is_rreq ξ ≡ case msg ξ of
    Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' ⇒
    { ξ(hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
      dsk := dsk', oip := oip', osn := osn', sip := sip') }
  | _ ⇒ {}"

lemma is_rreq_asm [dest!]:
assumes "ξ' ∈ is_rreq ξ"
shows "∃ hops' rreqid' dip' dsn' dsk' oip' osn' sip'.
    msg ξ = Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' ∧
    ξ' = ξ(hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
      dsk := dsk', oip := oip', osn := osn', sip := sip')"
using assms unfolding is_rreq_def
by (cases "msg ξ") simp_all

definition is_rrep
where "is_rrep ξ ≡ case msg ξ of
    Rrep hops' dip' dsn' oip' sip' ⇒ { ξ(hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip') }
  | _ ⇒ {}"

lemma is_rrep_asm [dest!]:
assumes "ξ' ∈ is_rrep ξ"
shows "∃ hops' dip' dsn' oip' sip'.
    msg ξ = Rrep hops' dip' dsn' oip' sip' ∧
using assms unfolding is_rrep_def
by (cases "msg ξ") simp_all

definition is_rerr
where "is_rerr ξ ≡ case msg ξ of
    Rerr dests' sip' ⇒ { ξ(dests := dests', sip := sip') }
  | _ ⇒ {}"

lemma is_rerr_asm [dest!]:
assumes "ξ' ∈ is_rerr ξ"
shows "∃ dests' sip'.
    msg ξ = Rerr dests' sip' ∧
    ξ' = ξ(dests := dests', sip := sip')"
using assms unfolding is_rerr_def
by (cases "msg ξ") simp_all

lemmas is_msg_defs =
is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def

lemma is_msg_inv_ip [simp]:
"ξ' ∈ is_newpkt ξ" ⟹ ip ξ' = ip ξ"
"ξ' ∈ is_pkt ξ" ⟹ ip ξ' = ip ξ"
"ξ' ∈ is_rreq ξ" ⟹ ip ξ' = ip ξ"
"ξ' ∈ is_pkt ξ  ⇒  ip ξ' = ip ξ"
"ξ' ∈ is_newpkt ξ  ⇒  ip ξ' = ip ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)

lemma is_msg_inv_sn [simp]:
"ξ' ∈ is_rerr ξ  ⇒  sn ξ' = sn ξ"
"ξ' ∈ is_rrep ξ  ⇒  sn ξ' = sn ξ"
"ξ' ∈ is_rreq ξ  ⇒  sn ξ' = sn ξ"
"ξ' ∈ is_pkt ξ  ⇒  sn ξ' = sn ξ"
"ξ' ∈ is_newpkt ξ  ⇒  sn ξ' = sn ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)

lemma is_msg_inv_rt [simp]:
"ξ' ∈ is_rerr ξ  ⇒  rt ξ' = rt ξ"
"ξ' ∈ is_rrep ξ  ⇒  rt ξ' = rt ξ"
"ξ' ∈ is_rreq ξ  ⇒  rt ξ' = rt ξ"
"ξ' ∈ is_pkt ξ  ⇒  rt ξ' = rt ξ"
"ξ' ∈ is_newpkt ξ  ⇒  rt ξ' = rt ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)

lemma is_msg_inv_rreqs [simp]:
"ξ' ∈ is_rerr ξ  ⇒  rreqs ξ' = rreqs ξ"
"ξ' ∈ is_rrep ξ  ⇒  rreqs ξ' = rreqs ξ"
"ξ' ∈ is_rreq ξ  ⇒  rreqs ξ' = rreqs ξ"
"ξ' ∈ is_pkt ξ  ⇒  rreqs ξ' = rreqs ξ"
"ξ' ∈ is_newpkt ξ  ⇒  rreqs ξ' = rreqs ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)

lemma is_msg_inv_store [simp]:
"ξ' ∈ is_rerr ξ  ⇒  store ξ' = store ξ"
"ξ' ∈ is_rrep ξ  ⇒  store ξ' = store ξ"
"ξ' ∈ is_rreq ξ  ⇒  store ξ' = store ξ"
"ξ' ∈ is_pkt ξ  ⇒  store ξ' = store ξ"
"ξ' ∈ is_newpkt ξ  ⇒  store ξ' = store ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)

lemma is_msg_inv_sip [simp]:
"ξ' ∈ is_pkt ξ  ⇒  sip ξ' = sip ξ"
"ξ' ∈ is_newpkt ξ  ⇒  sip ξ' = sip ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)

2.3.3 The protocol process

datatype pseqp =
  PAodv | PNewPkt | PPkt | PRreq | PRrep | PRerr

fun nat_of_seqp :: "pseqp ⇒ nat"
where
  "nat_of_seqp PAodv = 1"
  "nat_of_seqp PNewPkt = 2"
  "nat_of_seqp PPkt = 3"
  "nat_of_seqp PRreq = 4"
  "nat_of_seqp PRrep = 5"
instantiation "pseqp" :: ord
begin
definition less_eq_seqp [iff]: "l1 ≤ l2 = (nat_of_seqp l1 ≤ nat_of_seqp l2)"
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
instance ..
end

abbreviation AODV where
"AODV ≡ λ_. [clear_locals] call(PAodv)"

abbreviation PKT where
"PKT args ≡
[ξ. let (data, dip, oip) = args ξ in
  (clear_locals ξ) (| data := data, dip := dip, oip := oip |)]
call(PPkt)"

abbreviation NEWPKT where
"NEWPKT args ≡
[ξ. let (data, dip) = args ξ in
  (clear_locals ξ) (| data := data, dip := dip |)]
call(PNewPkt)"

abbreviation RREQ where
"RREQ args ≡
[ξ. let (hops, rreqid, dip, dsn, dsk, oip, osn, sip) = args ξ in
  (clear_locals ξ) (| hops := hops, rreqid := rreqid, dip := dip,
  dsn := dsn, dsk := dsk, oip := oip,
  osn := osn, sip := sip |)]
call(PRreq)"

abbreviation RREP where
"RREP args ≡
[ξ. let (hops, dip, dsn, oip, sip) = args ξ in
  (clear_locals ξ) (| hops := hops, dip := dip, dsn := dsn,
  oip := oip, sip := sip |)]
call(PRrep)"

abbreviation RERR where
"RERR args ≡
[ξ. let (dests, sip) = args ξ in
  (clear_locals ξ) (| dests := dests, sip := sip |)]
call(PRerr)"

fun Γ_AODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
"Γ_AODV PAodv = labelled PAodv (receive(λmsg'. ξ. (| msg := msg' |)).
  (is_newpkt) NEWPKT(λξ. (data ξ, dip ξ))
  (is_pkt) PKT(λξ. (data ξ, dip ξ, oip ξ))
  (is_rreq)
  [ξ. ξ (| rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {}) |)]
  RREQ(λξ. (hops ξ, rreqid ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, sip ξ))
  (is_rrep)
  [ξ. ξ (| rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {}) |)]
  RREP(λξ. (hops ξ, dip ξ, dsn ξ, oip ξ, sip ξ))
  (is_rerr)
[\$\xi, \xi \{ rt := update (rt \xi) (sip \xi) (0, unk, val, 1, sip \xi, \{\}) \} \]
RERR(\$\lambda_\xi. (dests \xi, sip \xi))
\)
\[\{ \xi. dip := dip \} \}
\[\xi. dip \in qD(store \xi) \land vD(rt \xi) \}
unicast(\$\lambda_\xi. the (nhop (rt \xi) (dip \xi)), \$\lambda_\xi. pkt(data \xi, dip \xi, ip \xi)).
\[\xi. store := the (drop (dip \xi) (store \xi)) \}
AODV()
\)
\[\{ \xi. dests := (\lambda_{\text{rip. if}} (rip \in vD (rt \xi) \land nhop (rt \xi) rip = nhop (rt \xi) (dip \xi))
then Some (inc (sqn (rt \xi) rip)) else None) \}
\[\xi. rt := invalidate (rt \xi) (dests \xi) \]
\[\xi. store := setRRF (store \xi) (dests \xi) \]
\[\xi. pre := \bigcup \{ the (prec (rt \xi) rip) \mid rip \in dom (dests \xi) \} \]
\[\xi. dests := (\lambda_{\text{rip. if}} ((dests \xi) rip \neq None \land the (prec (rt \xi) rip) \neq \{\})
then (dests \xi) rip else None) \]
groupcast(\$\lambda_\xi. pre \xi, \$\lambda_\xi. rerr(dests \xi, ip \xi)). AODV()
\]
\[\{ \xi. dip := dip \} \}
\[\xi. dip \in qD(store \xi) \land vD(rt \xi) \land \sigma_{\text{flag}}(store \xi, dip \xi) = req \}
\[\xi. store := unsetRRF (store \xi) (dip \xi) \]
\[\xi. sn := inc (sn \xi) \]
\[\xi. rreqid := nrreqid (rreqs \xi) (ip \xi) \]
\[\xi. rreq := rreq \xi \cup \{(ip \xi, rreqid \xi)\} \]
broadcast(\$\lambda_\xi. rreq(0, rreqid \xi, dip \xi, sqn (rt \xi) (dip \xi), sqnf (rt \xi) (dip \xi),
ip \xi, sn \xi, ip \xi)). AODV()"}

"\$\Gamma_{\text{AODV}}\ PNewPkt = labelled \text{PNewPkt} (
\{\xi. dip \xi = ip \xi\}
deliver(\$\lambda_\xi. data \xi). AODV()
\}
\[\xi. dip \xi \neq ip \xi\]
(\[\xi. \{ dip := dip \}\]
\[\xi. \xi \in vD (rt \xi)\}
unicast(\$\lambda_\xi. the (nhop (rt \xi) (dip \xi)), \$\lambda_\xi. pkt(data \xi, dip \xi, oip \xi)). AODV()
\)
\[\xi. \xi \neq \xi \}
(\[\xi. dip \xi \in vD (rt \xi)\}
\[\xi. dip \xi \neq ip \xi\]
(\[\xi. dip \xi \in vD (rt \xi)\}
\[\xi. dip \xi \in iD (rt \xi)\}
\[\xi. dip \xi \neq iD (rt \xi)\}
\}
\[\xi. dip \xi \neq iD (rt \xi)\}
\)
)
"}

"\$\Gamma_{\text{AODV}}\ PPkt = labelled \text{PPkt} (\n\{\xi. dip \xi = ip \xi\}
deliver(\$\lambda_\xi. data \xi). AODV()
\})
\[\xi. dip \xi \neq ip \xi\]
(\[\xi. dip \xi \in vD (rt \xi)\}
unicast(\$\lambda_\xi. the (nhop (rt \xi) (dip \xi)), \$\lambda_\xi. pkt(data \xi, dip \xi, oip \xi)). AODV()
\)
\[\xi. \xi \neq \xi \}
(\[\xi. dip \xi \in vD (rt \xi)\}
\[\xi. dip \xi \neq ip \xi\]
(\[\xi. dip \xi \in iD (rt \xi)\}
\[\xi. dip \xi \neq iD (rt \xi)\}
\}
)
"

"\$\Gamma_{\text{AODV}}\ PRreq = labelled \text{PRreq} (\n\{\xi. (oip \xi, rreqid \xi) \in rreqs \xi\}
AODV()
\})
\[\xi. (oip \xi, rreqid \xi) \neq rreqs \xi\]
\[\xi. \xi \{ rt := update (rt \xi) (oip \xi) (osn \xi, kno, val, hops \xi + 1, sip \xi, \{\}) \}
\[\xi. rreqs := rreqs \xi \cup \{(oip \xi, rreqid \xi)\} \}]}
PRrep = labelled PRrep

unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, oip ξ, ip ξ)).AODV()

• ξ. dip ξ = ip ξ

AODV()

• ξ. dip ξ ≠ ip ξ

AODV()

broadcast(λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, max (sqn (rt ξ) (dip ξ)) (dsn ξ), dsk ξ, oip ξ, osn ξ, ip ξ)).AODV()
\[ \text{groupcast}(\lambda \xi. \text{pre} \xi, \lambda \xi. \text{rerr}(\text{dests} \xi, \text{ip} \xi)). \text{AODV}() \]

\[ \text{declare } \Gamma \text{AODV}.\text{simps} [\text{simp del, code del}] \]
\[ \text{lemmas } \Gamma \text{AODV}_\text{simps} [\text{simp, code}] = \Gamma \text{AODV}.\text{simps} [\text{simplified}] \]

fun \( \Gamma \text{AODV}_\text{skeleton} \)
where
\[ \Gamma \text{AODV}_\text{skeleton} \text{PAdov} = \text{seqp_skeleton} (\Gamma \text{AODV} \text{PAdov}) \]
\[ \Gamma \text{AODV}_\text{skeleton} \text{PNewPkt} = \text{seqp_skeleton} (\Gamma \text{AODV} \text{PNewPkt}) \]
\[ \Gamma \text{AODV}_\text{skeleton} \text{PPkt} = \text{seqp_skeleton} (\Gamma \text{AODV} \text{PPkt}) \]
\[ \Gamma \text{AODV}_\text{skeleton} \text{PRreq} = \text{seqp_skeleton} (\Gamma \text{AODV} \text{PRreq}) \]
\[ \Gamma \text{AODV}_\text{skeleton} \text{PRrep} = \text{seqp_skeleton} (\Gamma \text{AODV} \text{PRrep}) \]
\[ \Gamma \text{AODV}_\text{skeleton} \text{PRerr} = \text{seqp_skeleton} (\Gamma \text{AODV} \text{PRerr}) \]

lemma \( \Gamma \text{AODV}_\text{skeleton}_{\text{wf}} [\text{simp}]: \)
\[ \text{"wellformed } \Gamma \text{AODV}_\text{skeleton}" \]
proof (rule, intro allI)
fix \(pn \) \(pn'\)
show "\( \text{call}(pn') \notin \text{stermsl} (\Gamma \text{AODV}_\text{skeleton} pn) "
by (cases pn) simp_all
qed

declare \( \Gamma \text{AODV}_\text{skeleton}.\text{simps} [\text{del, code del}] \)
lemmas \( \Gamma \text{AODV}_\text{skeleton}_{\text{simps}} [\text{simp, code}] = \Gamma \text{AODV}_\text{skeleton}.\text{simps} [\text{simplified } \Gamma \text{AODV}_\text{skeleton}.\text{simps}] \)

lemma aodv_proc_cases [dest]:
fixes \(p \) \(pn\)
shows "\( p \in \text{ctermsl} (\Gamma \text{AODV} pn) \implies \)
\( (p \in \text{ctermsl} (\Gamma \text{AODV} \text{PAdov}) \lor \)
\( p \in \text{ctermsl} (\Gamma \text{AODV} \text{PNewPkt}) \lor \)
\( p \in \text{ctermsl} (\Gamma \text{AODV} \text{PPkt}) \lor \)
\( p \in \text{ctermsl} (\Gamma \text{AODV} \text{PRreq}) \lor \)
\( p \in \text{ctermsl} (\Gamma \text{AODV} \text{PRrep}) \lor \)
\( p \in \text{ctermsl} (\Gamma \text{AODV} \text{PRerr}) ) "
by (cases pn) simp_all

definition \( \sigma \text{AODV} :: \" \text{ip } \Rightarrow (\text{state } \times (\text{state, msg, pseqp, pseqp label}) \text{ seq}) \text{ set} \"
where "\( \sigma \text{AODV} i \equiv \{(\text{aodv}_\text{init} i, \Gamma \text{AODV} \text{PAdov})\} \"

abbreviation paodv
:: "\text{ip } \Rightarrow (\text{state } \times (\text{state, msg, pseqp, pseqp label}) \text{ seq}, \text{msg seq_action}) \text{ automaton} "
where "\( \text{paodv} i \equiv (\text{init} = \sigma \text{AODV} i, \text{trans} = \text{seqp_sos} \Gamma \text{AODV } )") "

lemma aodv_trans: "\( \text{trans (paodv i)} = \text{seqp_sos} \Gamma \text{AODV } "
by simp

lemma aodv_control_within [simp]: "\( \text{control_within } \Gamma \text{AODV} (\text{init (paodv i)}) "
unfolding \( \sigma \text{AODV}_{\text{def}} \) by (rule control_withinI) (auto simp del: \( \Gamma \text{AODV}_{\text{simps}} \))

lemma aodv wf [simp]:
proof (rule, intro allI)
  fix pn pn'
  show \( \text{call}(pn') \notin \text{stermsl} (\Gamma_{AODV} \ pn) \rangle 
  by (cases pn) simp_all
qed

lemmas aodv_labels_not_empty [simp] = labels_not_empty [OF aodv_wf]

lemma aodv_ex_label [intro]: \( \exists l. l \in \text{labels} \Gamma_{AODV} p \rangle 
by (metis aodv_labels_not_empty all_not_in_conv)

lemma aodv_ex_labelE [elim]:
  assumes \( \forall l \in \text{labels} \Gamma_{AODV} p. P l p \rangle 
  and \( \exists p l. P l p \implies Q \rangle 
  shows \( Q \rangle 
  using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: \( \text{simple_labels} \Gamma_{AODV} \rangle 
proof
  fix pn p
  assume \( p \in \text{subterms}(\Gamma_{AODV} p) \rangle 
  thus \( \exists !l. \text{labels} \Gamma_{AODV} p = \{l\} \rangle 
  by (cases pn) (simp_all cong: seqp_congs | elim disjE)+
qed

lemma \( \sigma_{AODV} \_labels \ [simp] \): \( (\xi, p) \in \sigma_{AODV} i \implies \text{labels} \Gamma_{AODV} p = \{PAodv-:0\} \rangle 
unfolding \( \sigma_{AODV} \_def \) by simp

lemma aodv_init_kD_empty [simp]:
  \( (\xi, p) \in \sigma_{AODV} i \implies kD (rt \xi) = {} \rangle 
unfolding \( kD \_def \) by simp

lemma aodv_init_sip_not_ip [simp]: \( \neg (\text{sip (aodv_init i) = i}) \rangle 
by simp

lemma aodv_init_sip_not_ip' [simp]:
  assumes \( (\xi, p) \in \sigma_{AODV} i \rangle 
  shows \( \text{sip} \xi \neq \text{ip} \xi \rangle 
  using assms unfolding \( \sigma_{AODV} \_def \) by simp

lemma aodv_init_sip_not_i [simp]:
  assumes \( (\xi, p) \in \sigma_{AODV} i \rangle 
  shows \( \text{sip} \xi \neq i \rangle 
  using assms unfolding \( \sigma_{AODV} \_def \) by simp

lemma clear_locals_sip_not_ip':
  assumes \( \text{ip} \xi = i \rangle 
  shows \( \neg (\text{sip (clear_locals} \xi) = i) \rangle 
  using assms by auto

Stop the simplifier from descending into process terms.

declare seqp_congs [cong]

Configure the main invariant tactic for AODV.

declare \( \Gamma_{AODV} \_simps \ [cterms\_env] \)
  aodv_proc_cases [cterms\_cases]
  seq_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms\_intros]
  seq_step_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms\_intros]
end
2.4 Invariant assumptions and properties

theory B_Aodv_Predicates
imports B_Aodv
begin

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

abbreviation not_Pkt :: "msg ⇒ bool"
  where "not_Pkt m ≡ case m of Pkt _ _ _ ⇒ False | _ ⇒ True"

definition msg_sender :: "msg ⇒ ip"
  where "msg_sender m ≡ case m of Rreq _ _ _ _ _ _ _ ipc ⇒ ipc
    | Rrep _ _ _ _ ipc ⇒ ipc
    | Rerr _ ipc ⇒ ipc
    | Pkt _ _ ipc ⇒ ipc"

lemma msg_sender_simps [simp]:
  "⋀ hops rreqid dip dsn dsk oip osn sip.
    msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip) = sip"
  "⋀ hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) = sip"
  "⋀ dests sip. msg_sender (Rerr dests sip) = sip"
  "⋀ dip sip. msg_sender (Pkt d dip sip) = sip"

unfolding msg_sender_def by simp_all

definition msg_zhops :: "msg ⇒ bool"
  where "msg_zhops m ≡ case m of
    Rreq hopsc _ _ _ _ oipc osnc _ _ _ ⇒ hopsc = 0 −→ oipc = sipc
    | Rrep hopsc _ _ _ _ _ _ sipc ⇒ hopsc = 0 −→ dipsc = sipc
    | _ ⇒ True"

lemma msg_zhops_simps [simp]:
  "⋀ hops rreqid dip dsn dsk oip osn sip.
    msg_zhops (Rreq hops rreqid dip dsn dsk oip osn sip) = (hops = 0 −→ dipc = sipc)"
  "⋀ hops dip dsn oip sip. msg_zhops (Rrep hops dip dsn oip sip) = (hops = 0 −→ dipsc = sipc)"
  "⋀ dests sip. msg_zhops (Rerr dests sip) = True"
  "⋀ dip sip. msg_zhops (Pkt d dip sip) = True"

unfolding msg_zhops_def by simp_all

definition rreq_rrep_sn :: "msg ⇒ bool"
  where "rreq_rrep_sn m ≡ case m of
    Rreq _ _ _ _ _ _ _ _ osnc _ _ _ ⇒ osnc ≥ 1
    | Rrep _ _ _ _ _ _ _ sipc ⇒ dipsc = 0 −→ dipsc = sipc
    | _ ⇒ True"

lemma rreq_rrep_sn_simps [simp]:
  "⋀ hops rreqid dip dsn dsk oip osn sip.
    rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip) = (osn ≥ 1)"
  "⋀ hops dip dsn oip sip. rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn ≥ 1)"
  "⋀ dests sip. rreq_rrep_sn (Rerr dests sip) = True"
  "⋀ dip sip. rreq_rrep_sn (Pkt d dip sip) = True"

unfolding rreq_rrep_sn_def by simp_all

definition rreq_rrep_fresh :: "rt ⇒ msg ⇒ bool"
  where "rreq_rrep_fresh crt m ≡ case m of
    Rreq hopsc _ _ _ _ _ _ _ _ oipc osnc ipcc _ _ _ ⇒ (ipcc ≠ oipc −→ oipc ∈ kD(crt) ∧ (sqn crt oipc > osnc))
    | Rrep _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ ⇒ (ipcc ≠ dipc −→ dipc ∈ kD(crt))
    | _ ⇒ True"

lemma rreq_rrep_fresh_simps [simp]:
  "⋀ hops rreqid dip dsn dsk oip osn sip.
    rreq_rrep_fresh (Rreq hops rreqid dip dsn dsk oip osn sip) = (osn ≥ 1)"
  "⋀ hops dip dsn oip sip. rreq_rrep_fresh (Rrep hops dip dsn oip sip) = (dsn ≥ 1)"
  "⋀ dests sip. rreq_rrep_fresh (Rerr dests sip) = True"
  "⋀ dip sip. rreq_rrep_fresh (Pkt d dip sip) = True"

unfolding rreq_rrep_fresh_def by simp_all

end
lemma rreq_rrep_fresh [simp]:
"\(\forall\) hops rreqid dip dsn dsk oip osn sip.
   rreq_rrep_fresh crt (Rreq hops rreqid dip dsn dsk oip osn sip) =
   (sip \neq oip \rightarrow oip \in kD(crt)
   \wedge (sqn crt oip > osn
   \vee (sqn crt oip = osn
   \wedge the (dhops crt oip) \leq hops
   \wedge the (flag crt oip) = val)))"

"\(\forall\) hops dip dsn oip sip.
   rreq_rrep_fresh crt (Rrep hops dip dsn oip sip) =
   (sip \neq dip \rightarrow dip \in kD(crt)
   \wedge sqn crt dip = dsn
   \wedge the (dhops crt dip) = hops
   \wedge the (flag crt dip) = val)"

"\(\forall\) dests sip.
   rreq_rrep_fresh crt (Rerr dests sip) = True"

"\(\forall\) d dip.
   rreq_rrep_fresh crt (Newpkt d dip) = True"

"\(\forall\) d dip sip.
   rreq_rrep_fresh crt (Pkt d dip sip) = True"

unfolding rreq_rrep_fresh_def by simp_all

definition rerr_invalid :: "rt \Rightarrow msg \Rightarrow bool"
where
"rerr_invalid crt m = case m of Rerr destsc _ \Rightarrow
   (\forall ripc \in dom(destsc).
    ripc \in iD(crt) \wedge the (destsc ripc) = sqn crt ripc)
   | _ \Rightarrow True"

lemma rerr_invalid [simp]:
"\(\forall\) hops rreqid dip dsn dsk oip osn sip.
   rerr_invalid crt (Rreq hops rreqid dip dsn dsk oip osn sip) = True"

"\(\forall\) hops dip dsn oip sip.
   rerr_invalid crt (Rrep hops dip dsn oip sip) = True"

"\(\forall\) dests sip.
   rerr_invalid crt (Rerr dests sip) = True"

"\(\forall\) d dip.
   rerr_invalid crt (Newpkt d dip) = True"

"\(\forall\) d dip sip.
   rerr_invalid crt (Pkt d dip sip) = True"

unfolding rerr_invalid_def by simp_all

definition initmissing :: "(nat \Rightarrow state option) \times 'a \Rightarrow (nat \Rightarrow state) \times 'a"
where
"initmissing \sigma = (\lambda i. case (fst \sigma) i of None \Rightarrow aodv_init i | Some s \Rightarrow s, snd \sigma)"

lemma not_in_net_ips_fst_init_missing [simp]:
assumes "i \notin net_ips \sigma"
shows "fst (initmissing (netmap fst \sigma)) i = aodv_init i"
using assms unfolding initmissing_def by simp

lemma fst_initmissing_netmap_pair_fst [simp]:
"fst (initmissing (netmap (\lambda(p, q). (fst (id p), snd (id p), q)) s)) =
   fst (initmissing (netmap fst s))"

unfolding initmissing_def by auto

We introduce a streamlined alternative to initmissing with netmap to simplify invariant statements and thus facilitate their comprehension and presentation.

lemma fst_initmissing_netmap_default_aodv_init_netlift:
"fst (initmissing (netmap fst s)) = default aodv_init (netlift fst s)"

unfolding initmissing_def default_def
by (simp add: fst_netmap_netlift del: One_nat_def)

definition netglobal :: "((nat \Rightarrow state) \Rightarrow bool) \Rightarrow ((state \times 'b) \times 'c) net_state \Rightarrow bool"
where
"netglobal P = (\lambda s. P (default aodv_init (netlift fst s)))"

end
2.5 Quality relations between routes

theory B_Fresher
imports B_Aodv_Data
begin

2.5.1 Net sequence numbers

On individual routes

definition
  nsqn :: "r ⇒ sqn"
where
  "nsqn r ≡ if π₄(r) = val ∨ π₂(r) = 0 then π₂(r) else (π₂(r) - 1)"

lemma nsqn_def':
  "nsqn r = (if π₄(r) = inv then π₂(r) - 1 else π₂(r))"
  unfolding nsqn_def by simp

lemma nsqn_zero [simp]:
  "⋀dsn dsk flag hops nhip pre. nsqn (0, dsk, flag, hops, nhip, pre) = 0"
  unfolding nsqn_def by clarsimp

lemma nsqn_val [simp]:
  "⋀dsn dsk hops nhip pre. nsqn (dsn, dsk, val, hops, nhip, pre) = dsn"
  unfolding nsqn_def by clarsimp

lemma nsqn_inv [simp]:
  "⋀dsn dsk hops nhip pre. nsqn (dsn, dsk, inv, hops, nhip, pre) = dsn - 1"
  unfolding nsqn_def by clarsimp

lemma nsqn_lte_dsn [simp]:
  "⋀dsn dsk flag hops nhip pre. nsqn (dsn, dsk, flag, hops, nhip, pre) ≤ dsn"
  unfolding nsqn_def by clarsimp

On routes in routing tables

definition
  nsqn :: "rt ⇒ ip ⇒ sqn"
where
  "nsqn ≡ λrt dip. case σ_route(rt, dip) of None ⇒ 0 | Some r ⇒ nsqn r (r)"

lemma nsqn_sqn_def:
  "⋀rt dip. nsqn rt dip = (if flag rt dip = Some val ∨ sqn rt dip = 0
   then sqn rt dip else sqn rt dip - 1)"
  unfolding nsqn_def sqn_def by (clarsimp split: option.split)

lemma not_in_kD_nsqn [simp]:
  assumes "dip /∈ kD(rt)"
  shows "nsqn rt dip = 0"
  using assms unfolding nsqn_def by simp

lemma kD_nsqn:
  assumes "dip ∈ kD(rt)"
  shows "nsqn rt dip = nsqn r (the (σ_route(rt, dip)))"
  using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

lemma nsqn_r_flag_pred [simp, intro]:
  fixes dsn dsk flag hops nhip pre
  assumes "P (nsqn, (dsn, dsk, val, hops, nhip, pre))"
  and "P (nsqn, (dsn, dsk, inv, hops, nhip, pre))"
  shows "P (nsqn, (dsn, dsk, flag, hops, nhip, pre))"
  using assms by (cases flag) auto

lemma nsqn_addpreRT_inv [simp]:

lemma sqn_sqn:
"∀ rt dip. sqn rt dip - 1 ≤ nsqn rt dip"
unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma nsqn_sqn: "nsqn rt dip ≤ sqn rt dip"
unfolding sqn_def nsqn_def by (cases "rt dip") auto

lemma val_nsqn_sqn [elim, simp]:
assumes "ip∈kD(rt)"
and "the (flag rt ip) = val"
shows "nsqn rt ip = sqn rt ip"
using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
assumes "ip∈vD(rt)"
shows "nsqn rt ip = sqn rt ip"
proof -
  from ⟨ip∈vD(rt)⟩ have "ip∈kD(rt)"
  and "the (flag rt ip) = val" by auto
  thus ?thesis ..
qed

lemma inv_nsqn_sqn [elim, simp]:
assumes "ip∈kD(rt)"
and "the (flag rt ip) = inv"
shows "nsqn rt ip = sqn rt ip - 1"
using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
assumes "ip∈iD(rt)"
shows "nsqn rt ip = sqn rt ip - 1"
proof -
  from ⟨ip∈iD(rt)⟩ have "ip∈kD(rt)"
  and "the (flag rt ip) = inv" by auto
  thus ?thesis ..
qed

lemma nsqn_update_changed_kno_val [simp]: "∀ rt ip dsn dsk hops nhip.
rt ≠ update rt ip (dsn, kno, val, hops, nhip, {}) \n⇒ nsqn (update rt ip (dsn, kno, val, hops, nhip, {})) ip = dsn"
unfolding nsqn_def update_def
by (clarsimp simp: kD_nsqn split: option.split_asm option.split if_split_asm)
  (metis fun_upd_triv)

lemma nsqn_addpreRT_inv [simp]:
"∀ rt dip npre dip'. dip ∈ kD(rt) \n⇒ nsqn (the (addpreRT rt dip npre)) dip' = nsqn rt dip"
unfolding addpreRT_def nsqn_def nsqn_def
by (frule kD_Some) (clarsimp split: option.split)

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip pre rt ip
assumes "dip ≠ ip"
shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip, pre)) dip = nsqn rt dip"
using assms unfolding nsqn_def
by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
assumes "dip ∈ kD(rt)"
and "dests dip = Some rsn"
shows "nsqn (invalidate rt dests) dip = rsn - 1"
using assms
proof -
  from assms obtain dsk hops nhip pre
  where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip, pre)"
  unfolding invalidate_def
  by auto
moreover from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp
ultimately show ?thesis
  using ⟨dests dip = Some rsn⟩ by simp
qed

lemma nsqn_invalidate_other [simp]:
  assumes "dip ∈ kD(rt)"
  and "dip /∈ dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
  using assms by (clarsimp simp add: kD_nsqn)

2.5.2 Comparing routes

definition fresher :: "r ⇒ r ⇒ bool" ("(_/ ⊑ _)" [51, 51] 50)
where
  "fresher r r' ≡ ((nsqn r, r < nsqn r, r') ∨ (nsqn r, r = nsqn r, r' ∧ π5(r) ≥ π5(r')))"

lemma fresherI1 [intro]:
  assumes "nsqn r, r < nsqn r, r'"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
  assumes "nsqn r, r = nsqn r, r'" and "π5(r) ≥ π5(r')"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI [intro]:
  assumes "(nsqn r, r < nsqn r, r') ∨ (nsqn r, r = nsqn r, r' ∧ π5(r) ≥ π5(r'))"
  shows "r ⊑ r'"
  unfolding fresher_def using assms .

lemma fresherE [elim]:
  assumes "r ⊑ r'"
  and "nsqn r, r < nsqn r, r' ⟹ P r r'"
  and "nsqn r, r = nsqn r, r' ∧ π5(r) ≥ π5(r') ⟹ P r r'"
  shows "P r r'"
  using assms unfolding fresher_def by auto

lemma fresher_refl [simp]: "r ⊑ r"
  unfolding fresher_def by simp

lemma fresher_trans [elim, trans]:
  "[ x ⊑ y; y ⊑ z ] ⟹ x ⊑ z"
  unfolding fresher_def by auto

lemma not_fresher_trans [elim, trans]:
  "[ ¬(x ⊑ y); ¬(z ⊑ x) ] ⟹ ¬(z ⊑ y)"
  unfolding fresher_def by auto

lemma fresher_dsn_flag_hops_const [simp]:
  fixes dsn dsk dsk' flag hops nhip nhip' pre pre'
  shows "(dsn, dsk, flag, hops, nhip, pre) ⊑ (dsn, dsk', flag, hops, nhip', pre')"
  unfolding fresher_def by (cases flag) simp_all
lemma addpre_fresher [simp]: "\( r \ npre. \ r \subseteq (\text{addpre } r \ npre)\)"
  by clarsimp

2.5.3 Comparing routing tables

definition
  rt_fresher :: "ip \Rightarrow rt \Rightarrow rt \Rightarrow bool"
where
  "rt_fresher \equiv \lambda dip rt rt'. (the \((\sigma_{\text{route}}(rt, dip))) \subseteq (the \((\sigma_{\text{route}}(rt', dip)))\)"

abbreviation
  rt_fresher_syn :: "rt \Rightarrow ip \Rightarrow rt \Rightarrow bool" ("(_/ \subseteq _)"
  [51, 999, 51] 50)
where
  "rt1 \subseteq ip rt2 \equiv rt_fresher i rt1 rt2"

lemma rt_fresher_def':
  "(rt1 \subseteq ip rt2) = (\text{nsqn } r \ (the \ (rt1 i)) < \text{nsqn } r \ (the \ (rt2 i)) \lor
  \text{nsqn } r \ (the \ (rt1 i)) = \text{nsqn } r \ (the \ (rt2 i)) \land \pi_5 \ (the \ (rt1 i)) \leq \pi_5 \ (the \ (rt1 i)))"
unfolding rt_fresher_def fresher_def
by (rule refl)

lemma single_rt_fresher [intro]:
assumes "the \ (rt1 ip) \subseteq the \ (rt2 ip)"
shows "rt1 \subseteq ip rt2"
using assms unfolding rt_fresher_def .

lemma rt_fresher_single [intro]:
assumes "rt1 \subseteq ip rt2"
shows "the \ (rt1 ip) \subseteq the \ (rt2 ip)"
using assms unfolding rt_fresher_def .

lemma rt_fresher_def2:
assumes "dip \in kD(rt1)"
and "dip \in kD(rt2)"
shows "\((\text{nsqn } r \ dip < \text{nsqn } r \ dip) \lor
(\text{nsqn } r \ dip = \text{nsqn } r \ dip) \land \pi_5 \ (dhop \ dip) \geq \pi_5 \ (dhop \ dip))"
using assms unfolding rt_fresher_def'
by (simp add: kD_nsqn proj5_eq_dhops)

lemma rt_fresherI1 [intro]:
assumes "dip \in kD(rt1)"
and "dip \in kD(rt2)"
and "\text{nsqn } r \ dip < \text{nsqn } r \ dip"
shows "rt1 \subseteq dip rt2"
unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherI2 [intro]:
assumes "dip \in kD(rt1)"
and "dip \in kD(rt2)"
and "\text{nsqn } r \ dip = \text{nsqn } r \ dip"
and "the \ (dhop \ dip) \geq \text{the } (dhop \ dip)"
shows "rt1 \subseteq dip rt2"
unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt_fresherE [elim]:
assumes "rt1 \subseteq dip rt2"
and "dip \in kD(rt1)"
and "dip \in kD(rt2)"
and "\[ \text{nsqn } r \ dip < \text{nsqn } r \ dip \] \implies P \ rt1 rt2 dip"
and "\[ \text{nsqn } r \ dip = \text{nsqn } r \ dip; \text{the } (dhop \ dip) \geq \text{the } (dhop \ dip) \] \implies P \ rt1 rt2 dip"
shows "P \ rt1 rt2 dip"
using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
using assms(4-5) by auto
lemma rt_fresher_refl [simp]: "rt ⊑ dip rt"
  unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt3"
  shows "rt1 ⊑ dip rt3"
  using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip) ⊑ r"
  shows "rt ⊑ dip (λip. if ip = dip then Some r else rt ip)"
  using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip ⇒ rt ⇒ rt ⇒ bool"
  where "rt_fresh_as ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ (rt2 ⊑ dip rt1)"

abbreviation rt_fresh_as_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/≈_)")
  where "rt1 ≈ i rt2 ≡ rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: "⋀rt dip. rt ⊑ dip rt"
  unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  "⋀rt1 rt2 rt3 dip. [ rt1 ⊑ dip rt2; rt2 ⊑ dip rt3 ] ⇒ rt1 ⊑ dip rt3"
  unfolding rt_fresh_as_def rt_fresher_def
  by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt1"
  shows "rt1 ≈ dip rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "the (rt1 dip) ⊑ the (rt2 dip)"
  and "the (rt2 dip) ⊑ the (rt1 dip)"
  shows "rt1 ≈ dip rt2"
  using assms unfolding rt_fresh_as_def
  by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip ∈ kD(rt)"
  and "dip ∈ kD(rt')"
  and "nsqn rt dip = nsqn rt' dip"
  and "π5(the (rt dip)) = π5(the (rt' dip))"
  shows "rt ⊑ dip rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip) ⊆ the (dhops rt dip)"
    by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt ⊑ dip rt'"
    by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip) ⊆ the (dhops rt' dip)"
    by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt' ⊑ dip rt"
    by (rule rt_fresherI2)
qed
lemma rt_fresh_asE [elim]:
assumes "rt1 ≈ dip rt2"
and "[ rt1 ⊑ dip rt2; rt2 ⊑ dip rt1 ] ⇒ P rt1 rt2 dip"
shows "P rt1 rt2 dip"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
assumes "rt1 ≈ dip rt2"
shows "rt1 ⊑ dip rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
assumes "rt1 ≈ dip rt2"
shows "rt2 ⊑ dip rt1"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
assumes "rt1 ≈ dip rt2"
shows "rt2 ≈ dip rt1"
using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
assumes "¬ (rt1 ⊑ dip rt2)"
shows "¬ (rt1 ≈ dip rt2)"
proof
assume "rt1 ≈ dip rt2"
hence "rt1 ⊑ dip rt2" ..
with ⟨¬ (rt1 ⊑ dip rt2)⟩ show False ..
qed

lemma not_rt_fresh_asI2 [intro]:
assumes "¬ (rt2 ⊑ dip rt1)"
shows "¬ (rt1 ≈ dip rt2)"
proof
assume "rt1 ≈ dip rt2"
hence "rt2 ⊑ dip rt1" ..
with ⟨¬ (rt2 ⊑ dip rt1)⟩ show False ..
qed

lemma not_single_rt_fresher [elim]:
assumes "¬ (the (rt1 ip) ⊑ the (rt2 ip))"
shows "¬ (rt1 ⊑ ip rt2)"
proof
assume "rt1 ⊑ ip rt2"
hence "the (rt1 ip) ⊑ the (rt2 ip)" ..
with ⟨¬ (the (rt1 ip) ⊑ the (rt2 ip))⟩ show False ..
qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
assumes "¬ (the (rt1 ip) ⊑ the (rt2 ip))"
shows "¬ (rt1 ⊑ ip rt2)"
proof
assume "the (rt1 ip) ⊑ the (rt2 ip)"
hence "rt1 ⊑ ip rt2" ..
with ⟨¬ (the (rt1 ip) ⊑ the (rt2 ip))⟩ show False ..
qed

lemma rt_fresh_as_nsqnr:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and \( \text{rt1} \approx_{\text{dip}} \text{rt2} \)

shows \( \text{nsqn_r} \left( \text{the (rt2 dip)} \right) = \text{nsqn_r} \left( \text{the (rt1 dip)} \right) \)

using \text{assms(3)} unfolding \text{rt_fresh_as_def}

by (auto simp: \text{rt_fresher_def2} [OF \langle dip \in kD(rt1) \rangle \langle dip \in kD(rt2) \rangle]
\text{rt_fresher_def2} [OF \langle dip \in kD(rt2) \rangle \langle dip \in kD(rt1) \rangle]
\text{kD_nsqn} [OF \langle dip \in kD(rt1) \rangle]
\text{kD_nsqn} [OF \langle dip \in kD(rt2) \rangle])

lemma \text{rt_fresher_mapupd} [intro!]:
assumes \( \text{dip} \in kD(rt) \)
and \( \text{the (rt dip)} \subseteq r \)
shows \( rt \subseteq \text{dip rt(dip \mapsto r)} \)
using \text{assms unfolding rt_fresher_def}
by simp

lemma \text{rt_fresher_map_update_other} [intro!]:
assumes \( \text{dip} \in kD(rt) \)
and \( \text{dip} \neq ip \)
shows \( rt \subseteq \text{dip rt(ip \mapsto r)} \)
using \text{assms unfolding rt_fresher_def}
by simp

lemma \text{rt_fresher_update_other} [simp]:
assumes \text{inkD: } \( \text{dip} \in kD(rt) \)
and \( \text{dip} \neq ip \)
shows \( rt \subseteq \text{dip update rt ip r} \)
using \text{assms unfolding update_def}
by (clarsimp split: option.split) (fastforce)

theorem \text{rt_fresher_update} [simp]:
assumes \( \text{dip} \in kD(rt) \)
and \( \text{the (dhops rt dip)} \geq 1 \)
and \( \text{update_arg_wf r} \)
shows \( rt \subseteq \text{dip update rt ip r} \)
proof
  (cases \( \text{dip} = ip \))
  assume \( \text{dip} \neq ip \)
  with \( \langle \text{dip} \in kD(rt) \rangle \)
  show ?thesis
  by (rule \text{rt_fresher_update_other})
  next
  assume \( \text{dip} = ip \)
  from \( \langle \text{dip} \in kD(rt) \rangle \)
  \text{rtn}
  where \text{rtn [simp]: } \( \text{the (rt dip)} = (dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n) \)
  by (metis \text{prod_cases6})
  with \( \langle \text{the (dhops rt dip)} \geq 1 \rangle \) and \( \langle \text{dip} \in kD(rt) \rangle \) have \( \text{hops_n} \geq 1 \)
  by (metis \text{proj5_eq_dhops proj(4)})
  from \( \langle \text{dip} \in kD(rt) \rangle \) \text{rtn}
  have \text{simp: } \( \text{sqn rt dip} = dsn_n \)
  and \text{simp: } \( \text{the (dhops rt dip)} = hops_n \)
  and \text{simp: } \( \text{the (flag rt dip)} = f_n \)
  by (simp add: \text{sqn_def proj5_eq_dhops [symmetric] proj4_eq_flag [symmetric]})
  from \( \langle \text{update_arg_wf r} \rangle \) have \( \langle dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n \rangle \)
  \( \subseteq \text{the ((update rt dip r) dip)} \)
  proof (rule \text{wf_r_cases})
  fix nhip pre
  from \( \langle \text{hops_n} \geq 1 \rangle \) have \( \langle \text{pre'}. (dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n) \)
  \( \subseteq \text{(dsn_n, unk, val, Suc 0, nhip, pre')} \)
  unfolding \text{fresher_def sqn_def}
  by (cases \text{f_n})
  thus \( \langle dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n \rangle \)
  \( \subseteq \text{the ((update rt dip (0, unk, val, Suc 0, nhip, pre) dip)} \)
  using \( \langle \text{dip} \in kD(rt) \rangle \)
  by - (rule \text{update_cases_kD, simp_all})
  next
  fix dsn :: \text{sqn and hops nhip pre}
  assume \( 0 < \text{dsn} \)
  show \( \langle dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n \rangle \)
  \( \subseteq \text{the ((update rt dip (dsn, kno, val, hops, nhip, pre) dip)} \)
proof (rule update_cases_kD [OF _ (dip∈kD(rt)), simp_all add: 0 < dsn])
  assume "dsn_n < dsn"
  thus "(dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)
\subseteq (dsn, kno, val, hops, nhip, pre ∪ pre_n)"
  unfolding fresher_def by auto
next
  assume "dsn_n = dsn"
  and "hops < hops_n"
  thus "(dsn, dsk_n, f_n, hops_n, nhip_n, pre_n)
\subseteq (dsn, kno, val, hops, nhip, pre ∪ pre_n)"
  unfolding fresher_def nsqn r_def by simp
next
  assume "dsn_n = dsn"
  with ⟨0 < dsn⟩ show "(dsn, dsk_n, inv, hops_n, nhip_n, pre_n)
\subseteq (dsn, kno, val, hops, nhip, pre ∪ pre_n)"
  unfolding fresher_def by simp
qed

hence "rt ⊑ dip update rt dip r"
  by (rule single_rt_fresher, simp)
with ⟨dip = ip⟩ show ?thesis by simp
qed

theorem rt_fresher_invalidate [simp]:
  assumes "dip∈kD(rt)"
  and indests: "∀ rip∈dom(dests). rip∈vD(rt) ∧ sqn rt rip < the (dests rip)"
  shows "rt ⊑ dip invalidate rt dests"
proof (cases "dip∈dom(dests)")
  assume "dip∉dom(dests)"
  with ⟨dip∈kD(rt)⟩ have "dip∈kD(invalidate rt dests)"
  by simp
  with ⟨dip∈kD(rt)⟩ show ?thesis
  by rule (simp_all add: ⟨dip∉dom(dests)⟩)
next
  assume "dip∈dom(dests)"
  moreover with indests have "dip∈vD(rt)"
    and "sqn rt dip < the (dests dip)"
    by auto
  ultimately show ?thesis
  unfolding invalidate_def sqn_def
  by (rule single_rt_fresher, auto simp: fresher_def)
qed

lemma nsqn_r_invalidate [simp]:
  assumes "dip∈kD(rt)"
  and "dip∈dom(dests)"
  shows "nsqn r (the (invalidate rt dests dip)) = the (dests dip) - 1"
using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
  assumes "dip∈kD(rt)"
  and "∀ rip∈dom(dests). rip∈vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
  shows "rt ≈ dip invalidate rt dests"
proof (cases "dip∈dom(dests)")
  assume "dip∉dom(dests)"
  with (dip∈kD(rt)) have "dip∈kD(invalidate rt dests)"
  by simp
  with (dip∈kD(rt)) show ?thesis
  by rule (simp_all add: ⟨dip∉dom(dests)⟩)
next
  assume "dip∈dom(dests)"
  with assms(2) have "dip∈vD(rt)"
    and "the (dests dip) = inc (sqn rt dip)" by auto
  from (dip∈vD(rt)) have "dip∈kD(rt)" by simp
  moreover then have "dip∈kD(invalidate rt dests)" by simp
ultimately show "thesis

proof (rule nsqn_rt_fresh_asI)
  from (dip∈vD(rt)) have "nsqn rt dip = sqn rt dip" by simp
  also have "sqn rt dip = nsqn, (the (invalidate rt dests dip))"
  proof -
    from (dip∈kD(rt)) have "nsqn, (the (invalidate rt dests dip)) = the (dests dip) - 1"
      using (dip∈dom(dests)) by (rule nsqn, invalidate)
    with (the (dests dip) = inc (sqn rt dip))
    show "sqn rt dip = nsqn, (the (invalidate rt dests dip))" by simp
  qed
  also have "sqn rt dip = nsqn r (the (invalidate rt dests dip))"
    proof -
      from (dip∈kD(invalidate rt dests))
      have "nsqn r (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
        by (simp add: kD_nsqn)
    finally show "nsqn rt dip = nsqn (invalidate rt dests) dip" .
  qed simp

lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]

lemma rt_fresh_as_addpreRT [simp]:
  assumes "ip∈kD(rt)"
  shows "rt ≈ dip the (addpreRT rt ip npre)"
    using assms [THEN kD_Some] by (auto simp: addpreRT_def)

lemmas rt_fresher_addpreRT [simp] = rt_fresh_as_addpreRT [THEN rt_fresh_asD1]

2.5.4 Strictly comparing routing tables

definition rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
  where "rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ ¬(rt1 ≈ dip rt2)"

abbreviation
  rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("_/<__" [51, 999, 51] 50)
  where "rt1 _<_ rt2 ≡ rt_strictly_fresher i rt1 rt2"

lemma rt_strictly_fresher_def':
  "rt1 _<_ rt2 = ((rt1 _<_ rt2) ∧ ¬(rt2 _<_ rt1))"
unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

lemma rt_strictly_fresherI' [intro]:
  assumes "rt1 _<_ rt2"
    and "¬(rt2 _<_ rt1)"
  shows "rt1 _<_ rt2"
    using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherE' [elim]:
  assumes "rt1 _<_ rt2"
    and "[ rt1 _<_ rt2; ¬(rt2 _<_ rt1) ] ⇒ P rt1 rt2 i"
  shows "P rt1 rt2 i"
    using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherI [intro]:
  assumes "rt1 _<_ rt2"
    and "¬(rt1 ≈ rt2)"
  shows "rt1 _<_ rt2"
    unfolding rt_strictly_fresher_def using assms ..

lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]

lemma rt_strictly_fresherE [elim]:
  assumes "rt1 _<_ rt2"
    and "[ rt1 _<_ rt2; ¬(rt1 ≈ rt2) ] ⇒ P rt1 rt2 i"
  shows "P rt1 rt2 i"

shows "P rt1 rt2 i"
using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
"rt1 ⊏ rt2 =
(nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i))
  ∨ (nsqn_r (the (rt1 i)) = nsqn_r (the (rt2 i)) ∧ π₅(the (rt1 i)) > π₅(the (rt2 i))))"
unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
assumes "rt1 ⊏ dip rt2"
shows "the (rt1 dip) ⊑ the (rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
assumes "rt1 ⊏ dip rt2"
shows "¬ rt1 ≈ dip rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1 ⊏ dip rt2"
and "rt2 ⊏ dip rt3"
shows "rt1 ⊏ dip rt3"
using assms proof -
from rt1 ⊏ dip rt2: obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
also from rt2 ⊏ dip rt3 obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
finally have "the (rt1 dip) ⊑ the (rt3 dip)".
moreover have "¬ (rt1 ≈ dip rt3)"
proof -
from rt1 ⊏ dip rt2 obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
also from rt2 ⊏ dip rt3 obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))".
thus ?thesis ..
qed
ultimately show "rt1 ⊏ dip rt3"..
qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏ dip rt)"
unfolding rt_strictly_fresher_def by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1 ⊏ dip rt2"
and "rt2 ⊏ dip rt3"
shows "rt1 ⊏ dip rt3"
proof -
from rt1 ⊏ dip rt2: have "rt1 ⊏ dip rt2"
  and "¬(rt2 ⊏ dip rt1)"
  unfolding rt_strictly_fresher_def'' by auto
from this(1) and ⟨rt2 ⊏ dip rt3⟩ have "rt1 ⊏ dip rt3" ..
moreover from ⟨¬(rt2 ⊏ dip rt1)⟩ have "¬(rt3 ⊏ dip rt1)"
proof (rule contrapos_nm)
  assume "rt3 ⊏ dip rt1"
  with ⟨rt2 ⊏ dip rt3⟩ show "rt2 ⊏ dip rt1" ..
  qed
ultimately show "rt1 ⊏ dip rt3"
  unfolding rt_strictly_fresher_def'' by auto
  qed
lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
assumes "rt1 ⊑ dip rt2"
and "rt2 ⊑ dip rt3"
says "rt1 ⊑ dip rt3"

proof -
from rt2 ⊑ dip rt3: have "rt2 ⊑ dip rt3"
  and "¬ (rt3 ⊑ dip rt2)"
  unfolding rt_strictly_fresher_def'' by (auto)
from rt1 ⊑ dip rt2: and this(1) have "rt1 ⊑ dip rt3" ..

moreover from (¬(rt3 ⊑ dip rt2)) have "¬(rt3 ⊑ dip rt1)"
proof (rule contrapos_nn)
  assume "rt3 ⊑ dip rt1"
  thus "rt3 ⊑ dip rt2" using ⟨rt1 ⊑ dip rt2⟩ ..
qed

ultimately show "rt1 ⊑ dip rt3"
unfolding rt_strictly_fresher_def'' by (auto)

qed

lemma rt_fresher_imp_nsqn_le:
  assumes "rt1 ⊑ dip rt2"
  and "ip ∈ kD rt1"
  and "ip ∈ kD rt2"
  shows "nsqn rt1 ip ≤ nsqn rt2 ip"
  using assms(1)
  by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

lemma rt_strictly_fresher_ltI [intro]:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1 ⊏ dip rt2"
proof
  from assms show "rt1 ⊑ dip rt2" ..
next
  show "¬ (rt1 ≈ dip rt2)"
  proof
    assume "rt1 ≈ dip rt2"
    hence "rt2 ⊑ dip rt1" ..
    hence "nsqn rt2 dip ≤ nsqn rt1 dip"
      using ⟨dip ∈ kD(rt2)⟩ ⟨dip ∈ kD(rt1)⟩
      by (rule rt_fresher_imp_nsqn_le)
    with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
      by simp
  qed
qed

lemma rt_strictly_fresher_eqI [intro]:
  assumes "i ∈ kD(rt1)"
  and "i ∈ kD(rt2)"
  and "nsqn rt1 i = nsqn rt2 i"
  and "π5(the (rt2 i)) < π5(the (rt1 i))"
  shows "rt1 ⊏ i rt2"
  using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]:
  "dests dip rt rt'. dests dip = None → (invalidate rt dests ⊑ dip rt') = (rt ⊑ dip rt')"
  unfolding invalidate_def rt_strictly_fresher_def'
  by (rule iffI) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
  assumes "dip ∈ vD(invalidate rt1 dests)"
  shows "(invalidate rt1 dests ⊑ dip rt2) = (rt1 ⊑ dip rt2)"
  proof (cases "dip ∈ dom(dests)")

207
assume "dip ∈ dom(dests)"
hence "dip /∈ vD(invalidate rt1 dests)"
  unfolding invalidate_def vD_def
  by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
with ⟨dip ∈ vD(invalidate rt1 dests)⟩ show ?thesis by simp

next
assume "dip /∈ dom(dests)"
hence "dests dip = None" by auto
moreover with ⟨dip ∈ vD(invalidate rt1 dests)⟩
  have "dip ∈ vD(rt1)"
  unfolding invalidate_def vD_def
  by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
ultimately show ?thesis
  unfolding invalidate_def rt_strictly_fresher_def' by auto
qed

lemma rt_strictly_fresher_update_other [elim!]:
  "⋀ dip ip rt r rt'.
  [ dip ≠ ip; rt ⊏ dip rt' ] \implies update rt ip r ⊏ dip rt'"
unfolding rt_strictly_fresher_def' by clarsimp

lemma addpreRT_strictly_fresher [simp]:
  assumes "dip ∈ kD(rt)"
  shows "the (addpreRT rt dip npre) ⊏ ip rt2 = (rt ⊏ ip rt2)"
using assms unfolding rt_strictly_fresher_def' by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
  assumes "dip ∈ vD (rt2 nhip)"
  and *: "osn < sqn (rt2 nhip) dip"
  and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "update rt dip (osn, kno, val, hops, nhip, {}) ⊏ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
  from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip, {})) dip = osn"
    by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
    have "nsqn, (the (update rt dip (osn, kno, val, hops, nhip, {})) dip) = osn"
      by (simp add: kD_nsqn)
  also have "osn < sqn (rt2 nhip) dip" by (rule *)
  also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
    unfolding nsqn_def using ⟨dip ∈ vD(rt2 nhip)⟩
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqn, (the (update rt dip (osn, kno, val, hops, nhip, {})) dip) < nsqn, (the (rt2 nhip dip))".
  qed

lemma dhops_le_hops_imp_update_strictly_fresher:
  assumes "dip ∈ vD (rt2 nhip)"
  and sqn: "sqn (rt2 nhip) dip = osn"
  and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
  and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip, {})"
  shows "update rt dip (osn, kno, val, Suc hops, nhip, {}) ⊏ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
  from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip = osn"
    by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
    have "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip) = osn"
      by (simp add: kD_nsqn)
  also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
  also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
    unfolding nsqn_def using ⟨dip ∈ vD(rt2 nhip)⟩
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip) = nsqn, (the (rt2 nhip dip))".
next
have "the (dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
also have "hops < hops + 1" by simp
also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)"
  using ** by simp
finally have "the (dhops (rt2 nhip) dip) < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)" .
thus "π₅ (the (rt2 nhip dip)) < π₅ (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)"
  using ⟨dip ∈ vD(rt2 nhip)⟩ by (simp add: proj_eq_dhops)
qed

lemma nsqn_invalidate:
  assumes "dip ∈ kD(rt)"
  and "∀ ip∈dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sqn rt ip)"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
proof -
  from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp
  from assms have "rt ≈ dip invalidate rt dests"
    by (rule rt_fresh_as_inc_invalidate)
  with ⟨dip ∈ kD(rt)⟩ ⟨dip ∈ kD(invalidate rt dests)⟩ show ?thesis
    by (simp add: kD_nsqn del: invalidate_kD_inv)
  (erule(2) rt_fresh_as_nsqnr)
qed

end

2.6 Invariant proofs on individual processes

theory B_Seq_Invariants
importsAWN.Invariants B_Aodv B_Aodv_Data B_Aodv_Predicates B_Fresher

begin

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

lemma sequence_number_increases:
  "paodv i |=A onll Γ AODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by inv_cterms

lemma sequence_number_one_or_bigger:
  "paodv i |= onl Γ AODV (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule onll_step_to_invariantI [OF sequence_number_increases])
  (auto simp: σ AODV_def)

We can get rid of the onl/onll if desired...

lemma sequence_number_increases':
  "paodv i |=A (λ((ξ, _), _. (ξ', _)). sn ξ ≤ sn ξ')"
  by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

lemma sequence_number_one_or_bigger':
  "paodv i |= (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

lemma sip_in_kD:
  "paodv i |= onl Γ AODV (λ(ξ, 1). 1 ∈ {PRreq-:16..PRreq-:18} ∪ {PRreq-:1..PRrep-:5} → dip ξ ∈ kD (rt ξ))"
  by inv_cterms

lemma addpreRT_partly_welldefined:
  "paodv i |=
  onl Γ AODV (λ(ξ, 1). 1 ∈ {PRreq-:16..PRreq-:18} ∪ {PRrep-:1..PRrep-:5} → dip ξ ∈ kD (rt ξ))"
\( l \in \{PRreq\:-3..PRreq\:-17\} \rightarrow oip \xi \in kD (rt \xi) \)"

by inv_cterms

Proposition 7.38

lemma includes_nhip:

"paodv i \models onl \Gamma_{\text{AODV}} (\lambda(\xi, l). \ \forall \text{dip} \in kD (rt \xi). \ \text{the (nhop (rt \xi) dip) \in kD (rt \xi))"

proof

- { fix ip and \( \xi \) :: state
  assume \( "\forall \text{dip} \in kD (rt \xi). \ \text{the (nhop (rt \xi) dip) \in kD (rt \xi)}" 
  and \( "\xi \in \{PRreq\:-3..PRreq\:-17\} \rightarrow oip \xi \in kD (rt \xi)" "\)
  hence \( "\forall \text{dip} \in kD (rt \xi). \ \text{the (nhop (rt \xi) dip) \in kD (rt \xi))" "\)
    by clarsimp (metis nhop_update_unk_val update_another)
  }

  note one_hop = this
  }

- { fix ip sip sn hops and \( \xi \) :: state
  assume \( "\forall \text{dip} \in kD (rt \xi). \ \text{the (nhop (rt \xi) dip) \in kD (rt \xi)}" 
  and \( "\xi \in \{PRreq\:-3..PRreq\:-17\} \rightarrow oip \xi \in kD (rt \xi)" "\)
  and \( "\text{sip} \in kD (rt \xi)" "\)
  hence \( "\forall \text{dip} \in kD (rt \xi). \ \text{the (nhop (rt \xi) dip) \in kD (rt \xi))" "\)
    by (metis kD_update_unchanged nhop_update_changed update_another)
  }

  note nhip_is_sip = this

  show \( \text{?thesis} "\)
    by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf sip_in_kD] onl_invariant_sterms [OF aodv_wf addpreRT_partly_welldefined] solve: one_hop nhip_is_sip)

  qed

Proposition 7.22: needed in Proposition 7.4

lemma addpreRT_welldefined:

"paodv i \models onl \Gamma_{\text{AODV}} (\lambda(\xi, l). \ \forall \text{dip} \in kD (rt \xi). \ \text{the (nhop (rt \xi) dip) \in kD (rt \xi)}"

(is "_ \models onl \Gamma_{\text{AODV}} ?P")

unfolding invariant_def

proof

  fix s
  assume "s \in reachable (paodv i) TT"
  then obtain \( \xi p \) where "s = (\xi, p)"
    and "(\xi, p) \in reachable (paodv i) TT"
    by (metis prod.exhaust)
  have "onl \Gamma_{\text{AODV}} ?P (\xi, p)"
    proof (rule on1)
      fix l
      assume l: "l \in labels \Gamma_{\text{AODV}} p"
      with \( (\xi, p) \in reachable (paodv i) TT "\)
        have I1: "l \in \{PRreq\:-16..PRreq\:-18\} \rightarrow \text{dip} \xi \in kD (rt \xi) "
          and I2: "l \ sulph \:17 \rightarrow \text{dip} \xi \in kD (rt \xi) "
          and I3: "l \ sperp\:-4 \rightarrow \text{dip} \xi \in kD (rt \xi) "
            by (auto dest!: invariantD [OF addpreRT_partly_welldefined])
    moreover from \( (\xi, p) \in reachable (paodv i) TT "\) l \in labels \Gamma_{\text{AODV}} p and I3
      have "l \ sulph \:5 \rightarrow (\text{the (nhop (rt \xi) dip)}) \in kD (rt \xi) "
        by (auto dest!: invariantD [OF includes_nhip])
    ultimately show "?P (\xi, 1)"
      by simp
    qed
  with \( s = (\xi, p) "\) show "onl \Gamma_{\text{AODV}} ?P s"
    by simp

  qed
Proposition 7.4

lemma known_destinations_increase:
  "paodv i |-A onll \( \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{kD}\ (rt\ \xi) \subseteq \text{kD}\ (rt\ \xi'))\)"
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]
    simp add: subset_insertI)

Proposition 7.5

lemma rreqs_increase:
  "paodv i |-A onll \( \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{rreqs\ } \xi \subseteq \text{rreqs\ } \xi')\)"
  by (inv_cterms simp add: subset_insertI)

lemma dests_bigger_than_sqn:
  "paodv i |- onll \( \Gamma_{AODV} (\lambda(\xi, l). l \in \{\text{PAodv-:15..PAodv-:19}\}
    \cup \{\text{PPkt-:7..PPkt-:11}\}
    \cup \{\text{PRreq-:9..PRreq-:13}\}
    \cup \{\text{PRreq-:21..PRreq-:25}\}
    \cup \{\text{PRrep-:9..PRrep-:13}\}
    \cup \{\text{PRerr-:1..PRerr-:5}\}
    \rightarrow (\forall ip \in \text{dom(dests}\ \xi). ip \in \text{kD}\ (rt\ \xi) \land \text{sqn}\ (rt\ \xi)\ ip \leq \text{the}\ (\text{dests}\ \xi\ ip))\)"
proof -
  have sqninv:
    "\(\forall ip \in \text{dom(dests}\ \xi). ip \in \text{kD}\ (rt\ \xi) \land \text{sqn}\ (rt\ \xi)\ ip \leq \text{the}\ (\text{dests}\ \xi\ ip)\)"
    by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
  have indests:
    "\(\forall ip \in \text{dom(dests}\ \xi). ip \in \text{kD}\ (rt\ \xi) \land \text{sqn}\ (rt\ \xi)\ ip \leq \text{rsn}\)"
    by (metis domI option.sel)
  show ?thesis
    by (inv_cterms
        (clarsimp split: if_split_asm option.split_asm
        elim!: sqninv indests)+
    qed

Proposition 7.6

lemma sqns_increase:
  "paodv i |- onll \( \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall ip. \text{sqn}\ (rt\ \xi)\ ip \leq \text{sqn}\ (rt\ \xi')\ ip)\)"
proof -
  { fix \( \xi \) :: state
    assume *: "\(\forall ip \in \text{dom(dests}\ \xi). ip \in \text{kD}\ (rt\ \xi) \land \text{sqn}\ (rt\ \xi)\ ip \leq \text{the}\ (\text{dests}\ \xi\ ip)\)"
    have "\(\forall ip. \text{sqn}\ (rt\ \xi)\ ip \leq \text{sqn}\ (\text{invalidate}\ (rt\ \xi)\ (\text{dests}\ \xi))\ ip\)"
      proof
        fix ip
        from * have "\(ip \in \text{dom(dests}\ \xi) \land \text{sqn}\ (rt\ \xi)\ ip \leq \text{the}\ (\text{dests}\ \xi\ ip)\)" by simp
        thus "\(\text{sqn}\ (rt\ \xi)\ ip \leq \text{sqn}\ (\text{invalidate}\ (rt\ \xi)\ (\text{dests}\ \xi))\ ip\)"
          by (metis domI invalidate_sqn option.sel)
      qed
    } note solve_invalidate = this
  show ?thesis
    by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]
      onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn]
      simp add: solve_invalidate)
  qed

Proposition 7.7

lemma ip_constant:
  "paodv i |- onll \( \Gamma_{AODV} (\lambda(\xi, _). ip\ \xi = i)\)"
  by (inv_cterms simp add: \sigma_{AODV_def})

Proposition 7.8

lemma sender_ip_valid':
"paodv i |=_A onll \Gamma_{AODV} (\lambda((\xi, \_), a, \_). anycast (\lambda m. not_Pkt m \rightarrow msg_sender m = ip \xi) a)" by inv_cterms

lemma sender_ip_valid:
"paodv i |=_A onll \Gamma_{AODV} (\lambda((\xi, \_), a, \_). anycast (\lambda m. not_Pkt m \rightarrow msg_sender m = i) a)" by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
(auto dest!: onllD onll1D)

lemma received_msg_inv:
"paodv i |= (recvmsg P \rightarrow) onll \Gamma_{AODV} (\lambda(\xi, 1). l \in \{PAodv-:1\} \rightarrow P (msg \xi))" by inv_cterms

Proposition 7.9

lemma sip_not_ip':
"paodv i |= (recvmsg (\lambda m. not_Pkt m \rightarrow msg_sender m \neq i) \rightarrow) onll \Gamma_{AODV} (\lambda((\xi, \_). sip \xi \neq ip \xi)" by (inv_cterms inv add: onll_invariant_sterms [OF aodv_wf received_msg_inv]
onll_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]]
simp add: clear_locals_sip_not_ip') clarsimp+

lemma sip_not_ip:
"paodv i |= (recvmsg (\lambda m. not_Pkt m \rightarrow msg_sender m \neq i) \rightarrow) onll \Gamma_{AODV} (\lambda((\xi, \_). sip \xi \neq i)" by (inv_cterms inv add: onll_invariant_sterms [OF aodv_wf received_msg_inv]
onll_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]]
simp add: clear_locals_sip_not_ip') clarsimp+

Neither sip_not_ip' nor sip_not_ip is needed to show loop freedom.

Proposition 7.10

lemma hop_count_positive:
"paodv i |= onll \Gamma_{AODV} (\lambda((\xi, \_). \forall ip \in kD (rt \xi). the (dhops (rt \xi) ip) \geq 1)" by (inv_cterms inv add: onll_invariant_sterms [OF aodv_wf addpreRT_welldefined])

lemma rreq_dip_in_vD_dip_eq_ip:
"paodv i |= onll \Gamma_{AODV} (\lambda(\xi, 1). l \in \{PRreq-:16..PRreq-:18\} \rightarrow dip \xi \in vD(rt \xi))
\land (1 \in \{PRreq-:5, PRreq-:6\} \rightarrow dip \xi = ip \xi)
\land (1 \in \{PRreq-:15..PRreq-:18\} \rightarrow dip \xi \neq ip \xi)"
proof (inv_cterms, elim conjE)
fix \xi pp p'
assume "((\xi, pp) \in reachable (paodv i)) TT"
and "\{PRreq-:17\}[\lambda \xi. \xi(rt := \text{the (addpreRT (rt \xi) (oip \xi) \text{the (nhop (rt \xi) (dip \xi)))})]\] p'
\in \text{sterms} \Gamma_{AODV} pp"
and "1 = PRreq-:17"
and "dip \xi \in vD (rt \xi)"
from this(1-3) have "oip \xi \in kD (rt \xi)"
by (auto dest: onll_invariant_sterms [OF aodv_wf addpreRT_welldefined, where l="PRreq-:17")
with (dip \xi \in vD (rt \xi) show "dip \xi \in vD (the (addpreRT (rt \xi) (oip \xi) \text{the (nhop (rt \xi) (dip \xi)))})" by simp
qed

lemma rreq_dip_in_vD:
"paodv i |= onll \Gamma_{AODV} (\lambda(\xi, 1). l \in \{PRreq-:4..PRreq-:6\} \rightarrow dip \xi \in vD(rt \xi))"
proof inv_cterms
fix \xi pp p'
assume "((\xi, pp) \in reachable (paodv i)) TT"
and "\{PRreq-:5\}[\lambda \xi. \xi(rt := \text{the (addpreRT (rt \xi) (the (nhop (rt \xi) (dip \xi))) \{the (nhop (rt \xi) (oip \xi)))})]\] p'
\in \text{sterms} \Gamma_{AODV} pp"
and "1 = PRreq-:5"
and "dip \xi \in vD (rt \xi)"
from this(1-3) have "the (nhop (rt \xi) (dip \xi)) \in kD (rt \xi)"
by (auto dest: onll_invariant_sterms [OF aodv_wf addpreRT_welldefined, where l="PRreq-:5")
with (dip \xi \in vD (rt \xi) show "dip \xi \in vD (the (addpreRT (rt \xi) (the (nhop (rt \xi) (dip \xi))) \{the (nhop (rt \xi) (oip \xi)))})" by simp

212
Proposition 7.11

lemma anycast_msg_zhops:
"∀msg_zhops. anycast msg_zhops (λ(a, _, _). anycast msg_zhops a)"

proof (inv_cterms inv add:
  onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD dip_eq_ip]
  onl_invariant_sterms [OF aodv_wf rrep_dip_in_vD]
  onl_invariant_sterms [OF aodv_wf hop_count_positive],
  elim conjE)

fix l ξ a pp p' pp'

assume "((ξ, pp) ∈ reachable (paodv i) TT"
  
and "∃(PRrep-:6)unicast(λζ. the (nhop (rt ξ) (oip ξ)),
                         λζ. Rrep (the (dhops (rt ξ) (ip ξ))) (dip ξ) (sqn (rt ξ) (dzns (ip ξ))) (oip ξ) (ip ξ)).
                         p' > pp' ∈ sterms AODV pp"

  
and "l = PRrep-:5"

next

fix l ξ a pp p' pp'

assume "((ξ, pp) ∈ reachable (paodv i) TT"
  
and "∃(PRreq-:n|n. True)unicast(λζ. the (nhop (rt ξ) (oip ξ)),
                         λζ. Rrep (the (dhops (rt ξ) (ip ξ))) (dip ξ) (sqn (rt ξ) (dzns (ip ξ))) (oip ξ) (ip ξ)).
                         p' > pp' ∈ sterms AODV pp"

  
and "l = PRreq-:6"

  
and "a = unicast (the (nhop (rt ξ) (oip ξ)))
      (Rrep (the (dhops (rt ξ) (ip ξ))) (dip ξ) (sqn (rt ξ) (dzns (ip ξ))) (oip ξ) (ip ξ))"

  
and "Suc 0 ≤ the (dhops (rt ξ) ip)"

  
and "dip ∈ vD (rt ξ)"

  
from dip ∈ vD (rt ξ) have "dip ∈ vD (rt ξ)"

  
by (rule vD_iD_gives_kD(1))

  
with * have "Suc 0 ≤ the (dhops (rt ξ) (ip ξ))" ..

  
thus "the (dhops (rt ξ) (ip ξ)) = 0 → dip = ip ξ"

  
by auto

qed

lemma hop_count_zero_oip_dip_sip:
"paodv i |= (recvmsg msg_zhops) onll Γ_AODV (λ(ξ, l).
                         (l ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-n|n. True} →
                          (hops ξ = 0 → oip ξ = sip ξ))
                         ∧
                         ((l ∈ {PAodv-:6, PAodv-:7} ∪ {PRrep-n|n. True} →
                          (hops ξ = 0 → dip ξ = sip ξ))))"

  
by (inv_cterms inv add: onll_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:
"paodv i |= (recvmsg rreq_rrep_sn) onll Γ_AODV (λ(ξ, l).
                         1 ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-n|n. True} → 1 ≤ osn ξ)"

  
by (inv_cterms inv add: onll_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq'::
"paodv i |= (recvmsg (λm. rreq_rrep_sn m & msg_zhops m)) onll Γ_AODV (λ(ξ, l).
                         1 ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-n|n. True} → 1 ≤ osn ξ)"

  
proof (rule invariant_weakenE [OF osn_rreq])

  
fix a

  
assume "recvmsg (λm. rreq_rrep_sn m & msg_zhops m) a" a

  
thus "recvmsg rreq_rrep_sn a"
lemma dsn_rrep:
"paodv i |= (recvmsg rreq_rrep_sn →) onl Γ_AODV (λ(ξ, 1).
1 ∈ {PAodv:-6, PAodv:-7} ∪ {PRrep::n|n. True} ---> 1 ≤ dsn ξ)
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma dsn_rrep':
"paodv i ||= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl Γ_AODV (λ(ξ, 1).
(1∈{PAodv:-4..PAodv:-5} ∪ {PRreq::n|n. True} --->
(hops ξ = 0 ---> oip ξ = sip ξ))
∧ ((1∈{PAodv:-6..PAodv:-7} ∪ {PRrep::n|n. True} --->
(hops ξ = 0 ---> dip ξ = sip ξ))))"
proof (rule invariant_weakenE [OF dsn_rrep])
fix a
assume "recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
thus "recvmsg rreq_rrep_sn a"
by (cases a) simp_all
qed

lemma hop_count_zero_oip_dip_sip':
"paodv i ||= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl Γ_AODV (λ(ξ, 1).
(l∈{PAodv-:4..PAodv-:5} ∪ {PRreq-:n|n. True} --->
(hops ξ = 0 ---> oip ξ = sip ξ)) ∧
((l∈{PAodv-:6..PAodv-:7} ∪ {PRrep-:n|n. True} --->
(hops ξ = 0 ---> dip ξ = sip ξ))))"
proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
fix a
assume "recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
thus "recvmsg msg_zhops a"
by (cases a) simp_all
qed

Proposition 7.12
lemma zero_seq_unk_hops_one':
"paodv i ||= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl Γ_AODV (λ(ξ, 1).
∀ dip :: state.
(sqn (invalidate (rt ξ) (dests ξ)) dip = 0 ---> sqnf (rt ξ) dip = unk)
∧ (sqnf (rt ξ) dip = unk ---> the (dhops (rt ξ) dip) = Suc 0)
∧ (the (dhops (rt ξ) dip) = 1 ---> the (nhop (rt ξ) dip) = dip))"
proof -
{ fix dip and ξ :: state and P
assume "sqn (invalidate (rt ξ) (dests ξ)) dip = 0"
and all: "∀ip. sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip"
and *: "sqn (rt ξ) dip = 0 ---> P ξ dip"
have "P ξ dip"
proof -
from all have "sqn (rt ξ) dip ≤ sqn (invalidate (rt ξ) (dests ξ)) dip" ..
with sqn (invalidate (rt ξ) (dests ξ)) dip = 0: have "sqn (rt ξ) dip = 0" by simp
thus "P ξ dip" by (rule *)
qed
} note sqn_invalidate_zero [elim!] = this

{ fix dsn hops :: nat and sip oip rt and ip dip :: ip
assume "∀ dip ∈ kD(rt).
(sqn rt dip = 0 ---> π3 (the (rt dip)) = unk) ∧
(π3 (the (rt dip)) = unk ---> the (dhops rt dip) = Suc 0) ∧
(the (dhops rt dip) = Suc 0 ---> the (nhop rt dip) = dip)"
and "hops = 0 ---> sip = dip"
and "Suc 0 ≤ dsn"
and "ip ≠ dip ---> ip ∈ kD(rt)"
hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) dip) = Suc 0 --->
the (nhop (update rt dip (dsn, kno, val, Suc hops, sip, {})) dip) = ip"
by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok1 [simp] = this
{ fix ip dsn hops sip oip rt dip
  assume "∀ dip ∈ kD(rt).
  (sqn rt dip = 0 → π₃ (the (rt dip)) = unk) ∧
  (π₃ (the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"

  and "Suc 0 ≤ dsn"
  and "ip ≠ dip → ip ∈ kD(rt)"
  hence "π₃ (the (update rt dip (dsn, kno, val, Suc hops, sip, {}) ip)) = unk →
  the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0"
  by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
} note prreq_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "∀ dip ∈ kD(rt).
  (sqn rt dip = 0 → π₃ (the (rt dip)) = unk) ∧
  (π₃ (the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"

  and "Suc 0 ≤ dsn"
  and "ip ≠ dip → ip ∈ kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip = 0 →
  π₃ (the (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = unk"
  by - (rule update_cases, auto simp add: sqn_def sqnf_def)
} note prreq_ok3 [simp] = this

{ fix rt sip
  assume "∀ dip ∈ kD rt.
  (sqn rt dip = 0 → π₃ (the (rt dip)) = unk) ∧
  (π₃ (the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
  (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"

  hence "∀ dip ∈ kD rt.
  (sqn (update rt sip (0, unk, val, Suc 0, sip, {})) dip = 0 →
  π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = unk) ∧
  (π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = unk →
  the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0) ∧
  (the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0 →
  the (nhop (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = dip)"

  by - (rule update_cases, simp_all add: sqn_def sqnf_def)
} note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "∀ sip rt.
  π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = unk →
  the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = Suc 0"
  by (rule update_cases) simp_all

have prreq_ok6 [simp]: "∀ sip rt.
  sqn (update rt sip (0, unk, val, Suc 0, sip, {})) sip = 0 →
  π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = unk"
  by (rule update_cases) simp_all

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip']
    seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans]
    onl_invariant_sterms [OF aodv_wf osn_rreq']
    onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+
qed

lemma zero_seq_unk_hops_one:
  "paodv i |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) → onl Γ_{AODV} (λ(ξ, _).
    ∀ dip ∈ kD rt. (sqn (rt dip) dip = 0 → (sqn (rt dip) dip = unk ∧
    the (dhops (rt dip) dip) = 1 ∧
    the (nhop (rt dip) dip) = dip)))"
  by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto
lemma kD_unk_or_atleast_one:
"paodv i |= (recvmsg rreq_rrep_sn ↦ onl Γ_{AODV} (λ(ξ, 1).
  ∀dip∈kD(rt ξ). π₃(the (rt ξ dip)) = unk ∨ 1 ≤ π₂(the (rt ξ dip)))"

proof -
{ fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
  assume "dsk1 = unk ∨ Suc 0 ≤ dsn2"
  hence "π₃(the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1, pre1) sip)) = unk
   ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) sip"
  unfolding update_def by (cases "dsk1 = unk") (clarsimp split: option.split)+
} note fromsip [simp] = this

{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
  assume allkd: "∀dip∈kD(rt). π₃(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip"
  and **: "dsk1 = unk ∨ Suc 0 ≤ dsn2"
  have "∀dip∈kD(rt). ?prop dip" (is "∀dip∈kD(rt). ?prop dip")
  proof
    fix dip
    assume "dip∈kD(rt)"
    thus ?thesis
    proof (cases "dip = sip")
      assume "dip = sip"
      with ** show ?thesis
      by simp
    next
      assume "dip ≠ sip"
      with ⟨dip∈kD(rt)⟩ allkd show ?thesis
      by simp
    qed
  qed
} note solve_update [simp] = this

{ fix dip rt dests
  assume *: "∀ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip)"
  and **: "∀ip∈kD(rt). π₃(the (rt ip)) = unk ∨ Suc 0 ≤ sqn rt ip"
  have "∀dip∈kD(rt). π₃(the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
  proof
    fix dip
    assume "dip∈kD(rt)"
    with ** have "π₃(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip" ..
    thus "π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
    proof
      assume "π₃(the (rt dip)) = unk" thus ?thesis ..
    next
      assume "Suc 0 ≤ sqn rt dip"
      have "Suc 0 ≤ sqn (invalidate rt dests) dip"
      proof (cases "dip∈dom(dests)")
        assume "dip∈dom(dests)"
        with * have "sqn rt dip ≤ the (dests dip)" by simp
        with ⟨Suc 0 ≤ sqn rt dip⟩ have "Suc 0 ≤ the (dests dip)" by simp
        with ⟨dip∈dom(dests)⟩ ⟨dip∈kD(rt)⟩ [THEN kD_Some] show ?thesis
        unfolding invalidate_def sqn_def by auto
      next
      assume "dip∈dom(dests)"
      with ⟨Suc 0 ≤ sqn rt dip⟩ ⟨dip∈kD(rt)⟩ [THEN kD_Some] show ?thesis
      unfolding invalidate_def sqn_def by auto
      qed
    thus ?thesis by (rule disjI2)
  qed
} note solve_invalidate [simp] = this

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn
THEN invariant_restrict_inD])
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
simp add: proj3_inv proj2_eq_sqn)

qed

Proposition 7.13

lemma rreq_rrep_sn_any_step_invariant:
"paodv i \models (\texttt{recvmsg rreq_rrep_sn} \rightarrow) onll \Gamma_{AODV} (\lambda(\_, \_, \_). \text{anycast rreq_rrep_sn a})"
proof -
  have sqnf_kno: "paodv i \models \Gamma_{AODV} (\lambda(\xi, \_). \text{anycast rreq_rrep_sn a})"
    (1 \in \{PRreq:-16 .. PRreq:-18\} \rightarrow sqnf (rt \xi) (dip \xi) = kno))"
    by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined])
  have rrep_sqn_greater_dsn: "paodv i \models (\texttt{recvmsg rreq_rrep_sn} \rightarrow) onll \Gamma_{AODV} (\lambda(\xi, \_). \text{anycast rreq_rrep_sn a})"
    (1 \in \{PRrep-:1 .. PRrep-:6\} \rightarrow 1 \leq sqn (rt \xi) (dip \xi))"
    by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aodv_wf dsn_rrep])
    (clarsimp simp: update_kno_dsn_greater_zero [simplified])
  show \?thesis
    by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aodv_wf sequence_number_one_or_bigger [THEN invariant_restrict_inD]
onl_invariant_sterms [OF aodv_wf kD_unk_or_atleast_one]
onl_invariant_sterms_TT [OF aodv_wf sqnf_kno]
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
onl_invariant_sterms [OF aodv_wf rrep_sqn_greater_dsn])
    (auto simp: proj2_eq_sqn)
  qed

Proposition 7.14

lemma rreq_rrep_fresh_any_step_invariant:
"paodv i \models (\texttt{recvmsg rreq_rrep_fresh} \rightarrow) onll \Gamma_{AODV} (\lambda(\xi, \_, \_). \text{anycast (rreq_rrep_fresh (rt \xi)) a})"
proof -
  have rreq_oip: "paodv i \models \Gamma_{AODV} (\lambda(\xi, \_). \text{anycast (rreq_rrep_fresh (rt \xi)) a})"
    (1 \in \{PRreq:-3, PRreq:-4, PRreq:-15, PRreq:-27\} \rightarrow oip \xi \in kD(rt \xi) 
    \land (sqn (rt \xi) (oip \xi) > (osn \xi)) 
    \lor (sqn (rt \xi) (oip \xi) = (osn \xi)) 
    \land the (dhops (rt \xi) (oip \xi)) \leq Suc (hops \xi) 
    \land the (flag (rt \xi) (oip \xi) = val)))"
    proof
      fix \xi \_ pp p'
      assume "((\xi, pp) \in \text{reachable} \ (paodv i) \text{ TT}"
      and "\{PRreq-:2\} \lambda. \xi[rt := update (rt \xi) (oip \xi) (osn \xi, kno, val, Suc (hops \xi), sip \xi, \})\] p' \in \text{sterms} \ \Gamma_{AODV} pp"
      and "1' = PRreq-:3"
      show "osn \xi < sqn update (rt \xi) (oip \xi) (osn \xi, kno, val, Suc (hops \xi), sip \xi, \}) (oip \xi) 
        \lor (sqn update (rt \xi) (oip \xi) (osn \xi, kno, val, Suc (hops \xi), sip \xi, \}) (oip \xi) = osn \xi 
        \land the (dhops update (rt \xi) (oip \xi) (osn \xi, kno, val, Suc (hops \xi), sip \xi, \}) (oip \xi)) \leq Suc (hops \xi) 
        \land the (flag update (rt \xi) (oip \xi) (osn \xi, kno, val, Suc (hops \xi), sip \xi, \}) (oip \xi)) = val)"
      unfolding update_def by (clarsimp split: option.split)
        (metis linorder_neqE_nat not_less)
    qed
  have rrep_prrep: "paodv i \models \Gamma_{AODV} (\lambda(\xi, \_). \text{anycast (rreq_rrep_fresh (rt \xi)) a})"
    (1 \in \{PRreq-:4 .. PRreq-:6\} \rightarrow (dip \xi \in kD(rt \xi) 
    \land the (flag (rt \xi) (dip \xi)) = val))"
by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aodv_wf sip_in_kD])

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip]
onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip]
onl_invariant_sterms [OF aodv_wf rrep_prrep])

qed

Proposition 7.15

lemma rerr_invalid_any_step_invariant:
"paodv i \models onl Γ\ AODV (λ(ξ, _, a, _). anycast (rerr_invalid (rt ξ)) a)"
proof -
  have dests_inv: "paodv i \models onll Γ\ AODV (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a)"
    by (inv_cterms inv add: onll_invariant_sterms [OF aodv_wf dests_inv]
onll_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]
onll_invariant_sterms [OF aodv_wf osn_rreq]
onll_invariant_sterms [OF aodv_wf dsn_rrep]
onll_invariant_sterms [OF aodv_wf addpreRT_welldefined [THEN invariant_restrict_inD]])
  show ?thesis
    by (inv_cterms inv add: onll_invariant_sterms [OF aodv_wf dests_inv])

qed

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs \( (rip, rsn) \) is assigned to the variable \( \text{dests} \) of type \( \text{ip} \rightarrow \text{sqn} \), or to the first argument of the function \( \text{rerr} \), this set is a partial function, i.e., there is at most one entry \( (rip, rsn) \) for each destination \( \text{rip} \): guaranteed by type.

lemma dests_vD_inc_sqn:
"paodv i \models onl Γ\ AODV (λ(ξ, l). (l ∈ {PAodv-:15, PPkt-:7, PRreq-:9, PRreq-:21, PRrep-:9, PRerr-:1} → (∀ip∈dom(dests ξ). ip∈vD(rt ξ)) ∧ (l ∈ {PAodv-:16..PAodv-:19} ∪ {PPkt-:8..PPkt-:11} ∪ {PRreq-:10..PRreq-:13} ∪ {PRreq-:22..PRreq-:25} ∪ {PRrep-:10..PRrep-:13} ∪ {PRerr-:2..PRerr-:5} → (∀ip∈dom(dests ξ). ip∈iD(rt ξ) ∧ the (dests ξ ip) = inc (sqn (rt ξ) ip))) ∧ (l = PRerr-:1 → (∀ip∈dom(dests ξ). ip∈vD(rt ξ) ∧ the (dests ξ ip) > sqn (rt ξ) ip))")
by inv_cterms (clarsimp split: if_split_asm option.split_asm simp: domIff)+

Proposition 7.27

lemma route_tables_fresher:
"paodv i \models onl Γ\ AODV (λ((ξ, _, _), (ξ', _)). ∀dip∈kD(rt ξ). rt ξ ⊑ dip rt ξ')"
proof (inv_cterms inv add:
onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
onl_invariant_sterms [OF aodv_wf addpreRT_welldefined [THEN invariant_restrict_inD]])

fix ξ pp' assume "((ξ, pp) ∈ reachable (paodv i) (recvmsg rreq rrep_sn))" and "(PRreq-2)[λξ. ξ(rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {}))]

218
p' ∈ \text{sterms } \Gamma_{AODV} pp

and "Suc 0 ≤ osn ξ"
and "∀ip∈kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"

show "∀ip∈kD (rt ξ). rt ξ ⊑ ip update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"

proof
fix ip
assume "ip∈kD (rt ξ)"
moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
moreover from Suc 0 ≤ osn ξ
have "update_arg_wf (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})" ...
ultimately show "rt ξ ⊑ ip update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"
by (rule rt_fresher_update)
qed

next
fix ξ pp p'
assume "(ξ, pp) ∈ reachable (paodv i) (recvmsg req_rrep_sn)"
and "p' ∈ \text{sterms } \Gamma_{AODV} pp"
and "Suc 0 ≤ dsn ξ"
and "∀ip∈kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"

show "∀ip∈kD (rt ξ). rt ξ ⊑ ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})"

proof
fix ip
assume "ip∈kD (rt ξ)"
moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
moreover from Suc 0 ≤ dsn ξ
have "update_arg_wf (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})" ...
ultimately show "rt ξ ⊑ ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})"
by (rule rt_fresher_update)
qed
qed

end

2.7 The quality increases predicate

theory B_Quality_Increases
imports B_Aodv_Predicates B_Fresher
begin

definition quality_increases :: "state ⇒ state ⇒ bool"
where "quality_increases ξ ξ' ≡ (∀dip∈kD(rt ξ). dip ∈ kD(rt ξ') ∧ rt ξ ⊑ dip rt ξ')
∧ (∀dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip)"

lemma quality_increasesI [intro!]:
assumes "∀dip. dip ∈ kD(rt ξ) ⇒ dip ∈ kD(rt ξ')"
and "∀dip. dip ∈ kD(rt ξ); dip ∈ kD(rt ξ') \implies rt ξ ⊑ dip rt ξ'"
and "∀dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip"
shows "quality_increases ξ ξ'"

unfolding quality_increases_def using assms by clarsimp

lemma quality_increasesE [elim]:
fixes dip
assumes "quality_increases ξ ξ'"
and "dip∈kD(rt ξ)"
and "[ dip ∈ kD(rt ξ'); dip ∈ kD(rt ξ') ] \implies rt ξ ⊑ dip rt ξ'"
and "[ dip ∈ kD(rt ξ'); sqn (rt ξ) dip ≤ sqn (rt ξ') dip ] \implies R dip ξ'"
shows "R dip ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rt_fresherD [dest]:
fixes ip
assumes "quality_increases ξ ξ'"
and "ip∈kD(rt ξ)"
shows "rt ξ ⊑ ip rt ξ'"
using assms by auto

lemma quality_increases_sqnE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
  and "sqn (rt ξ) dip ≤ sqn (rt ξ') dip ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
  using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases ξ ξ"
  by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
  fixes σ σ' dip
  assumes "rt (σ i) ⊏ dip rt (σ' nhip)"
  and qinc: "quality_increases (σ nhip) (σ' nhip)"
  and "dip ∈ kD(rt (σ nhip))"
  shows "rt (σ i) ⊏ dip rt (σ' nhip)"
  proof -
    from qinc have "rt (σ nhip) ⊑ dip rt (σ' nhip)" using ⟨dip ∈ kD(rt (σ nhip))⟩
    by auto
    with ⟨rt (σ i) ⊏ dip rt (σ' nhip)⟩ show ?thesis ..
    qed

lemma kD_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ')"
  using assms by auto

lemma kD_nsqn_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ') ∧ nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  proof -
    from assms have "i ∈ kD(rt ξ')" ..
    moreover with assms have "rt ξ ⊑ i rt ξ'" by auto
    ultimately have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" using ⟨i ∈ kD(rt ξ)⟩ by (erule(2) rt_fresher_imp_nsqn_le)
    with ⟨i ∈ kD(rt ξ')⟩ show ?thesis ..
    qed

lemma nsqn_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
  shows "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "s ≤ nsqn (rt ξ) i"
  and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ') ∧ s ≤ nsqn (rt ξ') i"
  proof
    from ⟨i ∈ kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ show "i ∈ kD(rt ξ')" ..
    next
    from ⟨i ∈ kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
    with ⟨s ≤ nsqn (rt ξ) i⟩ show "s ≤ nsqn (rt ξ') i" by (rule le_trans)
    qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
  and "s < nsqn (rt ξ) i"

shows "s < nsqn (rt ξ') i"
proof -
  from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with s < nsqn (rt ξ) i: show "s < nsqn (rt ξ') i" by simp
qed

lemma nsqn_quality_increases_dhops [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
  and "nsqn (rt ξ) i = nsqn (rt ξ') i"
  shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
  using assms unfolding quality_increases_def
  by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
  assumes "i ∈ kD(rt ξ)"
  and "quality_increases ξ ξ'"
  and "s = nsqn (rt ξ) i"
  shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
  using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

lemma quality_increases_rreq_rrep_props [elim]:
  fixes sn ip hops sip
  assumes qinc: "quality_increases (σ sip) (σ' sip)"
  and "1 ≤ sn"
  and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip ∧ (nsqn (rt (σ sip)) ip = sn)
           → (the (dhops (rt (σ sip)) ip) ≤ hops
            ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip ∧ (nsqn (rt (σ' sip)) ip = sn)
           → (the (dhops (rt (σ' sip)) ip) ≤ hops
            ∨ the (flag (rt (σ' sip)) ip) = inv))"
  (is "_ ∧ nsqnafter")
  proof -
    from * obtain "ip ∈ kD(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto
    from quality_increases (σ sip) (σ' sip)
    have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
    from quality_increases (σ sip) (σ' sip) and ip ∈ kD (rt (σ sip))
    have "ip ∈ kD (rt (σ' sip))" ..
    from sn ≤ nsqn (rt (σ sip)) ip have nsqnafter
    proof
      assume "sn < nsqn (rt (σ sip)) ip"
      also from ip ∈ kD(rt (σ sip)) and quality_increases (σ sip) (σ' sip)
      have "... ≤ nsqn (rt (σ' sip)) ip" ..
      finally have "sn < nsqn (rt (σ' sip)) ip" .
      thus ?thesis by simp
    next
      assume "sn = nsqn (rt (σ sip)) ip"
      with ip ∈ kD(rt (σ sip)) and quality_increases (σ sip) (σ' sip)
      have "sn < nsqn (rt (σ' sip)) ip ∨ (sn = nsqn (rt (σ' sip)) ip)
           ∧ the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip))" ..
      hence "sn < nsqn (rt (σ' sip)) ip ∨ (nsqn (rt (σ' sip)) ip = sn ∧ (the (dhops (rt (σ' sip)) ip) ≤ hops
               ∨ the (flag (rt (σ' sip)) ip) = inv))" proof
        assume "sn < nsqn (rt (σ' sip)) ip" thus ?thesis ..
      next
        assume "sn = nsqn (rt (σ' sip)) ip
               ∧ the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)"
        hence "sn = nsqn (rt (σ' sip)) ip"
and "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)" by auto

from * and ⟨sn = nsqn (rt (σ sip)) ip⟩ have "the (dhops (rt (σ sip)) ip) ≤ hops ∨ the (flag (rt (σ sip)) ip) = inv"

by simp
thus ?thesis
proof
assume "the (dhops (rt (σ sip)) ip) ≤ hops"
with ⟨the (dhops (rt (σ sip)) ip) ≤ the (dhops (rt (σ sip)) ip)⟩
have "the (dhops (rt (σ sip)) ip) ≤ hops" by simp
with ⟨sn = nsqn (rt (σ sip)) ip⟩ show ?thesis by simp

next
assume "the (flag (rt (σ sip)) ip) = inv"
with ⟨ip ∈ kD(rt (σ sip))⟩ have "nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1" ..

from ⟨ip ∈ kD(rt (σ sip))⟩ show ?thesis
proof (rule vD_or_iD)
assume "ip ∈ iD(rt (σ sip))"

hence "nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip" ..
with ⟨sn = nsqn (rt (σ sip)) ip⟩ have "sqn (rt (σ sip)) ip > 1" by simp

with ⟨sqn (rt (σ sip)) ip > 1⟩ have "nsqn (rt (σ sip)) ip ≥ sqn (rt (σ sip)) ip" by simp

with ⟨nsqn (rt (σ sip)) ip = sn -→ (the (dhops (rt (σ sip)) ip) ≤ hops ∨ the (flag (rt (σ sip)) ip) = inv))" shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
∧ (nsqn (rt (σ sip)) ip = sn

→ (the (dhops (rt (σ sip)) ip) ≤ hops ∨ the (flag (rt (σ sip)) ip) = inv))" by simp

thus ?thesis ..

qed

qed

thus ?thesis by (metis (mono_tags) le_cases not_le)

qed

with ⟨ip ∈ kD (rt (σ' sip))⟩ show "ip ∈ kD (rt (σ' sip)) ∧ nsqnafter" ..

qed

lemma quality_increases_rreq_rrep_props':

| fixes sn ip hops sip |
| assumes "∀ j. quality_increases (σ j) (σ' j)" |
| and "1 ≤ sn" |
| and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip |
| ∧ (nsqn (rt (σ sip)) ip = sn |
| → (the (dhops (rt (σ sip)) ip) ≤ hops |
| ∨ the (flag (rt (σ sip)) ip) = inv))" |
| shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip |
| ∧ (nsqn (rt (σ sip)) ip = sn |
| → (the (dhops (rt (σ' sip)) ip) ≤ hops |
| ∨ the (flag (rt (σ' sip)) ip) = inv))" |
| proof |
| from assms(1) have "quality_increases (σ sip) (σ' sip)" .. |
| thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props) |

qed

lemma rreq_quality_increases:
assumes "∀ j. j ≠ i −→ quality_increases (σ j) (σ' j)"
\and "rt (σ' i) = rt (σ i)"
shows "∀ j. quality_increases (σ j) (σ' j)"
using assms by clarsimp (metis order_refl quality_increasesI rt_fresher_refl)
definition msg_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"
where
"msg_fresh σ m ≡
case m of Rreq hops rreqid dip dsk oip osnc sipc ⇒ osnc ≥ 1 ∧ (sipc ≠ oipc →
oipc∈kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc
∧ (nsqn (rt (σ sipc)) oipc = osnc
→ (hops ≥ the (dhops (rt (σ sipc)) oipc)
∨ the (flag (rt (σ sipc)) oipc) = inv))))
| Rrep hops dip dsk oip osnc sipc ⇒ dsk ≥ 1 ∧ (sipc ≠ dipc →
dipc∈kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsk
∧ (nsqn (rt (σ sipc)) dipc = dsk
→ (hops ≥ the (dhops (rt (σ sipc)) dipc)
∨ the (flag (rt (σ sipc)) dipc) = inv))))
| Rerr destsc sipc ⇒ (∀ ripc∈dom(destsc). (ripc∈kD(rt (σ sipc))
∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))
| _ ⇒ True"

lemma msg_fresh [simp]:
"∀ hops rreq dip dsk oip osnc sipc. msg_fresh σ (Rreq hops rreqid dip dsk oip osnc sipc) =
(osn ≥ 1 ∧ (sipc ≠ oip → oipc∈kD(rt (σ sipc))
∧ nsqn (rt (σ sipc)) oipc ≥ osnc
∧ (nsqn (rt (σ sipc)) oipc = osnc
→ (hops ≥ the (dhops (rt (σ sipc)) oipc)
∨ the (flag (rt (σ sipc)) oipc) = inv))))"
"∀ hops dsk oip sipc. msg_fresh σ (Rrep hops dsk oip sipc) =
(dsk ≥ 1 ∧ (sipc ≠ dipc → dipc∈kD(rt (σ sipc))
∧ nsqn (rt (σ sipc)) dipc ≥ dsk
∧ (nsqn (rt (σ sipc)) dipc = dsk
→ (hops ≥ the (dhops (rt (σ sipc)) dipc)
∨ the (flag (rt (σ sipc)) dipc) = inv))))"
"∀ destsc sipc. msg_fresh σ (Rerr destsc sipc) =
(∀ ripc∈dom(destsc). (ripc∈kD(rt (σ sipc))
∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))"
"∀ d dipc. msg_fresh σ (Newpkt d dipc) = True"
"∀ d dipc. msg_fresh σ (Pkt d dipc sipc) = True"

unfolding msg_fresh_def by simp_all

lemma msg_fresh_inc_sn [simp, elim]:
"msg_fresh σ m =⇒ rreq_rrep_sn m" by (cases m) simp_all

lemma recv_msg_fresh_inc_sn [simp, elim]:
"orecvmsg (msg_fresh) σ m =⇒ recvmsg rreq_rrep_sn m" by (cases m) simp_all

lemma rreq_nsqn_is_fresh [simp]:
fixes σ msg hops rreqid dip dsk oip osnc sipc
assumes "rreq_rrep_fresh (rt (σ sipc)) (Rreq hops rreqid dip dsk oip osnc sipc)"
\and "rreq_rrep_sn (Rreq hops rreqid dip dsk oip osnc sipc)"
shows "msg_fresh σ (Rreq hops rreqid dip dsk oip osnc sipc)"
(is "msg_fresh σ ?msg")

proof -
let ?rt = "rt (σ sipc)"
from assms(2) have "1 ≤ osnc" by simp
thus ?thesis
unfolding msg_fresh_def
proof (simp only: msg.case, intro conjI impI)
assume "sipc ≠ oipc"
with assms(1) show "oipc ∈ kD(?rt)" by simp
assume "sip ≠ oip"
   and "nsqn ?rt oip = osn"
show "the (dhops ?rt oip) ≤ hops ∨ the (flag ?rt oip) = inv"
proof (cases "oip∈vD(?rt)"
   assume "oip∈vD(?rt)"
   hence "nsqn ?rt oip = sqn ?rt oip" ..
   with ⟨nsqn ?rt oip = osn⟩ have "sqn ?rt oip = osn" by simp
   with assms(1) and ⟨sip ≠ oip⟩ have "the (dhops ?rt oip) ≤ hops"
   by simp
   thus ?thesis ..
next
   assume "oip /∈ vD(?rt)"
   moreover from assms(1) and ⟨sip ≠ oip⟩ have "oip∈kD(?rt)" by simp
   ultimately have "oip∈iD(?rt)" by auto
   hence "the (flag ?rt oip) = inv" ..
   thus ?thesis ..
qed

next
   assume "sip ≠ oip"
   with assms(1) have "osn ≤ sqn ?rt oip" by auto
   thus "osn ≤ nsqn (rt (σ sip)) oip"
proof (rule nat_le_eq_or_lt)
   assume "osn < sqn ?rt oip"
   hence "osn ≤ sqn ?rt oip - 1" by simp
   also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
   finally show "osn ≤ nsqn ?rt oip" ..
next
   assume "osn = sqn ?rt oip"
   with assms(1) and ⟨sip ≠ oip⟩ have "oip∈kD(?rt)"
   and "the (flag ?rt oip) = val" by simp
   hence "nsqn ?rt oip = sqn ?rt oip" ..
   with ⟨osn = sqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
   thus "osn ≤ nsqn ?rt oip" by simp
qed

lemma rrep_nsqn_is_fresh [simp]:
fixes σ msg hops dip dsn oip sip
assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
   and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
   (is "msg_fresh σ ?msg")
proof -
   let ?rt = "rt (σ sip)"
   from assms have "sip ≠ dip → dip∈kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
      by simp
   hence "sip ≠ dip → dip∈kD(?rt) ∧ nsqn ?rt dip ≥ dsn" by clarsimp
   with assms show "msg_fresh σ ?msg"
      by clarsimp
qed

lemma rerr_nsqn_is_fresh [simp]:
fixes σ msg dests sip
assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
shows "msg_fresh σ (Rerr dests sip)"
   (is "msg_fresh σ ?msg")
proof -
   let ?rt = "rt (σ sip)"
   from assms have "∀ rip∈dom(dests). (rip∈iD(rt (σ sip))
      ∧ the (dests rip) = sqn (rt (σ sip)) rip)"
      by simp
   qed
by clarsimp
have "(\forall rip \in \text{dom}(dests). (\text{rip} \in \text{kd}(rt \ (\sigma \ sip)) \\
\hspace{1cm} \wedge \text{the}(\text{dests} \ \text{rip}) - 1 \leq \text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{rip}))"

proof
fix rip
assume "\text{rip} \in \text{dom dests}"
with * have "\text{rip} \in \text{id}(rt \ (\sigma \ sip))" and "\text{the}(\text{dests} \ \text{rip}) = \text{sqn} \ (rt \ (\sigma \ sip)) \ \text{rip}"
by auto
from this(2) have "the(\text{dests} \ \text{rip}) - 1 = \text{sqn} \ (rt \ (\sigma \ sip)) \ \text{rip} - 1"
by simp
also have "... \leq \text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{rip}" by (rule sqn_nsqn)
finally have "the(\text{dests} \ \text{rip}) - 1 \leq \text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{rip}".

thus "msg\_fresh \ \sigma \ ?msg"
by simp
qed

lemma quality\_increases\_msg\_fresh [elim]:
assumes qinc: "\forall j. \text{quality\_increases} \ (\sigma \ j) \ (\sigma' \ j)"
and "msg\_fresh \ \sigma \ m"
shows "msg\_fresh \ \sigma' \ m"
using assms(2)
proof (cases m)
fix hops rreqid dip dsn oip osn sip
assume [simp]: "m = \text{Rreq} \ \text{hops} \ \text{rreqid} \ \text{dip} \ \text{dsn} \ \text{oip} \ \text{osn} \ \text{sip}"
and "msg\_fresh \ \sigma \ m"
then have "osn \geq 1" and "sip = oip \lor (\text{oip} \in \text{kd}(rt \ (\sigma \ sip)) \ \wedge \ \text{osn} \leq \text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{oip} \\
\hspace{1cm} \wedge (\text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{oip} = \text{osn} \\
\hspace{1.5cm} \rightarrow (\text{the}(\text{dhops}(rt \ (\sigma \ sip)) \ \text{oip}) \leq \text{hops} \\
\hspace{2cm} \lor \ \text{the}(\text{flag}(rt \ (\sigma \ sip)) \ \text{oip}) = \text{inv}))"
by auto
from this(2) show ?thesis
proof
assume "sip = oip" with (osn \geq 1) show ?thesis by simp
next
assume "\text{oip} \in \text{kd}(rt \ (\sigma \ sip)) \ \wedge \ \text{osn} \leq \text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{oip} \\
\hspace{1cm} \wedge (\text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{oip} = \text{osn} \\
\hspace{2cm} \rightarrow (\text{the}(\text{dhops}(rt \ (\sigma \ sip)) \ \text{oip}) \leq \text{hops} \\
\hspace{3cm} \lor \ \text{the}(\text{flag}(rt \ (\sigma \ sip)) \ \text{oip}) = \text{inv}))"
moreover from qinc have "\text{quality\_increases}(\sigma \ sip) \ (\sigma' \ sip)" ..
ultimately have "\text{oip} \in \text{kd}(rt \ (\sigma' \ sip)) \ \wedge \ osn \leq \text{nsqn} \ (rt \ (\sigma' \ sip)) \ \text{oip} \\
\hspace{1cm} \wedge (\text{nsqn} \ (rt \ (\sigma' \ sip)) \ \text{oip} = \text{osn} \\
\hspace{2cm} \rightarrow (\text{the}(\text{dhops}(rt \ (\sigma' \ sip)) \ \text{oip}) \leq \text{hops} \\
\hspace{3cm} \lor \ \text{the}(\text{flag}(rt \ (\sigma' \ sip)) \ \text{oip}) = \text{inv}))"
using (osn \geq 1) by (rule quality\_increases\_rreq\_rrep\_props [rotated 2])
with (osn \geq 1) show "msg\_fresh \ \sigma' \ m"
by (clarsimp)
qed
next
fix hops dip dsn oip sip
assume [simp]: "m = \text{Rrep} \ \text{hops} \ \text{dip} \ \text{dsn} \ \text{oip} \ \text{sip}"
and "msg\_fresh \ \sigma \ m"
then have "dsn \geq 1" and "sip = dip \lor (\text{dip} \in \text{kd}(rt \ (\sigma \ sip)) \ \wedge \ dsn \leq \text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{dip} \\
\hspace{1cm} \wedge (\text{nsqn} \ (rt \ (\sigma \ sip)) \ \text{dip} = \text{dsn} \\
\hspace{2cm} \rightarrow (\text{the}(\text{dhops}(rt \ (\sigma \ sip)) \ \text{dip}) \leq \text{hops} \\
\hspace{3cm} \lor \ \text{the}(\text{flag}(rt \ (\sigma \ sip)) \ \text{dip}) = \text{inv}))"
by auto
from this(2) show ?thesis
proof
assume "sip = dip" with (dsn \geq 1) show ?thesis by simp
assume \( \text{dip} \in kD(rt (\sigma \text{sip})) \land dsn \leq nsqn (rt (\sigma \text{sip})) \text{dip} \)
\( \land (nsqn (rt (\sigma \text{sip})) \text{dip} = dsn \)
\( \to (\text{the (dhops (rt (\sigma \text{sip})) \text{dip})} \leq \text{hops} \)
\( \lor \text{the (flag (rt (\sigma \text{sip})) \text{dip})} = \text{inv}) \)

moreover from qinc have "quality_increases (\sigma \text{sip}) (\sigma' \text{sip})" ..
ultimately have "dip \in kD(rt (\sigma' \text{sip})) \land dsn \leq nsqn (rt (\sigma' \text{sip})) \text{dip} \)
\( \land (nsqn (rt (\sigma' \text{sip})) \text{dip} = dsn \)
\( \to (\text{the (dhops (rt (\sigma' \text{sip}) \text{dip})} \leq \text{hops} \)
\( \lor \text{the (flag (rt (\sigma' \text{sip})) \text{dip})} = \text{inv}) \)

using \( dsn \geq 1 \) by (rule quality_increases_rreq_rrep_props [rotated 2])
with \( dsn \geq 1 \) show "msg_fresh \sigma' m"
by clarsimp

next
fix dests sip
assume \[ \text{simp} \]: "m = Rerr dests sip"
and "msg_fresh \sigma m"
then have \(*\): "\forall \text{rip} \in \text{dom(dests)}. \text{rip} \in kD(rt (\sigma \text{sip}))
\land \text{the (dests rip) - 1} \leq \text{nsqn (rt (\sigma \text{sip}) \text{rip})}"
by simp
have "\forall \text{rip} \in \text{dom(dests)}. \text{rip} \in kD(rt (\sigma' \text{sip}))
\land \text{the (dests rip) - 1} \leq \text{nsqn (rt (\sigma' \text{sip}) \text{rip})}"
proof
fix rip
assume "\text{rip} \in \text{dom(dests)}"
with \(*\) have "\text{rip} \in kD(rt (\sigma \text{sip}))" and "\text{the (dests rip) - 1} \leq \text{nsqn (rt (\sigma \text{sip}) \text{rip})}"
by -(drule(1) bspec, clarsimp)+
moreover from qinc have "quality_increases (\sigma \text{sip}) (\sigma' \text{sip})" by simp
ultimately show "\text{rip} \in kD(rt (\sigma' \text{sip})) \land \text{the (dests rip) - 1} \leq \text{nsqn (rt (\sigma' \text{sip}) \text{rip})}" ..
qed
thus \(?\)thesis by simp
qed simp_all

2.8 The ‘open’ AODV model

theory B_OAodv
imports B_AodvAWN.OAWN_SOS_LabelsAWN.OAWN_Convert
begin
Definitions for stating and proving global network properties over individual processes.
definition \( \sigma_{AODV}' :: "((ip \Rightarrow \text{state}) \times ((\text{state, msg, pseqp, pseqp label}) \text{seqp})) \text{set}" \)
where "\( \sigma_{AODV}' \equiv \{ \lambda i. \text{aodv}_i \text{init i, } \Gamma_{AODV} \text{PAodv})\}"

abbreviation \( \text{opaodv} :: \text{ip} \Rightarrow ((\text{ip} \Rightarrow \text{state}) \times (\text{state, msg, pseqp, pseqp label}) \text{seqp}, \text{msg seq action}) \text{automaton} \)
where "\( \text{opaodv} i \equiv (\{ \text{init = } \sigma_{AODV}', \text{trans = } \text{oseqp sos } \Gamma_{AODV} \text{ i \} })\)"

lemma \( \text{initiali_aodv \ [intro!, simp]} \): "\text{initiali i (init (opaodv i)) (init (paodv i))}\"
unfolding \( \sigma_{AODV}'_{\text{def}} \) \( \sigma_{AODV}'_{\text{def by rule simp all}} \)

lemma \( \text{oaodv_control_within \ [simp]} \): "\text{control_within } \Gamma_{AODV} \ (\text{init (opaodv i)})"
unfolding \( \sigma_{AODV}'_{\text{def by (rule control_withinI)} (auto simp del: } \Gamma_{AODV}_{\text{simps}}) \)

lemma \( \text{oaodv_labels \ [simp]} \): "\((\sigma, p) \in \sigma_{AODV}' \Rightarrow \text{labels } \Gamma_{AODV} \ p = \{ \text{PAodv:-0}\}"
unfolding \( \sigma_{AODV}'_{\text{def by simp}} \) \( \sigma_{AODV}'_{\text{def by simp}} \)

lemma \( \text{oaodv_init_kD_empty \ [simp]} \): "\((\sigma, p) \in \sigma_{AODV}' \Rightarrow \text{kD (rt (\sigma i))} = \{\}"
unfolding \( \sigma_{AODV}'_{\text{def by simp}} \)
lemma oaodv_init_vD_empty [simp]:
"(σ, p) ∈ σ_AODV' \implies vD (rt (σ i)) = {}"
unfolding σ_AODV'_def vD_def by simp

lemma oaodv_trans: "trans (opaodv i) = oseqp_sos Γ_AODV i"
by simp

declare oseq_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]
end

2.9 Global invariant proofs over sequential processes

theory B_Global_Invariants
imports B_Seq_Invariants
B_Aodv_Predicates
B_Fresher
B_Quality_Increases
AWN.OAWN_Convert
B_OAodv
begin

lemma other_quality_increases [elim]:
assumes "other quality_increases I σ σ'"
shows "∀ j. quality_increases (σ j) (σ' j)"
using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
fixes m
assumes *: "otherwith P I (orecvmsg Q) σ σ' a"
and weakenP: "∀ σ m. P σ m =⇒ P' σ m"
and weakenQ: "∀ σ m. Q σ m =⇒ Q' σ m"
shows "otherwith P' I (orecvmsg Q') σ σ' a"
proof
  fix j
  assume "j /∈ I"
  with * have "P (σ j) (σ' j)" by auto
  thus "P' (σ j) (σ' j)" by (rule weakenP)
next
  from * have "orecvmsg Q σ a" by auto
  thus "orecvmsg Q' σ a" by rule (erule weakenQ)
qed

lemma oreceived_msg_inv:
assumes other: "∀ σ σ' m. [ [ P σ m; other Q {i} (σ σ' m) =⇒ P σ' m ] ] =⇒ P σ' m"
and local: "∀ σ m. P σ m =⇒ P (σ (i := σ i | msg := m)) m"
shows "opaodv i = (otherwith Q {i} (orecvmsg P), other Q {i} → onl Γ_AODV (λ(σ, l). l ∈ {PAodv-:1} =⇒ P σ (msg (σ i))))"
proof (inv_cterms, intro impl)
  fix σ σ' l
  assume "l = PAodv-:1 =⇒ P σ (msg (σ i))"
  and "l = PAodv-:1" and "other Q {i} σ σ'"
  from this(1-2) have "P σ (msg (σ i))" ..
  hence "P σ' (msg (σ i))" using (other Q {i} σ σ')
  by (rule other)
  moreover from (other Q {i} σ σ') have "σ' i = σ i" ..
  ultimately show "P σ' (msg (σ' i))" by simp
next
  fix σ σ' msg
  assume "otherwith Q {i} (orecvmsg P) σ σ' (receive msg)"
and \( \sigma' i = \sigma i (msg := msg) \)

from this(1) have \( \text{P} \, \sigma \, msg \)
and \( \forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j) \) by auto

from this(1) have \( \text{P} (\sigma (i := \sigma i (msg := msg))) \, msg \) by (rule local)

thus \( \text{P} \, \sigma' \, msg \)

proof (rule other)

from \( \langle \sigma' i = \sigma i (msg := msg) \rangle \) and \( \langle \forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j) \rangle \)

show \( \text{other} \, Q \, (i) \, (\sigma (i := \sigma i (msg := msg))) \, \sigma' \)" by (rule otherI, auto)

qed

(Equivalent to) Proposition 7.27

lemma local_quality_increases:

"opaodv i \models_A (orecvmsg \, rreq_rrep_sn \rightarrow) \, onll \, \Gamma_{AODV} \, (\lambda((\xi, \_), \_, (\xi', \_)). \, quality_increases \, (\xi', \_))"

proof (rule step_invariantI)

fix \( \sigma \, p \, l \, a \)

assume or: "(\sigma, p) \in \text{oreachable} \, (opaodv \, i) \, ?S \, (\text{other} \, \text{quality_increases} \, \{i\})"
and ll: "l \in \text{labels} \, \Gamma_{AODV} \, p"
and "?S \, \sigma \, \sigma' \, a"
and tr: "((\sigma, p), a, (\sigma', p')) \in \text{oseqp sos} \, \Gamma_{AODV} \, i"
and ll': "l' \in \text{labels} \, \Gamma_{AODV} \, p'"

from this(1-3) have "orecvmsg \, (\lambda_. \, rreq_rrep_sn) \, \sigma \, a"

by (auto dest!: oreachable_weakenE [where QS="act (recvmsg \, rreq_rrep_sn)"
and QU="other \, \text{quality_increases} \, \{i\}"
otherwith_actionD]
with or have orw: "(\sigma, p) \in \text{oreachable} \, (opaodv \, i) \, (act \, (recvmsg \, rreq_rrep_sn))"

(other \, \text{quality_increases} \, \{i\})"

by - (erule oreachable_weakenE, auto)

with tr \, ll \, ll' \, and "orecvmsg \, (\lambda_. \, rreq_rrep_sn) \, \sigma \, a" \, have "quality_increases \, (\sigma \, i) \, (\sigma' \, i)"

by - (drule onll_ostep_invariantD [OF local_quality_increases], auto simp: seqll_def)
with "?S \, \sigma \, \sigma' \, a" show "\forall j. \, quality_increases \, (\sigma \, j) \, (\sigma' \, j)"
lemma rreq_rrep_nsqn_fresh_any_step_invariant:
"opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} →)
onll Γ_AODV (λ((ξ, _), a, _). anycast (msg_fresh σ) a)"
proof (rule ostep_invariantI, simp del: act_simp)

fix σ p a σ' p'
assume or: "((σ, p), a, (σ', p')) ∈ osqp sos Γ_AODV i"
and recv: "act (recvmsg rreq_rrep_sn) σ σ' a"

obtain l l' where "l ∈ labels Γ_AODV p" and "l' ∈ labels Γ_AODV p'" by (metis aodv_ex_label)

from ⟦((σ, p), a, (σ', p')) ∈ osqp sos Γ_AODV i⟧
have tr: "((σ, p), a, (σ', p')) ∈ trans (opaodv i)" by simp

have "anycast (rreq_rrep_fresh (rt (σ i))) a"
proof -
have "opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} →)
onll Γ_AODV (seqll i (λ((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a))"
by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aodv, simplified seqll_onll_swap]] auto)
hence "onll Γ_AODV (seqll i (λ((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a)) ((σ, p), a, (σ', p'))"
using or tr recv by - (erule(4) ostep_invariantE)
thus ?thesis using ⟨l ∈ labels Γ_AODV p⟧ and ⟨l' ∈ labels Γ_AODV p'⟧ by auto

qed

moreover have "anycast (rerr_invalid (rt (σ i))) a"
proof -
have "opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} →)
onll Γ_AODV (seqll i (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a))"
by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aodv, simplified seqll_onll_swap]] auto)
hence "onll Γ_AODV (seqll i (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a)) ((σ, p), a, (σ', p'))"
using or tr recv by - (erule(4) ostep_invariantE)
thus ?thesis using ⟨l ∈ labels Γ_AODV p⟧ and ⟨l' ∈ labels Γ_AODV p'⟧ by auto

qed

moreover have "anycast rreq_rrep_sn a"
proof -
from or tr recv
have "onll Γ_AODV (seqll i (λ(ξ, a, _). anycast rreq_rrep_sn a)) ((σ, p), a, (σ', p'))"
by (rule ostep_invariantE [OF open_seq_step_invariant [OF rreq_rrep_sn_any_step_invariant initiali_aodv oaodv_trans aodv_trans, simplified seqll_onll_swap]])
thus ?thesis using ⟨l ∈ labels Γ_AODV p⟧ and ⟨l' ∈ labels Γ_AODV p'⟧ by auto

qed

moreover have "anycast (λm. not_Pkt m → msg_sender m = i) a"
proof -
have "opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} →)
onll Γ_AODV (seqll i (λ((ξ, _), a, _). anycast (λm. not_Pkt m → msg_sender m = i) a))"
by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF sender_ip_valid initiali_aodv, simplified seqll_onll_swap]] auto)
thus ?thesis using or tr recv ⟨l ∈ labels Γ_AODV p⟧ and ⟨l' ∈ labels Γ_AODV p'⟧ by auto

qed
ultimately have "anycast (msg_fresh σ) a"
by (simp_all add: anycast_def
del: msg_fresh
split: seq_action.split_asm msg.split_asm) simp_all
thus "onll Γ_AODV (λ((σ, _), a, _). anycast (msg_fresh σ) a) ((σ, p), a, (σ', p'))"
by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaodv i \|= \ (otherwith quality_increases \{i\} (orecvmsg msg_fresh),
other quality_increases \{i\} \rightarrow)
onl Γ_AODV (λ((σ, _), a, _). anycast (msg_fresh σ) a) ((σ, p), a, (σ', p'))"
proof (rule oreceived_msg_inv)
  fix σ σ' p l
  assume *: "msg_fresh σ m"
  and "other quality_increases \{i\} σ σ'"
  from this(2) have "∀ j. quality_increases (σ j) (σ' j)" ..
  thus "msg_fresh σ' m" using * ..
next
  fix σ m
  assume "msg_fresh σ m"
  thus "msg_fresh (σ(i := σ i(\|msg := m\|)) m)"
  proof (cases m)
    fix dests sip
    assume "m = Rerr dests sip"
    with (msg_fresh σ m) show "?thesis" by auto
  qed
  auto
qed

lemma oquality_increases_nsqn_fresh:
"opaodv i \|= \ (otherwith quality_increases \{i\} (orecvmsg msg_fresh),
other quality_increases \{i\} \rightarrow)
onl Γ_AODV (λ((σ', _, _), _, (σ', _)). ∀ j. quality_increases (σ j) (σ' j))"
yby (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
"opaodv i \|= \ (otherwith quality_increases \{i\} (orecvmsg msg_fresh),
other quality_increases \{i\} \rightarrow)
onll Γ_AODV (λ(ξ, l). l ∈ \{PAodv-:4, PAodv-:5\} ∪ \{PRreq-:n \in. True\} \rightarrow \leq osn ξ))"
yby (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])
(auto simp: seql_onl_swap)

lemma rreq_sip:
"opaodv i \|= \ (otherwith quality_increases \{i\} (orecvmsg msg_fresh),
other quality_increases \{i\} \rightarrow)
onl Γ_AODV (λ(ξ, l). (l ∈ \{PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2\} ∧ sip (σ i) ≠ oip (σ i))
  \rightarrow oip (σ i) ∈ kd(rt (σ (sip (σ i))))
  ∧ nsqn (rt (σ (sip (σ i)))) (oip (σ i)) ≥ osn (σ i)
  \land (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i))
  \rightarrow (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (oip (σ i)))
  ∨ the (flag (rt (σ (sip (σ i)))) (oip (σ i))) = inv))" (is "_ \|= \ (\Rightarrow ?S, ?U \rightarrow) _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh]
aodv_wf oodv_trans)
onl oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
onl oinvariant_sterms [OF aodv_wf oosn_rreq]
simp add: seqlsimp
simp del: [me_nat_def, rule impI]
fix σ σ' p l
assume "((σ, p) ∈ oreachable (opaodv i) ?S ?U)"
and "1 ∈ labels $\Gamma_{AODV}$ p"
and pre:

\[
\begin{align*}
(1 = PAodv:-4 \lor l = PAodv:-5 \lor l = PRreq:-0 \lor l = PRreq:-2) \land sip (\sigma i) \neq oip (\sigma i) \\
\rightarrow oip (\sigma i) \in kd (rt (\sigma (sip (\sigma i)))) \\
\land osn (\sigma i) \leq nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)) \\
\land (nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = osn (\sigma i) \\
\rightarrow \text{the (dhops (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) \leq hops (\sigma i)} \\
\lor \text{the (flag (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = inv})
\end{align*}
\]

and "other_quality_increases {i} \sigma \sigma’"
and hyp: "(1=PAodv:-4 \lor l=PAodv:-5 \lor l=PRreq:-0 \lor l=PRreq:-2) \land sip (\sigma i) \neq oip (\sigma i)"
is "?labels \land sip (\sigma i) \neq oip (\sigma i)"
from this(4) have "\sigma’ i = \sigma i" ..

with hyp have hyp': "?labels \land sip (\sigma i) \neq oip (\sigma i)" by simp

show "oip (\sigma i) \in kd (rt (\sigma (sip (\sigma i)))) \\
\land osn (\sigma i) \leq nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)) \\
\land (nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = osn (\sigma i) \\
\rightarrow \text{the (dhops (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) \leq hops (\sigma i)} \\
\lor \text{the (flag (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = inv})"

proof (cases "sip (\sigma i) = i")

assume "sip (\sigma i) \neq i"
from "other_quality_increases {i} \sigma \sigma’", have "quality_increases (\sigma (sip (\sigma i))) (\sigma i) = oip (\sigma i)"
by (rule otherE simp: clarsimp simp: simp (\sigma i) \neq i)

moreover from \langle(\sigma, p) \in ooreachable (opaodv i) \rangle \langle \sigma \rangle \langle \sigma \rangle \langle \text{labels}(\sigma i) \rangle \langle \text{labels}(\sigma i) \rangle and hyp
have "1 \leq osn (\sigma i)"
by (auto dest!: onl_oninvariant_weakend [OF oosn_rreq]
simp add: seqlessimp simp (\sigma i) = \sigma i)

moreover from "\text{labels}(\sigma i) \neq i" hyp' and pre
have "oip (\sigma i) \in kd (rt (\sigma (sip (\sigma i)))) \\
\land osn (\sigma i) \leq nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)) \\
\land (nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = osn (\sigma i) \\
\rightarrow \text{the (dhops (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) \leq hops (\sigma i)} \\
\lor \text{the (flag (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = inv})"

by (auto simp: simp (\sigma i) = \sigma i)

ultimately show ?thesis
by (rule quality_increases_rreq_rrep_props)

next
assume "sip (\sigma i) = i" thus ?thesis
using simp (\sigma i) = \sigma i hyp and pre by auto

qed

qed (auto elim!: quality_increases_rreq_rrep_props’)

lemma odsn_rrep:

"opaodv i \models (otherwith quality_increases {i} orecvmsg msg_fresh), other качества_increases {i} \rightarrow \onl{\Gamma_{AODV}} (seq i (\lambda(i, l). l \in \{PAodv:-6, PAodv:-7\} \cup \{PRreq:-nl\} True \rightarrow 1 \leq dsn \xi))"

by (rule oninvariant_weakend [OF open_seq_invariant [OF dsn_rrep_initiali_aodv]]
(auto simp: seql_onl_swap)

lemma rrep_sip:

"opaodv i \models (otherwith quality_increases {i} orecvmsg msg_fresh), other качества_increases {i} \rightarrow \onl{\Gamma_{AODV}} (\lambda(\sigma, i).

(1 \in \{PAodv:-6, PAodv:-7, PRreq:-0, PRreq:-1\} \land sip (\sigma i) \neq dip (\sigma i) \\
\rightarrow dip (\sigma i) \in kd(rt (\sigma (sip (\sigma i)))) \\
\land nsqn (rt (\sigma (sip (\sigma i)))) (dip (\sigma i)) \geq dsn (\sigma i) \\
\land (nsqn (rt (\sigma (sip (\sigma i)))) (dip (\sigma i))) = dsn (\sigma i) \\
\rightarrow (\text{hops (\sigma i) \geq the (dhops (rt (\sigma (sip (\sigma i)))) (dip (\sigma i)))} \\
\lor \text{the (flag (rt (\sigma (sip (\sigma i)))) (dip (\sigma i))) = inv})")

(is _ " | (?S, ?U \rightarrow _")

proof (inv_cterms inv add: oseq_step_invariant_stems [OF quality_increases_nsqn_fresh aodv_wf aodv_trans]
onl_oninvariant_stems [OF aodv_wf orecieved_rreq_rrep_nsqn_fresh_inv]
onl_oninvariant_stems [OF aodv_wf odsn_rrep]

231
simp del: One_nat_def, rule \text{impl})

fix \sigma', \rho \in \text{oreachable (opaodv \ i) ?S ?U} 
and "1 \in \text{labels } \Gamma_{AODV} \ p"
and \text{pre:}

\text{'(1 = PAdv--\:6 \lor 1 = PAdv--\:7 \lor 1 = PRrep--\:0 \lor 1 = PRrep--\:1) \land \text{dip (} \sigma \ i) \neq \text{dip (} \sigma' \ i)"

\text{\rightarrow dip (} \sigma' \ i) \in kD (rt (\sigma (\text{dip (} \sigma \ i)))\}
\land \text{dsn (} \sigma' \ i) \leq \text{nsqn (} \sigma (\text{dip (} \sigma \ i))) (\text{dip (} \sigma' \ i))\}
\land (\text{nsqn (} \sigma (\text{dip (} \sigma \ i))) (\text{dip (} \sigma' \ i)) = \text{dsn (} \sigma' \ i)\}
\text{\rightarrow the (dhops (} \sigma (\text{dip (} \sigma \ i))) (\text{dip (} \sigma' \ i)) \leq \text{hops (} \sigma' \ i)\}
\lor \text{the (flag (} \sigma (\text{dip (} \sigma \ i))) (\text{dip (} \sigma' \ i)) = \text{inv})"}\}

and "other quality_increases \{i\} \sigma \sigma'"

and \text{hyp: } "(l=PAodv--\:6 \lor l=PAodv--\:7 \lor l=PRrep--\:0 \lor l=PRrep--\:1) \land \text{dip (} \sigma \ i) \neq \text{dip (} \sigma' \ i)"
(is \{?labels \land \text{dip (} \sigma' \ i) \neq \text{dip (} \sigma \ i)\})"

from(4) have "\sigma' \ i = \sigma \ i" ...

with \text{hyp have hyp': } "\text{labels \land \text{dip (} \sigma' \ i) \neq \text{dip (} \sigma \ i)\} by simp

show "\text{dip (} \sigma' \ i) \in kD (rt (\sigma (\text{dip (} \sigma' \ i)))\}
\land \text{dsn (} \sigma' \ i) \leq \text{nsqn (} \sigma (\text{dip (} \sigma' \ i))) (\text{dip (} \sigma' \ i))\}
\land (\text{nsqn (} \sigma (\text{dip (} \sigma' \ i))) (\text{dip (} \sigma' \ i)) = \text{dsn (} \sigma' \ i)\}
\text{\rightarrow the (dhops (} \sigma (\text{dip (} \sigma' \ i))) (\text{dip (} \sigma' \ i)) \leq \text{hops (} \sigma' \ i)\}
\lor \text{the (flag (} \sigma (\text{dip (} \sigma' \ i))) (\text{dip (} \sigma' \ i)) = \text{inv})"\}

proof (cases "\text{dip (} \sigma \ i) = \text{i}"

assume "\text{dip (} \sigma \ i) \neq \text{i}"
from (other quality_increases \{i\} \sigma \sigma')
have "other quality_increases (\sigma (\text{dip (} \sigma \ i))) (\sigma' (\text{dip (} \sigma' \ i)))"
by (rule otherE) (clarsimp simp: \sigma \sigma' \neq \text{i})

moreover from (\sigma, \rho \in \text{oreachable (opaodv \ i) ?S ?U} \ i \in \text{labels } \Gamma_{AODV} \ p) and \text{hyp}
have "1 \leq \text{dsn (} \sigma' \ i)"
by (auto dest!: onl_oneinvariant_weakenD [OF odsn_rrep]
\text{simp add: sqnlsimp (} \sigma' \ i = \sigma \ i)\)

moreover from (\text{dip (} \sigma \ i) \neq \text{i}) \text{hyp' and pre}
have "\text{dip (} \sigma' \ i) \in kD (rt (\sigma (\text{dip (} \sigma' \ i)))\}
\land \text{dsn (} \sigma' \ i) \leq \text{nsqn (} \sigma (\text{dip (} \sigma' \ i))) (\text{dip (} \sigma' \ i))\}
\land (\text{nsqn (} \sigma (\text{dip (} \sigma' \ i))) (\text{dip (} \sigma' \ i)) = \text{dsn (} \sigma' \ i)\}
\text{\rightarrow the (dhops (} \sigma (\text{dip (} \sigma' \ i))) (\text{dip (} \sigma' \ i)) \leq \text{hops (} \sigma' \ i)\}
\lor \text{the (flag (} \sigma (\text{dip (} \sigma' \ i))) (\text{dip (} \sigma' \ i)) = \text{inv})"\}

by (auto simp: \sigma' \ i = \sigma \ i)\)

ultimately show \text{?thesis}
by (rule quality_increases_rreq_rrep_props)

next
assume "\text{dip (} \sigma \ i) = \text{i}" thus \text{?thesis
using \sigma' \ i = \sigma \ i \text{hyp and pre by auto
qed

qed (auto simp add: sqnlsimp elim!: quality_increases_rreq_rrep Props')

lemma rerr_sip:
"opaodv \ i \ | (\text{otherwise quality_increases \{i\} (orecmsg msg_fresh),
other quality_increases \{i\} \rightarrow)
\text{onl } \Gamma_{AODV} (\lambda (\sigma, \ i)\).
\i \in \{PAdv--\:8, PAdv--\:9, PRrep--\:0, PRrep--\:1\}
\rightarrow (\forall \text{ripc \in dom(dests (} \sigma \ i)). \ \text{ripc \in kD(rt (} \sigma (\text{dip (} \sigma \ i)))\}
\land \text{the (dests (} \sigma \ i) \ \text{ripc) - 1 \leq nsqn (rt (} \sigma (\text{dip (} \sigma \ i))) \ \text{ripc)})"\)

(is _ | (?S, \ \text{?U \rightarrow } _) )"

proof -
{ fix dests rip sip rsn and \sigma \sigma': "ip \Rightarrow state"
assume qinc: "\forall j. quality_increases (} \sigma \ j) (} \sigma' \ j)"
and *: "\forall \text{ripc \in dom dests. rip \in kD (rt (} \sigma \ sip))
\land \text{the (dests rip) - 1 \leq nsqn (rt (} \sigma \ sip)) rip"\}

and "dests rip = Some rsn"
from this(3) have "\text{ripc \in dom dests}" by auto
with * and (dests rip = Some rsn) have "\text{ripc \in kD(rt (} \sigma \ sip))"
and "rsn - 1 \leq nsqn (rt (} \sigma \ sip)) rip"

by (auto dest!: bspec)
from qinc have "other quality_increases (} \sigma \ sip) (} \sigma' \ sip) .."
have "rip ∈ kD(rt (σ' sip)) ∧ rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
proof
  from ⟨rip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  show "rip ∈ kD(rt (σ' sip))" ..
next
  from ⟨rip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ' sip)) rip" ..
  with ⟨rsn - 1 ≤ nsqn (rt (σ sip)) rip⟩ show "rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
  by (rule le_trans)
qed
} note partial = this

show ?thesis
by (inv_cterms inv add: oseq_step_invariant_sterms [OF equality_increases_nsqn_fresh aodv_wf aodv_trans]
  onl_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  other_quality_increases other_localD
  simp del: One_nat_def, intro conjI)

lemma prerr_guard: "paodv i ||
  onl Γ AODV (λ(ξ, l). (l = PRerr-:
  (∀ ip∈dom(dests ξ). ip∈vD(rt ξ)
  ∧ the (nhop (rt ξ) ip) = sip ξ
  ∧ sqn (rt ξ) ip < the (dests ξ ip))))"
by (inv_cterms) (clarsimp split: option.split_asm)

lemmas oaddpreRT_welldefined =
  open_seq_invariant [OF addpreRT_welldefined initiali_aodv oadv_trans aodv_trans,
  simplified seql_onl_swap,
  THEN oinvariant_anyact]

lemmas odests_vD_inc_sqn =
  open_seq_invariant [OF dests_vD_inc_sqn initiali_aodv oadv_trans aodv_trans,
  simplified seql_onl_swap,
  THEN oinvariant_anyact]

lemmas oprerr_guard =
  open_seq_invariant [OF prerr_guard initiali_aodv oadv_trans aodv_trans,
  simplified seql_onl_swap,
  THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other_quality_increases {i} → onl Γ AODV (λ(σ, _).
  ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
  in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →
  dip ∈ kD(rt (σ nhip)) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"
(is "_ |= (?S, ?U → _)")

proof -

{ fix nhop and σ σ' :: "ip ⇒ state"
  assume pre: "∀ dip∈kD(rt (σ i)). nhop dip ≠ dip →
    dip∈kD(rt (σ (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
  and qinc: "∀ j. quality_increases (σ j) (σ' j)"
  have "∀ dip∈kD(rt (σ i)). nhop dip ≠ dip →
    dip∈kD(rt (σ' (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip
    assume "dip∈kD(rt (σ i))"
    and "nhop dip ≠ dip"
    with pre have "dip∈kD(rt (σ (nhop dip)))"

    from ⟨rip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
    show "rip ∈ kD(rt (σ' sip))" ..
next
  from ⟨rip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ' sip)) rip" ..
  with ⟨rsn - 1 ≤ nsqn (rt (σ sip)) rip⟩ show "rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
  by (rule le_trans)
qed
and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
by auto
from qinc have qinc_nhop: "quality_increases (σ (nhop dip)) (σ' (nhop dip))"
with ⟨dip∈kD(rt (σ (nhop dip)))) have "dip∈kD (rt (σ' (nhop dip)))"
moreover have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
proof -
from ⟨dip∈kD(rt (σ (nhop dip)))⟩ qinc_nhop
have "nsqn (rt (σ (nhop dip))) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
with ⟨dip∈kD(rt (σ (nhop dip)))⟩ have "dip∈kD (rt (σ' (nhop dip)))"
proof -
from ⟨dip∈kD(rt (σ (nhop dip)))⟩ qinc_nhop
have "nsqn (rt (σ (nhop dip))) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
with ⟨dip∈kD(rt (σ (nhop dip)))⟩ show ?thesis
by simp
qed
ultimately show "dip∈kD(rt (σ' (nhop dip)))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
qed
}

{ fix nhop and σ σ' :: "ip ⇒ state"
assume pre: "∀ dip∈kD(rt (σ i)). nhop dip ≠ dip → dip∈kD(rt (σ (nhop dip)))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
and ndest: "∀ ripc∈dom (dests (σ i)). ripc ∈ kD (rt (σ (sip (σ i))))
∧ the (dests (σ i) ripc) - 1 ≤ nsqn (rt (σ (sip (σ i)))) ripc"
and issip: "∀ ip∈dom (dests (σ i)). nhop ip = sip (σ i)"
and qinc: "∀ j. quality_increases (σ j) (σ' j)"
have "∀ dip∈kD(rt (σ i)). nhop dip ≠ dip → dip∈kD(rt (σ' (nhop dip)))
∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
proof (intro ballI impI)
fix dip
assume "dip∈kD(rt (σ i))"
and "nhop dip ≠ dip"
with pre and qinc have "dip∈kD(rt (σ' (nhop dip)))"
and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
by (auto dest!: basic)

have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
proof (cases "dip∈dom (dests (σ i))")
assume "dip∈dom (dests (σ i))"
with ⟨dip∈kD(rt (σ i))) obtain dsn where "dests (σ i) dip = Some dsn"
by auto
with ⟨dip∈kD(rt (σ i))⟩ have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = dsn - 1"
by (rule nsqn_invalidate_eq)
moreover have "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
proof -
from ⟨dests (σ i) dip = Some dsn⟩ have "the (dests (σ i) dip) = dsn" by simp
with ⟨dip∈kD(rt (σ i))⟩ show "dip∈kD (rt (σ (sip (σ i))))"
"dsn - 1 ≤ nsqn (rt (σ (sip (σ i))))" dip"
by auto
moreover from issip and ⟨dip∈dom (dests (σ i))⟩ have "nhop dip = sip (σ i)"
ultimately have "dip∈kD (rt (σ (nhop dip))))"
and "dsn - 1 ≤ nsqn (rt (σ (nhop dip))) dip" by auto
with qinc show "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
by simp (metis kD_nsqn_quality_increases_trans)
qed
ultimately show ?thesis by simp
next
assume "dip /∈ dom (dests (σ i))"
with ⟨dip∈kD(rt (σ i)))⟩ have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = nsqn (rt (σ i)) dip"
by (rule nsqn_invalidate_other)
with ⟨dip∈kD(rt (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip⟩ show ?thesis by simp
qed
with ⟨dip∈kD(rt (σ' (nhop dip)))⟩ show "dip∈kD (rt (σ' (nhop dip)))"
\[
\wedge \text{nsqn } (\text{invalidate } (\text{rt } (\sigma i)) \text{ (dests } (\sigma i))) \text{ dip } \leq \text{nsqn } (\text{rt } (\sigma (\text{nhop dip}))) \text{ dip}
\]

qed

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}
qed
} note update1 = this

\{ fix σ σ' oip sip osn hops
  assume pre: "∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip →
  dip ∈ kD (rt (σ (the (nhop (rt (σ i)) dip)))) ∧
  nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀ j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip →
  osn ≤ nsqn (rt (σ sip)) oip ∧
  (nsqn (rt (σ sip)) oip = osn →
  the (dhops (rt (σ sip)) oip) ≤ hops
  ∨ the (flag (rt (σ sip)) oip) = inv)"

  from pre and qinc
  have pre': "∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip →
  dip ∈ kD (rt (σ' (the (nhop (rt (σ i)) dip)))) ∧
  nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"

  by (rule basic)
  have "∀ dip ∈ kD (rt (σ i)). the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip →
  dip ∈ kD (rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))) ∧
  nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))) dip"

  (is "∀ dip ∈ kD (rt (σ i)). _ _ → ?dip_in_kD dip ∧ ?nsqn_le_nsqn dip")

  proof (intro ballI impI, split update_rt_split_asm)
  fix dip
  assume "dip ∈ kD (rt (σ i))"
  and "the (nhop (rt (σ i)) dip) ≠ dip"
  and "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"

  with pre' show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip" by simp

  next

  fix dip
  assume "dip ∈ kD (rt (σ i))"
  and notdip: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"
  and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"

  show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip"

  proof (cases "dip = oip")
  assume "dip ≠ oip"
  with pre' (dip ∈ kD (rt (σ i))) notdip
  show ?thesis by clarsimp

  next

  assume "dip = oip"
  with rtnot qinc (dip ∈ kD (rt (σ i))) notdip *
  have "?dip_in_kD dip"

  by simp (metis kD_quality_increases)

  moreover from (dip = oip) rtnot qinc (dip ∈ kD (rt (σ i))) notdip *
  have "?nsqn_le_nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)

  ultimately show ?thesis ..

  qed

  qed

} note update2 = this

have "opaodv i |= (?S, ?U →) onl Γ_AODV (λ(σ, _).
  ∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip →
  dip ∈ kD (rt (σ (the (nhop (rt (σ i)) dip)))) ∧
  nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip)"

  by (inv_cterms inv add: oseq_step_invariant_sterms [OF quality_increases_nsqn_fresh aodv_wf oaoadv_trans]
  onl_oinvariant_sterms [OF aodv_wf oaddpreRT_welldefined]
  onl_oinvariant_sterms [OF aodv_wf odests_vD_inc_sqn]
  onl_oinvariant_sterms [OF aodv_wf oprerr_guard]

236
thus \( ?\text{thesis} \) unfolding Let_def by auto

qed

Proposition 7.30

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aodv,
  simplified seql_onl_swap]

lemmas ozero_seq_unk_hops_one =
  open_seq_invariant [OF zero_seq_unk_hops_one initiali_aodv,
  simplified seql_onl_swap]

lemma ororeachable_fresh_okD_unk_or_atleast_one:
  fixes dip
  assumes "\( (\sigma, p) \in \text{oreachable (opaodv } i) \)"
  (otherwith \( ((=)) \) \( \{i\} \) \( \text{orecvmsg } (\lambda \sigma \ m. \ \text{msg_fresh } \sigma \ m \land \ \text{msg_zhops } m) \)\)
  (other quality_increases \( \{i\} \)"
  and "\( \text{dip} \in kD(\text{rt } (\sigma \ i)) \)"
  shows "\( \pi_3(\text{the (rt } (\sigma \ i) \text{ dip})) = \text{unk} \lor 1 \leq \pi_2(\text{the (rt } (\sigma \ i) \text{ dip})) \)"
  (is "\(?P \ \text{dip}\)"

proof -
  have "\( \exists l. l \in \text{labels } \Gamma_{AODV} \ p \)" by (metis aodv_ex_label)
  with assms(1) have "\( \forall \text{dip} \in kD(\text{rt } (\sigma \ i)). \ ?P \ \text{dip} \)"
  by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaodv_trans aodv_trans]],
  auto dest!: oinvariant_prerr simp del: One_nat_def)
  with (\( \text{dip} \in kD(\text{rt } (\sigma \ i)) \)) show ?thesis by simp
qed

lemma ororeachable_fresh_ozero_seq_unk_hops_one:
  fixes dip
  assumes "\( (\sigma, p) \in \text{oreachable (opaodv } i) \)"
  (otherwith \( ((=)) \) \( \{i\} \) \( \text{orecvmsg } (\lambda \sigma \ m. \ \text{msg_fresh } \sigma \ m \land \ \text{msg_zhops } m) \)\)
  (other quality_increases \( \{i\} \)"
  and "\( \text{dip} \in kD(\text{rt } (\sigma \ i)) \)"
  shows "\( \text{sqn } (\text{rt } (\sigma \ i) \text{ dip}) = 0 \rightarrow \text{sqnf } (\text{rt } (\sigma \ i) \text{ dip}) = \text{unk} \land \text{the (dhops } (\text{rt } (\sigma \ i) \text{ dip})) = 1 \land \text{the (nhop } (\text{rt } (\sigma \ i) \text{ dip})) = \text{dip} \)"
  (is "\(?P \ \text{dip}\)"

proof -
  have "\( \exists l. l \in \text{labels } \Gamma_{AODV} \ p \)" by (metis aodv_ex_label)
  with assms(1) have "\( \forall \text{dip} \in kD(\text{rt } (\sigma \ i)). \ ?P \ \text{dip} \)"
  by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]],
  auto dest!: oinvariant_prerr simp del: One_nat_def)
  with (\( \text{dip} \in kD(\text{rt } (\sigma \ i)) \)) show ?thesis by simp
qed

lemma seq_nhopt_quality_increases':
  shows "\( \text{opaodv } i \models (\text{otherwith } ((=)) \) \( \{i\} \) \)
  \( \text{orecvmsg } (\lambda \sigma \ m. \ \text{msg_fresh } \sigma \ m \land \ \text{msg_zhops } m) , \)
  other quality_increases \( (\{i\} \rightarrow) \)
  onl \( \Gamma_{AODV} (\lambda(\sigma, _). \ \forall \text{dip} \text{. let nhp } = \text{the (nhop } (\text{rt } (\sigma \ i) \text{ dip})) \)
\(\text{in dip} \in \text{vD} (\text{rt (σ i)}) \cap \text{vD} (\text{rt (σ nhip)})\)
\[\land \text{nhip} \neq \text{dip} \rightarrow (\text{rt (σ i)}) \sqsubseteq \text{dip (rt (σ nhip))}\]

(is "_ _ \models(?S i, _ \rightarrow _) _")

proof -

have weaken:
\[
\forall I \ Q \ R \ P. p \models (\text{otherwith quality_increases I (orecmsg Q), other quality_increases I \rightarrow P} \implies p \models (\text{otherwith ((=)) I (orecmsg (λσ, m. Q σ m \land R σ m)), other quality_increases I \rightarrow P})
\]

by auto

\{ 
fix a and σ' :: "ip ⇒ state"
assume a1: "∀ dip. dip ∈ \text{vD} (\text{rt (σ i)})\)
\[\land \text{dip ∈ vD (rt (σ (σ i)) dip)), dip ∈ vD (rt (σ (σ i)) dip)) \neq dip \rightarrow rt (σ i) \sqsubseteq dip \text{rt (σ (dip (σ i) dip))}"\]

and ow: "?S i σ' a"

have "∀ dip. dip ∈ \text{vD} (\text{rt (σ i)}))\)
\[\land \text{dip ∈ vD (rt (σ (σ i)) dip)), dip ∈ vD (rt (σ (σ i)) dip)) \neq dip \rightarrow rt (σ i) \sqsubseteq dip \text{rt (σ (dip (σ i) dip))}"\]

\}

proof clarify

fix dip

assume a2: "dip ∈ \text{vD} (\text{rt (σ i)})"

and a3: "dip ∈ \text{vD} (rt (σ (σ i)) dip), dip ∈ vD (rt (σ (σ i)) dip)) \neq dip \rightarrow rt (σ i) \sqsubseteq dip rt (σ (dip (σ i) dip))"

and a4: "(the (nhop (rt (σ i)) dip)) \neq dip"

from ow have "∀ j. j \neq i \rightarrow σ j = σ' j" by auto

show "rt (σ i) \sqsubseteq dip rt (σ' (the (nhop (σ i)) dip))" by simp

proof (cases "(the (nhop (σ i)) dip) = i")

assume "((dip (σ i) dip)) = i"

with \( d dip \in \text{vD (rt (σ i))} \) have "dip ∈ \text{vD} (\text{rt (σ (σ i)) dip)), dip ∈ vD (rt (σ (σ i)) dip)) \neq dip\) by simp

with a1 a2 a4 have "rt (σ i) \sqsubseteq dip rt (σ (σ (σ i) dip)), dip ∈ vD (rt (σ (σ i)) dip))" by simp

with (the (nhop (σ i) dip)) = i have "rt (σ i) \sqsubseteq dip rt (σ (σ (σ i) dip)), dip ∈ vD (rt (σ (σ i) dip))" by simp

hence False by simp

thus ?thesis ..

next

assume "((dip (σ i) dip)) \neq i"

with ∀ j. j \neq i \rightarrow σ j = σ' j

have "σ (the (σ i)) dip) = σ' (the (σ i)) dip))" by simp

with \( dip ∈ \text{vD (rt (σ i))} \) have "dip ∈ \text{vD} (\text{rt (σ (σ i)) dip ), dip ∈ vD (rt (σ (σ i)) dip)) \neq dip\) by simp

with a1 a2 a4 have "rt (σ i) \sqsubseteq dip rt (σ (σ (σ i) dip)), dip ∈ vD (rt (σ (σ i) dip))" by simp

with * show ?thesis by simp

qed

qed

\{ 
fix σ σ' a dip i

assume a1: "∀ dip. dip ∈ \text{vD} (\text{rt (σ i)})\)
\[\land \text{dip ∈ vD (rt (σ (σ i)) dip)), dip ∈ vD (rt (σ (σ i)) dip)) \neq dip \rightarrow rt (σ i) \sqsubseteq dip rt (σ (dip (σ i) dip))"\]

and ow: "?S i σ' a"

have "∀ dip. dip ∈ \text{vD} (\text{update (rt (σ i)) dip), dip ∈ vD (rt (σ (σ i)) dip)), dip ∈ vD (rt (σ (σ i)) dip)) \neq dip \rightarrow update (σ i) dip) = i have "rt (σ i) \sqsubseteq dip rt (σ (dip (σ i) dip)), dip ∈ vD (rt (σ (σ i) dip))" by simp

proof clarify

fix dip

assume a2: "dip ∈ \text{vD} (\text{update (rt (σ i)) dip), dip ∈ vD (rt (σ (σ i)) dip)), dip ∈ vD (rt (σ (σ i)) dip)) \neq dip\) by simp

and a3: "(the (update (σ i) dip)) \neq dip\) by simp

and a4: "(the (update (σ i) dip)) \neq dip"

show "update (σ i) dip) = i have "rt (σ i) \sqsubseteq dip rt (σ (dip (σ i) dip)), dip ∈ vD (rt (σ (σ i) dip))" by simp

238
\[ dip \ rt (\sigma' (the (nhop (update \ rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {}))) dip) \]

proof (cases "dip = sip")
assume "dip = sip"
with \langle the (nhop (update \ rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip \rangle \neq dip
have False by simp
thus ?thesis ..

next
assume [simp]: "dip \neq sip"
from a2 have "dip\in VD(\rt (\sigma i)) \lor dip = sip"
  by (rule vD_update_val)
with \langle dip \neq sip \rangle have "dip\in VD(\rt (\sigma i))" by simp
moreover from a3 have "dip\in VD(\rt (\sigma' (the (nhop (\rt (\sigma i)) dip))))" by simp
moreover from a4 have "the (nhop (\rt (\sigma i)) dip) \neq dip" by simp
ultimately have "\rt (\sigma i) \sqsubseteq dip \ rt (\sigma' (the (nhop (\rt (\sigma i)) dip)))"

using a1 ow by - (drule(I) basic, simp)
with \langle dip \neq sip \rangle show ?thesis
by - (erule rt_strictly_fresher_update_other, simp)

qed

qed

\}

note update_0_unk = this

\{
fix \sigma a \sigma' nhop
assume pre: "\forall dip. dip\in VD(\rt (\sigma i)) \land dip\in VD(\rt (\sigma (nhop dip))) \land nhop dip \neq dip
  \rt (\sigma i) \sqsubseteq dip \ rt (\sigma (nhop dip))"

and ow: "?S i \sigma \sigma' a"

have "\forall dip. dip \in VD (invalidate (rt (\sigma i)) (dests (\sigma i)))
  \land dip \in VD (rt (\sigma' (nhop dip))) \land nhop dip \neq dip
  \rt (\sigma i) \sqsubseteq dip \ rt (\sigma' (nhop dip))"

proof clarify

fix dip

assume "dip\in VD(invalidate (\rt (\sigma i)) (dests (\sigma i)))"

and "dip\in VD(\rt (\sigma' (nhop dip)))"

and "nhop dip \neq dip"

from this(I) have "dip\in VD (\rt (\sigma i))"

by (clarsimp dest!: vD_invalidate_vD_not_dests)

moreover from ow have "\forall j. j \neq i \rightarrow \sigma j = \sigma' j" by auto

ultimately have "\rt (\sigma i) \sqsubseteq dip \ rt (\sigma' (nhop dip))"

using pre \langle dip \in VD (\rt (\sigma' (nhop dip))) \rangle \land nhop dip \neq dip

by metis

with \forall j. j \neq i \rightarrow \sigma j = \sigma' j: show "\rt (\sigma i) \sqsubseteq dip \ rt (\sigma' (nhop dip))"

by (metis rt_strictly_fresher_iqrefl)

qed

\}

note invalidate = this

\{
fix \sigma a \sigma' dip oip osn sip hops i
assume pre: "\forall dip. dip \in VD (\rt (\sigma i))
  \land dip \in VD (\rt (\sigma (the (nhop (\rt (\sigma i)) dip))))
  \land the (nhop (\rt (\sigma i)) dip) \neq dip
  \rightarrow \rt (\sigma i) \sqsubseteq dip \ rt (\sigma (the (nhop (\rt (\sigma i)) dip)))"

and ow: "?S i \sigma \sigma' a"

and "Suc 0 \leq osn"

and a6: "sip \neq oip \rightarrow oip \in kD (\rt (\sigma sip))"

\land osn \leq nsqn (\rt (\sigma sip)) oip

\land (nsqn (\rt (\sigma sip)) oip = osn
  \rightarrow the (dhops (\rt (\sigma sip)) oip) \leq hops
  \lor the (flag (\rt (\sigma sip)) oip) = inv"

and after: "\sigma' i = \sigma i [rt := update (\rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})]"

have "\forall dip. dip \in VD (update (\rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}))"

\land dip \in VD (\rt (\sigma' (the (nhop (update (\rt (\sigma i)) oip
  (osn, kno, val, Suc hops, sip, {})) dip))))

\land the (nhop (update (\rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip) \neq dip

\rightarrow update (\rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})

\sqsubseteq dip

\rt (\sigma' (the (nhop (update (\rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))"

239
proof clarify
  fix dip
  assume a2: "dip ∈ vD(update (rt (σ i)) oip (osn, kno, val, Suc (hops), sip, {}))"
  and a3: "dip ∈ vD(rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, {})) dip))))"
  and a4: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"

from ow have a5: "∀ j. j ≠ i ⟹ σ j = σ' j" by auto
show "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})
  ⪯ dip rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))" (is "?rt1 ⪯ dip ?rt2 dip")
proof (cases "?rt1 = rt (σ i)"
  assume nochange [simp]:
  "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {}) = rt (σ i)"
from after have "σ' i = σ i" by simp
with a5 have "∀ j. σ j = σ' j" by metis
from a2 have "dip ∈ vD (rt (σ i))" by simp
moreover from a3 have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))"
  using nochange and "∀ j. σ j = σ' j" by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) ⪯ dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
hence "rt (σ i) ⪯ dip rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))"
  using ⟨"∀ j. σ j = σ' j"⟩ by simp
thus "?thesis" by simp
next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip ∈ kD(rt (σ' i))" by auto
show ?thesis
proof (cases "dip = oip"
  assume "dip ≠ oip"
  with a2 have "dip ∈ vD (rt (σ i))" by auto
  moreover with a3 a5 after and ⟨dip ≠ oip⟩
    have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))" by simp metis
  moreover from a4 and ⟨dip ≠ oip⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
  ultimately have "rt (σ i) ⪯ dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
  with after and a5 and ⟨dip ≠ oip⟩ show ?thesis
    by simp (metis rt_strictly_fresher_update_other rt_strictly_fresher_irefl)
next
assume "dip = oip"
with a4 and change have "sip ≠ oip" by simp
with a6 have "oip ∈ kD(rt (σ sip))" and "osn ≤ nsqn (rt (σ sip)) oip" by auto
from a3 change ⟨dip = oip⟩ have "oip ∈ vD(rt (σ' sip))" by simp
hence "the (flag (rt (σ' sip)) oip) = val" by simp
from ⟨oip ∈ kD(rt (σ sip))⟩ have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
  ∧ the (dhops (rt (σ' sip)) oip) ≤ hops)" by simp
proof
  assume "oip ∈ vD(rt (σ sip))" hence "the (flag (rt (σ sip)) oip) = val" by simp
  with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn ⟹ the (dhops (rt (σ sip)) oip) ≤ hops"
by simp
show ?thesis
proof (cases "sip = i")
  assume "sip ≠ i"
  with a5 have "σ sip = σ' sip" by simp
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  and ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  show ?thesis by auto
next
  — alternative to using sip_not_ip
  assume [simp]: "sip = i"
  have "?rt1 = rt (σ i)"
  proof (rule update_cases_kD, simp)
    from ⟨Suc 0 ≤ osn⟩
    show "0 < osn" by simp
  next
    from ⟨oip ∈ vD (rt (σ sip))⟩
    and ⟨sip = i⟩
    show "oip ∈ kD (rt (σ i))" by simp
  next
    assume "sqn (rt (σ i)) oip < osn"
    also from ⟨Suc 0 ≤ osn⟩
    have "Suc 0 ≤ osn" by simp
    next
      from ⟨oip ∈ kD (rt (σ i))⟩
      and ⟨sip = i⟩
      show "oip ∈ kD (rt (σ i))" by simp
next
  assume "sqn (rt (σ i)) oip = osn"
  also from ⟨Suc 0 ≤ osn⟩
  have "Suc 0 ≤ osn" by simp
  next
    assume "the (flag (rt (σ i)) oip) = inv"
    with ⟨the (flag (rt (σ sip)) oip) = val⟩
    and a5
to have "sip = i" by (metis f.distinct(1) iD_flag_is_inv)
  from ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  and ⟨osn ≤ nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  unfolding update_def
  by (clarsimp split: option.split_asm if_split_asm)
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  have "osn < nsqn (rt (σ' sip)) oip"
by simp  
thus ?thesis ..  
qed  
thus ?thesis  
proof  
assume osnl't: "osn < nsqn (rt (σ' sip)) oip"  
from ⟨dip∈kD(rt (σ' i)) ⟩ and ⟨dip = oip⟩ have "dip ∈ kD (?rt1)" by simp  
moreover from a3 have "dip ∈ kD(?rt2 dip)" by simp  
morerover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"  
proof -  
  have "nsq?rt1 oip = osn" 
      by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])  
  also have "... < nsqn (rt (σ' sip)) oip" using osnl't .  
  also have "... = nsqn (?rt2 oip) oip" by (simp add: change)  
  finally show ?thesis 
      using ⟨dip = oip⟩ by simp  
qed  
ultimately show ?thesis  
by (rule rt_strictly_fresher_ltI)  
next  
assume osneq: "osn = nsqn (rt (σ' sip)) ∧ the (dhops (rt (σ' sip)) oip) ≤ hops"  
have "oip∈kD(?rt1)" by simp  
morerover from a3 ⟨dip = oip⟩ have "oip∈kD(?rt2 oip)" by simp  
morerover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"  
proof -  
  from osneq have "osn = nsqn (rt (σ' sip))" ..  
  also have "osn = nsqn ?rt1 oip" 
      by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])  
  also have "nsqn (rt (σ' sip)) oip = nsqn (?rt2 oip) oip" 
      by (simp add: change)  
  finally show ?thesis .  
qed  
morerover have "π5(the (?rt2 oip oip)) < π5(the (?rt1 oip))"  
proof -  
  from osneq have "the (dhops (rt (σ' sip)) oip) ≤ hops" ..  
morerover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip∈kD(rt (σ' sip))" by auto  
ultimately have "π5(the (rt (σ' sip) oip)) ≤ hops" 
      by (auto simp add: proj5_eq_dhops)  
also from change after have "hops < π5(the (rt (σ' i) oip))" 
      by (simp add: proj5_eq_dhops) (metis dhops_update_changed lessI)  
finally have "π5(the (rt (σ' sip) oip)) < π5(the (rt (σ' i) oip))" .  
with change after show ?thesis by simp  
qed  
ultimately have "?rt1 ⊏?rt2 oip" 
  by (rule rt_strictly_fresher_eqI)  
with ⟨dip = oip⟩ show ?thesis by simp  
qed  
qed  
qed  
} note rreq_rrep_update = this  

have "opaodv i ⊩ (otherwith ((=)) {i} (orecmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)), 
oth other quality_increases {i} →)
onl Γ_{AODV} 
(λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip)))) 
  ∧ (the (nhop (rt (σ i)) dip) ≠ dip 
  −→ rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip))))")  
proof (inv_cterms inv add: onl_oinvariant_sterms [OF aodv_wf rreq_sip [THEN weaken]]
\text{onl\_oinvariant\_sterms [OF aodv\_wf rrep\_sip [THEN weaken]]}
\text{onl\_oinvariant\_sterms [OF aodv\_wf rerr\_sip [THEN weaken]]}
\text{onl\_oinvariant\_sterms [OF aodv\_wf oosn\_rreq [THEN weaken]]}
\text{onl\_oinvariant\_sterms [OF aodv\_wf odsn\_rrep [THEN weaken]]}
\text{onl\_oinvariant\_sterms [OF aodv\_wf oadpreRT\_welldefined]}

solve: basic update\_0\_unk invalidate rreq\_rrep\_update
simp add: seqlsimp)

fix \(\sigma, \sigma'\) \(p, l\)
assume \(or: (\sigma, p) \in \text{oreachable (opaodv}\ i) (\?S i) (\text{other quality\_increases \{i\})}\)
and \(ll: l \in \text{labels} \Gamma_{AODV} AODV p\)
and \(pre: \forall dip. dip \in vD (rt (\sigma i)) \wedge dip \neq dip\)
\(\rightarrow\) \(\text{dip} \in kD (rt (\sigma nhip))\)
\(\wedge\) \(\text{sqn} (rt (\sigma i)) dip = \text{sqn} (rt (\sigma nhip)) dip\)

from this(1-2)
have \(or': (\sigma', p) \in \text{oreachable (opaodv}\ i) (\?S i) (\text{other quality\_increases \{i\})}\)
by - (rule oreachable\_other')

from or and ll have next\_hop: \(\forall dip. \text{let nhip} = \text{the (nhop (rt (\sigma i)) dip)}\)
\(\rightarrow\) \(\text{dip} \in kD (rt (\sigma i)) \wedge dip \neq dip\)
\(\rightarrow\) \(\text{dip} \in kD (rt (\sigma nhip))\)
\(\wedge\) \(\text{sqn} (rt (\sigma i)) dip \leq \text{sqn} (rt (\sigma nhip)) dip\)

by (auto dest!: onl\_oinvariant\_weakenD [OF seq\_compare\_next\_hop'])

from or and ll have unk\_hops\_one: \(\forall dip \in kD (rt (\sigma i)). \text{sqn} (rt (\sigma i)) dip = 0\)
\(\rightarrow\) \(\text{sqnf (rt (\sigma i)) dip = unk}\)
\(\wedge\) \(\text{the (dhops (rt (\sigma i)) dip)} = 1\)
\(\wedge\) \(\text{the (nhop (rt (\sigma i)) dip)} = dip\)
by (auto dest!: onl\_oinvariant\_weakenD [OF ozero\_seq\_unk\_hops\_one [OF oaodv\_trans aodv\_trans] otherwith\_actionD simp: seqlsimp])

from \(\text{other quality\_increases \{i\}} \sigma \sigma'\)
have "\(\sigma', i = \sigma i\)" by auto

with \(\text{other quality\_increases \{i\}} \sigma \sigma'\)
have "\(\forall j. \text{quality\_increases (} \sigma j\) (\sigma' j)\)"
by - (erule otherE, metis singleton\_iff)

show "\(\forall dip. dip \in vD (rt (\sigma i))\)
\(\wedge\) \(\text{dip} \in vD (rt (\sigma' i))\)
\(\wedge\) \(\text{the (nhop (rt (\sigma i)) dip)} \neq dip\)
\(\rightarrow\) \(\text{dip} \in kD (rt (\sigma i))\)
compute proof clarify
fix dip
assume "\(\text{dip} \in vD (rt (\sigma i))\)"
and "\(\text{dip} \in vD (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip))})\)"
and "\(\text{the (nhop (rt (\sigma i)) dip)} \neq dip\)"
from this(1) and \(\sigma' i = \sigma i\) have "\(\text{dip} \in vD (rt (\sigma i))\)"
and "\(\text{dip} \in kD (rt (\sigma i))\)"
by auto

from \(\text{the (nhop (rt (\sigma i)) dip)} \neq dip\) and \(\sigma' i = \sigma i\)
have "\(\text{the (nhop (rt (\sigma i)) dip)} \neq dip\)" (is "\(?nhip \neq _\)"") by simp

with \(\text{dip} \in kD (rt (\sigma i))\) and next\_hop
have "\(\text{dip} \in kD (rt (\sigma (\?nhip)))\)"
and nsqns: "\(\text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma ?nhip)) dip}\)"
by (auto simp: Let_def)

have "\(0 < \text{sqn (rt (\sigma i)) dip}\)"
compute proof (rule neq0\_conv [THEN iffD1, OF notI])
assume "\(\text{sqn (rt (\sigma i)) dip} = 0\)"
with \(\text{dip} \in kD (rt (\sigma i))\) and unk\_hops\_one
have "\(?nhip = dip\)" by simp
with (?nhip \neq dip) show False ..

qed
also have ". . . = nsqn (rt (\sigma i) dip)
by (rule vD_nsqn [OF \langle dip \in vD (rt (\sigma i)) \rangle], THEN sym]
also have ". . . \leq nsqn (rt (\sigma ?nhip) dip)
by (rule nsqns)
also have ". . . \leq sqn (rt (\sigma ?nhip) dip)
by (rule nsqn_sqn)
finally have "0 < sqn (rt (\sigma ?nhip)) dip" .

have "rt (\sigma i) \sqsubseteq dip rt (\sigma' ?nhip)"
proof (cases "dip \in vD(rt (\sigma ?nhip))")
assume "dip \in vD(rt (\sigma ?nhip))"
with pre \langle dip \in vD(rt (\sigma i)) \rangle and ?nhip \neq dip
have "rt (\sigma i) \sqsubseteq dip rt (\sigma ?nhip)" by auto
moreover from \forall j. quality_increases (\sigma j) (\sigma' j)
  have "quality_increases (\sigma ?nhip) (\sigma' ?nhip)" ..
ultimately show ?thesis
  using \langle dip \in kD(rt (\sigma ?nhip)) \rangle by (rule strictly_fresher_quality_increases_right)
next
assumet "dip \in kD(rt (\sigma ?nhip))"
with \langle dip \in kD(rt (\sigma ?nhip)) \rangle have "dip \in D(rt (\sigma ?nhip))" ..
hence "the (flag (rt (\sigma ?nhip)) dip) = inv"
  by auto
moreover from \langle \forall j. quality_increases (\sigma j) (\sigma' j) \rangle
  have "sqn (rt (\sigma ?nhip)) dip \leq sqn (rt (\sigma' ?nhip)) dip" ..
  with \langle 0 < sqn (rt (\sigma ?nhip)) dip \rangle show ?thesis by auto
qed
also have "... = nsqn (rt (\sigma ?nhip) dip)"
proof (rule vD_nsqn_sqn [THEN sym])
  from \langle dip \in vD(rt (\sigma' (the (nhop (rt (\sigma' i) dip)))) \rangle and \langle \sigma' i = \sigma i \rangle
  show "dip \in vD(rt (\sigma' ?nhip))" by simp
qed
finally have "nsqn (rt (\sigma i) dip) < nsqn (rt (\sigma' ?nhip) dip)" .

moreover from \langle dip \in vD(rt (\sigma' (the (nhop (rt (\sigma' i) dip)))) \rangle and \langle \sigma' i = \sigma i \rangle
have "dip \in kD(rt (\sigma' ?nhip))" by auto
ultimately show "rt (\sigma i) \sqsubseteq dip rt (\sigma' ?nhip)"
  using \langle dip \in kD(rt (\sigma i)) \rangle by - (rule rt_strictly_fresher_ltI)
  qed
with \langle \sigma' i = \sigma i \rangle show "rt (\sigma' i) \sqsubseteq dip rt (\sigma' (the (nhop (rt (\sigma' i) dip))))" by simp
  qed
  qed
thus ?thesis unfolding Let_def .
  qed

lemma seq_compare_next_hop:
  fixes w
shows "opaodv i \models (otherwith ((\equiv)) \langle i \rangle (orecmsg msg_fresh),
  other quality_increases \langle i \rangle \rightarrow)
  global (\lambda \sigma. \forall dip. let nhip = the (nhop (rt (\sigma i) dip))
    in dip \in D(rt (\sigma i)) \land nhip \neq dip \rightarrow
    dip \in D(rt (\sigma nhip))
    \land nsqn (rt (\sigma i) dip) \leq nsqn (rt (\sigma nhip) dip)"
by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onlD)

lemma seq_nhop_quality_increases:
shows "opaodv i | = (otherwith ((=)) {i} (orecmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)), other quality_increases {i} ⇒)
global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip) in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip → (rt (σ i)) ⊏ dip (rt (σ nhip)))"
by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onlD)

end

2.10 Routing graphs and loop freedom

theory B_Loop_Freedom
imports B_Aodv_Predicates B_Fresher
begin

Define the central theorem that relates an invariant over network states to the absence of loops in the associate routing graph.

definition
  rt_graph :: "(ip ⇒ state) ⇒ ip ⇒ ip rel"
where
  "rt_graph σ = (λdip. {(ip, ip') | ip ip' dsn dsk hops pre.
    ip ≠ dip ∧ rt (σ ip) dip = Some (dsn, dsk, val, hops, ip', pre))"

Given the state of a network σ, a routing graph for a given destination ip address dip abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

lemma rt_graphE [elim]:
  fixes n dip ip ip'
  assumes "(ip, ip') ∈ rt_graph σ dip"
  shows "ip ≠ dip ∧ (∃r. rt (σ ip) = r ∧ (∃dsn dsk hops pre. r dip = Some (dsn, dsk, val, hops, ip', pre)))"
  using asms unfolding rt_graph_def by auto

lemma rt_graph_vD [dest]:
  "∀ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ dip ∈ vD(rt (σ ip))"
  unfolding rt_graph_def vD_def by auto

lemma rt_graph_vD_trans [dest]:
  "∀ip ip' σ dip. (ip, ip') ∈ (rt_graph σ dip)⁺ ⇒ dip ∈ vD(rt (σ ip))"
  by (erule converse_tranclE) auto

lemma rt_graph_not_dip [dest]:
  "∀ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ ip ≠ dip"
  unfolding rt_graph_def by auto

lemma rt_graph_not_dip_trans [dest]:
  "∀ip ip' σ dip. (ip, ip') ∈ (rt_graph σ dip)⁺ ⇒ ip ≠ dip"
  by (erule converse_tranclE) auto

NB: the property below cannot be lifted to the transitive closure

lemma rt_graph_nhip_is_nhop [dest]:
  "∀ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ ip' = the (nhop (rt (σ ip)) dip)"
  unfolding rt_graph_def by auto

theorem inv_to_loop_freedom:
  assumes "∀i dip. let nhip = the (nhop (rt (σ i)) dip) in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip → (rt (σ i)) ⊏ dip (rt (σ nhip))"
  shows "∀dip. irrefl ((rt_graph σ dip)⁺)"

end
using assms proof (intro allI)
fix σ :: "ip ⇒ state" and dip
assume inv: "∀ip dip. let nhip = the (nhop (rt (σ ip)) dip)
in dip ∈ vD(rt (σ ip)) \cap vD(rt (σ nhip)) ∧
nhip ≠ dip → rt (σ ip) ⊑ dip rt (σ nhip)"

{ fix ip ip'
assume "(ip, ip') ∈ (rt_graph σ dip)++" and "dip ∈ vD(rt (σ ip))"
and "ip' ≠ dip"
hence "rt (σ ip) ⊑ dip rt (σ ip')"
proof induction
fix nhip
assume "(ip, nhip) ∈ rt_graph σ dip" and "dip ∈ vD(rt (σ nhip))" and "nhip ≠ dip"
from ⟨(ip, nhip) ∈ rt_graph σ dip⟩ have "dip ∈ vD(rt (σ ip))" and "nhip = the (nhop (rt (σ ip)) dip)" by auto
from ⟨dip ∈ vD(rt (σ ip))⟩ and ⟨dip ∈ vD(rt (σ nhip))⟩ have "dip ∈ vD(rt (σ ip)) \cap vD(rt (σ nhip))" ..
with ⟨nhip = the (nhop (rt (σ ip)) dip)⟩ and inv
show "rt (σ ip) ⊑ dip rt (σ nhip)"
by (clarsimp simp: Let_def)
next
fix nhip nhip'
assume "(ip, nhip) ∈ (rt_graph σ dip)++" and "(nhip, nhip') ∈ rt_graph σ dip" and IH: "[\[ dip ∈ vD(rt (σ nhip)); nhip ≠ dip \]] ⇒ rt (σ ip) ⊑ dip rt (σ nhip)"
and "dip ∈ vD(rt (σ nhip'))" and "nhip' ≠ dip"
from ⟨(nhip, nhip') ∈ rt_graph σ dip⟩ have 1: "dip ∈ vD(rt (σ nhip))" and 2: "nhip ≠ dip" and "nhip' = the (nhop (rt (σ nhip)) dip)" by auto
from 1 2 have "rt (σ nhip) ⊑ dip rt (σ nhip')" by (rule IH)
also have "rt (σ nhip) ⊑ dip rt (σ nhip')" proof -
from ⟨dip ∈ vD(rt (σ nhip))⟩ and ⟨dip ∈ vD(rt (σ nhip'))⟩ have "dip ∈ vD(rt (σ nhip)) \cap vD(rt (σ nhip'))" ..
with ⟨nhip' ≠ dip⟩ and ⟨nhip' = the (nhop (rt (σ nhip)) dip)⟩ and inv
show "rt (σ nhip) ⊑ dip rt (σ nhip')"
by (clarsimp simp: Let_def)
qed
finally show "rt (σ ip) ⊑ dip rt (σ nhip')".
qed } note fresher = this

show "irrefl ((rt_graph σ dip)++)"
unfolding irrefl_def proof (intro allI notI)
fix ip
assume "(ip, ip) ∈ (rt_graph σ dip)++" moreover then have "dip ∈ vD(rt (σ ip))" and "ip ≠ dip"
by auto
ultimately have "rt (σ ip) ⊑ dip rt (σ ip)" by (rule fresher)
thus False by simp
qed
qed

end
2.11 Lift and transfer invariants to show loop freedom

theory B_Aodv_Loop_Freedom
imports AWN.OClosed_Transfer AWN.Qmsg_Lifting B_Global_Invariants B_Loop_Freedom
begin

2.11.1 Lift to parallel processes with queues

lemma par_step_no_change_on_send_or_receive:
fixes σ s a σ' s'
assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos Γ$_{AODV}$ i) (seqp_sos Γ$_{QMSG}$)"
and "a ≠ τ"
shows "σ' i = σ i"
using assms by (rule qmsg_no_change_on_send_or_receive)

lemma par_nhop_quality_increases:
shows "opaodv i ⟨⟨ i qmsg |= (otherwith ((=)) {i}) (orecvmsg (λσ. msg_fresh σ m ∧ msg_zhops m)),
other quality_increases {i} →)\n\nglobal (λσ. ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ νD (rt (σ i)) ∩ νD (rt (σ nhip)) ∧ nhip ≠ dip
→ (rt (σ i)) ⊏ dip (rt (σ nhip)))"
proof (rule lift_into_qmsg [OF seq_nhop_quality_increases])
show "opaodv i \langle\langle i qmsg |\!=\! A (otherwith ((=)) {i}) (orecvmsg (λ_. rreq_rrep_sn)) σ σ',
other (λ_ _. True) →)\n\nglobala (λ(σ, _). quality_increases (σ i) (σ' i))"
proof (rule ostep_invariant_weakenE [OF olocal_quality_increases], simp_all)
fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
assume "onll Γ$_{AODV}$ (λ((σ, _), _, (σ', _)). quality_increases (σ j) (σ' j)) t"
thus "quality_increases (fst (fst t) i) (fst (snd (snd t)) i)"
by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)
next
fix σ σ' a
assume "otherwith ((=)) {i} (orecvmsg (λ_. rreq_rrep_sn)) σ σ', a"
thus "otherwith quality_increases (i) (orecvmsg (λ_. rreq_rrep_sn)) σ σ', a"
by - (erule weaken_otherwith, auto)
qed

lemma par_rreq_rrep_sn_quality_increases:
"opaodv i ⟨⟨ i qmsg |= A (otherwith ((=)) {i}) (orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)\n\nglobala (λ(σ, _). quality_increases (σ i) (σ' i))"
proof -
have "opaodv i |= A (λσ _. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)\n\nglobala (λ(σ, _). quality_increases (σ i) (σ' i))"
by (rule ostep_invariant_weakenE [OF olocal_quality_increases])
(auto dest!: onll1D seq11D elim!: aodv_ex_labelE)
thesis by rule auto
qed

lemma par_rreq_rrep_nsqn_fresh_any_step:
shows "opaodv i ⟨⟨ i qmsg |= A (λσ _. orecvmsg (λ_. rreq_rrep_sn)) σ,\nother (λ_ _. True) {i} →)\n\nglobala (λ(σ, a, σ'). anycast (msg_fresh σ a))"
proof -
have "opaodv i |= A (λσ _. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)\n\nglobala (λ(σ, a, σ'). anycast (msg_fresh σ a))"
proof (rule ostep_invariant_weakenE [OF oreq_rreq_rrep_nsqn_fresh_any_step_invariant])
fix t
by auto
qed

next

...
lemma \(\text{par\_anycast\_msg\_zhops}:)\) shows \(\text{par\_anycast\_msg\_zhops}:)\) thus \(\text{par\_anycast\_msg\_zhops}:)\) qed auto

by (rule lift\_step\_into\_qmsg\_statelessassm) simp_all

thus \(\text{thesis}\) by rule auto

2.11.2 Lift to nodes

lemma \(\text{node\_nhop\_quality\_increases}:)\) shows \(\text{node\_nhop\_quality\_increases}:)\) thus \(\text{node\_nhop\_quality\_increases}:)\) by (rule node\_lift\_anycast\_statelessassm) simp_all

thus \(\text{thesis}\) by rule auto

2.11.2 Lift to nodes

lemma \(\text{node\_step\_no\_change\_on\_send\_or\_receive}:)\) assumes \(\text{node\_step\_no\_change\_on\_send\_or\_receive}:)\) and \(\text{node\_step\_no\_change\_on\_send\_or\_receive}:)\) by (cases a) (auto elim!: par\_step\_no\_change\_on\_send\_or\_receive)

lemma \(\text{node\_quality\_increases}:)\) shows \(\text{node\_quality\_increases}:)\) by (rule node\_lift\_anycast\_statelessassm\_fresh\_any\_step) simp

thus \(\text{thesis}\) by rule auto

248
lemma node_anycast_msg_zhops:
  shows "(i : opaodv i ((qmsg : R_1)_o) |=A
            (λσ _, oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
            globala (λ(σ _, a, _). castmsg msg_zhops a)"
by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

lemma node_silent_change_only:
  shows "(i : opaodv i ((qmsg : R_1)_o) |=A
            (λσ _, oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
            globala (λ(λ(σ, a, _). a) → σ _ → σ' i = σ i)"
proof (rule ostep_invariantI, simp (no_asm), rule impl)
fix σ ζ a σ' ζ'
assume or: "(σ, ζ) ∈ orreachable ((i : opaodv i ((qmsg : R_1)_o))
            (λσ _, oarrivemsg (λ_. rreq_rrep_sn) σ)
            other (λ_ _. True) {i})"
and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans ((i : opaodv i ((qmsg : R_1)_o))
            and "a ≠ τ n"
from or obtain p R where "ζ = NodeS i p R"
by - (drule node_net_state, metis)
with tr have "((σ, i, NodeS i p R), a, (σ', ζ'))
            ∈ onode_sos (oparp_sos i (trans (opaodv i))) (trans qmsg))" 
  by simp 
thus "σ' i = σ i" using (a ≠ τ n)
  by (cases rule: onode_sos_cases)
  (auto elim: qmsg_no_change_on_send_or_receive)
qed

2.11.3 Lift to partial networks

lemma arrive_rreq_rrep_nsqn_fresh_inc_sn [simp]:
  assumes "oarrivemsg (λσ m. msg_fresh σ m ∧ P σ m) σ m"
  shows "oarrivemsg (λ_. rreq_rrep_sn) σ m"
using asms by (cases (m) auto)

lemma opnet_nhop_quality_increases:
  shows "opnet (λi. opaodv i ((qmsg : p))) p =
        (otherwith ((=)) (net_tree_ips p))
        (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
        other_quality_increases (net_tree_ips p) →)
        globala (λσ. ∀i∈net_tree_ips p. ∀dip.
            let nhip = the (nhop (rt (σ i)) dip).
            in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
            → (rt (σ i)) ⊏ dip (rt (σ nhip)))"
proof (rule pnet_lift [OF node_nhop_quality_increases])
fix i R
have "i : opaodv i ((qmsg : R_1)_o) |=A
        (λσ _, oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
        globala (λ(σ, a, _). castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
proof (rule ostep_invariantI, simp (no_asm))
fix σ s a σ' s'
assume or: "(σ, s) ∈ orreachable ((i : opaodv i ((qmsg : R_1)_o))
            (λσ _, oarrivemsg (λ_. rreq_rrep_sn) σ)
            other (λ_ _. True) {i})"
and tr: "((σ, s), a, (σ', s')) ∈ trans ((i : opaodv i ((qmsg : R_1)_o))
            and am: "oarrivemsg (λ_. rreq_rrep_sn) σ a"
from or tr am have "castmsg (msg_fresh σ) a"
  by (auto dest!: ostep_invariantD [OF node_rreq_rrep_nsqn_fresh_any_step])
moreover from or tr am have "castmsg (msg_zhops) a"
  by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
ultimately show "castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a"
  by (case_tac a) auto
qed
thus "i : opaodv i ((qmsg : R_1)_o) |=A
        (λσ _, oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
  by (rule pnet_lift [OF node_nhop_quality_increases])
  (auto dest!: ostep_invariantD [OF node_rreq_rrep_nsqn_fresh_any_step])
moreover from or tr am have "castmsg (msg_zhops) a"
  by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
ultimately show "castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a"
  by (case_tac a) auto
 qed
other quality_increases \{i\} \rightarrow \text{globa} (\lambda (\sigma, a, \_).
\text{castmsg} (\lambda m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) a)"

by rule auto

next

fix \( i \ \text{R} \)

show "(i : \text{opaodv \langle\langle i \ qmsg : R\rangle\rangle}_{o}) \models A
(\lambda \sigma \_\_. \text{oarrivemsg} (\lambda \sigma m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) \sigma,
other quality_increases \{i\} \rightarrow \text{globa} (\lambda (\sigma, a, \_).
a \neq \tau \land (\forall d. a \neq i: \text{deliver}(d) \rightarrow \sigma i = \sigma' i))"

by (rule ostep\_invariant\_weakenE [OF node\_silent\_change\_only]) auto

next

fix \( i \ \text{R} \)

show "(i : \text{opaodv \langle\langle i \ qmsg : R\rangle\rangle}_{o}) \models A
(\lambda \sigma \_\_. \text{oarrivemsg} (\lambda \sigma m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) \sigma,
other quality_increases \{i\} \rightarrow \text{globa} (\lambda (\sigma, a, \_).
a = \tau \lor (\exists d. a = i: \text{deliver}(d) \rightarrow \text{quality\_increases} (\sigma i) (\sigma' i)))"

by (rule ostep\_invariant\_weakenE [OF node\_quality\_increases]) auto

qed simp_all

2.11.4 Lift to closed networks

lemma onet\_nhop\_quality\_increases:
shows \( \text{oclosed} (\text{opnet} (\lambda i. \text{opaodv \langle\langle i \ qmsg \rangle\rangle}_{p})) \models (\text{otherwith (\_\_\_)} \text{net\_tree\_ips} p \text{ inoclosed, ?U \_\_\_}) \text{?inv}"

proof (rule inclosed\_closed)

from opnet\_nhop\_quality\_increases

show "\text{opnet} (\lambda i. \text{opaodv \langle\langle i \ qmsg \rangle\rangle}_{p}) \models (\text{otherwith ((\_\_\_)} \text{net\_tree\_ips} p \text{ inoclosed, ?U \_\_\_}) \text{?inv}"

proof (rule oinvariant\_weakenE)

fix \( \sigma \ \sigma' :: \text{ip} \Rightarrow \text{state} \) and \( a :: \text{msg node\_action} \)

assume "otherwith ((\_\_\_)} \text{net\_tree\_ips} p \text{ inoclosed} \ \sigma \ \sigma' \ a"

thus "otherwith ((\_\_\_)} \text{net\_tree\_ips} p)
\text{oarrivemsg} (\lambda \sigma m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) \ \sigma \ a"

proof (cases a)

fix ii ni ms

assume "a = ii\_ni: \text{arrive}(ms)"

moreover with \( \langle \text{inoclosed} \ \sigma \ a \rangle \) obtain d di where "ms = \text{newpkt}(d, di)"

by (cases ms) auto

ultimately show "?thesis" by simp

qed simp_all

qed

qed

2.11.5 Transfer into the standard model

interpretation aodv\_openproc: \text{openproc paodv opaodv id}
rewrites "aodv\_openproc.\text{initmissing = initmissing}"

proof -

show "\text{openproc paodv opaodv id}"

proof unfold\_locales

fix i :: ip

have "\langle(\sigma, \zeta). (\sigma i, \zeta) \in \sigma AODV i \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst} ' \sigma AODV j)\rangle \subseteq \sigma AODV'"

unfolding \( \sigma AODV\_\text{def} \sigma AODV'\_\text{def} 

proof (rule equalityD1)
show "\( \forall p. \{ (\sigma, \zeta). (\sigma i, \zeta) \in \{(f i, p)\} \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst} \ ' \{(f j, p)\}) \} = \{(f, p)\} \)"
by (rule set_eqI) auto

qed

thus "\{(\sigma, \zeta) \mid \sigma \zeta s. s \in \text{init} (\text{paodv} i) \land (\sigma i, \zeta) = \text{id} s \land (\forall j. j \neq i \rightarrow \sigma j \in (\text{fst o id} \ ' \text{init} (\text{paodv} j))) \subseteq \text{init} (\text{opaodv} i)\)"
by simp

next

show "\( \forall j. \text{init} (\text{paodv} j) \neq \{\} \)"

unfolding \( \sigma_{\text{AODV}}\_\text{def} \) by simp

next

fix i s a s' \( \sigma \sigma' \)
assume "\(\sigma i = \text{fst} (\text{id} s)\)"
and "\(\sigma' i = \text{fst} (\text{id} s')\)"
and "\((s, a, s') \in \text{trans} (\text{paodv} i)\)"

then obtain q q' where "\(s = (\sigma i, q)\)"
and "\(s' = (\sigma' i, q')\)"
and "\(((\sigma i, q), a, (\sigma' i, q')) \in \text{trans} (\text{paodv} i)\)"

by (cases s, cases s') auto

from this(3)

have "\(\langle \sigma, \text{snd} (\text{id} s) \rangle, a, (\sigma', \text{snd} (\text{id} s')) \rangle \in \text{trans} (\text{opaodv} i)\)"

by simp

qed

then interpret opn: openproc \( \text{paodv} \ \text{opaodv} \ \text{id} \).

have (simp): "\(\forall i. (\text{SOME} x. x \in (\text{fst o id} \ ' \text{init} (\text{paodv} i))) = \text{aodv_init} i\)"

unfolding \( \sigma_{\text{AODV}}\_\text{def} \) by simp

hence "\(\forall i. \text{openproc.initmissing} (\lambda i. \text{paodv} i \langle\langle qmsg) (\lambda(p, q). \text{fst} (\text{id} p), \text{snd} (\text{id} p), q)) \sigma = \text{initmissing} \ \sigma\)"

unfolding opn.initmissing_def opn.someinit_def initmissing_def

unfolding \( \sigma_{\text{AODV}}\_\text{def} \ \sigma_{\text{QMSG}}\_\text{def} \) by (clarsimp cong: option.case_cong)

thus "\(\text{openproc.initmissing} (\lambda i. \text{paodv} i \langle\langle qmsg) (\lambda(p, q). \text{fst} (\text{id} p), \text{snd} (\text{id} p), q)) = \text{initmissing} \ \sigma\)"

by (rule ext)

have im: "\(\forall \sigma. \text{openproc.netglobal} (\lambda i. \text{paodv} i \langle\langle qmsg) (\lambda(p, q). \text{fst} (\text{id} p), \text{snd} (\text{id} p), q)) \ \sigma \ \text{P} \ \sigma = \text{netglobal} \ \sigma\)"

unfolding opn.netglobal_def netglobal_def opn.someinit_def initmissing_def

unfolding \( \sigma_{\text{AODV}}\_\text{def} \ \sigma_{\text{QMSG}}\_\text{def} \) by (clarsimp cong: option.case_cong)

thus "\(\text{openproc.netglobal} (\lambda i. \text{paodv} i \langle\langle qmsg) (\lambda(p, q). \text{fst} (\text{id} p), \text{snd} (\text{id} p), q)) = \text{netglobal} \ \sigma\)"

by (clarsimp cong: option.case_cong)

thus "\(\forall \sigma. \text{openproc.netglobal} (\lambda i. \text{paodv} i \langle\langle qmsg) (\lambda(p, q). \text{fst} (\text{id} p), \text{snd} (\text{id} p), q)) = \text{netglobal} \ \sigma\)"

by auto

qed

lemma net_nhops_quality_increases:

assumes "\(\text{wf}_\text{net_tree} \ n\)"
shows \( \text{"closed } (\text{pnet } (\forall i. \text{paodv } i \langle\langle \text{qmsg} \rangle n) \| \| = \text{netglobal } (\lambda \sigma. \forall i \in \text{net_tree_ips } n. \forall \text{dip. irrefl } (\text{rt_graph } \sigma \text{dip}) \| \| ) \)" } \\
(\text{is \"} _\| \| = \text{netglobal } (\lambda \sigma. \forall i. ?\text{inv } \sigma i\)\"") \\
proof - \\
\text{from } (\text{wf_net_tree } n) \\
\text{have proto: \"closed } (\text{pnet } (\forall i. \text{paodv } i \langle\langle \text{qmsg} \rangle n) \| \| = \text{netglobal } (\lambda \sigma. \forall i \in \text{net_tree_ips } n. \forall \text{dip. irrefl } (\text{rt_graph } \sigma \text{dip}) \| \| ) \)" \\
by (\text{rule aodv_openproc_par_qmsg.close_opnet } [\text{OF } _\| \| = \text{netglobal } (\lambda \sigma. \forall i. ?\text{inv } \sigma i\)\""]) \\
show ?thesis \\
unfolding \text{invariant_def opnet_sos.opnet_tau1} \\
proof (\text{rule, simp only: aodv_openproc_par_qmsg.netglobalsimp} \\
\text{fst_initmissing_netgmap_pair_fst, rule allI}) \\
fix \sigma i \\
assume sr: \"\sigma \in \text{reachable } (\text{closed } (\text{pnet } (\forall i. \text{paodv } i \langle\langle \text{qmsg} \rangle n) ) \| \| \| ) \| \| ) \) TT" \\
hence \"\forall i \in \text{net_tree_ips } n. ?\text{inv } (\text{fst } (\text{initmissing } \text{netgmap } \text{fst } \sigma)) i\" \\
by - (\text{drule invariantD } [\text{OF proto}], \\
\text{simp only: aodv_openproc_par_qmsg.netglobalsimp} \\
\text{fst_initmissing_netgmap_pair_fst}) \\
thus \"?\text{inv } (\text{fst } (\text{initmissing } \text{netgmap } \text{fst } \sigma)) i\"
proof (\text{cases } \"i \in \text{net_tree_ips } n\") \\
assume "i \in \text{net_tree_ips } n" \\
from sr have \"\sigma \in \text{reachable } (\text{pnet } (\forall i. \text{paodv } i \langle\langle \text{qmsg} \rangle n) ) \| \| \| ) \) TT" .. \\
hence \"\text{net_ips } \sigma = \text{net_tree_ips } n\" .. \\
with \"i \in \text{net_tree_ips } n\" have \"i \in \text{net_ips } \sigma\" by \text{simp} \\
hence \"(\text{fst } (\text{initmissing } \text{netgmap } \text{fst } \sigma)) i = \text{aodv_init } i\" \\
by \text{simp} \\
thus ?thesis by \text{simp} \\
qed metis \\
qed

2.11.6 Loop freedom of AODV

\text{theorem aodv_loop_freedom:} \\
\text{assumes } \"\text{wf_net_tree } n\" \\
\text{shows } \"\text{closed } (\text{pnet } (\forall i. \text{paodv } i \langle\langle \text{qmsg} \rangle n) ) \| \| = \text{netglobal } (\lambda \sigma. \forall \text{dip. irrefl } (\text{rt_graph } \sigma \text{dip}) \| \| )\"
\text{using assms by } (\text{rule aodv_openproc_par_qmsg.netglobal_weakenE} \\
[\text{OF } \text{net_nhop_quality_increases } \text{inv_to_loop_freedom}]) \\
end
Chapter 3

Variant C: From Groupcast to Broadcast

Explanation [4, §10.4]: A node maintains a set of ‘precursor nodes’ for each of its valid routes. If the link to a route’s next hop is lost, an error message is groupcast to the associated precursor nodes. The idea is to reduce the number of messages received and handled. However, precursor lists are incomplete. They are updated only when a RREP message is sent. This can lead to packet loss. A possible solution is to abandon precursors and to replace every groupcast by a broadcast. At first glance this strategy seems to need more bandwidth, but this is not the case. Sending error messages to a set of precursors is implemented at the link layer by broadcasting the message anyway; a node receiving such a message then checks the header to determine whether it is one of the intended recipients. Instead of analysing the header only, a node can just as well read the message and decide whether the information contained in the message is of use. To be more precise: an error message is useful for a node if the node has established a route to one of the nodes listed in the message, and the next hop to a listed node is the sender of the error message. In case a node finds useful information inside the message, it should update its routing table and distribute another error message.

3.1 Predicates and functions used in the AODV model

theory C_Aodv_Data
imports C_Gtobcast
begin

3.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

definition inc :: "sqn ⇒ sqn"
  where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ inc x"
  unfolding inc_def by simp

lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp

lemma inc_never_one' [simp, intro]: "inc x ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
  by simp

3.1.2 Modelling Routes

A route is a 5-tuple, \( (dsn, dsk, flag, hops, nhip) \) where \( dsn \) is the ‘destination sequence number’, \( dsk \) is the ‘destination-sequence-number status’, \( flag \) is the route status, \( hops \) is the number of hops to the destination, and \( nhip \) is the next hop toward the destination. In this variant, the set of ‘precursor nodes’ is not modelled.

type_synonym r = "sqn × k × f × nat × ip"
definition proj2 :: "r ⇒ sqn" ("π_2")
  where "π_2 ≡ λ(dsn, _, _, _, _). dsn"

definition proj3 :: "r ⇒ k" ("π_3")
  where "π_3 ≡ λ(_, dsk, _, _, _). dsk"

definition proj4 :: "r ⇒ f" ("π_4")
  where "π_4 ≡ λ(_, _, flag, _, _). flag"

definition proj5 :: "r ⇒ nat" ("π_5")
  where "π_5 ≡ λ(_, _, _, hops, _). hops"

definition proj6 :: "r ⇒ ip" ("π_6")
  where "π_6 ≡ λ(_, _, _, _, nhip). nhip"

lemma projs [simp]:
  "π_2(dsn, dsk, flag, hops, nhip) = dsn"
  "π_3(dsn, dsk, flag, hops, nhip) = dsk"
  "π_4(dsn, dsk, flag, hops, nhip) = flag"
  "π_5(dsn, dsk, flag, hops, nhip) = hops"
  "π_6(dsn, dsk, flag, hops, nhip) = nhip"
  by (clarsimp simp: proj2_def proj3_def proj4_def proj5_def proj6_def)+

lemma proj3_pred [intro]: "[ P kno; P unk ] ⇒ P (π_3 x)"
  by (rule k.induct)

lemma proj4_pred [intro]: "[ P val; P inv ] ⇒ P (π_4 x)"
  by (rule f.induct)

lemma proj6_pair_snd [simp]:
  fixes dsn' r
  shows "π_6 (dsn', snd (r)) = π_6(r)"
  by (cases r) simp

3.1.3 Routing Tables
Routing tables map ip addresses to route entries.

type_synonym rt = "ip ⇒ r"

syntax
  "_Sigma_route" :: "rt ⇒ ip ⇒ r" ("σ_route(_, _')")
translations
  "σ_route(rt, dip)" ⇒ "rt dip"

definition sqn :: "rt ⇒ ip ⇒ sqn"
  where "sqn rt dip ≡ case σ_route(rt, dip) of Some r ⇒ π_2(r) | None ⇒ 0"

definition sqnf :: "rt ⇒ ip ⇒ k"
  where "sqnf rt dip ≡ case σ_route(rt, dip) of Some r ⇒ π_3(r) | None ⇒ unk"

abbreviation flag :: "rt ⇒ ip ⇒ f"
  where "flag rt dip ≡ map_option π_4 (σ_route(rt, dip))"

abbreviation dhops :: "rt ⇒ ip ⇒ nat"
  where "dhops rt dip ≡ map_option π_5 (σ_route(rt, dip))"

abbreviation nhop :: "rt ⇒ ip ⇒ ip"
  where "nhop rt dip ≡ map_option π_6 (σ_route(rt, dip))"

definition vD :: "rt ⇒ ip set"
  where "vD rt ≡ {dip. flag rt dip = Some val}"
definition \( iD : rt \Rightarrow ip \, set \)
  where \( "iD \, rt \equiv \{ dip. \, flag \, rt \, dip = Some \, inv \}" \)

definition \( kD : rt \Rightarrow ip \, set \)
  where \( "kD \, rt \equiv \{ dip. \, rt \, dip \neq None \}" \)

lemma \( kD_is_vD_and_iD : kD \, rt = vD \, rt \cup iD \, rt \)
  unfolding kD_def vD_def iD_def by auto

lemma \( vD_iD_gives_kD \, [simp] : \)
  \( \forall ip \, rt. \, ip \in vD \, rt \implies ip \in kD \, rt \)
  \( \forall ip \, rt. \, ip \in iD \, rt \implies ip \in kD \, rt \)
  unfolding kD_is_vD_and_iD by simp_all

lemma \( kD_Some \, [dest] : \)
  fixes dip rt
  assumes "dip \in kD \, rt"
  shows "\( \exists dsn \, dsk \, flag \, hops \, nhip. \)
         \( \sigma_{route}(rt, dip) = Some \, (dsn, \, dsk, \, flag, \, hops, \, nhip)\)"
  using assms unfolding kD_def by simp

lemma \( kD_None \, [dest] : \)
  fixes dip rt
  assumes "dip \notin kD \, rt"
  shows "\( \sigma_{route}(rt, dip) = None \)"
  using assms unfolding kD_def by (metis (mono_tags) mem_Collect_eq)

lemma \( vD_Some \, [dest] : \)
  fixes dip rt
  assumes "dip \in vD \, rt"
  shows "\( \exists dsn \, dsk \, hops \, nhip. \)
         \( \sigma_{route}(rt, dip) = Some \, (dsn, \, dsk, \, val, \, hops, \, nhip)\)"
  using assms unfolding vD_def by simp

lemma \( vD_empty \, [simp] : vD \, Map.empty = {} \)
  unfolding vD_def by simp

lemma \( iD_Some \, [dest] : \)
  fixes dip rt
  assumes "dip \in iD \, rt"
  shows "\( \exists dsn \, dsk \, flag \, hops \, nhip. \)
         \( \sigma_{route}(rt, dip) = Some \, (dsn, \, dsk, \, inv, \, hops, \, nhip)\)"
  using assms unfolding iD_def by simp

lemma \( val_is_vD \, [elim] : \)
  fixes ip rt
  assumes "ip \in kD(rt)"
  and "the (flag rt ip) = val"
  shows "ip \in vD(rt)"
  using assms unfolding vD_def by auto

lemma \( inv_is_iD \, [elim] : \)
  fixes ip rt
  assumes "ip \in kD(rt)"
  and "the (flag rt ip) = inv"
  shows "ip \in iD(rt)"
  using assms unfolding iD_def by auto

lemma \( iD_flag_is_inv \, [elim, simp] : \)
  fixes ip rt
  assumes "ip \in iD(rt)"
  shows "the (flag rt ip) = inv"
  proof -
from \( \text{ip} \in \text{id}(\text{rt}) \) have "\( \text{ip} \in \text{kd}(\text{rt}) \)" by auto
with assms show \( \? \text{thesis} \) unfolding id_def by auto
qed

lemma \( \text{kd\_but\_not\_vd\_is\_id} \) [elim]:
  fixes ip rt
  assumes "\( \text{ip} \in \text{kd}(\text{rt}) \)"
  and "\( \text{ip} \in \text{vd}(\text{rt}) \)"
  shows "\( \text{ip} \in \text{id}(\text{rt}) \)"
proof -
  from \( \langle \text{ip} \in \text{kd}(\text{rt}) \rangle \) obtain dsn dsk f hops nhop
  where \( \text{rtip} : \text{rt ip} = \text{some } (dsn, dsk, f, hops, nhop) \)
  by (metis kd_Some)
  from \( \langle \text{ip} \not\in \text{vd}(\text{rt}) \rangle \) have "\( f \neq \text{val} \)"
proof (rule contrapos_nn)
    assume "\( f = \text{val} \)"
    with \( \text{rtip} \) have "\( \text{the (flag rt ip)} = \text{val} \)"
    by simp
    with \( \langle \text{ip} \in \text{kd}(\text{rt}) \rangle \) show "\( \text{ip} \in \text{vd}(\text{rt}) \)" ..
  qed
  with \( \text{rtip} \) have "\( \text{the (flag rt ip)} = \text{inv} \)"
  by simp
  with \( \langle \text{ip} \in \text{kd}(\text{rt}) \rangle \) show "\( \text{ip} \in \text{id}(\text{rt}) \)" ..
  qed

lemma \( \text{vd\_or\_id} \) [elim]:
  fixes ip rt
  assumes "\( \text{ip} \in \text{kd}(\text{rt}) \)"
  and "\( \text{ip} \in \text{vd}(\text{rt}) \) = \( \Rightarrow \text{P rt ip} \)"
  and "\( \text{ip} \in \text{id}(\text{rt}) \) = \( \Rightarrow \text{P rt ip} \)"
  shows "\( \text{P rt ip} \)"
proof -
  from \( \langle \text{ip} \in \text{kd}(\text{rt}) \rangle \) have "\( \text{ip} \in \text{vd}(\text{rt}) \cup \text{id}(\text{rt}) \)"
  by (simp add: kd_is_vd_and_id)
  thus \( \? \text{thesis} \) by (auto elim: assms(2-3))
  qed

lemma \( \text{proj5\_eq\_dhops} \) "\( \forall \text{dip rt. dip} \in \text{kd}(\text{rt}) \Rightarrow \pi_5(\text{the (rt dip)}) = \text{the (dhops rt dip)} \)"
unfolding sqn_def by (drule kd_Some) clarsimp

lemma \( \text{proj4\_eq\_flag} \) "\( \forall \text{dip rt. dip} \in \text{kd}(\text{rt}) \Rightarrow \pi_4(\text{the (rt dip)}) = \text{the (flag rt dip)} \)"
unfolding sqn_def by (drule kd_Some) clarsimp

lemma \( \text{proj2\_eq\_sqn} \) "\( \forall \text{dip rt. dip} \in \text{kd}(\text{rt}) \Rightarrow \pi_2(\text{the (rt dip)}) = \text{sqn rt dip} \)"
unfolding sqn_def by (drule kd_Some) clarsimp

lemma \( \text{kd\_sqnf\_is\_proj3} \) [simp]:
  "\( \forall \text{ip rt. ip} \in \text{kd}(\text{rt}) \Rightarrow \text{sqnf rt ip} = \pi_3(\text{the (rt ip)}) \)"
unfolding sqnf_def by auto

lemma \( \text{vd\_flag\_val} \) [simp]:
  "\( \forall \text{dip rt. dip} \in \text{vd}(\text{rt}) \Rightarrow \text{the (flag rt dip)} = \text{val} \)"
unfolding vd_def by clarsimp

lemma \( \text{kd\_update} \) [simp]:
  "\( \forall \text{rt nip v. kd (rt(nip } \mapsto \text{ v}) = \text{insert nip (kd rt)} \)"
unfolding kd_def by auto

lemma \( \text{kd\_empty} \) [simp]: "\( \text{kd Map.empty} = \{\} \)"
unfolding kd_def by simp

lemma \( \text{ip\_equal\_or\_known} \) [elim]:
  fixes rt ip ip'
  assumes "\( \text{ip} = \text{ip'} \lor \text{ip} \in \text{kd}(\text{rt}) \)"
  and "\( \text{ip} = \text{ip'} \Rightarrow \text{P rt ip ip'} \)"
  and "\( [\text{ip} \neq \text{ip'}; \text{ip} \in \text{kd}(\text{rt})] \Rightarrow \text{P rt ip ip'} \)"
3.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

Updating route entries

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = es (the (rt dip))"
  using assms [THEN kD_Some] by auto

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip /∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = en"
  using assms [THEN kD_None] by auto

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip /∈ kD(rt)"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def by simp

definition update_arg_wf :: "r ⇒ bool"
where "update_arg_wf r ≡ π₄(r) = val ∧
(π₂(r) = 0) = (π₃(r) = unk) ∧
(π₃(r) = unk → π₅(r) = 1)"

lemma update_arg_wf_gives_cases:
  "∀ r. update_arg_wf r ⇒ (π₂(r) = 0) = (π₃(r) = unk)"
  unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples [simp]:
  "∀ nhip. update_arg_wf (0, unk, val, Suc 0, nhip)"
  "∀ n hops nhip. update_arg_wf (Suc n, kno, val, hops, nhip)"
  unfolding update_arg_wf_def by auto

lemma update_arg_wf_tuples' [elim]:
  "∀ n hops nhip. Suc 0 ≤ n ⇒ update_arg_wf (n, kno, val, hops, nhip)"
  unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: "∀ nhip. P (0, unk, val, Suc 0, nhip)"
  and c2: "∀ dsn hops nhip. dsn > 0 ⇒ P (dsn, kno, val, hops, nhip)"
  shows "P r" 
  proof -
    obtain dsn dsk flag hops nhip
    where *: "r = (dsn, dsk, flag, hops, nhip)" by (cases r)
    with update_arg_wf r have wf1: "flag = val"
    and wf2: "(dsn = 0) = (dsk = unk)"
and \( wf3: \ dsk = unk \rightarrow (hops = 1) \)

unfolding update_arg_wf_def by auto

have "\( P (dsn, dsk, flag, hops, nhip) \)"

proof (cases dsk)
  assume "dsk = unk"
  moreover with \( wf2 \) \( wf3 \) have "dsn = 0" and "hops = Suc 0" by auto
  ultimately show \( ?thesis \) using \( \langle flag = val \rangle \) by simp (rule c1)

next
  assume "dsk = kno"
  moreover with \( wf2 \) have "dsn > 0" by simp
  ultimately show \( ?thesis \) using \( \langle flag = val \rangle \) by simp (rule c2)

qed

with \( * \) show "\( P r \)" by simp

qed

definition update :: "rt \Rightarrow ip \Rightarrow r \Rightarrow rt"
where
  "update rt ip r \equiv \case \sigma_{route}(rt, ip) of
  None \Rightarrow rt (ip \mapsto r)
  | Some s \Rightarrow
    \begin{cases}
      \text{if } \pi_2(s) < \pi_2(r) \text{ then } rt (ip \mapsto r) \\
      \text{else if } \pi_2(s) = \pi_2(r) \land (\pi_3(s) > \pi_3(r) \lor \pi_4(s) = \text{inv}) \text{ then } rt (ip \mapsto r) \\
      \text{else if } \pi_3(r) = unk \text{ then } rt (ip \mapsto (\pi_2(s), \text{snd (r)})) \\
      \text{else } rt (ip \mapsto s)
    \end{cases}\"

lemma update_simps [simp]:
  fixes r s nrt nr' ns rt ip
  defines "s \equiv \text{the } \sigma_{route}(rt, ip)"
  and "nr' \equiv (\pi_2(s), \pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r))"
  shows
  "[\{ \langle ip \notin kD(rt) \rangle \}] \Rightarrow \text{update rt ip r = rt (ip \mapsto r)}"
  "[\{ \langle ip \in kD(rt); \text{sqn rt ip} < \pi_2(r) \rangle \}] \Rightarrow \text{update rt ip r = rt (ip \mapsto r)}"
  "[\{ \langle ip \in kD(rt); \text{sqn rt ip} = \pi_2(r); \text{flag rt ip} = \text{Some inv} \rangle \}] \Rightarrow \text{update rt ip r = rt (ip \mapsto r)}"
  "[\{ \langle ip \in kD(rt); \pi_3(r) = unk; (\pi_2(r) = 0) = (\pi_3(r) = \text{unk}) \rangle \}] \Rightarrow \text{update rt ip r = rt (ip \mapsto nr')}"
  "[\{ \langle ip \in kD(rt); \text{sqn rt ip} \geq \pi_2(r); \pi_3(r) = \text{kno}; \text{sqn rt ip} = \pi_2(r) \Rightarrow \text{the (dhops rt ip)} \leq \pi_5(r) \land \text{the (flag rt ip) = val} \}] \Rightarrow \text{update rt ip r = rt (ip \mapsto s)}"

proof -
  assume "ip \notin kD(rt)"
  hence "\( \sigma_{route}(rt, ip) = \text{None} \)" ..
  thus "update rt ip r = rt (ip \mapsto r)"

  unfolding update_def by simp

next
  assume "ip \in kD(rt)"
  and "\( \text{sqn rt ip} < \pi_2(r) \)"
  from this(1) obtain dsn dsk fl hops nhip
  where "rt ip = \text{Some (dsn, dsk, fl, hops, nhip)}"
    by (metis kD_Some)
  with \( \langle \text{sqn rt ip} < \pi_2(r) \rangle \) show "update rt ip r = rt (ip \mapsto r)"
  unfolding update_def s_def by auto

next
  assume "ip \in kD(rt)"
  and "\( \text{sqn rt ip} = \pi_2(r) \)"
  and "\( \text{the (dhops rt ip)} > \pi_5(r) \)"
  from this(1) obtain dsn dsk fl hops nhip
  where "rt ip = \text{Some (dsn, dsk, fl, hops, nhip)}"
    by (metis kD_Some)
  with \( \langle \text{sqn rt ip} = \pi_2(r) \rangle \) and \( \langle \text{the (dhops rt ip)} > \pi_5(r) \rangle \)
show "update rt ip r = rt (ip ↦ r)"
  unfolding update_def s_def by auto

next
assume "ip ∈ kD(rt)"
and "sqn rt ip = π₂(r)"
and "flag rt ip = Some inv"
from this(1) obtain dsn dsk fl hops nhip
  where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
  by (metis kD_Some)
with (sqn rt ip = π₂(r)) and (flag rt ip = Some inv)
show "update rt ip r = rt (ip ↦ r)"
  unfolding update_def s_def by auto

next
assume "ip ∈ kD(rt)"
and "π₃(r) = unk"
and "(π₂(r) = 0) = (π₃(r) = unk)"
from this(1) obtain dsn dsk fl hops nhip
  where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
  by (metis kD_Some)
with ((π₂(r) = 0) = (π₃(r) = unk)) and (π₃(r) = unk)
show "update rt ip r = rt (ip ↦ nr')"
  unfolding update_def nr'_def s_def
  by (cases r) simp

next
assume "ip ∈ kD(rt)"
and otherassms: "sqn rt ip ≥ π₂(r)"
  "π₃(r) = kno"
  "sqn rt ip = π₂(r) ⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val"
from this(1) obtain dsn dsk fl hops nhip
  where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
  by (metis kD_Some)
with otherassms show "update rt ip r = rt (ip ↦ s)"
  unfolding update_def s_def by auto

qed

lemma update_cases [elim]:
  assumes "(π₂(r) = 0) = (π₃(r) = unk)"
  and c1: "[\[ \[ \text{ip } \notin \text{kD(rt)} \] \] \] ⇒ P (rt (ip ↦ r))"
  and c2: "[\[ \[ \text{ip } \in \text{kD(rt)}; \text{sqn rt ip } < \pi₂(r) \] \] \] ⇒ P (rt (ip ↦ r))"
  and c3: "[\[ \[ \text{ip } \in \text{kD(rt)}; \text{sqn rt ip } = \pi₂(r); \text{the (dhops rt ip)} > \pi₅(r) \] \] \] ⇒ P (rt (ip ↦ r))"
  and c4: "[\[ \[ \text{ip } \in \text{kD(rt)}; \text{sqn rt ip } = \pi₂(r); \text{the (flag rt ip)} = \text{inv} \] \] \] ⇒ P (rt (ip ↦ r))"
  and c5: "[\[ \[ \text{ip } \in \text{kD(rt)}; \pi₃(r) = unk \] \] \] ⇒ P (rt (ip ↦ (π₃(\text{the \( \sigma \) route (rt, ip))}, π₃(r), π₄(r), π₅(r), π₆(r))))"
  and c6: "[\[ \[ \text{ip } \in \text{kD(rt)}; \text{sqn rt ip } ≥ \pi₂(r); \pi₃(r) = kno; \text{sqn rt ip } = \pi₂(r) ⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val \] \] \] ⇒ P (rt (ip ↦ the \( \sigma \) route (rt, ip))))"
shows "(P (update rt ip r))"

proof (cases "\text{ip } \in \text{kD(rt)}")
assume "\text{ip } \notin \text{kD(rt)}"
with c1 show ?thesis
  by simp

next
assume "\text{ip } \in \text{kD(rt)}"
moreover then obtain dsn dsk fl hops nhip
  where \text{rt eq: "rt ip = Some (dsn, dsk, fl, hops, nhip)"}
  by (metis kD_Some)
moreover obtain dsn' dsk' fl' hops' nhip'
  where \text{req: "r = \( \text{dsn}', \text{dsk}', \text{fl}', \text{hops}', \text{nhip}' \)"
  by (cases r) metis

259
ultimately show thesis
using \((\pi_2(r) = 0) = (\pi_3(r) = \text{unk})\):
\[c2 \ [\text{OF } ip \in kD(rt)]\]
\[c3 \ [\text{OF } ip \in kD(rt)]\]
\[c4 \ [\text{OF } ip \in kD(rt)]\]
\[c5 \ [\text{OF } ip \in kD(rt)]\]
\[c6 \ [\text{OF } ip \in kD(rt)]\]
unfolding update_def sqn_def by auto
qed

lemma update_cases_kD:
assumes \"(\pi_2(r) = 0) = (\pi_3(r) = \text{unk})\"
and \"ip \in kD(rt)\"
and \text{c2}: \"sqn rt ip < \pi_2(r) = \Rightarrow P (rt (ip \mapsto r))\"
and \text{c3}: \"[sqn rt ip = \pi_2(r); the (dhops rt ip) > \pi_5(r)] = \Rightarrow P (rt (ip \mapsto r))\"
and \text{c4}: \"[sqn rt ip = \pi_2(r); the (flag rt ip) = inv] = \Rightarrow P (rt (ip \mapsto r))\"
and \text{c5}: \"(\pi_3(r) = \text{unk} = \Rightarrow P (rt (ip \mapsto (\pi_2(\text{route}(rt, ip)), \pi_3(r), 
\pi_4(r), \pi_5(r), \pi_6(r))))\"
and \text{c6}: \"[sqn rt ip \geq \pi_2(r); \pi_3(r) = \text{kno}; 
sqn rt ip = \pi_2(r) = \Rightarrow \text{dhops rt ip} \leq \pi_5(r) \land \text{flag rt ip} = \text{val}] = \Rightarrow P (rt (ip \mapsto \text{route}(rt, ip)))\"

shows \"(P (update rt ip r))\"
using assms(1) proof (rule update_cases)
assume \"sqn rt ip < \pi_2(r)\"
thus \"P (rt(ip \mapsto r))\" by (rule c2)
next
assume \"sqn rt ip = \pi_2(r)\"
and \"the (dhops rt ip) > \pi_5(r)\"
thus \"P (rt(ip \mapsto r))\"
by (rule c3)
next
assume \"sqn rt ip = \pi_2(r)\"
and \"the (flag rt ip) = inv\"
thus \"P (rt(ip \mapsto r))\"
by (rule c4)
next
assume \"\pi_3(r) = \text{unk}\"
thus \"P (rt (ip \mapsto (\pi_2(\text{route}(rt, ip)), \pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r))))\"
by (rule c5)
next
assume \"sqn rt ip \geq \pi_2(r)\"
and \"\pi_3(r) = \text{kno}\"
and \"sqn rt ip = \pi_2(r) = \Rightarrow \text{dhops rt ip} \leq \pi_5(r) \land \text{flag rt ip} = \text{val}\"
thus \"P (rt (ip \mapsto (\text{rt ip}))\"
by (rule c6)
qed (simp add: \langle ip \in kD(rt) \rangle)

lemma in_kD_after_update [simp]:
fixes rt nip dsn dsk flag hops nhip
shows \"kD (update rt nip (dsn, dsk, flag, hops, nhip)) = insert nip (kD rt)\"
unfolding update_def
by (cases \"rt nip\") auto

lemma nhop_of_update [simp]:
fixes rt dip dsn dsk flag hops nhip
assumes \"rt \neq update rt dip (dsn, dsk, flag, hops, nhip)\"
shows \"the (nhop (update rt dip (dsn, dsk, flag, hops, nhip)) dip) = nhip\"
proof -
from assms
have update_neq: \"\forall v. rt dip = Some v \Rightarrow update rt dip (dsn, dsk, flag, hops, nhip) 
\neq rt dip \mapsto (the (rt dip))\"
by auto
show ?thesis
  proof (cases "rt dip = None")
    assume "rt dip = None"
    thus "?thesis" unfolding update_def by clarsimp
  next
    assume "rt dip \neq None"
    then obtain v where "rt dip = Some v" by (metis not_None_eq)
    with update_neq [OF this] show ?thesis
      unfolding update_def by auto
  qed
qed

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn (λx. if x = rip then Some v else rt x) ip
         = (if ip = rip then π2(v) else sqn rt ip)"
  unfolding sqn_def by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip)) dip"
  proof (rule update_cases)
    show "(π2 (dsn, dsk, val, hops, nhip) = 0) = (π3 (dsn, dsk, val, hops, nhip) = unk)"
      by simp (rule assms)
  qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip
  assumes "1 ≤ hops"
  shows "sqn rt ip ≤ sqn (update rt ip' (dsn, dsk, flag, hops, nhip)) ip"
  using assms unfolding update_def sqn_def
    by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops nhip ip rip
  assumes "Suc 0 ≤ dhops rt ip" (is _: "\forall ip\in kD rt. dhops rt ip \geq 1")
  shows "Suc 0 ≤ dhops rt ip - 1" (is _: "\forall ip\in kD rt. dhops rt ip = Suc 0")
  using ip proof
    assume "Suc 0 ≤ dhops rt ip" thus ?thesis
    unfolding update_def using ex
    by (cases "rip \in kD rt") (drule(1) bspec, auto)
  next
    assume "Suc 0 ≤ dhops rt ip - 1" thus ?thesis
    using ex unfolding update_def
    by (cases "rip \in kD rt") auto
  qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip \neq dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip)) ip = rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip \neq dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)
lemma dhops_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip \neq dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  "\rt ip dsn dsk flag hops nhip. sqn (rt(ip \mapsto v)) ip = \pi_2(v)"
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt \neq update rt dip (osn, kno, val, hops, nhip)"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip)) dip) = hops"
  using assms unfolding update_def
  by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma nhop_update_unk_val [simp]:
  "\rt dip dsn dsk flg hops sip. the (nhop (update rt dip (dsn, unk, val, hops, sip)) dip) = sip"
  unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip) \neq rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip)) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.splits if_split_asm) auto

lemma update_rt_split_asm:
  "\rt ip dsn dsk flag hops sip.
   P (update rt ip (dsn, dsk, flag, hops, sip)) =
   (\neg (rt = update rt ip (dsn, dsk, flag, hops, sip) \land \neg P rt
   \lor \neg update rt ip (dsn, dsk, flag, hops, sip)
   \land \neg P (update rt ip (dsn, dsk, flag, hops, sip))))"
  by auto

lemma sqn_update [simp]: "\rt dip dsn dsk flg hops sip.
   rt \neq update rt dip (dsn, kno, flg, hops, sip)
   \implies sqn (update rt dip (dsn, kno, flg, hops, sip)) dip = dsn"
  unfolding update_def by (clarsimp split: option.split if_split_asm) auto

lemma sqnf_update [simp]: "\rt dip dsn dsk flg hops sip.
   rt \neq update rt dip (dsn, dsk, flg, hops, sip)
   \implies sqnf (update rt dip (dsn, dsk, flg, hops, sip)) dip = dsk"
  unfolding sqnf_def
  by (clarsimp split: option.splits if_split_asm) auto

lemma update_kno_dsn_greater_zero:
  "\rt dip ip dsn hops. 1 \leq dsn \implies 1 \leq (sqn (update rt dip (dsn, kno, val, hops, ip)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: "\rt dip dsn dsk flg hops sip.
   rt \neq update rt dip (dsn, dsk, flg, hops, sip)
   \implies \pi_3 (the (update rt dip (dsn, dsk, flg, hops, sip)) dip) = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits if_split_asm) auto

lemma nhop_update_changed_kno_val [simp]: "\rt ip dsn dsk hops nhip.
   rt \neq update rt ip (dsn, kno, val, hops, nhip)
   \implies the (nhop (update rt ip (dsn, kno, val, hops, nhip)) ip) = nhip"
unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma flag_update [simp]: "\rt dip dsn flg hops sip.
rt \neq update rt dip (dsn, kno, flg, hops, sip)
\implies the (flag (update rt dip (dsn, kno, flg, hops, sip)) dip) = flg"
unfolding update_def
by (clarsimp split: option.split if_split_asm) auto

lemma the_flag_Some [dest!]:
fixes ip rt
assumes "the (flag rt ip) = x"
and "ip \in kD rt"
shows "flag rt ip = Some x"
using assms by auto

lemma kD_update_unchanged [dest]:
fixes rt dip dsn dsk flag hops nhip
assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip)"
shows "dip \in kD(rt)"
proof -
have "dip \in kD(update rt dip (dsn, dsk, flag, hops, nhip))" by simp
with assms show thesis by simp
qed

lemma nhop_update [simp]: "\rt dip dsn dsk flg hops sip.
rt \neq update rt dip (dsn, dsk, flg, hops, sip)
\implies the (nhop (update rt dip (dsn, dsk, flg, hops, sip)) dip) = sip"
unfolding update_def sqn_def
by (clarsimp split: option.splits if_split_asm) auto

lemma sqn_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip
assumes "ip \neq dip"
shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip
assumes "ip \neq dip"
shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:
"\rt dip dsn dsk hops nhip.
dip \in vD(update rt dip (dsn, dsk, val, hops, nhip)) \implies (dip \in vD(rt) \lor dip=dip')"
unfolding update_def vD_def by (clarsimp split: option.split_asm if_split_asm)

Invalidating route entries

definition invalidate :: "rt \Rightarrow (ip \Rightarrow sqn) \Rightarrow rt"
where "invalidate rt dests \equiv 
\lambda ip. case (rt ip, dests ip) of
  (None, _) \Rightarrow None
  | (Some s, None) \Rightarrow Some s
  | (Some (_, dsk, _, hops, nhip), Some rsn) \Rightarrow Some (rsn, dsk, inv, hops, nhip)"

lemma proj3_invalidate [simp]:
"\rt dip dsn dsk hops nhip.
\pi_3 (the ((invalidate rt dests) dip)) = \pi_3 (the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)
lemma proj5_invalidate [simp]:
"\( \wedge dip. \pi_5(\text{the } (\text{invalidate rt dests) dip}) = \pi_5(\text{the } (rt dip)) \)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
"\( \wedge dip. \pi_6(\text{the } (\text{invalidate rt dests) dip}) = \pi_6(\text{the } (rt dip)) \)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_kD_inv [simp]:
"\( \wedge rt dests. kD (\text{invalidate rt dests}) = kD rt \)"
unfolding invalidate_def kD_def
by (simp split: option.split)

lemma invalidate_sqn:
fixes rt dip dests
assumes "\( \forall rsn. \text{dests dip = Some rsn } \rightarrow sqn rt dip \leq rsn \)"
shows "sqn rt dip \leq sqn (\text{invalidate rt dests) dip}"
proof (cases "dip \notin kD(rt)"
assume "\( \neg dip \notin kD(rt) \)"
then obtain dsn dsk flag hops nhip pre where "rt dip = Some (dsn, dsk, flag, hops, nhip)"
by (metis kD_Some)
with assms show "sqn rt dip \leq sqn (\text{invalidate rt dests) dip}"
by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
qed simp

lemma sqn_invalidate_in_dests [simp]:
fixes dests ipa rsn rt
assumes "dests ipa = Some rsn" and "ipa \in kD(rt)"
shows "sqn (\text{invalidate rt dests) ipa = rsn}"
unfolding invalidate_def sqn_def
using assms(1) assms(2) [THEN kD_Some]
by clarsimp

lemma dhops_invalidate [simp]:
"\( \wedge dip. \text{the } (\text{dhops (invalidate rt dests) dip}) = \text{the } (dhops rt dip) \)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
"\( \wedge dip. sqnf (\text{invalidate } (rt \xi) \text{ (dests } \xi) \text{) dip} = sqnf (rt \xi) \text{ dip} \)"
unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
"\( \wedge dip. \text{the } (\text{nhop (invalidate rt } \xi \text{ (dests } \xi \text{) dip}) = \text{the } (nhop (rt } \xi \text{ dip}) \)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
fixes rt dests dip
assumes "dip \notin \text{dom(dests)}"
shows "\text{invalidate rt dests dip} = rt dip"
using assms unfolding invalidate_def
by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
fixes rt dests dip
assumes "dip \notin kD(rt)"
shows "\text{invalidate rt dests dip} = None"
using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
"\( \wedge dip rt dests. dip \in vD(\text{invalidate rt dests) \implies dip \in vD(rt) \wedge dests dip = None} \)"
unfolding invalidate_def vD_def
by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
fixes dests dip rt
assumes "dip ∉ dom(dests)"
shows "sqn (invalidate rt dests) dip = sqn rt dip"
using asms unfolding sqn_def by simp

lemma invalidate_changes:
fixes rt dests dip dsn dsk flag hops nhip
assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip)"
shows "dsn = (case dests dip of None ⇒ π₂(the (rt dip)) | Some rsn ⇒ rsn)
∧ dsk = π₃(the (rt dip))
∧ flag = (if dests dip = None then π₄(the (rt dip)) else inv)
∧ hops = π₅(the (rt dip))
∧ nhip = π₆(the (rt dip)))"
using asms unfolding invalidate_def
by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "∀ dip rt dests. dip ∈ kD (rt) ⇒ π₃(the (invalidate rt dests dip)) = π₃(the (rt dip))"
by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
assumes "dests ip = Some rsn"
and "ip ∈ kD(rt)"
shows "ip ∈ iD(invalidate rt dests)"
using asms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
by (clarsimp split: option.split)

3.1.5 Route Requests

Generate a fresh route request identifier.

definition nrreqid :: "(ip × rreqid) set ⇒ ip ⇒ rreqid"
where "nrreqid rreqs ip = Max ({n. (ip, n) ∈ rreqs} ∪ {0}) + 1"

3.1.6 Queued Packets

Functions for sending data packets.

type synonym store = "ip ⇒ (p × data list)"

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("σ_queue(_, _)"
where "σ_queue(store, dip) ≡ case store dip of None ⇒ [] | Some (p, q) ⇒ q"

definition qD :: "store ⇒ ip set"
where "qD ≡ dom"

definition add :: "data ⇒ ip ⇒ store ⇒ store"
where "add d dip store ≡ case store dip of
None ⇒ store (dip := None)
| Some (p, q) ⇒ store (dip := (p, q @ [d])))"

lemma qD_add [simp]:
fixes d dip store
shows "qD(add d dip store) = insert dip (qD store)"
unfolding add_def Let_def qD_def
by (clarsimp split: option.split)

definition drop :: "ip ⇒ store ⇒ store"
where "drop dip store ≡
map_option (λ(p, q). if tl q = [] then store (dip := None)
else store (dip := (p, tl q))) (store dip)"

definition $\sigma_p$-flag :: "store $\Rightarrow$ ip $\Rightarrow$ p" ("$\sigma_p$-flag'(_, _, _)"
  where "$\sigma_p$-flag(store, dip) \equiv map\_option\_fst (store dip)"

definition unsetRRF :: "store $\Rightarrow$ ip $\Rightarrow$ store"
  where "unsetRRF store dip \equiv case store dip of
  None $\Rightarrow$ store
  | Some (p, q) $\Rightarrow$ store (dip $\mapsto$ (noreq, q))"

definition setRRF :: "store $\Rightarrow$ (ip $\Rightarrow$ sqn) $\Rightarrow$ store"
  where "setRRF store dests \equiv \lambda dip. if dests dip = None then store dip
  else map\_option (\lambda(_, q). (req, q)) (store dip)"

3.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. nhop is partial, thus a ‘the’ is needed, similarly for dhops and addpreRT.
2. precs is partial.
3. $\sigma_p$-flag(store, dip) is partial.
4. The routing table (rt) is modelled as a map (ip $\Rightarrow$ r option) rather than a set of 7-tuples, likewise, the r
   is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address (dip) is taken from the argument to the function,
   rather than a part of the result. Well-definedness then follows from the structure of the type and
   more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the dests mapping passed to invalidate, and store.

3.2 AODV protocol messages

theory C_Aodv_Message
imports C_Stobcast
begin

datatype msg =
  Rreq nat rreqid ip sqn k ip sqn ip
  | Rrep nat ip sqn ip ip
  | Rerr "ip $\mapsto$ sqn" ip
  | Newpkt data ip
  | Pkt data ip ip

instantiation msg :: msg
begin
  definition newpkt_def [simp]: "newpkt \equiv \lambda (d, dip). Newpkt d dip"
  definition eq_newpkt_def: "eq\_newpkt m \equiv case m of Newpkt d dip \Rightarrow True | _ \Rightarrow False"

  instance by intro\_classes (simp add: eq\_newpkt\_def)
end

The msg type models the different messages used within AODV. The instantiation as a msg is a technicality due to
the special treatment of newpkt messages in the AWN SOS rules. This use of classes allows a clean separation of
the AWN-specific definitions and these AODV-specific definitions.

definition rreq :: "nat \times rreqid \times ip \times sqn \times k \times ip \times sqn \times ip \Rightarrow msg"
  where "rreq \equiv \lambda (hops, rreqid, dip, dsn, dsk, oip, osn, sip).
  Rreq hops rreqid dip dsn dsk oip osn sip"

lemma rreq\_simp [simp]:
  "rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip) = Rreq hops rreqid dip dsn dsk oip osn sip"
  unfolding rreq\_def by simp

266
definition rrep :: "nat × ip × sqn × ip × ip ⇒ msg"
  where "rrep ≡ λ(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"

lemma rrep_simp [simp]:
  "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
  unfolding rrep_def by simp

definition rerr :: "(ip ↪ sqn) × ip ⇒ msg"
  where "rerr ≡ λ(dests, sip). Rerr dests sip"

lemma rerr_simp [simp]:
  "rerr(dests, sip) = Rerr dests sip"
  unfolding rerr_def by simp

lemma not_eq_newpkt_rreq [simp]:
  "¬ eq_newpkt (Rreq hops rreqid dip dsn dsk oip osn sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rrep [simp]:
  "¬ eq_newpkt (Rrep hops dip dsn oip sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rerr [simp]:
  "¬ eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_pkt [simp]:
  "¬ eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp

definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"

lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp

end

3.3 The AODV protocol

theory C_Aodv
imports C_Aodv_Data C_Aodv_Message
  AWN.AWN_SOS_Labels AWN.AWN_Invariants
begin

3.3.1 Data state

record state =
  ip :: "ip"
  sn :: "sqn"
  rt :: "rt"
  rreqs :: "(ip × rreqid) set"
  store :: "store"
  msg :: "msg"
  data :: "data"
  dests :: "ip ↪ sqn"
  rreqid :: "rreqid"
  dip :: "ip"
  oip :: "ip"
  hops :: "nat"
  dsn :: "sqn"
  dsk :: "k"
  osn :: "sqn"
  sip :: "ip"
abbreviation aodv_init :: "ip ⇒ state"
  where "aodv_init i ≡ 
    ip = i,
    sn = 1,
    rt = Map.empty,
    rreqs = {},
    store = Map.empty,
    msg = (SOME x. True),
    data = (SOME x. True),
    dests = (SOME x. True),
    rreqid = (SOME x. True),
    dip = (SOME x. True),
    oip = (SOME x. True),
    hops = (SOME x. True),
    dsn = (SOME x. True),
    dsk = (SOME x. True),
    osn = (SOME x. True),
    sip = (SOME x. x ≠ i)
  "

lemma some_neq_not_eq [simp]: "¬((SOME x :: nat. x ≠ i) = i)"
  by (subst some_eq_ex) (metis zero_neq_numeral)

definition clear_locals :: "state ⇒ state"
  where "clear_locals ξ = ξ 
    msg := (SOME x. True),
    data := (SOME x. True),
    dests := (SOME x. True),
    rreqid := (SOME x. True),
    dip := (SOME x. True),
    oip := (SOME x. True),
    hops := (SOME x. True),
    dsn := (SOME x. True),
    dsk := (SOME x. True),
    osn := (SOME x. True),
    sip := (SOME x. x ≠ ip ξ)
  "

lemma clear_locals_sip_not_ip [simp]: "¬(sip (clear_locals ξ) = ip ξ)"
  unfolding clear_locals_def by simp

lemma clear_locals_but_not_globals [simp]:
  "ip (clear_locals ξ) = ip ξ"
  "sn (clear_locals ξ) = sn ξ"
  "rt (clear_locals ξ) = rt ξ"
  "rreqs (clear_locals ξ) = rreqs ξ"
  "store (clear_locals ξ) = store ξ"
  unfolding clear_locals_def by auto

3.3.2 Auxilliary message handling definitions

definition is_newpkt
  where "is_newpkt ξ ≡ case msg ξ of
    Newpkt data' dip' ⇒ { ξ\{data := data', dip := dip'\} }
    _ ⇒ {}"

definition is_pkt
  where "is_pkt ξ ≡ case msg ξ of
    Pkt data' dip' oip' ⇒ { ξ\{data := data', dip := dip', oip := oip'\} }
    _ ⇒ {}"

definition is_rreq
  where "is_rreq ξ ≡ case msg ξ of
\begin{align*}
Rreq \text{ hops' } rreqid' \text{ dip' } dsn' \text{ dsnk' } oip' \text{ osnk' } sip' \Rightarrow & \\
\_ \Rightarrow \{ \}
\end{align*}

**Lemma is_rreq_asm [dest!]:**

- Assumes \text{\"\xi\" \in is_rreq \\xi\"}
- Shows \text{\"(\exists hops' rreqid' dip' dsn' dsnk' osnk' sip'. \\
  msg \xi = Rreq \text{ hops' rreqid' dip' dsn' dsnk' oip' osnk' sip' } \land \xi' = \xi(\xi' \text{ hops := hops', rreqid := rreqid', dip := dip', dsn := dsn', } \\
  \text{ dsnk := dsnk', oip := oip', osnk := osnk', sip := sip' \})\"}

Using \text{assms unfolding is_rreq_def}

By (cases "msg \xi") simp_all

**Definition is_rrep**

- Where \text{\"is_rrep \xi \equiv case msg \xi of \n  Rrep \text{ hops' dip' dsn' oip' sip' } \Rightarrow \\
  \{ \xi(\xi' \text{ hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' \} \} \\
  \_ \Rightarrow \{ \} \"}

**Lemma is_rrep_asm [dest!]:**

- Assumes \text{\"\xi' \in is_rrep \xi\"}
- Shows \text{\"(\exists hops' dip' dsn' oip' sip'. \\
  msg \xi = Rrep \text{ hops' dip' dsn' oip' sip' } \land \xi' = \xi(\xi' \text{ hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' \})\"}

Using \text{assms unfolding is_rrep_def}

By (cases "msg \xi") simp_all

**Definition is_rerr**

- Where \text{\"is_rerr \xi \equiv case msg \xi of \n  Rerr \text{ dests' sip' } \Rightarrow \\
  \{ \xi(\xi' \text{ dests := dests', sip := sip' } \} \} \\
  \_ \Rightarrow \{ \} \"}

**Lemma is_rerr_asm [dest!]:**

- Assumes \text{\"\xi' \in is_rerr \xi\"}
- Shows \text{\"(\exists dests' sip'. \\
  msg \xi = Rerr \text{ dests' sip' } \land \xi' = \xi(\xi' \text{ dests := dests', sip := sip' } \})\"}

Using \text{assms unfolding is_rerr_def}

By (cases "msg \xi") simp_all

**Lemmas is_msg_defs =**

- is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def

**Lemma is_msg_inv_ip [simp]:**

- \text{\"\xi' \in is_rerr \xi \Rightarrow ip \xi' = ip \xi\"}
- \text{\"\xi' \in is_rrep \xi \Rightarrow ip \xi' = ip \xi\"}
- \text{\"\xi' \in is_rreq \xi \Rightarrow ip \xi' = ip \xi\"}
- \text{\"\xi' \in is_pkt \xi \Rightarrow ip \xi' = ip \xi\"}
- \text{\"\xi' \in is_newpkt \xi \Rightarrow ip \xi' = ip \xi\"}

Unfolding \text{is_msg_defs}

By (cases "msg \xi", clarsimp+)

**Lemma is_msg_inv_sn [simp]:**

- \text{\"\xi' \in is_rerr \xi \Rightarrow sn \xi' = sn \xi\"}
- \text{\"\xi' \in is_rrep \xi \Rightarrow sn \xi' = sn \xi\"}
- \text{\"\xi' \in is_rreq \xi \Rightarrow sn \xi' = sn \xi\"}
- \text{\"\xi' \in is_pkt \xi \Rightarrow sn \xi' = sn \xi\"}
- \text{\"\xi' \in is_newpkt \xi \Rightarrow sn \xi' = sn \xi\"}

Unfolding \text{is_msg_defs}

By (cases "msg \xi", clarsimp+)

**Lemma is_msg_inv_rt [simp]:**

- \text{\"\xi' \in is_rerr \xi \Rightarrow rt \xi' = rt \xi\"}
\[ \xi' \in \text{is}\_rrep \xi \implies \text{rt}\xi' = \text{rt}\xi \]
\[ \xi' \in \text{is}\_rreq \xi \implies \text{rt}\xi' = \text{rt}\xi \]
\[ \xi' \in \text{is}\_pkt \xi \implies \text{rt}\xi' = \text{rt}\xi \]
\[ \xi' \in \text{is}\_\text{newpkt} \xi \implies \text{rt}\xi' = \text{rt}\xi \]

unfolding \text{is}\_\text{msg_defs}
by (cases "\text{msg} \xi", clarsimp+)

lemma \text{is}\_\text{msg_inv}\_\text{rreqs} [simp]:
\[ \xi' \in \text{is}\_\text{rerr} \xi \implies \text{rreqs}\xi' = \text{rreqs}\xi \]
\[ \xi' \in \text{is}\_\text{rrep} \xi \implies \text{rreqs}\xi' = \text{rreqs}\xi \]
\[ \xi' \in \text{is}\_\text{rreq} \xi \implies \text{rreqs}\xi' = \text{rreqs}\xi \]
\[ \xi' \in \text{is}\_\text{pkt} \xi \implies \text{rreqs}\xi' = \text{rreqs}\xi \]
\[ \xi' \in \text{is}\_\text{newpkt} \xi \implies \text{rreqs}\xi' = \text{rreqs}\xi \]

unfolding \text{is}\_\text{msg_defs}
by (cases "\text{msg} \xi", clarsimp+)

lemma \text{is}\_\text{msg_inv}\_\text{store} [simp]:
\[ \xi' \in \text{is}\_\text{rerr} \xi \implies \text{store}\xi' = \text{store}\xi \]
\[ \xi' \in \text{is}\_\text{rrep} \xi \implies \text{store}\xi' = \text{store}\xi \]
\[ \xi' \in \text{is}\_\text{rreq} \xi \implies \text{store}\xi' = \text{store}\xi \]
\[ \xi' \in \text{is}\_\text{pkt} \xi \implies \text{store}\xi' = \text{store}\xi \]
\[ \xi' \in \text{is}\_\text{newpkt} \xi \implies \text{store}\xi' = \text{store}\xi \]

unfolding \text{is}\_\text{msg_defs}
by (cases "\text{msg} \xi", clarsimp+)

lemma \text{is}\_\text{msg_inv}\_\text{sip} [simp]:
\[ \xi' \in \text{is}\_\text{pkt} \xi \implies \text{sip}\xi' = \text{sip}\xi \]
\[ \xi' \in \text{is}\_\text{newpkt} \xi \implies \text{sip}\xi' = \text{sip}\xi \]

unfolding \text{is}\_\text{msg_defs}
by (cases "\text{msg} \xi", clarsimp+)

3.3.3 The protocol process

datatype \text{pseqp} =
\begin{align*}
\text{PAodv} \\
\text{PNewPkt} \\
\text{PPkt} \\
\text{PRreq} \\
\text{PRrep} \\
\text{PRerr}
\end{align*}

fun \text{nat_of_seqp} :: \text{pseqp} \Rightarrow \text{nat}
where
\begin{align*}
\text{nat_of_seqp PAodv} &= 1 \\
\text{nat_of_seqp PPkt} &= 2 \\
\text{nat_of_seqp PNewPkt} &= 3 \\
\text{nat_of_seqp PRreq} &= 4 \\
\text{nat_of_seqp PRrep} &= 5 \\
\text{nat_of_seqp PRerr} &= 6
\end{align*}

instantiation \text{pseqp} :: \text{ord}
begin
\begin{align*}
definition \text{less_eq_seqp} [iff]: & "\text{nat_of_seqp} \text{PAodv} \leq 12 = (\text{nat_of_seqp} 11 \leq \text{nat_of_seqp} 12)" \\
definition \text{less_seqp} [iff]: & "\text{nat_of_seqp} \text{PAodv} < 12 = (\text{nat_of_seqp} 11 < \text{nat_of_seqp} 12)"
\end{align*}
end

abbreviation \text{AODV}
where
\[ \text{AODV} \equiv \lambda_. \text{[clear_locals] call(PAodv)} \]

abbreviation \text{PKT}
where
\[ \text{PKT} \text{ args} \equiv \]
abbreviation NEWPKT

where

"NEWPKT args ≡
  [ξ. let (data, dip, oip) = args ξ in
   (clear_locals ξ) (data := data, dip := dip, oip := oip)]
call(PPkt)"

abbreviation RREQ

where

"RREQ args ≡
  [ξ. let (hops, rreqid, dip, dsn, dsk, oip, osn, sip) = args ξ in
   (clear_locals ξ) (hops := hops, rreqid := rreqid, dip := dip,
   dsn := dsn, dsk := dsk, oip := oip,
   osn := osn, sip := sip)]
call(PRreq)"

abbreviation RREP

where

"RREP args ≡
  [ξ. let (hops, dip, dsn, oip, sip) = args ξ in
   (clear_locals ξ) (hops := hops, dip := dip, dsn := dsn,
   oip := oip, sip := sip)]
call(PRrep)"

abbreviation RERR

where

"RERR args ≡
  [ξ. let (dests, sip) = args ξ in
   (clear_locals ξ) (dests := dests, sip := sip)]
call(PRerr)"

fun \( \Gamma_{AODV} \) :: "(state, msg, pseqp, pseqp label) seqp_env"

where

\( \Gamma_{AODV} \) PAodv = labelled PAodv (receive(\( \lambda \text{msg} \cdot \xi. \xi (\text{msg := msg'}))\).
  (\( \text{is_newpkt} \cdot \text{NEWPKT}(\lambda \xi. (\text{data} \xi, \text{ip} \xi)) \))
  (\( \text{is_pkt} \cdot \text{PKT}(\lambda \xi. (\text{data} \xi, \text{dip} \xi, \text{oip} \xi)) \))
  (\( \text{is_rreq} \cdot \text{RREQ}(\lambda \xi. (\text{hops} \xi, \text{rreqid} \xi, \text{dip} \xi, \text{dsn} \xi, \text{dsk} \xi, \text{oip} \xi, \text{osn} \xi, \text{sip} \xi)) \))
  (\( \text{is_rrep} \cdot \text{RREP}(\lambda \xi. (\text{hops} \xi, \text{dip} \xi, \text{dsn} \xi, \text{oip} \xi, \text{sip} \xi)) \))
  (\( \text{is_rrerr} \cdot \text{RERR}(\lambda \xi. (\text{dests} \xi, \text{sip} \xi)) \))

\( \xi \cdot *\ (\text{dip := dip}) / \text{dip. dip} \in \text{qD}(\text{store} \xi) \cap \text{vD}(\text{rt} \xi) \)\)
\( \xi \cdot *\ (\text{data := hd}(\sigma_{\text{queue}}(\text{store} \xi, \text{dip} \xi)) \))
\( \text{unicast}(\lambda \xi. \text{the}(\text{nhop}(\text{rt} \xi) (\text{dip} \xi), \lambda \xi. \text{pkt}(\text{data} \xi, \text{dip} \xi, \text{ip} \xi)).\)
\( \xi \cdot *\ (\text{store := the}(\text{drop}(\text{dip} \xi) (\text{store} \xi)) \))\]
\( \text{AODV}() \)
\( \xi \cdot *\ (\text{dests := } (\lambda \text{rip. if } (\text{rip} \in \text{vD}(\text{rt} \xi ) \land \text{nhop}(\text{rt} \xi) \text{ rip} = \text{nhop}(\text{rt} \xi) (\text{dip} \xi)) \text{ then Some (inc (sqn (rt \xi) \text{ rip})) else None)}) \))
\( \xi \cdot *\ (\text{rt := invalidate}(\text{rt} \xi) (\text{dests} \xi)) \))
\( \xi \cdot *\ (\text{store := setRRF}(\text{store} \xi) (\text{dests} \xi))\]
\( \text{broadcast}(\lambda \xi. \text{rerr}(\text{dests} \xi, \text{ip} \xi)). \text{AODV}() \)
\( \xi \cdot *\ (\text{dip := dip}) \)
\( \text{dip. dip} \in \text{qD}(\text{store} \xi) \land \text{vD}(\text{rt} \xi ) \land (\sigma_{\text{p-flag}}(\text{store} \xi, \text{dip})) = \text{req} \)\)
\( \xi \cdot *\ (\text{store := unsetRRF}(\text{store} \xi) (\text{dip} \xi)) \)
[[ξ. ξ (| sn := inc (sn ξ) |)]]
[[ξ. ξ (| rreqid := nrreqid (rreqs ξ) (ip ξ) |)]]
[[ξ. ξ (| rreqs := rreqs ξ ∪ {(ip ξ, rreqid ξ)} |)]]
broadcast( λξ. rreq(0, rreqid ξ, dip ξ, sqn (rt ξ) (dip ξ), sqnf (rt ξ) (dip ξ), ip ξ, sn ξ,
ip ξ)). AODV())"
|

" ΓAO D V PNewPkt = labelled PNewPkt (
hξ. dip ξ = ip ξi
deliver( λξ. data ξ).AODV()
⊕ hξ. dip ξ 6= ip ξi
[[ξ. ξ (| store := add (data ξ) (dip ξ) (store ξ) |)]]
AODV())"

| " ΓAO D V PPkt = labelled PPkt (
hξ. dip ξ = ip ξi
deliver( λξ. data ξ).AODV()
⊕ hξ. dip ξ 6= ip ξi
(
hξ. dip ξ ∈ vD (rt ξ) i
unicast( λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, oip ξ)).AODV()
.
[[ξ. ξ (| dests := ( λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
then Some (inc (sqn (rt ξ) rip)) else None) |)]]
[[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]]
[[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]]
broadcast( λξ. rerr(dests ξ, ip ξ)).AODV()
⊕ hξ. dip ξ ∈
/ vD (rt ξ) i
(
hξ. dip ξ ∈ iD (rt ξ) i
broadcast( λξ. rerr([dip ξ 7→ sqn (rt ξ) (dip ξ)], ip ξ)). AODV()
⊕ hξ. dip ξ ∈
/ iD (rt ξ) i
AODV()
)
))"
| " ΓAO D V PRreq = labelled PRreq (
hξ. (oip ξ, rreqid ξ) ∈ rreqs ξi
AODV()
⊕ hξ. (oip ξ, rreqid ξ) ∈
/ rreqs ξi
[[ξ. ξ (| rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, hops ξ + 1, sip ξ) |)]]
[[ξ. ξ (| rreqs := rreqs ξ ∪ {(oip ξ, rreqid ξ)} |)]]
(
hξ. dip ξ = ip ξi
[[ξ. ξ (| sn := max (sn ξ) (dsn ξ) |)]]
unicast( λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, oip ξ, ip ξ)).AODV()
.
[[ξ. ξ (| dests := ( λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
then Some (inc (sqn (rt ξ) rip)) else None) |)]]
[[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]]
[[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]]
broadcast( λξ. rerr(dests ξ, ip ξ)).AODV()
⊕ hξ. dip ξ 6= ip ξi
(
hξ. dip ξ ∈ vD (rt ξ) ∧ dsn ξ ≤ sqn (rt ξ) (dip ξ) ∧ sqnf (rt ξ) (dip ξ) = kno i
unicast( λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
AODV()
.
[[ξ. ξ (| dests := ( λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
then Some (inc (sqn (rt ξ) rip)) else None) |)]]
[[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]]
[[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]]
broadcast( λξ. rerr(dests ξ, ip ξ)).AODV()
⊕ hξ. dip ξ ∈
/ vD (rt ξ) ∨ sqn (rt ξ) (dip ξ) < dsn ξ ∨ sqnf (rt ξ) (dip ξ) = unk i

272


broadcast(\(\lambda\xi. \text{rreq}(\text{hops }\xi + 1, \text{rreqid }\xi, \text{dip }\xi, \max (\text{sqn}(\text{rt }\xi)(\text{dip }\xi)), \text{dsn }\xi, \text{dsk }\xi, \text{oip }\xi, \text{osn }\xi, \text{ip }\xi)).\))

AODV()

\(\langle\xi, \text{rt }\xi \neq \text{update}(\text{rt }\xi)(\text{dip }\xi) (\text{dsn }\xi, \text{kno}, \text{val}, \text{hops }\xi + 1, \text{sip }\xi) \rangle\)

\(\langle\xi, \text{rt }\xi = \text{update}(\text{rt }\xi)(\text{dip }\xi) (\text{dsn }\xi, \text{kno}, \text{val}, \text{hops }\xi + 1, \text{sip }\xi) \rangle\)

\(\langle\xi, \text{oip }\xi = \text{ip }\xi \rangle\)

\(\langle\xi, \text{oip }\xi \neq \text{ip }\xi \rangle\)

\(\langle\xi, \text{oip }\xi \in \text{vD}(\text{rt }\xi)\rangle\)

unicast(\(\lambda\xi. \text{the}(\text{nhop}(\text{rt }\xi)(\text{oip }\xi)), \lambda\xi. \text{rrep}(\text{hops }\xi + 1, \text{dip }\xi, \text{dsn }\xi, \text{oip }\xi, \text{ip }\xi)).\))

AODV()

\(\langle\xi, \text{rt }\xi = \text{update}(\text{rt }\xi)(\text{dip }\xi) (\text{dsn }\xi, \text{kno}, \text{val}, \text{hops }\xi + 1, \text{sip }\xi) \rangle\)

\(\langle\xi, \text{rt }\xi \neq \text{update}(\text{rt }\xi)(\text{dip }\xi) (\text{dsn }\xi, \text{kno}, \text{val}, \text{hops }\xi + 1, \text{sip }\xi) \rangle\)

\(\langle\xi, \text{oip }\xi \neq \text{ip }\xi \rangle\)

\(\langle\xi, \text{oip }\xi \in \text{vD}(\text{rt }\xi)\rangle\)

unicast(\(\lambda\xi. \text{the}(\text{nhop}(\text{rt }\xi)(\text{oip }\xi)), \lambda\xi. \text{rrep}(\text{hops }\xi + 1, \text{dip }\xi, \text{dsn }\xi, \text{oip }\xi, \text{ip }\xi)).\))

AODV()

\(\langle\xi, \text{rt }\xi = \text{update}(\text{rt }\xi)(\text{dip }\xi) (\text{dsn }\xi, \text{kno}, \text{val}, \text{hops }\xi + 1, \text{sip }\xi) \rangle\)

\(\langle\xi, \text{rt }\xi \neq \text{update}(\text{rt }\xi)(\text{dip }\xi) (\text{dsn }\xi, \text{kno}, \text{val}, \text{hops }\xi + 1, \text{sip }\xi) \rangle\)

\(\langle\xi, \text{oip }\xi = \text{ip }\xi \rangle\)

\(\langle\xi, \text{oip }\xi \neq \text{ip }\xi \rangle\)

\(\langle\xi, \text{oip }\xi \in \text{vD}(\text{rt }\xi)\rangle\)

unicast(\(\lambda\xi. \text{the}(\text{nhop}(\text{rt }\xi)(\text{oip }\xi)), \lambda\xi. \text{rrep}(\text{hops }\xi + 1, \text{dip }\xi, \text{dsn }\xi, \text{oip }\xi, \text{ip }\xi)).\))

AODV()

\(\langle\xi, \text{rt }\xi = \text{update}(\text{rt }\xi)(\text{dip }\xi) (\text{dsn }\xi, \text{kno}, \text{val}, \text{hops }\xi + 1, \text{sip }\xi) \rangle\)

\(\langle\xi, \text{rt }\xi \neq \text{update}(\text{rt }\xi)(\text{dip }\xi) (\text{dsn }\xi, \text{kno}, \text{val}, \text{hops }\xi + 1, \text{sip }\xi) \rangle\)

\(\langle\xi, \text{oip }\xi = \text{ip }\xi \rangle\)

\(\langle\xi, \text{oip }\xi \neq \text{ip }\xi \rangle\)

\(\langle\xi, \text{oip }\xi \in \text{vD}(\text{rt }\xi)\rangle\)

unicast(\(\lambda\xi. \text{the}(\text{nhop}(\text{rt }\xi)(\text{oip }\xi)), \lambda\xi. \text{rrep}(\text{hops }\xi + 1, \text{dip }\xi, \text{dsn }\xi, \text{oip }\xi, \text{ip }\xi)).\))

AODV()
fix pn pn'  
  show "call(pn') \notin stermsl (\Gamma_{AODV-skeleton} pn)"
  by (cases pn) simp_all
  qed

declare \Gamma_{AODV-skeleton}.simps [simp del, code del]
lemmas \Gamma_{AODV-skeleton-simps} [simp, code]
  = \Gamma_{AODV-skeleton}.simps [simplified \Gamma_{AODV-simps} seqp_skeleton.simpss]

lemma aodv_proc_cases [dest]:  
  fixes p pn  
  shows "p \in \ctermsl (\Gamma_{AODV} pn) \Longrightarrow
  (p \in \ctermsl (\Gamma_{AODV} PAodv) \lor
  p \in \ctermsl (\Gamma_{AODV} PNewPkt) \lor
  p \in \ctermsl (\Gamma_{AODV} PPkt) \lor
  p \in \ctermsl (\Gamma_{AODV} PRreq) \lor
  p \in \ctermsl (\Gamma_{AODV} PRrep) \lor
  p \in \ctermsl (\Gamma_{AODV} PRerr))"
  by (cases pn) simp_all

definition \sigma_{AODV} :: "ip \Rightarrow (state \times (state, msg, pseqp, pseqp label) seqp) set"
  where "\sigma_{AODV} i \equiv \{(aodv_init i, \Gamma_{AODV} PAodv)\}"

abbreviation paodv :: "ip \Rightarrow (state \times (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
  where "paodv i \equiv (init = \sigma_{AODV} i, trans = seqp_sos \Gamma_{AODV})"

lemma aodv_trans: "trans (paodv i) = seqp_sos \Gamma_{AODV}"
  by simp

lemma aodv_control_within [simp]: "control_within \Gamma_{AODV} (init (paodv i))"
  unfolding \sigma_{AODV} def by (rule control_withinI) (auto simp del: \Gamma_{AODV-simps})

lemma aodv_wf [simp]:  
  "wellformed \Gamma_{AODV}"
  proof (rule, intro allI)
    fix pn pn'
    show "call(pn') \notin stermsl (\Gamma_{AODV} pn)"
      by (cases pn) simp_all
    qed

lemmas aodv_labels_not_empty [simp] = labels_not_empty [OF aodv_wf]

lemma aodv_ex_label [intro]: "\exists l. l \in labels \Gamma_{AODV} p"
  by (metis aodv_labels_not_empty all_not_in_conv)

lemma aodv_ex_labelE [elim]:  
  assumes "\forall l \in labels \Gamma_{AODV} P. P l p"
    and "\exists p l. P l p \Longrightarrow Q"
  shows "Q"
  using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "simple_labels \Gamma_{AODV}"
  proof
    fix pn p
    assume "p \in subterms(\Gamma_{AODV} pn)"
    thus "\exists l. labels \Gamma_{AODV} p = \{l\}"  
      by (cases pn) simp_all cong: seqp_congs | elim disjE)+
    qed

lemma \sigma_{AODV}-labels [simp]: "(\xi, p) \in \sigma_{AODV} i \Longrightarrow labels \Gamma_{AODV} p = \{PAodv-\0\}"
  unfolding \sigma_{AODV} def by simp
lemma aodv_init_kD_empty [simp]:
"(ξ, p) ∈ σ_{AODV} i ⇒ kD (rt ξ) = {}"
unfolding σ_{AODV} def kD_def by simp

lemma aodv_init_sip_not_ip [simp]: "¬(sip (aodv_init i) = i)" by simp

lemma aodv_init_sip_not_ip' [simp]:
assumes "(ξ, p) ∈ σ_{AODV} i"
shows "sip ξ ≠ ip ξ"
using assms unfolding σ_{AODV} def by simp

lemma aodv_init_sip_not_ip'' [simp]:
assumes "(ξ, p) ∈ σ_{AODV} i"
shows "sip ξ ≠ i"
using assms unfolding σ_{AODV} def by simp

lemma clear_locals_sip_not_ip':
assumes "ip ξ = i"
shows "¬(sip (clear_locals ξ) = i)"
using assms by auto

Stop the simplifier from descending into process terms.
declare seqp_congs [cong]

Configure the main invariant tactic for AODV.
declare Γ_{AODV} simps [cterms_env]
 aodv_proc_cases [ctermsl_cases]
 seq_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms_intros]
 seq_step_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms_intros]

end

3.4 Invariant assumptions and properties

theory C_Aodv_Predicates
imports C_Aodv
begin

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

abbreviation not_Pkt :: "msg ⇒ bool"
where "not_Pkt m ≡ case m of Pkt _ _ _ ⇒ False | _ ⇒ True"
definition msg_sender :: "msg ⇒ ip"
where "msg_sender m ≡ case m of Rreq _ _ _ _ _ _ _ ipc ⇒ ipc
  | Rrep _ _ _ _ ipc ⇒ ipc
  | Rerr _ ipc ⇒ ipc
  | Pkt _ _ ipc ⇒ ipc"

lemma msg_sender_simps [simp]:
"⋀hops rreqid dip dsn dsk oip osn sip. msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip) = sip"
"⋀hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) = sip"
"⋀dests sip. msg_sender (Rerr dests sip) = sip"
"⋀d dip sip. msg_sender (Pkt d dip sip) = sip"

unfolding msg_sender_def by simp_all
definition msg_zhops :: "msg ⇒ bool"
where "msg_zhops m ≡ case m of Rreq hopsc _ _ _ dip oip osn sip ⇒ hopsc = 0 → oipc = sipc
  | Rrep _ _ _ _ ipc ⇒ hopsc = 0 → dipc = sipc
  | Rerr _ ipc ⇒ dipc = sipc
  | Pkt _ _ ipc ⇒ dipc = sipc"
lemma msg_zhops_simps [simp]:
 "\hops \reqid \dsn \dsk \oip \osn \sip.
  msg_zhops (Rreq \hops \reqid \dsn \dsk \oip \osn \sip) = (\hops = 0 \implies \oip = \sip)"
 "\hops \dsn \oip \sip. msg_zhops (Rrep \hops \dsn \oip \sip) = (\hops = 0 \implies \dsn = \sip)"
 "\dests \sip. msg_zhops (Rerr \dests \sip) = True"
 "\d dip. msg_zhops (Newpkt \d dip) = True"
 "\d dip \sip. msg_zhops (Pkt \d dip \sip) = True"
 unfolding msg_zhops_def by simp_all

definition rreq_rrep_sn :: "msg \Rightarrow \text{bool}"
 where "rreq_rrep_sn m \equiv case m of
 Rreq _ _ _ _ _ _ \osnc _ \Rightarrow \osnc \geq 1
 | Rrep _ _ \dsnc _ _ \Rightarrow \dsnc \geq 1
 | _ \Rightarrow True"

lemma rreq_rrep_sn_simps [simp]:
 "\hops \reqid \dsn \oip \osn \sip.
  rreq_rrep_sn (Rreq \hops \reqid \dsn \oip \osn \sip) = (\osnc \geq 1)"
 "\hops \dsn \oip \sip. rreq_rrep_sn (Rrep \hops \dsn \oip \sip) = (\dsn \geq 1)"
 "\dests \sip. rreq_rrep_sn (Rerr \dests \sip) = True"
 "\d dip. rreq_rrep_sn (Newpkt \d dip) = True"
 "\d dip \sip. rreq_rrep_sn (Pkt \d dip \sip) = True"
 unfolding rreq_rrep_sn_def by simp_all

definition rreq_rrep_fresh :: "rt \Rightarrow \text{msg} \Rightarrow \text{bool}"
 where "rreq_rrep_fresh \crt m \equiv case m of
 Rreq \hops _ _ _ _ \oipc \osnc \ipcc \Rightarrow \ipcc \neq \oipc \rightarrow
 \oipc \in KD(\crt) \land (\sqn \crt \oipc > \osnc \lor
 \sqn \crt \oipc = \osnc \land \the (dhops \crt \oipc) \leq \hops
 \land \the (flag \crt \oipc) = \val))"
 | Rrep \hops _ _ _ _ \dipc _ \ipcc \Rightarrow \ipcc \neq \dipc \rightarrow
 \dipc \in KD(\crt) \land \sqn \crt \dipc = \dsn \land
 \the (dhops \crt \dipc) = \hops \land
 \the (flag \crt \dipc) = \val)
 | _ \Rightarrow True"

lemma rreq_rrep_fresh [simp]:
 "\hops \reqid \dsn \dsk \oip \osn \sip.
  rreq_rrep_fresh \crt (Rreq \hops \reqid \dsn \dsk \oip \osn \sip) =
 (\sip \neq \oip \rightarrow \oipc \in KD(\crt) \land
 \sqn \crt \oipc > \osnc \lor
 \sqn \crt \oipc = \osnc \land
 \the (dhops \crt \oipc) \leq \hops
 \land \the (flag \crt \oipc) = \val))"
 "\hops \dsn \oip \sip. rreq_rrep_fresh \crt (Rrep \hops \dsn \oip \sip) =
 (\sip \neq \dipc \rightarrow \dipc \in KD(\crt) \land
 \sqn \crt \dipc = \dsn \land
 \the (dhops \crt \dipc) = \hops \land
 \the (flag \crt \dipc) = \val))"
 "\dests \sip. rreq_rrep_fresh \crt (Rerr \dests \sip) = True"
 "\d dip. rreq_rrep_fresh \crt (Newpkt \d dip) = True"
 "\d dip \sip. rreq_rrep_fresh \crt (Pkt \d dip \sip) = True"
 unfolding rreq_rrep_fresh_def by simp_all

definition rerr_invalid :: "rt \Rightarrow \text{msg} \Rightarrow \text{bool}"
 where "rerr_invalid \crt m \equiv \forall \ripc \in \text{dom}(\dests).\ripc \in iD(\crt) \land \the (dests \ripc) = \sqn \crt \ripc"
 | _ \Rightarrow True"

lemma rerr_invalid [simp]:
 "\hops \reqid \dsn \dsk \oip \osn \sip.
  rerr_invalid \crt (Rreq \hops \reqid \dsn \dsk \oip \osn \sip) = True"
"\hops dip dsn oip sip. \rerr_invalid crt (Rrep \hops dip dsn oip sip) = True"
"\dests sip. \rerr_invalid crt (Rerr \dests sip) = (\\rip \in \dom(\dests).
\rip \in d\id(crt) \land \the(\dests \\rip) = \sqn \crt \\rip)"
"\d dip. \rerr_invalid crt (Newpkt \d dip) = True"
"\d dip sip. \rerr_invalid crt (Pkt \d dip sip) = True"
unfolding \rerr_invalid_def by simp_all

definition
\initmissing :: "(nat \Rightarrow state option) \times 'a \Rightarrow (nat \Rightarrow state) \times 'a"
where
"\initmissing \sigma = (\lambda i. case (fst \sigma) i of None \Rightarrow \aodv_init i | Some s \Rightarrow s, snd \sigma)"

lemma not_in_net_ips_fst_init_missing [simp]:
assumes "i /\in net_ips \sigma"
shows "fst (\initmissing (netgmap fst \sigma)) i = \aodv_init i"
using assms unfolding \initmissing_def by simp

lemma fst_initmissing_netgmap_pair_fst [simp]:
"fst (\initmissing (netgmap (\lambda (p, q). (fst (id p), snd (id p), q)) \sigma))
= fst (\initmissing (netgmap fst \sigma))"
unfolding \initmissing_def by auto

We introduce a streamlined alternative to \initmissing with netgmap to simplify invariant statements and thus facilitate their comprehension and presentation.

lemma fst_initmissing_netgmap_default_aodv_init_netlift:
"fst (\initmissing (netgmap fst \sigma)) = default \aodv_init (netlift fst \sigma)"
unfolding initmissing_def default_def
by (simp add: fst_netgmap_netlift del: One_nat_def)

definition
\netglobal :: "((nat \Rightarrow state) \Rightarrow bool) \Rightarrow ((state \times 'b) \times 'c) \net_state \Rightarrow bool"
where
"\netglobal P \equiv (\lambda s. P (default \aodv_init (netlift fst s)))"

end

3.5 Quality relations between routes

theory C_Fresher
imports C_Aodv_Data
begin

3.5.1 Net sequence numbers

On individual routes

definition
\nsqn_r :: "r \Rightarrow \sqn"
where
"\sqn, r \equiv if \pi_4(r) = \val \lor \pi_2(r) = 0 then \pi_2(r) else (\pi_2(r) - 1)"

lemma nsqnr_def':
"\sqn, r = (if \pi_4(r) = \inv then \pi_2(r) - 1 else \pi_2(r))"
unfolding nsqnr_def by simp

lemma nsqn_r_zero [simp]:
"\dsn \dsk \flag \hops \nhip. \sqn, (0, \dsk, \flag, \hops, \nhip) = 0"
unfolding nsqnr_def by clarsimp

lemma nsqn_r_val [simp]:
"\dsn \dsk \hops \nhip. \sqn, (\dsn, \dsk, \val, \hops, \nhip) = \dsn"
unfolding nsqnr_def by clarsimp

lemma nsqn_r_inv [simp]:
"∀dsn dsk hops nhip. nsqn_r (dsn, dsk, inv, hops, nhip) = dsn - 1"
unfolding nsqn_r_def by clarsimp

lemma nsqn_r_lte_dsn [simp]:
"∀dsn dsk flag hops nhip. nsqn_r (dsn, dsk, flag, hops, nhip) ≤ dsn"
unfolding nsqn_r_def by clarsimp

On routes in routing tables

definition
nsqn :: "rt ⇒ ip ⇒ sqn"
where
"nsqn ≡ λrt dip. case σ_route(rt, dip) of None ⇒ 0 | Some r ⇒ nsqn_r (r)"

lemma nsqn_sqn_def:
"∀rt dip. nsqn rt dip = (if flag rt dip = Some val ∨ sqn rt dip = 0
then sqn rt dip else sqn rt dip - 1)"
unfolding nsqn_def sqn_def by (clarsimp split: option.split)

lemma not_in_kD_nsqn [simp]:
assumes "dip /∈ kD(rt)"
shows "nsqn rt dip = 0"
using assms unfolding nsqn_def by simp

lemma kD_nsqn:
assumes "dip ∈ kD(rt)"
shows "nsqn rt dip = nsqn_r (the (σ_route(rt, dip)))"
using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

lemma nsqn_r_flag_pred [simp, intro]:
fixes dsn dsk flag hops nhip
assumes "P (nsqn_r (dsn, dsk, val, hops, nhip))"
and "P (nsqn_r (dsn, dsk, inv, hops, nhip))"
shows "P (nsqn_r (dsn, dsk, flag, hops, nhip))"
using assms by (cases flag) auto

lemma sqn_nsqn:
"∀rt dip. sqn rt dip - 1 ≤ nsqn rt dip"
unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma nsqn_sqn: "nsqn rt dip ≤ sqn rt dip"
unfolding sqn_def nsqn_def by (cases "rt dip") auto

lemma val_nsqn_sqn [elim, simp]:
assumes "ip∈kD(rt)"
and "the (flag rt ip) = val"
shows "nsqn rt ip = sqn rt ip"
using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
assumes "ip∈vD(rt)"
shows "nsqn rt ip = sqn rt ip"
proof -
from ip∈vD(rt) have "ip∈kD(rt)"
and "the (flag rt ip) = val" by auto
thus ?thesis ..
qed

lemma inv_nsqn_sqn [elim, simp]:
assumes "ip∈kD(rt)"
and "the (flag rt ip) = inv"
shows "nsqn rt ip = sqn rt ip - 1"
using assms unfolding nsqn_sqn_def by auto
lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
proof -
  from \{ip ∈ iD(rt)\} have "ip ∈ kD(rt)"
    and "the (flag rt ip) = inv" by auto
thus ?thesis ..
qed

lemma nsqn_update_changed_kno_val [simp]: "∀ rt ip dsn dsk hops nhip.
  rt ≠ update rt ip (dsn, kno, val, hops, nhip) ⇒ nsqn (update rt ip (dsn, kno, val, hops, nhip)) ip = dsn"
unfolding nsqn_def update_def
by (clarsimp simp: kD_nsqn split: option.split_asm option.split if_split_asm)
  (metis fun_upd_triv)

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip rt ip
  assumes "dip ≠ ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip)) dip = nsqn rt dip"
using assms unfolding nsqn_def
by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
  assumes "dip ∈ kD(rt)"
  and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
using assms
proof -
  from assms obtain dsk hops nhip pre
    where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip)"
    unfolding invalidate_def
    by auto
  moreover from \{dip ∈ kD(rt)\} have "dip ∈ kD(invalidate rt dests)" by simp
  ultimately show ?thesis
    using \{dests dip = Some rsn\} by simp
qed

lemma nsqn_invalidate_other [simp]:
  assumes "dip∈kD(rt)"
  and "dip∉dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
using assms by (clarsimp simp add: kD_nsqn)

3.5.2 Comparing routes

definition
  fresher :: "r ⇒ r ⇒ bool" ("<_/ ⊑ _") [51, 51] 50
where
  "fresher r r' ≡ (nsqn r < nsqn r') ∨ (nsqn r = nsqn r' ∧ π₅(r) ≥ π₅(r'))"

lemma fresherI1 [intro]:
  assumes "nsqn r < nsqn r'"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
  assumes "nsqn r = nsqn r'" 
  and "π₅(r) ≥ π₅(r')"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI [intro]:
  assumes "(nsqn r < nsqn r') ∨ (nsqn r = nsqn r' ∧ π₅(r) ≥ π₅(r'))"
shows "r ⊆ r'"
unfolding fresher_def using assms.

lemma fresherE [elim]:
  assumes "r ⊆ r'"
  and "nsqn_r < nsqn_r' \Rightarrow P r r'"
  and "nsqn_r = nsqn_r' \land \pi_5(r) \geq \pi_5(r') \Rightarrow P r r'"
  shows "P r r'"
using assms unfolding fresher_def by auto

lemma fresher_refl [simp]: "r ⊆ r"
unfolding fresher_def by simp

lemma fresher_trans [elim, trans]:
  "[x ⊆ y; y ⊆ z] \Rightarrow x ⊆ z"
unfolding fresher_def by auto

lemma not_fresher_trans [elim, trans]:
  "[¬(x ⊆ y); ¬(z ⊆ x)] \Rightarrow ¬(z ⊆ y)"
unfolding fresher_def by auto

lemma fresher_dsn_flag_hops_const [simp]:
  fixes dsn dsd dsd' flag hops nhip nhip'
  shows "(dsn, dsd, flag, hops, nhip) ⊑ (dsn, dsd', flag, hops, nhip')"
unfolding fresher_def by (cases flag) simp_all

3.5.3 Comparing routing tables

definition
  rt_fresher :: "ip \Rightarrow rt \Rightarrow rt \Rightarrow bool"
where
  "rt_fresher ≡ λdip rt rt'. (the (σ_route(rt, dip))) ⊑ (the (σ_route(rt', dip)))"
abbreviation
  rt_fresher_syn :: "rt \Rightarrow ip \Rightarrow rt \Rightarrow bool" ("(_/ ⊑ _/ _)"
  [51, 999, 51] 50)
where
  "rt1 ⊑_i rt2 ≡ rt_fresher i rt1 rt2"

lemma rt_fresher_def':
  "(rt1 ⊑_i rt2) = (nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i)) \lor
    nsqn_r (the (rt1 i)) = nsqn_r (the (rt2 i)) \land \pi_5 (the (rt2 i)) \leq \pi_5 (the (rt1 i)))"
unfolding rt_fresher_def fresher_def by (rule refl)

lemma single_rt_fresher [intro]:
  assumes "the (rt1 ip) ⊑ the (rt2 ip)"
  shows "rt1 ⊑_ip rt2"
using assms unfolding rt_fresher_def.

lemma rt_fresher_single [intro]:
  assumes "rt1 ⊑_ip rt2"
  shows "the (rt1 ip) ⊑ the (rt2 ip)"
using assms unfolding rt_fresher_def.

lemma rt_fresher_def2: 
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  shows "(rt1 ⊑_dip rt2) = (nsqn rt1 dip < nsqn rt2 dip \lor
    (nsqn rt1 dip = nsqn rt2 dip \land the (dhops rt1 dip) \geq the (dhops rt2 dip)))"
using assms unfolding rt_fresher_def by (simp add: kD_nsqn proj5_eq_dhops)

lemma rt_fresherI1 [intro]:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"

280
and "nsqn rt1 dip < nsqn rt2 dip"
shows "rt1 ⊑ dip rt2"
unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherI2 [intro]:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "nsqn rt1 dip = nsqn rt2 dip"
and "the (dhops rt1 dip) ≥ the (dhops rt2 dip)"
shows "rt1 ⊑ dip rt2"
unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt_fresherE [elim]:
assumes "rt1 ⊑ dip rt2"
and "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "[ nsqn rt1 dip < nsqn rt2 dip ] ⇒ P rt1 rt2 dip"
and "[ nsqn rt1 dip = nsqn rt2 dip; the (dhops rt1 dip) ≥ the (dhops rt2 dip) ] ⇒ P rt1 rt2 dip"
shows "P rt1 rt2 dip"
using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "rt ⊑ dip rt"
unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
assumes "rt1 ⊑ dip rt2"
and "rt2 ⊑ dip rt3"
shows "rt1 ⊑ dip rt3"
using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
assumes "the (rt dip) ⊑ r"
shows "rt ⊑ dip (λip. if ip = dip then Some r else rt ip)"
using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip ⇒ rt ⇒ rt ⇒ bool"
where "rt_fresh_as ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ (rt2 ⊑ dip rt1)"

abbreviation rt_fresh_as_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/≈_)") [51, 999, 51] 50
where "rt1 ≈ i rt2 ≡ rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: "∀ rt dip. rt ≈ dip rt"
unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
"∀ rt1 rt2 rt3 dip. [ rt1 ≈ dip rt2; rt2 ≈ dip rt3 ] ⇒ rt1 ≈ dip rt3"
unfolding rt_fresh_as_def rt_fresher_def
by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
assumes "rt1 ⊑ dip rt2"
and "rt2 ⊑ dip rt1"
shows "rt1 ≈ dip rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "the (rt1 dip) ⊑ the (rt2 dip)"
and "the (rt2 dip) ⊑ the (rt1 dip)"
shows "rt1 ≈ dip rt2" using assms unfolding rt_fresh_as_def
by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI: assumes "dip ∈ kD(rt)"
and "dip ∈ kD(rt')"
and "nsqn rt dip = nsqn rt' dip"
and "Π₅(π₅(rt dip)) = Π₅(π₅(rt' dip))"
shows "rt ≈ dip rt'"

proof from assms(1-2,4) have dhops': "the (dhops rt' dip) ≤ the (dhops rt dip)"
by (simp add: proj5_eq_dhops)
with assms(1-3) show "rt ⊑ dip rt'"
by (rule rt_fresherI2)
next from assms(1-2,4) have dhops: "the (dhops rt dip) ≤ the (dhops rt' dip)"
by (simp add: proj5_eq_dhops)
with assms(2,1) assms(3) [symmetric] show "rt' ⊑ dip rt"
by (rule rt_fresherI2)

qed

lemma rt_fresh_asE [elim]: assumes "rt1 ≈ dip rt2"
and "[ rt1 ⊑ dip rt2; rt2 ⊑ dip rt1 ] ⇒ P rt1 rt2 dip"
shows "P rt1 rt2 dip" using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]: assumes "rt1 ≈ dip rt2"
shows "rt1 ⊑ dip rt2" using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]: assumes "rt1 ≈ dip rt2"
shows "rt2 ⊑ dip rt1" using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym: assumes "rt1 ≈ dip rt2"
shows "rt2 ≈ dip rt1" using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]: assumes "¬ (rt1 ⊑ dip rt2)"
shows "¬ (rt1 ≈ dip rt2)"

proof assume "rt1 ≈ dip rt2"
  hence "rt1 ⊑ dip rt2" ..
  with ⟨¬ (rt1 ⊑ dip rt2)⟩ show False ..

qed

lemma not_rt_fresh_asI2 [intro]: assumes "¬ (rt2 ⊑ dip rt1)"
shows "¬ (rt1 ≈ dip rt2)"

proof assume "rt1 ≈ dip rt2"
  hence "rt2 ⊑ dip rt1" ..
  with ⟨¬ (rt2 ⊑ dip rt1)⟩ show False ..

qed

lemma not_single_rt_fresher [elim]: assumes "¬(the (rt1 ip) ⊑ the (rt2 ip))"

282
shows "¬(rt1 ⊑ ip rt2)"
proof
  assume "rt1 ⊑ ip rt2"
  hence "the (rt1 ip) ⊑ the (rt2 ip)" ..
  with ¬(the (rt1 ip) ⊑ the (rt2 ip)) show False ..
qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
  assumes "¬(rt1 ⊑ ip rt2)"
  shows "¬(the (rt1 ip) ⊑ the (rt2 ip))"
proof
  assume "the (rt1 ip) ⊑ the (rt2 ip)"
  hence "rt1 ⊑ ip rt2" ..
  with ¬(rt1 ⊑ ip rt2) show False ..
qed

lemma rt_fresh_as_nsqnr:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "rt1 ≈ dip rt2"
  shows "nsqn (the (rt2 dip)) = nsqn (the (rt1 dip))"
using assms(3)
unfolding rt_fresh_as_def
by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨dip ∈ kD(rt2)⟩]
kD_nsqn [OF ⟨dip ∈ kD(rt1)⟩]
kD_nsqn [OF ⟨dip ∈ kD(rt2)⟩])

lemma rt_fresher_mapupd [intro!]:
  assumes "dip ∈ kD(rt)"
  and "the (rt dip) ⊑ r"
  shows "rt ⊑ dip rt(dip ↦→ r)"
using assms
unfolding rt_fresher_def
by simp

lemma rt_fresher_map_update_other [intro!]:
  assumes "dip ∈ kD(rt)"
  and "dip ≠ ip"
  shows "rt ⊑ dip update rt ip r"
using assms
unfolding rt_fresher_def
by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip ∈ kD(rt)"
  and "dip ≠ ip"
  shows "rt ⊑ dip update rt ip r"
using assms
unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip ∈ kD(rt)"
  and "the (dhops rt dip) ≥ 1"
  and "update_arg_wf r"
  shows "rt ⊑ dip update rt ip r"
proof (cases "dip = ip")
  assume "dip ≠ ip" with ⟨dip ∈ kD(rt)⟩ show ?thesis
  by (rule rt_fresher_update_other)
next
  assume "dip = ip"
  from ⟨dip ∈ kD(rt)⟩ obtain dsn n dsk n hops n nhip n
      where rtn [simp]: "the (rt dip) = (dsn n, dsk n, f n, hops n, nhip n)"
      by (metis prod_cases5)
  with ⟨the (dhops rt dip) ≥ 1⟩ and ⟨dip ∈ kD(rt)⟩ have "hops n ≥ 1"
by (metis proj5_eq_dhops proj4(4))
from ⟨dip∈kD(rt)⟩ have [simp]: "sqn rt dip = dsn_n" and [simp]: "the (dhops rt dip) = hops_n" and [simp]: "the (flag rt dip) = f_n" by (simp add: sqn_def proj5_eq_dhops [symmetric] proj4_eq_flag [symmetric])

from ⟨update_arg_wf r⟩ have "(dsn_n, dsk_n, f_n, hops_n, nhip_n) ⊑ the ((update rt dip r) dip)"
proof (rule wf_r_cases)
  fix nhip pre
  from ⟨hops_n ≥ 1⟩ have "∀ pre'. (dsn_n, dsk_n, f_n, hops_n, nhip_n) ⊑ (dsn_n, unk, val, Suc 0, nhip)"
    unfolding fresher_def sqn_def by (cases f_n) auto
  thus "(dsn_n, dsk_n, f_n, hops_n, nhip_n) ⊑ the (update rt dip (0, unk, val, Suc 0, nhip) dip)"
    using ⟨dip∈kD(rt)⟩ by - (rule update_cases_kD, simp_all)
next
  fix dsn :: sqn and hops nhip pre
  assume "0 < dsn" and "hops < hops_n"
  thus "(dsn_n, dsk_n, f_n, hops_n, nhip_n) ⊑ (dsn, kno, val, hops, nhip)"
    unfolding fresher_def nsqn_r_def by simp
next
  assume "dsn_n = dsn" with ⟨0 < dsn⟩ show "(dsn, dsk_n, inv, hops_n, nhip_n) ⊑ (dsn, kno, val, hops, nhip)"
    unfolding fresher_def by simp
qed

hence "rt ⊑ dip update rt dip r"
  by - (rule single_rt_fresher, simp)
  with ⟨dip = ip⟩ show ?thesis by simp
qed

theorem rt_fresher_invalidate [simp]:
  assumes "dip∈kD(rt)"
  and indests: "∀ rip∈dom(dests). rip∈vD(rt) ∧ sqn rt rip < the (dests rip)"
  shows "rt ⊑ dip invalidate rt dests"
proof (cases "dip∈dom(dests)"
  assume "dip∉dom(dests)"
  thus ?thesis using ⟨dip∈kD(rt)⟩ by - (rule single_rt_fresher, simp)
next
  assume "dip∈dom(dests)"
  moreover with indests have "dip∈vD(rt)" and "sqn rt dip < the (dests dip)"
    by auto
  ultimately show ?thesis
    unfolding invalidate_def sqn_def by - (rule single_rt_fresher, auto simp: fresher_def)
qed
lemma nsqn_r_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
  and "dip ∈ dom(dests)"
  shows "nsqn_r (the (invalidate rt dests dip)) = the (dests dip) - 1"
  using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
  and "∀ rip ∈ dom(dests). rip ∈ VD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
  shows "rt ≈ dip invalidate rt dests"
  proof
    (cases "dip ∈ dom(dests)"
      assume "dip /∈ dom(dests)"
      with ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)"
        by simp
      with ⟨dip ∈ kD(rt)⟩ show ?thesis
        by rule (simp_all add: ⟨dip /∈ dom(dests)⟩)
    next
      assume "dip ∈ dom(dests)"
      with assms(2) have "dip ∈ kD(rt)" by simp
      moreover then have "dip ∈ kD(invalidate rt dests)" by simp
      ultimately show ?thesis
      proof
        (rule nsqn_rt_fresh_asI)
        from ⟨dip ∈ kD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
        also have "sqn rt dip = nsqn (invalidate rt dests) dip"
          proof
            from ⟨dip ∈ kD(invalidate rt dests)⟩ have "nsqn_r (the (invalidate rt dests dip)) = the (dests dip) - 1"
              using ⟨dip ∈ dom(dests)⟩ by (rule nsqn_r_invalidate)
            with ⟨the (dests dip) = inc (sqn rt dip)⟩ show "sqn rt dip = nsqn_r (the (invalidate rt dests dip))" by simp
          qed
          also from ⟨dip ∈ kD(invalidate rt dests)⟩ have "nsqn_r (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
            by (simp add: kD_nsqn)
          finally show "nsqn rt dip = nsqn (invalidate rt dests) dip" .
        qed
      qed
    qed

lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]

3.5.4 Strictly comparing routing tables

definition rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
  "rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ ¬(rt2 ⊑ dip rt1)"

abbreviation
  rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊏ _)" [51, 999, 51] 50)
where
  "rt1 ⊏ i rt2 ≡ rt_strictly_fresher i rt1 rt2"

lemma rt_strictly_fresher_def':
  "rt1 ⊏ i rt2 = ((rt1 ⊑ i rt2) ∧ ¬(rt2 ⊑ i rt1))"
  unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

lemma rt_strictly_fresherI' [intro]:
  assumes "rt1 ⊑ i rt2"
  and "¬(rt2 ⊑ i rt1)"
  shows "rt1 ⊏ i rt2"
  using assms unfolding rt_strictly_fresher_def' by simp

lemma rt_strictly_fresherE' [elim]:
  assumes "rt1 ⊏ i rt2"
  shows "rt1 ⊑ i rt2"
and "[ rt1 ⊆₁ rt2; ¬(rt2 ⊆₁ rt1) ] ⇒ P rt1 rt2 i"
shows "P rt1 rt2 i"
using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherI [intro]:
assumes "rt1 ⊑ i rt2"
and "¬(rt1 ≈ i rt2)"
shows "rt1 ⊏ i rt2"
unfolding rt_strictly_fresher_def using assms ..

lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]

lemma rt_strictly_fresherE [elim]:
assumes "rt1 ⊏ i rt2"
and "[ [ rt1 ⊑ i rt2; ¬(rt1 ≈ i rt2) ] ⇒ P rt1 rt2 i ]" 
shows "P rt1 rt2 i"
using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
"rt1 ⊏ i rt2 =
(nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i))
∨ (nsqn_r (the (rt1 i)) = nsqn_r (the (rt2 i)) ∧ π_5(the (rt1 i)) > π_5(the (rt2 i))))"
unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
assumes "rt1 ⊏ dip rt2"
shows "the (rt1 dip) ⊑ the (rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
assumes "rt1 ⊏ dip rt2"
shows "¬ rt1 ≈ dip rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1 ⊏ dip rt2"
and "rt2 ⊏ dip rt3"
shows "rt1 ⊏ dip rt3"
using assms proof -
from rt1 ⊏ dip rt2: obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
also from rt2 ⊏ dip rt3: obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
finally have "the (rt1 dip) ⊑ the (rt3 dip)" .
moreover have "¬ (rt1 ≈ dip rt3)"
proof -
from rt1 ⊏ dip rt2: obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
also from rt2 ⊏ dip rt3: obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))" .
thus ?thesis ..
qed
ultimately show "rt1 ⊏ dip rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏ dip rt)"
unfolding rt_strictly_fresher_def
by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1 ⊏ dip rt2"
and "rt2 ⊏ dip rt3"
shows "rt1 ⊏ dip rt3"
proof -
from rt1 ⊏ dip rt2: have "rt1 ⊆ dip rt2"
and \((\neg (rt2 \sqsubseteq dip rt1))\)
unfolding \(rt\text{\textunderscore}\text{\textunderscore}\text{\textunderscore}fresher\text{\textunderscore}def'\) by auto
from this(1) and \((rt2 \sqsubseteq dip rt3)\) have "\(rt1 \sqsubseteq dip rt3\) ..
moreover from \((\neg (rt2 \sqsubseteq dip rt1))\) have "\((\neg (rt3 \sqsubseteq dip rt1))\)
proof (rule contrapos_nn)
assume "rt3 \sqsubseteq dip rt1"
with \((rt2 \sqsubseteq dip rt3)\) show "rt2 \sqsubseteq dip rt1" ..
qed
ultimately show "rt1 \sqsubseteq dip rt3"
unfolding \(rt\text{\textunderscore}\text{\textunderscore}\text{\textunderscore}fresher\text{\textunderscore}def'\) by auto
qed

lemma \(rt\text{\textunderscore}fresher\text{\textunderscore}trans\text{\textunderscore}rt\text{\textunderscore}strictly\text{\textunderscore}fresher'\) [elim, trans]:
assumes "rt1 \sqsubseteq dip rt2"
and "rt2 \sqsubseteq dip rt3"
shows "rt1 \sqsubseteq dip rt3"
proof -
from \((rt2 \sqsubseteq dip rt3)\) have "rt2 \sqsubseteq dip rt3"
and "\((\neg (rt3 \sqsubseteq dip rt2))\)"
unfolding \(rt\text{\textunderscore}\text{\textunderscore}\text{\textunderscore}fresher\text{\textunderscore}def'\) by auto
from \((rt1 \sqsubseteq dip rt2)\) and this(1) have "rt1 \sqsubseteq dip rt3" ..
moreover from \((\neg (rt3 \sqsubseteq dip rt2))\) have "\((\neg (rt3 \sqsubseteq dip rt1))\)
proof (rule contrapos_nn)
assume "rt3 \sqsubseteq dip rt1"
thus "rt3 \sqsubseteq dip rt2" using \((rt1 \sqsubseteq dip rt2)\) ..
qed
ultimately show "rt1 \sqsubseteq dip rt3"
unfolding \(rt\text{\textunderscore}\text{\textunderscore}\text{\textunderscore}fresher\text{\textunderscore}def'\) by auto
qed

lemma \(rt\text{\textunderscore}fresher\text{\textunderscore}imp\text{\textunderscore}nsqn\text{\textunderscore}le\):
assumes "rt1 \sqsubseteq dip rt2"
and "ip \in kD rt1"
and "ip \in kD rt2"
shows "nsqn rt1 ip \leq nsqn rt2 ip"
using assms(1)
by (auto simp add: rt\text{\textunderscore}fresher\text{\textunderscore}def2 [OF assms(2-3)])

lemma \(rt\text{\textunderscore}strictly\text{\textunderscore}fresher\text{\textunderscore}ltI\) [intro]:
assumes "dip \in kD rt1"
and "dip \in kD rt2"
and "nsqn rt1 dip < nsqn rt2 dip"
shows "rt1 \sqsubseteq dip rt2"
proof
from assms show "rt1 \sqsubseteq dip rt2" ..
next
show "\((\neg (rt1 \approx dip rt2))\)"
proof
assume "rt1 \approx dip rt2"
thus "rt2 \sqsubseteq dip rt1" ..
hence "nsqn rt2 dip \leq nsqn rt1 dip"
using \((dip \in kD rt2)\) \((dip \in kD rt1)\)
by (rule rt\text{\textunderscore}fresher\text{\textunderscore}imp\text{\textunderscore}nsqn\text{\textunderscore}le)
with \((\neg (rt1 \approx dip rt2))\) show "False"
by simp
qed
qed

lemma \(rt\text{\textunderscore}strictly\text{\textunderscore}fresher\text{\textunderscore}eqI\) [intro]:
assumes "i \in kD rt1"
and "i∈kD(rt2)"
and "nsqn rt1 i = nsqn rt2 i"
and "π5(the (rt2 i)) < π5(the (rt1 i))"
shows "rt1 ⊏i rt2"

using assm unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]:
"\ destinets dip rt rt'. destinets dip = None \Longrightarrow (invalidate rt destinets ⊏ dip rt') = (rt ⊏ dip rt')"
unfolding invalidate_def rt_strictly_fresher_def'
by (rule iffI) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
assumes "dip ∈ vD(invalidate rt1 destinets)"
shows "(invalidate rt1 destinets ⊏ dip rt2) = (rt1 ⊏ dip rt2)"
proof (cases "dip ∈ dom(destinets)")
  assume "dip ∈ dom(destinets)"
  hence "dip /∈ vD(invalidate rt1 destinets)"
  unfolding invalidate_def vD_def
  by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
  with ⟨dip ∈ vD(invalidate rt1 destinets)⟩
  show ?thesis by simp
  next
  assume "dip /∈ dom(destinets)"
  hence "destinets dip = None"
  by auto
  moreover with ⟨dip ∈ vD(invalidate rt1 destinets)⟩
  have "dip ∈ vD(rt1)"
  unfolding invalidate_def vD_def
  by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
  ultimately show ?thesis
  unfolding invalidate_def rt_strictly_fresher_def'
  by auto
  qed

lemma rt_strictly_fresher_update_other [elim!]:
"\ dip ip rt r rt'. \[ \[ dip \neq ip; rt ⊏ dip rt' \] \] = \Longrightarrow update rt ip r ⊏ dip rt'"
unfolding rt_strictly_fresher_def'
by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
assumes "dip ∈ vD (rt2 nhip)"
and *: "osn < sqn (rt2 nhip) dip"
and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip)"
shows "update rt dip (osn, kno, val, hops, nhip) ⊏ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
  from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip)) dip = osn"
  by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
  have "nsqn, (the (update rt dip (osn, kno, val, hops, nhip)) dip) = osn"
  by (simp add: kD_nsqn)
  also have "osn < sqn (rt2 nhip) dip" by (rule *)
  also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
  unfolding nsqn_def using ⟨dip ∈ vD (rt2 nhip)⟩
  by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_id_gives_kD(1))
  finally show "nsqn, (the (update rt dip (osn, kno, val, hops, nhip) dip))
  < nsqn, (the (rt2 nhip dip))".
  qed

lemma dhops_le_hops_imp_update_strictly_fresher:
assumes "dip ∈ vD (rt2 nhip)"
and sqn: "sqn (rt2 nhip) dip = osn"
and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip)"
shows "update rt dip (osn, kno, val, Suc hops, nhip) ⊏ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
  from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip)) dip = osn"
  by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
  have "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip)) dip) = osn"
  by (simp add: kD_nsqn)
  also have "osn < sqn (rt2 nhip) dip" by (rule *)
  also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
  unfolding nsqn_def using ⟨dip ∈ vD (rt2 nhip)⟩
  by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_id_gives_kD(1))
  finally show "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip) dip))
  < nsqn, (the (rt2 nhip dip))".
  qed
with \( \langle \text{dip} \in vD(rt2 nhip) \rangle \)
have "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip) dip)) = osn"
  by (simp add: kD_nsqn)
also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
  unfolding nsqn_def using \( \langle \text{dip} \in vD(rt2 nhip) \rangle \)
by (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip) dip))
  = nsqn, (the (rt2 nhip dip)).
next
have "the (dhops (rt2 nhip) dip) \leq hops" by (rule hop)
also have "hops < hops + 1" by simp
finally have "the (dhops (rt2 nhip) dip)
  < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip)) dip)."
thus "\( \pi_5 \) (the (rt2 nhip dip)) < \( \pi_5 \) (the (update rt dip (osn, kno, val, Suc hops, nhip) dip))"
  using \( \langle \text{dip} \in vD(rt2 nhip) \rangle \) by (simp add: proj5_eq_dhops)
qed

lemma nsqn_invalidate:
assumes "\( \text{dip} \in kD(rt) \)"
  and "\( \forall ip \in \text{dom(dests)}. ip \in vD(rt) \land the (dests ip) = inc (sqn rt ip) \)"
shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
proof -
  from assms have "rt \approx dip in validate rt dests"
    by (rule rt_fresh_as_inc_invalidate)
  with \( \langle \text{dip} \in kD(rt) \rangle \) show ?thesis
    by (simp add: kD_nsqn del: invalidate_kD_inv)
  (erule(2) rt_fresh_as_nsqnr)
qed

3.6 Invariant proofs on individual processes

theory C_Seq_Invariants
imports AWN.Invariants C_Aodv C_Aodv_Data C_Aodv_Predicates C_Fresher
begin

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

lemma sequence_number_increases:
  "paodv i \|= \_ onl1 I_AODV (\lambda((\xi, _), _, (\xi', _)). \text{sn } \xi \leq \text{sn } \xi')"
  by inv_cterms

lemma sequence_number_one_or_bigger:
  "paodv i \|= onl I_AODV (\lambda(\xi, _). 1 \leq \text{sn } \xi)"
  by (rule onl1_step_to_invariantI [OF sequence_number_increases])
    (auto simp: \sigma_AODV_def)

We can get rid of the onl/onll if desired...

lemma sequence_number_increases':
  "paodv i \|= A (\lambda((\xi, _), _, (\xi', _)). \text{sn } \xi \leq \text{sn } \xi')"
  by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

lemma sequence_number_one_or_bigger':
  "paodv i \|= (\lambda(\xi, _). 1 \leq \text{sn } \xi)"
  by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

end
lemma sip_in_kD: 
"\(\text{paodv i \|= onl } \Gamma_{AODV} (\lambda(\xi, 1). 1 \in \{\text{PAodv-:7} \cup \{\text{PAodv-:5} \cup \{\text{PRrep-:0..PRrep-:1} \cup \{\text{PRreq-:0..PRreq-:3}} \rightarrow \text{sip } \xi \in kD (rt \xi)\))\)"
by inv_cterms

lemma rrep_1_update_changes: 
"\(\text{paodv i \|= onl } \Gamma_{AODV} (\lambda(\xi, 1). (1 = \text{PRrep-:1} \rightarrow rt \xi \neq \text{update (rt } \xi) (\text{dsn } \xi, \text{kno, val, hops } \xi + 1, \text{sip } \xi)))\)"
by inv_cterms

Proposition 7.38

lemma includes_nhip: 
"\(\text{paodv i \|= onl } \Gamma_{AODV} (\lambda(\xi, l). \forall dip \in kD(rt \xi). \text{nhip (rt } \xi) dip \in kD(rt \xi))\)"
proof -
{ fix ip and \(\xi, \xi'::\text{state} \)
assume "\((\forall dip \in kD(rt \xi). \text{the (nhop (rt } \xi) dip \in kD(rt \xi)\) and "\(\xi' = \xi\{(rt := \text{update (rt } \xi) ip (0, unk, val, Suc 0, ip))\)"

hence "\((\forall dip \in kD(rt \xi). \text{the (nhop (update (rt } \xi) ip (0, unk, val, Suc 0, ip)) dip = ip \lor the (nhop (update (rt } \xi) ip (0, unk, val, Suc 0, ip)) dip) \in kD(rt \xi)\)"
by clarsimp (metis nhop_update_unk_val update_another)
}
ote one_hop = this
{ fix ip sip sn hops and \(\xi, \xi'::\text{state} \)
assume "\((\forall dip \in kD(rt \xi). \text{the (nhop (rt } \xi) dip \in kD(rt \xi)\) and "\(\xi' = \xi\{(rt := \text{update (rt } \xi) ip (sn, kno, val, Suc hops, sip))\)"

and "\(\text{sip } \in kD(rt \xi)\)"

hence "\((\forall dip \in kD(rt \xi). \text{the (nhop (update (rt } \xi) ip (sn, kno, val, Suc hops, sip)) dip = ip \lor the (nhop (update (rt } \xi) ip (sn, kno, val, Suc hops, sip)) dip) \in kD(rt \xi)\)"
by (metis kD_update_unchanged nhop_update_changed update_another)
}
ote nhip_is_sip = this
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf sip_in_kD]
solve: one_hop nhip_is_sip)
qed

Proposition 7.4

lemma known_destinations_increase: 
"\(\text{paodv i \|= onl } \Gamma_{AODV} (\lambda((\xi, _), (_, (\xi', _))). kD (rt \xi) \subseteq kD (rt \xi'))\)"
by (inv_cterms simp add: subset_insertI)

Proposition 7.5

lemma rreqs_increase: 
"\(\text{paodv i \|= onl } \Gamma_{AODV} (\lambda((\xi, _), (_, (\xi', _))). \text{rreqs } \xi \subseteq \text{rreqs } \xi')\)"
by (inv_cterms simp add: subset_insertI)

lemma dests_bigger_than_sqn: 
"\(\text{paodv i \|= onl } \Gamma_{AODV} (\lambda(\xi, 1). 1 \in \{\text{PAodv-:15..PAodv-:17} \cup \{\text{PPkt-:7..PPkt-:9} \cup \{\text{PRreq-:9..PRreq-:11} \cup \{\text{PRreq-:17..PRreq-:19} \cup \{\text{PRrep-:8..PRrep-:10} \cup \{\text{PRrep-:1..PRrep-:4} \cup \{\text{PRrep-:6} \rightarrow (\forall ip \in dom(dests \xi). ip \in kD(rt \xi) \land \text{sqn (rt } \xi) ip \leq \text{the (dests } \xi\ ip))\)\)"
proof -
have sqninv:
"\(\forall ip \in dom(dests \xi). ip \in kD(rt \xi) \land \text{sqn (rt } \xi) ip \leq \text{the (dests } \xi\ ip)\)"
by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
have indeq:
"\(\forall ip \in dom(dests \xi). ip \in kD(rt \xi) \land \text{sqn (rt } \xi) ip \leq \text{the (dests } \xi\ ip)\)"

∀ \( \text{ip} \in \text{dom(dests)}. \text{ip} \in kD(rt) \land \text{sqn rt ip} \leq \text{the (dests ip)} \) \\
\implies \text{ip} \in kD(rt) \land \text{sqn rt ip} \leq \text{rsn} \\
by (metis \text{domI option.sel})

show \(?\text{thesis}\) 
by \text{inv_cterms} 
(clarsimp split: if_split_asm option.split_asm 
  elim!: sqninv indests)+

qed

Proposition 7.6

lemma sqns_increase:
"\text{paodv i} \models_{A} \text{onll} \Gamma_{AODV} (\lambda((\xi, \_), (\xi', \_)). \forall \text{ip}. \text{sqn (rt } \xi \text{) ip} \leq \text{sqn (rt } \xi' \text{) ip})"
proof - 
{ fix \(\xi \:: \text{state} \)
  assume \(*\): "\forall \text{ip} \in \text{dom(dests } \xi). \text{ip} \in kD (rt \xi) \land \text{sqn (rt } \xi \text{) ip} \leq \text{the (dests } \xi \text{) ip}"
  have "\forall \text{ip}. \text{sqn (rt } \xi \text{) ip} \leq \text{sqn (invalidate (rt } \xi \text) (dests } \xi\text{)) ip}"
  proof 
    fix \text{ip} 
    from \(*\) have "\text{ip} \notin \text{dom(dests } \xi) \lor \text{sqn (rt } \xi \text{) ip} \leq \text{the (dests } \xi \text{ ip)}" 
    by simp 
    thus "\text{sqn (rt } \xi \text{) ip} \leq \text{sqn (invalidate (rt } \xi \text) (dests } \xi\text{)) ip}"
    by (metis \text{domI invalidate_sqn option.sel})
  qed 
  note \text{solve_invalidate} = this 
  show \(?\text{thesis}\) 
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf \text{dests_bigger_than_sqn}] 
    simp add: \text{solve_invalidate})
  qed

Proposition 7.7

lemma ip_constant:
"\text{paodv i} \models_{A} \text{onll} \Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{ip } \xi = i)"
by (inv_cterms simp add: \text{σ}_{AODV} \_\text{def})

Proposition 7.8

lemma sender_ip_valid':
"\text{paodv i} \models_{A} \text{onll} \Gamma_{AODV} ((\lambda((\xi, \_), a, \_). \text{anycast (λm. not_Pkt m } \rightarrow \text{msg_sender m = ip } \xi \text{) a})"
by inv_cterms

lemma sender_ip_valid:
"\text{paodv i} \models_{A} \text{onll} \Gamma_{AODV} ((\lambda((\xi, \_), a, \_). \text{anycast (λm. not_Pkt m } \rightarrow \text{msg_sender m = i}) a)"
by (rule step_invariant_weaken_with_invariantE [OF \text{ip_constant} sender_ip_valid']) 
(auto \text{dest!: onlD onllD})

lemma received_msg_inv:
"\text{paodv i} \models (\text{recvmsg P } \rightarrow) \text{onll} \Gamma_{AODV} (\lambda((\xi, 1). \text{l } \in \text{PAdv}:-1 } \rightarrow \text{P (msg } \xi\text{)"
by inv_cterms

Proposition 7.9

lemma sip_not_ip':
"\text{paodv i} \models (\text{recvmsg (λm. not_Pkt m } \rightarrow \text{msg_sender m } \neq i) } \rightarrow \text{onll} \Gamma_{AODV} (\lambda((\xi, \_). \text{ sip } \xi \neq ip \xi)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
  onl_invariant_sterms [OF aodv_wf \text{ip_constant} \_THEN invariant_restrict_inD]) 
  simp add: clear_locals_sip_not_ip' clarsimp+

lemma sip_not_ip:
"\text{paodv i} \models (\text{recvmsg (λm. not_Pkt m } \rightarrow \text{msg_sender m } \neq i) } \rightarrow \text{onll} \Gamma_{AODV} (\lambda((\xi, \_). \text{ sip } \xi \neq i)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
  onl_invariant_sterms [OF aodv_wf \text{ip_constant} \_THEN invariant_restrict_inD]) 
  simp add: clear_locals_sip_not_ip' clarsimp+

Neither sip_not_ip' nor sip_not_ip is needed to show loop freedom.

Proposition 7.10
lemma hop_count_positive:
"paodv i |= onl \( \Gamma_{AODV} (\lambda(\xi, \_). \forall ip \in kD (rt \xi). \text{the (dhops (rt \xi) ip) } \geq 1)\)"
by (inv_cterms) auto

lemma rreq_dip_in_vD_dip_eq_ip:
"paodv i |= onl \( \Gamma_{AODV} (\lambda(\xi, \_). \forall ip \in kD (rt \xi). \text{the (dhops (rt \xi) ip)}) \geq 1)\)"
by (inv_cterms) auto

Proposition 7.11
lemma anycast_msg_zhops:
"\bigwedge rreqid dip dsn dsk oip osn sip. paodv i||\( \Gamma_{AODV} (\lambda(_, a, \_). \text{anycast msg}_zhops a)\)"
proof (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip [THEN invariant_restrict_inD]] onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]], elim conjE)
fix l \( \xi a pp p' pp' \)
assume "(\( \xi, pp \)) \in \text{reachable (paodv i) TT}"
and "\{PRreq-:14\}unicast(\( \lambda_\xi. \text{the (nhop (rt \xi) (oip \xi))}, \lambda_\xi. \text{Rrep (the (dhops (rt \xi) (dip \xi)) (sqn (rt \xi) (dip \xi))) (oip \xi) (ip \xi))}. p' \triangleq pp' \in \text{sterms } \Gamma_{AODV} pp"
and "l = PRreq-:14"
and "a = unicast (the (nhop (rt \xi) (oip \xi))) (Rrep (the (dhops (rt \xi) (dip \xi)))) (dip \xi) (sqn (rt \xi) (dip \xi)) (oip \xi) (ip \xi))"
and *: "\forall ip \in kD (rt \xi). \text{Suc 0 }\leq \text{the (dhops (rt \xi) ip)}"
and "dip \xi \in vD (rt \xi)"
from dip \( \xi \in vD (rt \xi)\) have "dip \( \xi \in kD (rt \xi)\)"
by (rule vD_iD_gives_kD(1))
with * have "Suc 0 \leq \text{the (dhops (rt \xi) (dip \xi))} ... thus "0 < \text{the (dhops (rt \xi) (dip \xi))}" by simp
qed

lemma hop_count_zero_oip_dip_sip:
"paodv i |= (recvmsg msg_zhops \(\rightarrow\)) onl \( \Gamma_{AODV} (\lambda(\xi, \_). \forall ip \in kD (rt \xi). \text{the (dhops (rt \xi) ip))} \geq 1\)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:
"paodv i |= (recvmsg \( \text{rreq_rrep_sn} \rightarrow\)) onl \( \Gamma_{AODV} (\lambda(\xi, \_). 1 \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-n\mid n. \text{True}\} \rightarrow \text{osn \( \xi \leq 1\)\})"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq':
"paodv i |= (recvmsg \( \lambda m. \text{rreq_rrep_sn} m \land \text{msg_zhops m} \rightarrow\)) onl \( \Gamma_{AODV} (\lambda(\xi, \_). 1 \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-n\mid n. \text{True}\} \rightarrow \text{osn \( \xi \leq 1\)\})"
proof (rule invariant_weakenE [OF osn_rreq])
fix a
assume "recvmsg \( \lambda m. \text{rreq_rrep_sn} m \land \text{msg_zhops m} \) a"
thus "recvmsg \( \lambda m. \text{rreq_rrep_sn} m \) a"
by (cases a) simp_all
qed

lemma dsn_rrep:
"paodv i |= (recvmsg \( \text{rreq_rrep_sn} \rightarrow\)) onl \( \Gamma_{AODV} (\lambda(\xi, \_). 1 \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-n\mid n. \text{True}\} \rightarrow \text{dsn \( \xi \leq 1\)\})"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp
\[ \text{proof (rule invariant_weakenE [OF dsn_rrep])} \]
fix a
assume "recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) a"
thus "recvmsg rreq_rrep_sn a"
by (cases a) simp_all
qed

\begin{enumerate}
\item \textbf{Proposition 7.12}
\item \textbf{lemma hop_count_zero_oip_dip_sip'}:
\[ \text{proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])} \]
\item \textbf{Proposition 7.12}
\item \textbf{lemma zero_seq_unk_hops_one'}:
\[ \text{proof} \]
\begin{enumerate}
\item \textbf{Proposition 7.12}
\item \textbf{lemma prreq_ok1 [simp] = this}
\end{enumerate}
\end{enumerate}
by (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)

{ fix ip dsn hops sip oip rt dip
  assume "\(\forall dip \in kD(rt). (sqn rt dip = 0 \rightarrow \pi_3 (\text{the (rt dip)}) = \text{unk}) \land
          (\pi_3 (\text{the (rt dip)}) = \text{unk} \rightarrow (\text{dhops rt dip}) = \text{Suc 0}) \land
          (\text{the (dhops rt dip)} = \text{Suc 0} \rightarrow (\text{nhop rt dip}) = \text{dip})""
  and "\(\text{Suc 0} \leq dsn""
  and "ip \neq dip \rightarrow ip \in kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip)) ip = 0 \rightarrow
          \pi_3 (\text{the (update rt dip (dsn, kno, val, Suc hops, sip) ip)}) = \text{unk}" by (rule update_cases, auto simp add: sqn_def dest!: bspec)
}

note prreq_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "\(\forall dip \in kD(rt). (sqn rt dip = 0 \rightarrow \pi_3 (\text{the (rt dip)}) = \text{unk}) \land
          (\pi_3 (\text{the (rt dip)}) = \text{unk} \rightarrow (\text{dhops rt dip}) = \text{Suc 0}) \land
          (\text{the (dhops rt dip)} = \text{Suc 0} \rightarrow (\text{nhop rt dip}) = \text{dip})""
  and "Suc 0 \leq dsn""
  and "ip \neq dip \rightarrow ip \in kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip)) ip = 0 \rightarrow
          \pi_3 (\text{the (update rt dip (dsn, kno, val, Suc hops, sip) ip)}) = \text{unk}" by (rule update_cases, auto simp add: sqn_def dest!: bspec)
}

note prreq_ok3 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "\(\forall dip \in kD(rt). (sqn rt dip = 0 \rightarrow \pi_3 (\text{the (rt dip)}) = \text{unk}) \land
          (\pi_3 (\text{the (rt dip)}) = \text{unk} \rightarrow (\text{dhops rt dip}) = \text{Suc 0}) \land
          (\text{the (dhops rt dip)} = \text{Suc 0} \rightarrow (\text{nhop rt dip}) = \text{dip})""
  and "Suc 0 \leq dsn""
  and "ip \neq dip \rightarrow ip \in kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip)) ip = 0 \rightarrow
          \pi_3 (\text{the (update rt dip (dsn, kno, val, Suc hops, sip) ip)}) = \text{unk}" by (rule update_cases, auto simp add: sqn_def dest!: bspec)
}

note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "\(\forall sip rt. \pi_3 (\text{the (update rt sip (0, unk, val, Suc 0, sip) sip)}) = \text{unk} \rightarrow
          \text{the (dhops (update rt sip (0, unk, val, Suc 0, sip) sip)} = \text{Suc 0}" by (rule update_cases) simp_all

have prreq_ok6 [simp]: "\(\forall sip rt. sqn (update rt sip (0, unk, val, Suc 0, sip)) sip = 0 \rightarrow
          \pi_3 (\text{the (update rt sip (0, unk, val, Suc 0, sip) sip)}) = \text{unk}" by (rule update_cases) simp_all

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip'] seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans] onl_invariant_sterms [OF aodv_wf osn_rreq'] onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+

qed

lemma zero_seq_unk_hops_one:
  "paodv i ||= (recvmsg (\(\lambda m. \text{rreq_rrep_sn m} \land \text{msg_zhops m}) \rightarrow onl \Gamma_{AODV} (\lambda (\xi, _). \forall dip \in kD(rt \xi) \cdot (sqn (rt \xi) dip = 0 \rightarrow (sqnf (rt \xi) dip) = \text{unk}
          \land \text{the (dhops (rt \xi) dip)} = \text{Suc 0} \land \text{the (nhop (rt \xi) dip)} = \text{dip}))""
  by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
  "paodv i ||= (recvmsg \text{rreq_rrep_sn} \rightarrow onl \Gamma_{AODV} (\lambda (\xi, _). \forall dip \in kD(rt \xi) . \pi_3 (\text{the (rt \xi) dip}) = \text{unk} \lor 1 \leq \pi_2 (\text{the (rt \xi) dip}))""
proof -
  { fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
    assume "dsk1 = unk \lor Suc 0 \leq dsn2"
    hence "\(\pi_3 (\text{the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1) sip)}) = \text{unk}
          \lor Suc 0 \leq sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2)) sip" unfolding update_def by (cases "dsk1 = unk") (clarsimp split: option.split)+
  } note fromsip [simp] = this

294
fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
assume allkd: "∀ dip ∈ kD(rt). π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip"
and **: "dsk1 = unk ∨ Suc 0 ≤ dsn2"
have "∀ dip ∈ kD(rt). π₃ (the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1) dip)) = unk
      ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2)) dip"
is "∀ dip ∈ kD(rt). ?prop dip")

proof
fix dip
assume "dip ∈ kD(rt)"
thus "?prop dip"
proof (cases "dip = sip")
  assume "dip = sip"
  with ** show ?thesis
  by simp
next
  assume "dip ≠ sip"
  with ⟨dip ∈ kD(rt)⟩ allkd show ?thesis
  by simp
qed
qed

note solve_update [simp] = this

fix dip rt dests
assume *: "∀ ip ∈ dom(dests). ip ∈ kD(rt) ∧ sqn rt ip ≤ the (dests ip)"
and **: "∀ ip ∈ kD(rt). π₃ (the (rt ip)) = unk ∨ Suc 0 ≤ sqn rt ip"
have "∀ dip ∈ kD(rt). π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"

proof
fix dip
assume "dip ∈ kD(rt)"
with ** have "π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip" ..
  thus "π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
  proof
    assume "π₃ (the (rt dip)) = unk" thus ?thesis ..
  next
    assume "Suc 0 ≤ sqn rt dip"
    have "Suc 0 ≤ sqn (invalidate rt dests) dip"
      proof (cases "dip ∈ dom(dests)")
        assume "dip ∈ dom(dests)"
        with * have "sqn rt dip ≤ the (dests dip)" by simp
        with ⟨Suc 0 ≤ sqn rt dip⟩ have "Suc 0 ≤ the (dests dip)" by simp
        with ⟨dip ∈ dom(dests)> ⟨dip ∈ kD(rt)> [THEN kd_Some] show ?thesis
          unfolding invalidate_def sqn_def by auto
      next
        assume "dip ∉ dom(dests)"
        with ⟨Suc 0 ≤ sqn rt dip⟩ ⟨dip ∈ kD(rt)> [THEN kd_Some] show ?thesis
          unfolding invalidate_def sqn_def by auto
    qed
    thus ?thesis by (rule disjI2)
  qed
qed

note solve_invalidate [simp] = this

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn
          [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aodv_wf osn_rreq]
    onl_invariant_sterms [OF aodv_wf dsn_rrep]
    simp add: proj3_inv proj2_eq_sqn)
qed

Proposition 7.13

lemma rreq_rrep_sn_any_step_invariant:
  "paodv i |=ₐ (recvmsg rreq_rrep_sn → onll ΓₐOADV (λ(_, a, _). anycast rreq_rrep_sn a)"

295
proof -

have sqnf_kno: "paodv i ⪰ onl _ADV (λ(ξ, 1).
(1 ∈ {PRreq−:14} → dip ξ ∈ kD (rt ξ) ∧ sqn (rt ξ) (dip ξ) = kno)"
by (inv_cterms)
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf sequence_number_one_or_bigger
THEN invariant_restrict_inD]
onl_invariant_sterms [OF aodv_wf kD_unk_or_atleast_one]
onl_invariant_sterms [OF aodv_wf sqnf_kno]
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep])
(auto simp: proj2_eq_sqn)
qed

Proposition 7.14

lemma rreq_rrep_fresh_any_step_invariant:
"paodv i ⪰ onll _ADV (λ(((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a)"
proof -

have rreq_oip: "paodv i ⪰ onl _ADV (λ((ξ, l). (l ∈ {PRreq−:3, PRreq−:4, PRreq−:13, PRreq−:21} → oip ξ ∈ kD (rt ξ)
∧ (sqn (rt ξ) (oip ξ) > (osn ξ)
∨ (sqn (rt ξ) (oip ξ) = (osn ξ)
∧ the (dhops (rt ξ) (oip ξ)) ≤ Suc (hops ξ)
∧ the (flag (rt ξ) (oip ξ)) = val))))"

proof inv_cterms
fix 1 l' pp p'
assume "((ξ, pp) ∈ reachable (paodv i) TT"
and "{PRreq−:2}[[ξ. (1:[rt :=
(update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)]] p' ∈ sterms _ADV pp"
and "l' = PRreq−:3"

show "osn ξ < sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)
∨ (sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ) = osn ξ
∧ the (dhops (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)) ≤ Suc (hops ξ)
∧ the (flag (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)) = val)"

unfolding update_def by (clarsimp split: option.split)
(metis linorder_neqE_nat not_less)

qed

Proposition 7.15

lemma rerr_invalid_any_step_invariant:
"paodv i ⪰ onll _ADV (λ((ξ, _, a, _). anycast (rerr_invalid (rt ξ)) a)"

qed

Lemma 7.14

lemma rreq_rrep_fresh_any_step_invariant:
"paodv i ⪰ onll _ADV (λ(((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a)"

proof -

have rreq_oip: "paodv i ⪰ onl _ADV (λ((ξ, l). (l ∈ {PRreq−:3, PRreq−:4, PRreq−:13, PRreq−:21} → oip ξ ∈ kD (rt ξ)
∧ (sqn (rt ξ) (oip ξ) > (osn ξ)
∨ (sqn (rt ξ) (oip ξ) = (osn ξ)
∧ the (dhops (rt ξ) (oip ξ)) ≤ Suc (hops ξ)
∧ the (flag (rt ξ) (oip ξ)) = val))))"

proof inv_cterms
fix 1 l' pp p'
assume "((ξ, pp) ∈ reachable (paodv i) TT"
and "{PRreq−:2}[[ξ. (1:[rt :=
(update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)]] p' ∈ sterms _ADV pp"
and "l' = PRreq−:3"

show "osn ξ < sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)
∨ (sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ) = osn ξ
∧ the (dhops (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)) ≤ Suc (hops ξ)
∧ the (flag (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)) = val)"

unfolding update_def by (clarsimp split: option.split)
(metis linorder_neqE_nat not_less)

qed

Lemma 7.15

lemma rerr_invalid_any_step_invariant:
"paodv i ⪰ onll _ADV (λ((ξ, _, a, _). anycast (rerr_invalid (rt ξ)) a)"

proof -

have rreq_oip: "paodv i ⪰ onl _ADV (λ((ξ, l). (l ∈ {PRreq−:3, PRreq−:4, PRreq−:13, PRreq−:21} → oip ξ ∈ kD (rt ξ)
∧ (sqn (rt ξ) (oip ξ) > (osn ξ)
∨ (sqn (rt ξ) (oip ξ) = (osn ξ)
∧ the (dhops (rt ξ) (oip ξ)) ≤ Suc (hops ξ)
∧ the (flag (rt ξ) (oip ξ)) = val))))"

proof inv_cterms
fix 1 l' pp p'
assume "((ξ, pp) ∈ reachable (paodv i) TT"
and "{PRreq−:2}[[ξ. (1:[rt :=
(update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)]] p' ∈ sterms _ADV pp"
and "l' = PRreq−:3"

show "osn ξ < sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)
∨ (sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ) = osn ξ
∧ the (dhops (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)) ≤ Suc (hops ξ)
∧ the (flag (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)) (oip ξ)) = val)"

unfolding update_def by (clarsimp split: option.split)
(metis linorder_neqE_nat not_less)

qed
\[
\forall ip \in \text{dom}(\text{dests } \xi). \ ip \in vD(rt \ \xi)) \\
\land (l \in \{\text{PAodv}:-:16..\text{PAodv}:-:17\} \\
\cup \{\text{PPkt}:-:8..\text{PPkt}:-:9\} \\
\cup \{\text{PRreq}:-:10..\text{PRreq}:-:11\} \\
\cup \{\text{PRreq}:-:18..\text{PRreq}:-:19\} \\
\cup \{\text{PRrep}:-:9..\text{PRrep}:-:10\} \\
\cup \{\text{PRerr}:-:2..\text{PRerr}:-:4\} \rightarrow (\forall ip \in \text{dom}(\text{dests } \xi). \ ip \in iD(rt \ \xi) \\
\land \ \text{the (dests } \xi \ ip) = \text{sqn (rt } \ \xi \ ip)) \\
\land (l = \text{PPkt}:-:12 \rightarrow \text{dip } \xi \in iD(rt \ \xi)))"
\]

by \(\text{inv_cterms (clarsimp split: if_split_asm option.split_asm simp: domIff)}\)+

\[
\text{show } \text{thesis} \text{ by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_inv])}
\]

qed

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs \((rip, rsn)\) is assigned to the variable \(\text{dests}\) of type \(\text{ip} \rightarrow \text{sqn}\), or to the first argument of the function \(\text{rerr}\), this set is a partial function, i.e., there is at most one entry \((rip, rsn)\) for each destination \(rip\): guaranteed by type.

Lemma \text{dests_vD_inc_sqn}:

\[
\text{paodv } i \models \text{onl } \Gamma_{AODV} (\lambda(\xi, l). (l \in \{\text{PAodv}:-:15, \text{PPkt}:-:7, \text{PRreq}:-:9, \text{PRreq}:-:17, \text{PRrep}:-:8\} \\
\land (l = \text{PRerr}:-:1 \rightarrow (\forall ip \in \text{dom}(\text{dests } \xi). \ ip \in vD(rt \ \xi) \\
\land \ \text{the (dests } \xi \ ip) > \text{sqn (rt } \ \xi \ ip))))
\]

by \(\text{inv_cterms (clarsimp split: if_split_asm option.split_asm)}\)+

Proposition 7.27

Lemma \text{route_tables_fresher}:

\[
\text{paodv } i \models_A (\text{recvmsg } \text{rreq_rrep_sn} \rightarrow \text{onll } \Gamma_{AODV} (\lambda((\xi, _), (\xi', _)). \\
\forall dip \in kD (rt \ \xi). \ rt \ \xi \subseteq dip \ rt \ \xi'))
\]

proof \(\text{(inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]})\)+

\[
\text{fix } \xi \ pp \ p'
\]

assume 
\(\{(\xi, pp) \in \text{reachable (paodv } i) (\text{recvmsg } \text{rreq_rrep_sn})\}"

and \(\{\text{PRreq}:-:2\}[(\lambda \xi. (\xi, _))(\text{rt} := \text{update (rt } \ \xi) (\text{oip } \xi) \ (\text{osn } \xi, \ \text{kno, val, Suc (hops } \xi, \ \text{sip } \xi))] \)

\(p' \in \text{sterms } \Gamma_{AODV} pp\)

and \(\text{Suc } 0 \leq \text{osn } \xi\)

and \(\text{*: } \forall ip \in kD (rt \ \xi). \ \text{Suc } 0 \leq \text{the (dhops (rt } \ \xi \ ip))\)

show \(\forall ip \in kD (rt \ \xi). \ rt \ \xi \subseteq dip \ \text{update (rt } \ \xi) (oip \ \xi) \ (\text{osn } \xi, \ \text{kno, val, Suc (hops } \xi, \ \text{sip } \xi))\)

proof \(\text{fix ip}\)

assume \(\text{ip} \in kD (rt \ \xi)\)

moreover with \(\text{*} \) have \("1 \leq \text{the (dhops (rt } \ \xi \ ip)\)" by simp

moreover from \(\text{Suc } 0 \leq \text{osn } \xi\)

have "\text{update_arg_wf (osn } \xi, \ \text{kno, val, Suc (hops } \xi, \ \text{sip } \xi)" ..

ultimately show "\text{rt } \ \xi \subseteq dip \ \text{update (rt } \ \xi) (oip \ \xi) \ (\text{osn } \xi, \ \text{kno, val, Suc (hops } \xi, \ \text{sip } \xi)" by (rule rt_fresher_update)

qed

next

\[
\text{fix } \xi \ pp \ p'
\]

assume 
\(\{(\xi, pp) \in \text{reachable (paodv } i) (\text{recvmsg } \text{rreq_rrep_sn})\}"


and \( \{PR\text{-rep-}:1\}[\Lambda \xi. \xi][rt := \text{update}(rt \xi) (\text{dip} \xi) (\text{dsn} \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops} \xi), \text{sip} \xi)] \)
\( p' \in \text{sterms} \Gamma_{AODV} \text{ pp} \)
and \( \text{Suc} 0 \leq \text{dsn} \xi \)
and \( \forall \text{ip} \in \text{KD}(rt \xi). \text{Suc} 0 \leq \text{the}(\text{dhops}(rt \xi) \text{ ip}) \)

show \( \forall \text{ip} \in \text{KD}(rt \xi). \text{rt} \xi \sqsubseteq \text{ip} \) \( \text{update}(rt \xi) (\text{dip} \xi) (\text{dsn} \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops} \xi), \text{sip} \xi) \)

proof
fix \text{ip}
assume \( \text{ip} \in \text{KD}(rt \xi) \)
moreover with \( * \) have \( 1 \leq \text{the}(\text{dhops}(rt \xi) \text{ ip}) \)
by simp
moreover from \( \langle \text{Suc} 0 \leq \text{dsn} \xi \rangle \) have \( \text{update}\_\text{arg}\_\text{wf}(\text{dsn} \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops} \xi), \text{sip} \xi) \)
ultimately show \( \text{rt} \xi \sqsubseteq \text{ip} \) \( \text{update}(rt \xi) (\text{dip} \xi) (\text{dsn} \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops} \xi), \text{sip} \xi) \)
by (rule rt\_fresher\_update)
qed
qed

end

3.7 The quality increases predicate

theory C_Quality_Increases
imports C_Aodv_Predicates C_Fresher
begin

definition quality_increases :: "state \Rightarrow state \Rightarrow bool"
where
"quality_increases \xi \xi' \equiv \( \forall \text{dip} \in \text{KD}(rt \xi). \text{dip} \in \text{KD}(rt \xi') \land rt \xi \sqsubseteq dip \xi'
\land \forall \text{dip}. \text{sqn}(rt \xi) dip \leq \text{sqn}(rt \xi') dip \)"

lemma quality_increasesI [intro!]:
assumes \( \forall \text{dip}. \text{dip} \in \text{KD}(rt \xi) \equiv \Rightarrow \text{dip} \in \text{KD}(rt \xi') \)
and \( \forall \text{dip}. \forall \text{dip} \in \text{KD}(rt \xi); \text{dip} \in \text{KD}(rt \xi'). \Rightarrow rt \xi \sqsubseteq dip \xi' \)
and \( \forall \text{dip}. \text{sqn}(rt \xi) dip \leq \text{sqn}(rt \xi') dip \)
shows "quality_increases \xi \xi'"
unfolding quality_increases_def using assms by clarsimp

lemma quality_increasesE [elim]:
fixes \text{dip}
assumes "quality_increases \xi \xi'" 
and "\text{dip} \in \text{KD}(rt \xi)" 
and "\[ \text{dip} \in \text{KD}(rt \xi'); \text{rt} \xi \sqsubseteq dip \xi'; \text{sqn}(rt \xi) dip \leq \text{sqn}(rt \xi') dip \] \Rightarrow \text{R dip \xi'}" 
shows "\text{R dip \xi'}"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rtfresherD [dest]:
fixes \text{ip}
assumes "quality_increases \xi \xi'" 
and "\text{ip} \in \text{KD}(rt \xi)" 
shows "rt \xi \sqsubseteq dip \xi'" 
using assms by auto

lemma quality_increases_sqnE [elim]:
fixes \text{dip}
assumes "quality_increases \xi \xi'" 
and "\text{sqn}(rt \xi) dip \leq \text{sqn}(rt \xi') dip \Rightarrow \text{R dip \xi'}" 
shows "\text{R dip \xi'}"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases \xi \xi" 
by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
fixes \sigma \sigma' \text{ dip}
assumes "rt (\sigma \text{ i}) \sqsubseteq dip rt (\sigma \text{ nhip})" 
and qinc: "quality_increases (\sigma \text{ nhip}) (\sigma' \text{ nhip})"
and "dip∈kD(rt (σ nhip))"
shows "rt (σ i) □ dip rt (σ' nhip)"
proof -
  from qinc have "rt (σ nhip) □ dip rt (σ' nhip)" using ⟨dip∈kD(rt (σ nhip))⟩
  by auto
  with ⟨rt (σ i) □ dip rt (σ nhip)⟩ show ?thesis ..
qed

lemma kD_quality_increases [elim]:
assumes "i∈kD(rt ξ)"
  and "quality_increases ξ ξ'"
s shows "i∈kD(rt ξ')"
using assms by auto

lemma kD_nsqn_quality_increases [elim]:
assumes "i∈kD(rt ξ)"
  and "quality_increases ξ ξ'"
s shows "i∈kD(rt ξ') ∧ nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
proof -
  from assms have "i∈kD(rt ξ')" ..
  moreover with assms have "rt ξ □ i rt ξ'" by auto
  ultimately have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  using ⟨i∈kD(rt ξ)⟩ by - (erule(2) rt_fresher_imp_nsqn_le)
  with ⟨i∈kD(rt ξ')⟩ show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
assumes "i∈kD(rt ξ)"
  and "quality_increases ξ ξ'"
s shows "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
assumes "i∈kD(rt ξ)"
  and "s ≤ nsqn (rt ξ) i"
  and "quality_increases ξ ξ'"
s shows "i∈kD(rt ξ') ∧ s ≤ nsqn (rt ξ') i"
proof -
  from ⟨i∈kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ show "i∈kD(rt ξ')" ..
next
  from ⟨i∈kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with ⟨s ≤ nsqn (rt ξ) i⟩ show "s ≤ nsqn (rt ξ') i" by (rule le_trans)
qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
assumes "i∈kD(rt ξ)"
  and "quality_increases ξ ξ'"
  and "s < nsqn (rt ξ) i"
  and "s < nsqn (rt ξ') i"
shows "s < nsqn (rt ξ') i"
proof -
  from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with ⟨s < nsqn (rt ξ) i⟩ show "s < nsqn (rt ξ') i" by simp
qed

lemma nsqn_quality_increases_dhops [elim]:
assumes "i∈kD(rt ξ)"
  and "quality_increases ξ ξ'"
  and "nsqn (rt ξ) i = nsqn (rt ξ') i"
shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
using assms unfolding quality_increases_def
by (clarsimp) (erule(1) bepec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
assumes "i∈kD(rt ξ)"
and "quality_increases ξ ξ'"
and "s = nsqn (rt ξ) i"
shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

lemma quality_increases_rreq_rrep_props [elim]:
fixes sn ip hops sip
assumes qinc: "quality_increases (σ sip) (σ' sip)"
and "1 ≤ sn"
*: "ip∈kd(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
\rightarrow (the (dhops (rt (σ sip)) ip) ≤ hops
\lor the (flag (rt (σ sip)) ip) = inv))"
shows "ip∈ kd(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
\rightarrow (the (dhops (rt (σ' sip)) ip) ≤ hops
\lor the (flag (rt (σ' sip)) ip) = inv))"
(is "_ ∧ ?nsqnafter")
proof -
from * obtain "ip∈kd(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto
from (quality_increases (σ sip) (σ' sip))
  have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
from (quality_increases (σ sip) (σ' sip)) and (ip∈kd (rt (σ sip)))
  have "ip∈kd (rt (σ' sip))" ..

from (sn ≤ nsqn (rt (σ sip)) ip) have ?nsqnafter
proof
  assume "sn < nsqn (rt (σ sip)) ip"
  also from (ip∈kd(rt (σ sip))) and (quality_increases (σ sip) (σ' sip))
    have "...
    hence (sn < nsqn (rt (σ sip)) ip) ..
  finally have "sn < nsqn (rt (σ' sip)) ip" .
thus ?thesis by simp

next
  assume "sn = nsqn (rt (σ sip)) ip"
with (ip∈kd(rt (σ sip))) and (quality_increases (σ sip) (σ' sip))
  have "sn < nsqn (rt (σ' sip)) ip
  \lor (sn = nsqn (rt (σ' sip)) ip
    \lor (the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ' sip)) ip))) ..
  hence (sn < nsqn (rt (σ' sip)) ip
  \lor (nsqn (rt (σ' sip)) ip = sn ∧ (the (dhops (rt (σ' sip)) ip) ≤ hops
    \lor the (flag (rt (σ' sip)) ip) = inv))"
proof
  assume "sn < nsqn (rt (σ' sip)) ip" thus ?thesis ..
next
  assume "sn = nsqn (rt (σ' sip)) ip
  ∧ the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)"
  hence "sn = nsqn (rt (σ' sip)) ip"
  and "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)" by auto
from * and (sn = nsqn (rt (σ sip)) ip) have "the (dhops (rt (σ sip)) ip) ≤ hops
  \lor the (flag (rt (σ sip)) ip) = inv"
by simp
thus ?thesis

proof
  assume "the (dhops (rt (σ sip)) ip) ≤ hops"
with (the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip))
  have "the (dhops (rt (σ' sip)) ip) ≤ hops" by simp
with (sn = nsqn (rt (σ' sip)) ip) show ?thesis by simp
next
  assume "the (flag (rt (σ sip)) ip) = inv"
with (ip∈kd(rt (σ sip))) have "nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1" ..
with (sn ≥ 1) and (sn = nsqn (rt (σ sip)) ip)
have "sqn (rt (σ sip)) ip > 1" by simp

from ⟨ip∈kD(rt (σ' sip))⟩ show ?thesis
proof (rule vD_or_iD)
  assume "ip∈iD(rt (σ' sip))"
  hence "the (flag (rt (σ' sip)) ip) = inv" ..
  with ⟨sn = nsqn (rt (σ' sip)) ip⟩ show ?thesis
    by simp
next

  assume "ip∈vD(rt (σ' sip))"
  hence "nsqn (rt (σ' sip)) ip = sqn (rt (σ' sip)) ip" ..
  with ⟨sn = nsqn (rt (σ' sip)) ip⟩ show ?thesis
  by simp

with ⟨sn = nsqn (rt (σ sip)) ip⟩
  have "sqn (rt (σ sip)) ip > 1" by simp

thus ?thesis ..
qed
qed

lemma quality_increases_rreq_rrep_props':
  fixes sn ip hops sip
  assumes "∀ j. quality_increases (σ j) (σ' j)"
  and "1 ≤ sn"
  and *: "ip∈kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
    ∧ (nsqn (rt (σ sip)) ip = sn
    → (the (dhops (rt (σ sip)) ip) ≤ hops
    ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip∈kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
    ∧ (nsqn (rt (σ' sip)) ip = sn
    → (the (dhops (rt (σ' sip)) ip) ≤ hops
    ∨ the (flag (rt (σ' sip)) ip) = inv))"
proof -
  from assms(1) have "quality_increases (σ sip) (σ' sip)" ..
  thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props)
qed

lemma rteq_quality_increases:
  assumes "∀ j. j ≠ i → quality_increases (σ j) (σ' j)"
  and "rt (σ' i) = rt (σ i)"
  shows "∀ j. quality_increases (σ j) (σ' j)"
  using assms by clarsimp (metis order_refl quality_increasesI rt_fresher_refl)

definition msg_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"
where "msg_fresh σ m ≡
  case m of Rreq hopsc _ _ _ oipc osnc sipc ⇒ osnc ≥ 1 ∧ (sipc ≠ oipc →
    oipc∈kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc
    ∧ (nsqn (rt (σ sipc)) oipc = osnc
    → (hopsc ≥ the (dhops (rt (σ sipc)) oipc)
    ∨ the (flag (rt (σ sipc)) oipc) = inv))
  | Rrep hopsc dipc dsnc _ sipc ⇒ dsnc ≥ 1 ∧ (sipc ≠ dipc →
    dipc∈kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc
    ∧ (nsqn (rt (σ sipc)) dipc = dsnc
    → (hopsc ≥ the (dhops (rt (σ sipc)) dipc)

301
lemma msg_fresh [simp]:
"\forall hops rreqid dip dsn dsk oip osn sip.
msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip) =
(osn \geq 1 \land (sip \neq oip \rightarrow oip \in kD(rt (σ sip)))
\land nsqn (rt (σ sip)) oip \geq osn
\land (nsqn (rt (σ sip)) oip = osn
\rightarrow (hops \geq the (dhops (rt (τ σ)) oip)
\lor the (flag (rt (σ sip)) oip) = inv)))"

"\forall hops dip dsn oip sip. msg_fresh σ (Rrep hops dip dsn oip sip) =
(dsn \geq 1 \land (sip \neq dip \rightarrow dip \in kD (rt (σ sip)))
\land nsqn (rt (σ sip)) dip \geq dsn
\land (nsqn (rt (σ sip)) dip = dsn
\rightarrow (hops \geq the (dhops (rt (τ σ)) dip))
\lor the (flag (rt (σ sip)) dip) = inv)))"

"\forall dests sip. msg_fresh σ (Rerr dests sip) =
(\forall ripc \in dom(dests). (ripc \in kD(rt (σ sip)))
\land the (dests ripc) - 1 \leq nsqn (rt (σ sip)) ripc))"

"\forall d dip. msg_fresh σ (Newpkt d dip) = True"

"\forall dip sip. msg_fresh σ (Pkt d dip sip) = True"

unfolding msg_fresh_def by simp_all

lemma msg_fresh_inc_sn [simp, elim]:
"msg_fresh σ m =⇒ rreq_rrep_sn m"
by (cases m) simp_all

lemma recv_msg_fresh_inc_sn [simp, elim]:
"orecvmsg (msg_fresh) σ m =⇒ recvmsg rreq_rrep_sn m"
by (cases m) simp_all

lemma rreq_nsqn_is_fresh [simp]:
fixes σ msg hops rreqid dip dsn dsk oip osn sip
assumes "rreq_rrep_fresh (rt (σ sip)) (Rreq hops rreqid dip dsn dsk oip osn sip)"
and "rreq_rrep_sn (Rreq hops rreqid dip dsn oip osn sip)"
shows "msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip)"
(is "msg_fresh σ ?msg")

proof -
let ?rt = "rt (σ sip)"
from assms(2) have "1 \leq osn" by simp
thus ?thesis
unfolding msg_fresh_def
proof (simp only: msg.case, intro conj1 impI)
assume "sip \neq oip"
with assms(1) show "oip \in kD(?rt)" by simp
next
assume "sip \neq oip"
and "nsqn ?rt oip = osn"
show "the (dhops ?rt oip) \leq hops \lor the (flag ?rt oip) = inv"
proof (cases "oip \in vD(?rt)")
assume "oip \in vD(?rt)"
	hence "nsqn ?rt oip = sqn ?rt oip" ..
with "\nsqn ?rt oip = osn" have "sqn ?rt oip = osn" by simp
with assms(1) and "\n sip \neq oip" have "the (dhops ?rt oip) \leq hops"
by simp
thus ?thesis ..
next
assume "oip \notin vD(?rt)"
moreover from assms(1) and "\n sip \neq oip" have "oip \in kD(?rt)" by simp
ultimately have "\n ip \in iD(?rt)" by auto
hence "the (flag ?rt oip) = inv" ..
thus \( ?\text{thesis} \).

qed

next

assume "\( \text{sip} \neq \text{oip} \)"

with assms(1) have "\( \text{osn} \leq \text{sqn} \ (\text{rt} (\sigma \ \text{sip})) \ \text{oip} \)"

thus "\( \text{osn} \leq \text{nsqn} (\text{rt} (\sigma \ \text{sip})) \ \text{oip} \)"

proof (rule nat_le_eq_or_lt)

assume "\( \text{osn} < \text{sqn} \ (\text{rt} \ \text{oip}) \)"

hence "\( \text{osn} \leq \text{sqn} \ (\text{rt} \ (\text{rt} \ (\text{rt} \ \text{oip}) - 1)) \)" by simp

also have "\( \ldots \leq \text{nsqn} \ (\text{rt} \ (\text{rt} \ \text{oip})) \)" by (rule sqn_nsqn)

finally show "\( \text{osn} \leq \text{nsqn} \ (\text{rt} \ \text{oip}) \)".

next

assume "\( \text{osn} = \text{sqn} \ (\text{rt} \ \text{oip}) \)"

with assms(1) and \( \langle \text{sip} \neq \text{oip} \rangle \) have "\( \text{oip} \in \text{kD} (\text{rt}) \)"

and "\( \text{the (flag}\ (\text{rt} \ \text{oip}) = \text{val} \)"

by auto

hence "\( \text{osn} \neq \text{dip} \rightarrow \text{dip} \in \text{kD} (\text{rt}) \)"

and "\( \text{nsqn} \ (\text{rt} \ \text{oip}) = \text{sqn} \ (\text{rt} \ \text{oip}) \)" ..

thus "\( \text{osn} \leq \text{nsqn} \ (\text{rt} \ \text{oip}) \)" by simp

qed

qed simp

qed

lemma rrep_nsqn_is_fresh [simp]:

fixes \( \sigma \ \text{msg} \ \text{hops} \ \text{dip} \ \text{dsn} \ \text{oip} \ \text{sip} \)

assumes "\( \text{rreq_rrep_fresh (rt (\sigma \ \text{sip})) (Rrep \ \text{hops} \ \text{dip} \ \text{dsn} \ \text{oip} \ \text{sip})} \)"

and "\( \text{rreq_rrep_sn (Rrep \ \text{hops} \ \text{dip} \ \text{dsn} \ \text{oip} \ \text{sip})} \)"

shows "\( \text{msg_fresh } \sigma \ (\text{Rrep} \ \text{hops} \ \text{dip} \ \text{dsn} \ \text{oip} \ \text{sip}) \)"

(is "\( \text{msg_fresh } \sigma \ ?\text{msg} \)"

proof -

let \( \text{rt} = \text{rt (\sigma \ \text{sip})} \)

from assms have "\( \text{sip} \neq \text{dip} \rightarrow \text{dip} \in \text{kD} (\text{rt}) \land \text{sqn} \ (\text{rt} \ \text{dip}) = \text{dsn} \land \text{the (flag}\ (\text{rt} \ \text{dip}) = \text{val} \)"

by simp

hence "\( \text{sip} \neq \text{dip} \rightarrow \text{dip} \in \text{kD} (\text{rt}) \land \text{sqn} \ (\text{rt} \ \text{dip}) \geq \text{dsn} \)"

by clarsimp

with assms show "\( \text{msg_fresh } \sigma \ ?\text{msg} \)"

by clarsimp

qed

lemma rerr_nsqn_is_fresh [simp]:

fixes \( \sigma \ \text{msg} \ \text{dests} \ \text{sip} \)

assumes "\( \text{rerr_invalid (rt (\sigma \ \text{sip})) (Rerr \ \text{dests} \ \text{sip})} \)"

shows "\( \text{msg_fresh } \sigma \ (\text{Rerr} \ \text{dests} \ \text{sip}) \)"

(is "\( \text{msg_fresh } \sigma \ ?\text{msg} \)"

proof -

let \( \text{rt} = \text{rt (\sigma \ \text{sip})} \)

from assms have \( \langle \forall \text{rip} \in \text{dom(dests)}. (\text{rip} \in \text{iD}(\text{rt (\sigma \ \text{sip}))} \)

\( \land \text{the}\ (\text{dests}\ \text{rip}) = \text{sqn} (\text{rt (\sigma \ \text{sip}))\ \text{rip})\)""

by clarsimp

have \( \langle \forall \text{rip} \in \text{dom(dests)}. (\text{rip} \in \text{kD}(\text{rt (\sigma \ \text{sip}))} \)

\( \land \text{the}\ (\text{dests}\ \text{rip}) - 1 \leq \text{nsqn} (\text{rt (\sigma \ \text{sip}))\ \text{rip})\)""

proof

fix rip

assume "\( \text{rip} \in \text{dom dests} \)"

with \( \langle \text{dests}\ \text{rip} \rangle \) have "\( \text{rip} \in \text{iD}(\text{rt (\sigma \ \text{sip}))} \land \text{the (dests rip) = sqn (rt (\sigma sip)) rip} \)"

by auto

from this(2) have "\( \text{the (dests rip) - 1 = sqn (rt (\sigma \ \text{sip}))\ rip - 1} \)" by simp

also have \( \langle \text{\ldots \leq sqn (rt (\sigma \ \text{sip}))\ rip} \rangle \) by (rule sqn_nsqn)

finally have "\( \text{the (dests rip) - 1 \leq nsqn (rt (\sigma \ \text{sip}))\ rip} \)" .

with \( \langle \text{rip} \in \text{iD}(\text{rt (\sigma \ \text{sip}))} \rangle \)

show "\( \text{rip} \in \text{kD}(\text{rt (\sigma \ \text{sip}))} \land \text{the (dests rip) - 1 \leq nsqn (rt (\sigma \ \text{sip}))\ rip} \)"

by clarsimp

303


Thus "msg_fresh σ ?msg"

by simp

qed


lemma quality_increases_msg_fresh [elim]:

assumes qinc: "∀j. quality_increases (σ j) (σ' j)"

and "msg_fresh σ m"

shows "msg_fresh σ' m"

using assms(2)

proof (cases m)

fix hops rreqid dip dsn dsk oip osn sip

assume [simp]: "m = Rreq hops rreqid dip dsn dsk oip osn sip"

and "msg_fresh σ m"

then have "osn ≥ 1" and "sip = oip ∨ (dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip

∧ (nsqn (rt (σ sip)) dip = dsn

→ (the (dhops (rt (σ sip)) dip) ≤ hops

∨ the (flag (rt (σ sip)) dip) = inv)))"

by auto

from this(2) show ?thesis

proof

assume "sip = oip" with ⟨osn ≥ 1⟩ show ?thesis by simp

next

assume "oip∈kD(rt (σ sip)) ∧ osn ≤ nsqn (rt (σ sip)) oip

∧ (nsqn (rt (σ sip)) oip = osn

→ (the (dhops (rt (σ sip)) oip) ≤ hops

∨ the (flag (rt (σ sip)) oip) = inv)))"

moreover from qinc have "quality_increases (σ sip) (σ' sip)"..

ultimately have "oip∈kD(rt (σ' sip)) ∧ osn ≤ nsqn (rt (σ' sip)) oip

∧ (nsqn (rt (σ' sip)) oip = osn

→ (the (dhops (rt (σ' sip)) oip) ≤ hops

∨ the (flag (rt (σ' sip)) oip) = inv)))"

using ⟨osn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2])

with ⟨osn ≥ 1⟩ show "msg_fresh σ' m"

by (clarsimp)

qed

next

fix hops dip dsn oip sip

assume [simp]: "m = Rrep hops dip dsn oip sip"

and "msg_fresh σ m"

then have "dsn ≥ 1" and "sip = dip ∨ (dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip

∧ (nsqn (rt (σ sip)) dip = dsn

→ (the (dhops (rt (σ sip)) dip) ≤ hops

∨ the (flag (rt (σ sip)) dip) = inv)))"

by auto

from this(2) show "?thesis"

proof

assume "sip = dip" with ⟨dsn ≥ 1⟩ show ?thesis by simp

next

assume "dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip

∧ (nsqn (rt (σ sip)) dip = dsn

→ (the (dhops (rt (σ sip)) dip) ≤ hops

∨ the (flag (rt (σ sip)) dip) = inv)))"

moreover from qinc have "quality_increases (σ sip) (σ' sip)"..

ultimately have "dip∈kD(rt (σ' sip)) ∧ dsn ≤ nsqn (rt (σ' sip)) dip

∧ (nsqn (rt (σ' sip)) dip = dsn

→ (the (dhops (rt (σ' sip)) dip) ≤ hops

∨ the (flag (rt (σ' sip)) dip) = inv)))"

using ⟨dsn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2])

with ⟨dsn ≥ 1⟩ show "msg_fresh σ' m"

byclarsimp

qed

next

fix dests sip
assume [simp]: "m = Rerr dests sip"  
and "msg_fresh σ m"  
then have "∀ rip∈dom(dests). rip∈kD(rt (σ sip)) 
∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip" 
by simp  
have "∀ rip∈dom(dests). rip∈kD(rt (σ' sip)) 
∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" 
proof  
fix rip  
assume "rip∈dom(dests)"  
with * have "rip∈kD(rt (σ sip))" and "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip" 
by - (drule(1) bspec, clarsimp)+  
moreover from qinc have "quality_increases (σ sip) (σ' sip)" by simp  
ultimately show "rip∈kD(rt (σ' sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" .. 
qed  
thus ?thesis by simp  
qed simp_all  
end  

3.8 The ‘open’ AODV model

theory C_OAodv  
imports C_Aodv AWN.OAWN_SOS_Labels AWN.OAWN_Convert  
begins

Definitions for stating and proving global network properties over individual processes.

definition σ_AODV ′ :: "((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp)) set" where "σ_AODV ′ ≡ {((λ i. aodv_init i, Γ_AODV PAodv))}"

abbreviation opaodv :: "ip ⇒ ((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp, msg seq_action)) automaton" where "opaodv i ≡ (init = σ_AODV ′, trans = oseqp_sos Γ_AODV i)"

lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))" unfolding σ_AODV ′_def by rule simp_all

lemma oaoedv_control_within [simp]: "control_within Γ_AODV (init (opaodv i))" unfolding σ_AODV ′_def by (rule control_withinI) (auto simp del: Γ_AODV _simps)

lemma σ_AODV ′_labels [simp]: "(σ, p) ∈ σ_AODV ′ ⇒ labels Γ_AODV p = {PAodv−:0}" unfolding σ_AODV ′_def by simp

lemma oaoedv_init_kD_empty [simp]: "(σ, p) ∈ σ_AODV ′ ⇒ kD (rt (σ i)) = {}" unfolding σ_AODV ′_def kD_def by simp

lemma oaoedv_init_vD_empty [simp]: "(σ, p) ∈ σ_AODV ′ ⇒ vD (rt (σ i)) = {}" unfolding σ_AODV ′_def vD_def by simp

lemma oaoedv_trans: "trans (opaodv i) = oseqp_sos Γ_AODV i" by simp

declare oseq_invariant_ctermsI [OF aodv_wf oaoedv_control_within aodv_simple_labels oaoedv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf oaoedv_control_within aodv_simple_labels oaoedv_trans, cterms_intros]

end
3.9 Global invariant proofs over sequential processes

theory C_Global_Invariants
imports C_Seq_Invariants
  C_Aodv_Predicates
  C_Fresher
  C_Quality_Increases
  AWN.OAWN_Convert
  C_OAodv
begin

lemma other_quality_increases [elim]:
  assumes "other quality_increases I σ σ'"
  shows "∀ j. quality_increases (σ j) (σ' j)"
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q) σ σ' a" and 
  weakenP: "∀ σ m. P σ m =⇒ P' σ m" and
  weakenQ: "∀ σ m. Q σ m =⇒ Q' σ m"
  shows "otherwith P' I (orecvmsg Q') σ σ' a"
  proof
    fix j
    assume "j ∉ I" with * have "P (σ j) (σ' j)" by auto
    thus "P' (σ j) (σ' j)" by (rule weakenP)
  next
    from * have "orecvmsg Q σ a" by auto
    thus "orecvmsg Q' σ a" by rule (erule weakenQ)
  qed

lemma oreceived_msg_inv:
  assumes other: "∀ σ σ' m. [ [ P σ m; other Q {i} σ σ' ] ] =⇒ P σ' m" and
  local: "∀ σ m. P σ m =⇒ P (σ(i := σ i | msg := m)) m"
  shows "opaodv i | = (otherwith Q {i} (orecvmsg P), other Q {i} →) onl Γ_AODV (λ(σ, l). l ∈ {PAodv-:1} −→ P σ (msg (σ i)))"
  proof
    (inv_cterms, intro impI)
    fix σ σ' l
    assume "l = PAodv-:1 =⇒ P σ (msg (σ i))"
    and "l = PAodv-:1"
    and "other Q {i} σ σ'"
    from this(1-2) have "P σ (msg (σ i))" ..
    hence "P σ' (msg (σ i))" using 'other Q {i} σ σ' by (rule other)
    moreover from 'other Q {i} σ σ' have "σ' i = σ i" ..
    ultimately show "P σ' (msg (σ' i))" by simp
  next
    fix σ σ' msg
    assume "otherwith Q {i} (orecvmsg P) σ σ' (receive msg)" and
    "σ' i = σ i(msg := msg)"
    from this(1) have "P σ msg" and "j ≠ i =⇒ Q (σ j) (σ' j)" by auto
    from this(1) have "P (σ(i := σ i(msg := msg))) msg" by (rule local)
    thus "P σ' msg" by (rule other)
    proof (rule other)
      from "σ' i = σ i(msg := msg)" and "j ≠ i =⇒ Q (σ j) (σ' j)"
      show "other Q {i} (σ(i := σ i(msg := msg))) σ'" by - (rule otherI, auto)
    qed
  qed

(Equivalent to) Proposition 7.27
lemma local_quality_increases:  
"paodv i \vdash_A (recvmsg rreq_rrep_sn \rightarrow) onll \Gamma_{AODV} (\lambda((\xi, \_), \_)(\xi', \_)). quality_increases (\xi \xi')"  
proof (rule step_invariantI)  
  fix s a s'  
  assume sr: "s \in reachable (paodv i) (recvmsg rreq_rrep_sn)"  
  and tr: "(s, a, s') \in trans (paodv i)"  
  and rm: "recvmsg rreq_rrep_sn a"  
  from sr have srTT: "s \in reachable (paodv i) TT" ..  
  from route_tables_fresher sr tr rm  
  have "onll \Gamma_{AODV} (\lambda((\xi, \_), \_), \_). quality_increases \xi \xi'"  
    by (rule step_invariantD)  
  moreover from known_destinations_increase srTT tr TT_True  
  have "onll \Gamma_{AODV} (\lambda((\xi, \_), \_), \_). quality_increases \xi \xi'"  
    by (rule step_invariantD)  
  moreover from sqns_increase srTT tr TT_True  
  have "onll \Gamma_{AODV} (\lambda((\xi, \_), \_), \_). quality_increases \xi \xi'"  
    by (rule step_invariantD)  
  ultimately show "onll \Gamma_{AODV} (\lambda((\xi, \_), \_), \_). quality_increases \xi \xi'"  
    unfolding onll_def by auto  
  qed  
  lemmas olocal_quality_increases =  
  open_seq_step_invariant [OF local_quality_increases initiali_aodv oaodv_trans aodv_trans,  
  simplified seqll_onll_swap]  
  lemma oquality_increases:  
  "opaodv i \vdash_A (otherwith quality_increases {i} (orecvmsg (\lambda_. rreq_rrep_sn)),  
  other quality_increases {i} \rightarrow) onll \Gamma_{AODV} (\lambda((\sigma, \_), \_), (\sigma', \_)). quality_increases (\sigma \sigma')"  
  (is "\vdash_A (?S, \_ \rightarrow) \_")  
  proof (rule onll_ostep_invariantI, simp)  
    fix \sigma p a \sigma' p' l'  
    assume or: "((\sigma, p), a, (\sigma', p')) \in osqp_sos \Gamma_{AODV} i"  
    and tr: "((\sigma, p), a, (\sigma', p')) \in osqp_sos \Gamma_{AODV} i"  
    and ll: "l \in labels \Gamma_{AODV} p"  
    and ll': "l' \in labels \Gamma_{AODV} p'"  
    from this(1-3) have "orecvmsg (\lambda_. rreq_rrep_sn) \sigma a"  
      by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
        and QU="other quality_increases {i}"], simp: seqll_def)  
    with ⟨?S \sigma \sigma' a⟩ show "\forall j. quality_increases (\sigma j) (\sigma' j)"  
      by (auto dest!: otherwith_syncD)  
    qed  
  lemmas olocal_quality_increases =  
  open_seq_step_invariant [OF local_quality_increases initiali_aodv oaodv_trans aodv_trans,  
  simplified seqll_onll_swap]  
  lemma quality_increases:  
  "opaodv i \vdash_A (otherwith quality_increases {i} (orecvmsg (\lambda_. rreq_rrep_sn)),  
  other quality_increases {i} \rightarrow) onll \Gamma_{AODV} (\lambda((\sigma, \_), \_), (\sigma', \_)). quality_increases (\sigma \sigma')"  
  (is "\vdash_A (?S, \_ \rightarrow) \_")  
  proof (rule onll_ostep_invariantI, simp)  
    fix \sigma p l a \sigma' p' l'  
    assume or: "((\sigma, p) \in reachable (opaodv i) ?S (other quality_increases {i}))"  
    and tr: "((\sigma, p), a, (\sigma', p')) \in osqp_sos \Gamma_{AODV} i"  
    and ll: "l \in labels \Gamma_{AODV} p"  
    and ll': "l' \in labels \Gamma_{AODV} p'"  
    from this(1-3) have "orecvmsg (\lambda_. rreq_rrep_sn) \sigma a"  
      by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
        and QU="other quality_increases {i}"], simp: seqll_def)  
    with ⟨?S \sigma \sigma' a⟩ show "\forall j. quality_increases (\sigma j) (\sigma' j)"  
      by (auto dest!: otherwith_syncD)  
  qed  
  lemma rreq_rrep_nsqn_fresh_any_step_invariant:  
  "opaodv i \vdash_A (act (recvmsg rreq_rrep_sn), other A (i) \rightarrow) onll \Gamma_{AODV} (\lambda((\sigma, \_), a, \_). anycast (msg_fresh \sigma) a)"  
  proof (rule ostep_invariantI, simp del: act_simp)  
    fix \sigma p a \sigma' p' l'  
    assume or: "((\sigma, p) \in reachable (opaodv i) ?S (other quality_increases {i}))"  
    and tr: "((\sigma, p), a, (\sigma', p')) \in osqp_sos \Gamma_{AODV} i"  
    and ll: "l \in labels \Gamma_{AODV} p"  
    and ll': "l' \in labels \Gamma_{AODV} p'"  
    from this(1-3) have "orecvmsg (\lambda_. rreq_rrep_sn) \sigma a"  
      by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
        and QU="other quality_increases {i}"], simp: seqll_def)  
    with ⟨?S \sigma \sigma' a⟩ show "\forall j. quality_increases (\sigma j) (\sigma' j)"  
      by (auto dest!: otherwith_syncD)  
  qed  
  lemma rreq_rrep_nsqn_fresh_any_step_invariant:  
  "opaodv i \vdash_A (act (recvmsg rreq_rrep_sn), other A (i) \rightarrow) onll \Gamma_{AODV} (\lambda((\sigma, \_), a, \_). anycast (msg_fresh \sigma) a)"  
  proof (rule ostep_invariantI, simp del: act_simp)  
    fix \sigma p a \sigma' p' l'  
    assume or: "((\sigma, p) \in reachable (opaodv i) ?S (other quality_increases {i}))"  
    and tr: "((\sigma, p), a, (\sigma', p')) \in osqp_sos \Gamma_{AODV} i"  
    and ll: "l \in labels \Gamma_{AODV} p"  
    and ll': "l' \in labels \Gamma_{AODV} p'"  
    from this(1-3) have "orecvmsg (\lambda_. rreq_rrep_sn) \sigma a"  
      by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
        and QU="other quality_increases {i}"], simp: seqll_def)  
    with ⟨?S \sigma \sigma' a⟩ show "\forall j. quality_increases (\sigma j) (\sigma' j)"  
      by (auto dest!: otherwith_syncD)  
  qed
have "anycast \(rreq_rrep_fresh\ (\text{rt} \ (\sigma \ i))\) a"
proof -
  have "opaodv i \|=A\ (\text{act} \ (\text{recvmsg} \ rreq_rrep_sn), \text{other} \ A \ \{i\} \rightarrow)"
      \(\text{onll} \ \Gamma_{AODV} \ (\text{seqll} \ i \ (\lambda((\xi, \ _), \ a, \ _). \ \text{anycast} \ (rreq_rrep_fresh \ (\text{rt} \ \xi)) \ a))\)"
      by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto
  hence "\(\text{onll} \ \Gamma_{AODV} \ (\text{seqll} \ i \ (\lambda((\xi, \ _), \ a, \ _). \ \text{anycast} \ (rreq_rrep_fresh \ (\text{rt} \ \xi)) \ a))\) \((\sigma, \ p), \ a, (\sigma', \ p')\)"
  using or tr recv by - (erule(4) ostep_invariantE)
  thus \?thesis using \langle \ l\in\text{labels} \ \Gamma_{AODV} \ p \rangle \ and \ \langle \ l'\in\text{labels} \ \Gamma_{AODV} \ p' \rangle \ by \ auto
qed

moreover have "anycast \(rerr_invalid\ (\text{rt} \ (\sigma \ i))\) a"
proof -
  have "opaodv i \|=A\ (\text{act} \ (\text{recvmsg} \ rreq_rrep_sn), \text{other} \ A \ \{i\} \rightarrow)"
      \(\text{onll} \ \Gamma_{AODV} \ (\text{seqll} \ i \ (\lambda((\xi, \ _), \ a, \ _). \ \text{anycast} \ (rerr_invalid \ (\text{rt} \ \xi)) \ a))\)"
      by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto
  hence "\(\text{onll} \ \Gamma_{AODV} \ (\text{seqll} \ i \ (\lambda((\xi, \ _), \ a, \ _). \ \text{anycast} \ (rerr_invalid \ (\text{rt} \ \xi)) \ a))\) \((\sigma, \ p), \ a, (\sigma', \ p')\)"
  using or tr recv by - (erule(4) ostep_invariantE)
  thus \?thesis using \langle \ l\in\text{labels} \ \Gamma_{AODV} \ p \rangle \ and \ \langle \ l'\in\text{labels} \ \Gamma_{AODV} \ p' \rangle \ by \ auto
qed

moreover have "anycast \(rreq_rrep_sn\) a"
proof -
  from or tr recv
  have "\(\text{onll} \ \Gamma_{AODV} \ (\text{seqll} \ i \ (\lambda((\xi, \ _), \ a, \ _). \ \text{anycast} \ rreq_rrep_sn \ a))\) \((\sigma, \ p), \ a, (\sigma', \ p')\)"
  by (rule ostep_invariantE [OF open_seq_step_invariant [OF rreq_rrep_sn_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto
  thus \?thesis using \langle \ l\in\text{labels} \ \Gamma_{AODV} \ p \rangle \ and \ \langle \ l'\in\text{labels} \ \Gamma_{AODV} \ p' \rangle \ by \ auto
qed

moreover have "anycast \(\lambda m. \ not_Pkt \ m \rightarrow msg_sender \ m = i) \ a"
proof -
  from or tr recv
  have "\(\text{onll} \ \Gamma_{AODV} \ (\text{seqll} \ i \ (\lambda((\xi, \ _), \ a, \ _). \ \text{anycast} \ (\lambda m. \ not_Pkt \ m \rightarrow msg_sender \ m = i) \ a))\)"
  by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rreq_rrep_sn_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto
  thus \?thesis using or tr recv \langle \ l\in\text{labels} \ \Gamma_{AODV} \ p \rangle \ and \ \langle \ l'\in\text{labels} \ \Gamma_{AODV} \ p' \rangle \ by \ auto
qed

ultimately have "anycast \(msg_fresh \ \sigma) \ a"
by (simp_all add: anycast_def del: msg_fresh split: seq_action.split_asm msg.split_asm) simp_all
thus "\(\text{onll} \ \Gamma_{AODV} \ (\lambda((\sigma, \ _), \ a, \ _). \ \text{anycast} \ (\text{msg_fresh} \ \sigma) \ a)\) \((\sigma, \ p), \ a, (\sigma', \ p')\)"
by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaodv i \|= (otherwith quality_increases \{i\} \ (orecvmsg \ msg_fresh),"
lemma quality_increases_nsqn_fresh:
  assumes sigma: "msg_fresh σ m"
  shows "msg_fresh σ' m" using * ...

next

fix σ σ'
assume "msg_fresh σ m"
thus "msg_fresh σ' m" using * ...

by (rule ostep_invariant_weakenE [OF quality_increases]) auto

lemma oquality_increases
  assumes sigma: "msg_fresh σ m"
  shows "msg_fresh σ' m" using * ...

by (rule ostep_invariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])

qed

lemma oosn_rreq:
  assumes sigma: "msg_fresh σ m"
  shows "msg_fresh σ' m" using * ...

by (rule ostep_invariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])

(auto simp: seql_onl_swap)

lemma rreq_sip:
  assumes sigma: "msg_fresh σ m"
  shows "msg_fresh σ' m" using * ...

by (rule ostep_invariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])

(auto simp: seql_onl_swap)

309
show "oip (σ' i) ∈ kD (rt (σ' (sip (σ' i))))
∧ osn (σ' i) ≤ nsqn (rt (σ' (sip (σ' i)))) (oip (σ' i))
∧ (nsqn (rt (σ' (sip (σ' i)))) (oip (σ' i)) = osn (σ' i)
    → (the (dhops (rt (σ' (sip (σ' i)))) (oip (σ' i))) ≤ hops (σ' i)
        ∨ the (flag (rt (σ' (sip (σ' i)))) (oip (σ' i))) = inv)"

proof (cases "sip (σ i) = i")
assume "sip (σ i) ≠ i"
from (other quality increases {i} σ σ')
have "quality_increases (σ (sip (σ i))) (σ' (sip (σ' i)))"
    by (rule quality_increases_rreq_rrep_props)
moreover from \(⟨σ, p⟩\) ∈ or reachable (opaodv i) ?S ?U \(1 \in\) labels \(Γ_{AODV}\) and hyp
have "1 ≤ osn (σ' i)"
    by (auto dest!: onl_oinvariant_weakenD [OF oosn_rreq]
simp add: seqlsimp)
moreover from sip (σ i) ≠ i hyp' and pre
have "oip (σ' i) ∈ kD (rt (σ (sip (σ i))))
∧ osn (σ' i) ≤ nsqn (rt (σ (sip (σ i)))) (oip (σ' i))
∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ' i)) = osn (σ' i)
    → (the (dhops (rt (σ (sip (σ i)))) (oip (σ' i))) ≤ hops (σ' i)
        ∨ the (flag (rt (σ (sip (σ i)))) (oip (σ' i))) = inv)"
    by (auto simp: σ' i = σ i)
ultimately show ?thesis
    by (rule quality_increases_rreq_rrep Props)
next
assume "sip (σ i) = i" thus ?thesis
    using σ' i = σ i hyp and pre by auto
qed

qd (auto elim!: quality_increases_rreq_rrep_props')

lemma odsn_rrep:
"opaodv i |= (otherwith quality increases {i} (orecvmsg msg_fresh),
other quality increases {i} →)
    onl Γ_{AODV} (seq i (\(λ(i, l). \ l \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-:n|n. True\} \rightarrow 1 ≤ \(dsn \ x\))")
    by (rule orinvariant_weakenE [OF open_seq_invariant [OF dsn_rrep_initiali_aodv]])
(auto simp: seql_onl_swap)

lemma rrep_sip:
"opaodv i |= (otherwith quality increases {i} (orecvmsg msg_fresh),
other quality increases {i} →)
    onl Γ_{AODV} (\(λ(\(σ, i\). \ l \in \{PAodv-:6, PAodv-:7, PRrep-:0, PRrep-:1\} \land sip (σ i) ≠ dip (σ i))
        \rightarrow dip (σ i) ∈ kD (rt (σ (sip (σ i))))
        \land nsqn (rt (σ (sip (σ i)))) (dip (σ i)) ≥ dsn (σ i)
        \land (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dsn (σ i)
            → (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (dip (σ i))) (dip (σ i))
                ∨ the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv))")
(is "\(\_\) |= (?S, ?U → \(\_\))")

proof (inv cterms inv add: oseq_step_invariant_sterms [OF quality_increases nsqn_fresh aodv_wf aodv_trans]
onl_orinvariant_sterms [OF aodv_wf oreceived_rreq_rrep nsqn_fresh_inv]
onl_orinvariant_sterms [OF aodv_wf dsn_rrep]
simp del: One_nat_def, rule impl)

fix σ σ' p l
assume "\((σ, p)\) ∈ or reachable (opaodv i) ?S ?U"
and "\(l \in\) labels Γ_{AODV} p" and pre:
"\((l = PAodv-:6 \lor l = PAodv-:7 \lor l = PRrep-:0 \lor l = PRrep-:1) \land sip (σ i) ≠ dip (σ i)
    \rightarrow dip (σ i) ∈ kD (rt (σ (sip (σ i))))
    \land dsn (σ i) ≤ nsqn (rt (σ (sip (σ i)))) (dip (σ i))
    \land (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dsn (σ i)
        → (dhops (rt (σ (sip (σ i)))) (dip (σ i))) ≤ hops (σ i)
            ∨ the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv)"
and "other quality increases {i} σ σ'"
and hyp: "\((l = PAodv-:6 \lor l = PAodv-:7 \lor l = PRrep-:0 \lor l = PRrep-:1) \land sip (σ' i) ≠ dip (σ' i)"

310
(is "؟labels ∧ sip (σ i) ≠ dip (σ i)"
from this(4) have "σ i = σ i" ..
with hyp have hyp': "؟labels ∧ sip (σ i) ≠ dip (σ i)" by simp
show "dip (σ i) ∈ KD (rt (σ (sip (σ i))))
∧ dns (σ i) ≤ nsqn (rt (σ (sip (σ i)))) (dip (σ i))
∧ (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dns (σ i)
→ the (dips (rt (σ (sip (σ i)))) (dip (σ i))) ≤ hops (σ i)
∨ the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv"
proof (cases "sip (σ i) = i")
assume "sip (σ i) ≠ i"
from (other quality_increases {i} σ σ')
have "quality_increases (σ (sip (σ i))) (σ' (sip (σ i)))"
by (rule otherE) (clarsimp simp: sip i ≠ i)
moreover from ⟨σ, p⟩ ∈ ooreachable (opaodv i) ?S ?U | l ∈ labels Γ AODV p and hyp
have "1 ≤ dns (σ i)"
by (auto dest!: onl_invariant_weakenD [OF odsn_rrep]
simp add: seqlsimp σ i = σ i)
moreover from 'sip (σ i) ≠ i' hyp' and pre
have "dip (σ i) ∈ KD (rt (σ (sip (σ i))))
∧ dns (σ i) ≤ nsqn (rt (σ (sip (σ i)))) (dip (σ i))
∧ (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dns (σ i)
→ the (dips (rt (σ (sip (σ i)))) (dip (σ i))) ≤ hops (σ i)
∨ the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv"
by (auto simp: 'σ i = σ i)
ultimately show ?thesis
by (rule quality_increases_req_rrep_props)
next
assume "sip (σ i) = i" thus ?thesis
using ⟨σ i = σ i⟩ hyp and pre by auto
qed
qed (auto simp add: seqlsimp elim!: quality_increases_req_rrep_props')

lemma rerr_sip:
"opaodv i ⊢ (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onl Γ AODV (λ(σ, l).
1 ∈ {PBadv, PBadv9, PRerr, PRerr1} →
(∀ ripc∈dom(dests (σ i)). ripc∈KD(rt (σ (sip (σ i)))) ∧
the (dests (σ i) ripc) − 1 ≤ nsqn (rt (σ (sip (σ i)))) ripc))"
(is "_ ⊢ (?S, ?U →) _")
proof -
{ fix dests rip sip rsn and σ σ' :: "ip ⇒ state"
assume qinc: "∀ j. quality_increases (σ j) (σ' j)"
and *: "∀ rip∈dom dests. rip ∈ KD (rt (σ sip))
∧ the (dests rip) − 1 ≤ nsqn (rt (σ sip)) rip"
and "dests rip = Some rsn"
from this(3) have "rip∈dom dests" by auto
with * and 'dests rip = Some rsn' have "rip∈KD(rt (σ sip))"
and "rsn − 1 ≤ nsqn (rt (σ sip)) rip"
by (auto dest!: bspec)
from qinc have "quality_increases (σ sip) (σ' sip)" ..
have "rip ∈ KD(rt (σ sip)) ∧ rsn − 1 ≤ nsqn (rt (σ sip)) rip"
proof
from 'rip∈KD(rt (σ sip))' and 'quality_increases (σ sip) (σ' sip)
show "rip ∈ KD(rt (σ sip))" ..
next
from 'rip∈KD(rt (σ sip))' and 'quality_increases (σ sip) (σ' sip)
have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ sip)) rip" ..
with 'rsn − 1 ≤ nsqn (rt (σ sip)) rip' show "rsn − 1 ≤ nsqn (rt (σ sip)) rip"
by (rule le_trans)
qed
} note partial = this

show ?thesis
by (inv_cterms inv add: oseq_step_invariant_sterms [OF equality_increases_nsqn_fresh aodv_wf aodv_trans]
onl_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
other_quality_increases other_localD
simp del: One_nat_def, intro conjI)
(clarsimp simp del: One_nat_def split: if_split_asm option.split_asm, erule(2) partial)+
qed

lemma prerr_guard: "paodv i \|= 
onl \Gamma_{AODV} (\lambda(\xi, l). (l = PRerr-:1 
\rightarrow (\forall ip \in dom(dests \xi). ip \in \nuD(rt \xi) 
\land the (nhop (rt \xi) ip) = sip \xi 
\land sqn (rt \xi) ip < the (dests \xi ip))))"
by (inv_cterms) (clarsimp split: option.split_asm if_split_asm)
lemmas odests_vD_inc_sqn =
open_seq_invariant [OF dests_vD_inc_sqn initiali_aodv aodv_trans, 
simplified seql_onl_swap, 
THEN oinvariant_anyact]
lemmas oprerr_guard =
open_seq_invariant [OF prerr_guard initiali_aodv aodv_trans, 
simplified seql_onl_swap, 
THEN oinvariant_anyact]

Proposition 7.28
lemma seq_compare_next_hop':
"opaodv i \|= (otherwith quality_increases \{i\} (orecmsg msg_fresh),
other_quality_increases \{i\} \rightarrow onl \Gamma_{AODV} (\lambda(\sigma, ._).
\forall dip. let nhip = the (nhop (rt (\sigma i)) dip) 
in dip \in kD(rt (\sigma i)) \land nhip \neq dip \rightarrow 
dip \in kD(rt (\sigma nhip)) \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma nhip)) dip)"
(is "_ \|= (?S, ?U \rightarrow ._) _")
proof -
{ fix nhop and \sigma \sigma' :: "ip \Rightarrow state"
  assume pre: "\forall dip \in kD(rt (\sigma i)). nhop dip \neq dip \rightarrow 
dip \in kD(rt (\sigma (nhop dip))) \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip"
  and qinc: "\forall j. quality_increases (\sigma j) (\sigma' j)"
  have "\forall dip \in kD(rt (\sigma i)). nhop dip \neq dip \rightarrow 
dip \in kD(rt (\sigma (nhop dip))) \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip"
    by auto
  from qinc have qinc_nhop: "quality_increases (\sigma (nhop dip)) (\sigma' (nhop dip))" ..
  with \( \forall dip \in kD(rt (\sigma (nhop dip))) \) have "\forall dip \in kD(rt (\sigma' (nhop dip)))" ..
  moreover have "nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"
  proof -
    from \( \forall dip \in kD(rt (\sigma (nhop dip))) \) qinc_nhop
     have "nsqn (rt (\sigma (nhop dip))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip" ..
    with \( \forall dip \in kD(rt (\sigma (nhop dip))) \) dip \show ?thesis
    by simp
  qed
  ultimately show "\forall dip \in kD(rt (\sigma' (nhop dip))) 
  \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip" ..
  qed
} note basic = this

312
\[
\begin{align*}
\{ & \text{fix nhop and } \sigma \sigma' :: \text{"ip } \Rightarrow \text{state"} \\
& \text{assume pre: } "\forall \text{dip} \in kD(rt (\sigma i)). \text{nhop dip } \neq \text{dip } \rightarrow \text{dip} \in kD(rt (\text{nhop dip})) \\
& \quad \land \text{nsqn (rt (} \sigma i) \text{) dip } \leq \text{nsqn (rt (} \sigma' \text{ (nhop dip))) dip} \\
& \quad \land \text{ndest: } "\forall \text{ IPC} \in \text{dom (dests (} \sigma i)\text{). IPC } \in \text{kD (rt (} \sigma (\text{sip (} \sigma i)\text{)))} \\
& \quad \quad \land \text{the (dests (} \sigma i\text{) IPC) } - 1 \leq \text{nsqn (rt (} \sigma (\text{sip (} \sigma i)\text{))) IPC"} \\
& \quad \land \text{issip: } "\forall \text{j. quality_increases (} \sigma j\text{) (} \sigma' j\text{)"} \\
& \quad \text{have } "\forall \text{dip} \in kD(rt (} \sigma i)\text{). nhop dip } \neq \text{dip } \rightarrow \text{dip } \in \text{kD (rt (} \sigma' \text{ (nhop dip)))} \\
& \quad \quad \land \text{nsqn (invalidate (rt (} \sigma i\text{)) (dests (} \sigma i\text{))) dip } \leq \text{nsqn (rt (} \sigma' \text{ (nhop dip))) dip} \}
\end{align*}
\]

proof (intro ballI impI)
fix dip
assume "dip \in kD(rt (\sigma i))"
and "nhop dip \neq dip"
with pre and qinc have "dip \in kD(rt (\sigma' (nhop dip)))"
and "nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"
by (auto dest: basic)

have "nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"
proof (cases "dip \in dom (dests (\sigma i))")
assume "dip \in dom (dests (\sigma i))"
with \(\text{dip} \in kD(rt (\sigma i))\) have "\text{nsqn (invalidate (rt (} \sigma i\text{)) (dests (} \sigma i\text{))) dip } = \text{dsn - 1}" 
by (rule nsqn_invalidate_eq)
moreover have "dsn - 1 \leq nsqn (rt (\sigma' (nhop dip))) dip"
proof -
from \(\text{dip} \in kD(rt (\sigma i))\) have "\text{the (dests (} \sigma i\text{)) dip } = \text{dsn}" by simp
with ndest and \(\text{dip} \in dom (dests (\sigma i))\) have "dip \in kD(rt (\sigma (\text{sip (} \sigma i)\text{))))" 
"dsn - 1 \leq nsqn (rt (\sigma (\text{sip (} \sigma i)\text{))) dip"
by auto
moreover from issip and \(\text{dip} \in dom (dests (\sigma i))\) have "nhop dip = sip (\sigma i)" ..
ultimately have "dip \in kD (rt (\sigma (nhop dip)))" 
and "dsn - 1 \leq nsqn (rt (\sigma (nhop dip))) dip" by auto
with qinc show "dsn - 1 \leq nsqn (rt (\sigma' (nhop dip))) dip" 
by simp (metis kD_nsqn_quality_increases_trans)
qed
ultimately show \(?\text{thesis by simp}\)
next
assume "dip \notin dom (dests (\sigma i))"
with \(\text{dip} \in kD(rt (\sigma i))\) have "\text{nsqn (invalidate (rt (} \sigma i\text{)) (dests (} \sigma i\text{))) dip } = \text{nsqn (rt (} \sigma i\text{)) dip}"
by (rule nsqn_invalidate_other)
with \(\text{nsqn (rt (} \sigma i\text{)) dip } \leq \text{nsqn (rt (} \sigma' \text{ (nhop dip))) dip}\) show \(?\text{thesis by simp}\)
qed
with \(\text{dip} \in kD(rt (\sigma' (nhop dip)))\)
show "dip \in kD (rt (\sigma' (nhop dip)))" 
\quad \land \text{nsqn (invalidate (rt (} \sigma i\text{)) (dests (} \sigma i\text{))) dip } \leq \text{nsqn (rt (} \sigma' \text{ (nhop dip))) dip}" ..
qed

\} note basic_prerr = this

\{ fix \sigma \sigma' :: \text{"ip } \Rightarrow \text{state"} \\
assume a1: "\forall \text{dip} \in kD(rt (\sigma i)). \text{the (nhop (rt (} \sigma i\text{)) dip) } \neq \text{dip } \rightarrow \text{dip} \in kD(rt (\sigma (the (nhop (rt (} \sigma i)\text{ dip))))\) dip" \\
\quad \land \text{nsqn (rt (} \sigma i\text{)) dip } \leq \text{nsqn (rt (} \sigma (\text{the (nhop (rt (} \sigma i)\text{ dip))))\) dip)"} \\
\quad \land \text{a2: } "\forall \text{j. quality_increases (} \sigma j\text{) (} \sigma' j\text{)"} \\
\quad \text{have } "\forall \text{dip} \in kD(rt (\sigma i)). \text{the (nhop (update (rt (} \sigma i\text{)) (sip (} \sigma i\text{)) (0, unk, val, Suc 0, sip (} \sigma i\text{))) dip) } \neq \text{dip } \rightarrow \text{dip} \in kD(rt (\sigma' (the (nhop (update (rt (} \sigma i)\text{) (sip (} \sigma i)\text{) (0, unk, val, Suc 0, sip (} \sigma i)\text{)}) dip)))\) dip" \\
\quad \land \text{nsqn (update (rt (} \sigma i\text{)) (sip (} \sigma i\text{)) (0, unk, val, Suc 0, sip (} \sigma i)\text{)) dip } \leq \text{nsqn (rt (} \sigma' \text{ (the (nhop (update (rt (} \sigma i)\text{) (sip (} \sigma i)\text{) (0, unk, val, Suc 0, sip (} \sigma i)\text{)}) dip)))\) dip)\}
\[
\text{dip} \quad (\text{is } \forall \text{ dip} \in kD(\Gamma (\sigma i)). \ ?P \ \text{dip})
\]

proof
fix dip
assume "\text{dip} \in kD(\Gamma (\sigma i))"
with a1 and a2
have "(the (nhop (\Gamma (\sigma i)) dip) \neq dip \longrightarrow dip \in kD(\Gamma (\sigma' (the (nhop (\Gamma (\sigma i)) dip))))
\& nsqn (\Gamma (\sigma i)) dip \leq nsqn (\Gamma (\sigma' (the (nhop (\Gamma (\sigma i)) dip)))) dip"
by - (drule(1) basic, auto)
thus "?P dip" by (cases "dip = sip (\sigma i)") auto
qed}

} note nhop_update_sip = this

{ fix \sigma \sigma' oip sip osn hops
assume pre: "\forall \text{ dip} \in kD (\Gamma (\sigma i)). \ (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip}) \neq \text{ dip} \longrightarrow \text{ dip} \in kD (\Gamma (\sigma' (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip}))))
\& nsqn (\Gamma (\sigma i)) \text{ dip} \leq \text{ nsqn} (\Gamma (\sigma' (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip})))) \text{ dip}"
and qinc: "\forall j. \text{ quality} \_ \text{ increases} (\sigma j) (\sigma' j)"
and *: "\text{ sip} \neq \text{ oip} \longrightarrow \text{ oip} \in kD (\Gamma (\sigma \text{ sip}))
\& osn \leq \text{ nsqn} (\Gamma (\sigma \text{ sip})) \text{ oip}
\& (\text{ nsqn} (\Gamma (\sigma \text{ sip})) \text{ oip} = \text{ osn}
\longrightarrow \text{ the} (\text{ dhops} (\Gamma (\sigma \text{ sip})) \text{ oip}) \leq \text{ hops}
\vee \text{ the} (\text{ flag} (\Gamma (\sigma \text{ sip})) \text{ oip}) = \text{ inv})"

from pre and qinc
have pre': "\forall \text{ dip} \in kD (\Gamma (\sigma i)). \ (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip}) \neq \text{ dip} \longrightarrow \text{ dip} \in kD (\Gamma (\sigma' (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip}))))
\& nsqn (\Gamma (\sigma i)) \text{ dip} \leq \text{ nsqn} (\Gamma (\sigma' (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip})))) \text{ dip}"
by (rule basic)
have "(the (nhop (update (\Gamma (\sigma i)) oip (osn, kno, val, Suc hops, sip)) oip) \neq \text{ oip} 
\longrightarrow oip \in kD (\Gamma (\sigma' (the (nhop (update (\Gamma (\sigma i)) oip
(osn, kno, val, Suc hops, sip)) oip))))
\& nsqn (update (\Gamma (\sigma i)) oip (osn, kno, val, Suc hops, sip)) oip
\leq nsqn (\Gamma (\sigma') (the (nhop (update (\Gamma (\sigma i)) oip
(osn, kno, val, Suc hops, sip)) oip)))) oip)
(is "?nhop\_not\_oip \longrightarrow ?oip\_in\_kD \& ?nsqn\_le\_nsqn")

proof (rule, split update rt split asm)
assume "\Gamma (\sigma i) = update (\Gamma (\sigma i)) oip (osn, kno, val, Suc hops, sip)"
and "the (nhop (\Gamma (\sigma i)) oip) \neq \text{ oip}"
with pre' show "?oip\_in\_kD \& ?nsqn\_le\_nsqn" by auto
next
assume rtnot: "\Gamma (\sigma i) \neq update (\Gamma (\sigma i)) oip (osn, kno, val, Suc hops, sip)"
and notoip: ?nhop not oip
with * qinc have ?oip in kD
by (clarsimp elim!: kD quality increases)
moreover with * pre qinc rtnot notoip have ?nsqn le nsqn
by simp (metis kD nsqn quality increases trans)
ultimately show "?oip\_in\_kD \& ?nsqn\_le\_nsqn" ..
qed}

} note update1 = this

{ fix \sigma \sigma' oip sip osn hops
assume pre: "\forall \text{ dip} \in kD (\Gamma (\sigma i)). \ (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip}) \neq \text{ dip} 
\longrightarrow dip \in kD (\Gamma (\sigma' (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip}))))
\& nsqn (\Gamma (\sigma i)) dip \leq nsqn (\Gamma (\sigma' (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip})))) dip"
and qinc: "\forall j. \text{ quality} \_ \text{ increases} (\sigma j) (\sigma' j)"
and *: "\text{ sip} \neq \text{ oip} \longrightarrow oip \in kD (\Gamma (\sigma \text{ sip}))
\& osn \leq \text{ nsqn} (\Gamma (\sigma \text{ sip})) \text{ oip}
\& (\text{ nsqn} (\Gamma (\sigma \text{ sip})) \text{ oip} = \text{ osn}
\longrightarrow \text{ the} (\text{ dhops} (\Gamma (\sigma \text{ sip})) \text{ oip}) \leq \text{ hops}
\vee \text{ the} (\text{ flag} (\Gamma (\sigma \text{ sip})) \text{ oip}) = \text{ inv})"

from pre and qinc
have pre': "\forall \text{ dip} \in kD (\Gamma (\sigma i)). \ (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip}) \neq \text{ dip} 
\longrightarrow dip \in kD (\Gamma (\sigma' (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip}))))
\& nsqn (\Gamma (\sigma i)) dip \leq nsqn (\Gamma (\sigma' (\text{the} (\text{nhop} (\Gamma (\sigma i)) \text{ dip})))) dip"
by (rule basic)

314
have \( \forall dip \in kD(rt (\sigma i)). \)
\( \text{the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip) \neq dip} \)
\( \rightarrow \text{dip} \in kD(rt (\sigma' (the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip)))} \)
\( \land \text{nsqn (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip} \leq \text{nsqn (rt (\sigma' (the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip))) dip}} \)
(is \( \forall dip \in kD(rt (\sigma i)). \)_ \( \rightarrow \text{dip} \in kD rt (\sigma i) \land \text{nsqn (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip} \)
\text{proof (intro ballI impI, split update rt_split_asm)}

fix dip
assume "dip \in kD(rt (\sigma i))"
and "the (nhop (rt (\sigma i)) dip) \neq dip"
and "rt (\sigma i) = update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)"
with pre' show "\( ?\text{dip in kD dip} \land ?\text{nsqn le nsqn dip} \)" by simp

next
fix dip
assume "dip \in kD(rt (\sigma i))"
and notdip: "the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip) \neq dip"
and rtnot: "rt (\sigma i) \neq update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)"
show "\( ?\text{dip in kD dip} \land ?\text{nsqn le nsqn dip} \)" proof (cases "dip = oip")
assume "dip \neq oip"
with pre' (dip \in kD(rt (\sigma i))) notdip *
show \( ?\text{thesis} \) by clarsimp

next
assume "dip = oip"
with rtnot qinc (dip \in kD(rt (\sigma i))) notdip *

have \( ?\text{dip in kD dip} \) by simp (metis kD_quality_increases)
moreover from (dip = oip) rtnot qinc (dip \in kD(rt (\sigma i))) notdip *

have \( ?\text{nsqn le nsqn dip} \) by simp (metis kD_nsqn_quality_increases_trans)
ultimately show \( ?\text{thesis} \) ..

done

have "\text{opaodv i \( \models \) \( (?S, \_ \rightarrow \text{onl}) \Gamma_{AODV} \lambda(\sigma, \_). \)
\( \forall dip \in kD(rt (\sigma i)). \) the (nhop (rt (\sigma i)) dip) \neq dip 
\( \rightarrow \text{dip} \in kD(rt (\sigma (the (nhop (rt (\sigma i)) dip))))} \)
\( \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (the (nhop (rt (\sigma i)) dip)))) dip)" by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases nsqn_fresh aodv_wf oaoev_trans]
onl_oinvariant_sterms [OF aodv_wf odests_vD_inc sqn]
onl_oinvariant_sterms [OF aodv_wf oprerr_guard]
onl_oinvariant_sterms [OF aodv_wf rreq sip]
onl_oinvariant_sterms [OF aodv_wf rrrep sip]
onl_oinvariant_sterms [OF aodv_wf rerr sip]
other_quality_increases
other_localD
solve: basic basic_prerr
simp add: seqlsimp nsqn invalidate nhop_update_sip
simp del: One_nat_def)

thus \( ?\text{thesis} \) unfolding Let_def by auto

 qed

Proposition 7.30
lemmas okD_unk_or_atleast_one =
open_seq_invariant [OF kD_unk_or_atleast_one initiali_aodv, simplified seql_onl_swap]

lemmas ozero_seq_unk_hops_one =
lemma oreachable_fresh_okD_unk_or_atleast_one:
  fixes dip
  assumes "(σ, p) ∈ oreachable (opaodv i)
  (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
  ∧ msg_zhops m)))
  (other_quality_increases {i})"
  and "dip ∈ kD(rt (σ i))"
  shows "σ_3 (the (rt (σ i) dip)) = unk ∨ 1 ≤ σ_2 (the (rt (σ i) dip))"
  (is "?P dip")
proof -
  have "∃ l ∈ labels Γ AODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaodv_trans aodv_trans]],
    auto dest!: otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

lemma oreachable_fresh_ozero_seq_unk_hops_one:
  fixes dip
  assumes "(σ, p) ∈ oreachable (opaodv i)
  (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
  ∧ msg_zhops m)))
  (other_quality_increases {i})"
  and "dip ∈ kD(rt (σ i))"
  shows "sqn (rt (σ i)) dip = 0 −→ sqnf (rt (σ i)) dip = unk ∧ the (dhops (rt (σ i)) dip) = 1 ∧ the (nhop (rt (σ i)) dip) = dip"
  (is "?P dip")
proof -
  have "∃ l ∈ labels Γ AODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]],
    auto dest!: onlD otherwith_actionD simp: seqlsimp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

lemma seq_nhop_quality_increases':
  shows "opaodv i |= (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
  ∧ msg_zhops m)),
  other_quality_increases {i}) →
  onl Γ AODV (λ(σ, _). ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip))
∧ nhip ≠ dip
−→ (rt (σ i)) ⊏ dip (rt (σ nhip)))"
  (is "_ |= (?S i, _ −→ _)")
proof -
  have weaken:
    "∀ P Q R P. p |= (otherwith quality_increases I (orecvmsg Q), other quality_increases I →) P
    −→ p |= (otherwith ((=)) I (orecvmsg (λσ m. Q σ m ∧ R σ m)), other quality_increases I →) P"
    by auto
  { fix ia and σ σ' :: "ip ⇒ state"
    assume ai: "∀ dip. dip ∈ vD (rt (σ i))
    ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ (the (nhop (rt (σ i)) dip)) ≠ dip
    −→ rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip)))"
    and ov: "?S i σ σ' a"
    have "∀ dip. dip ∈ vD (rt (σ i))
    ∧ dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))
    ∧ (the (nhop (rt (σ i)) dip)) ≠ dip

316
proof clarify
  fix dip
  assume a2: "dip ∈ vD(rt (σ i))"
  and a3: "dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))"
  and a4: "(the (nhop (rt (σ i)) dip)) ≠ dip"
  from ow have "∀ j. j ≠ i → σ j = σ' j" by auto
  show "rt (σ i) ⊏ dip rt (σ' (the (nhop (rt (σ i)) dip)))"
  proof (cases "(the (nhop (rt (σ i)) dip)) = i")
    assume "(the (nhop (rt (σ i)) dip)) = i"
    with ⟨dip ∈ vD(rt (σ i))⟩ have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))" by simp
    with a1 a2 a4 have "rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
    with ⟨(the (nhop (rt (σ i)) dip)) = i⟩ have "rt (σ i) ⊏ dip rt (σ i)" by simp
    hence False by simp
    thus ?thesis ..
  next
    assume "(the (nhop (rt (σ i)) dip)) ≠ i"
    with ⟨∀ j. j ≠ i → σ j = σ' j ⟩
      have "σ (the (nhop (rt (σ i)) dip)) = σ' (the (nhop (rt (σ i)) dip))" by simp
    with ⟨dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))⟩
      have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))" by simp
    with a1 a2 a4 have "rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
    with ⟨∀ i. i ≠ σ' a ⟩
      show ?thesis by simp
    qed
  qed

{ fix σ σ' a dip sip i
  assume a1: "∀ dip. dip ∈ vD(rt (σ i))
    ∧ dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip)))"
  and ov: "∀ i. i ≠ σ' a"
  have "∀ dip ∈ vD(update (rt (σ i)) sip (0, unk, val, Suc 0, sip))
    ∧ dip ∈ vD(rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip))))
    ∧ the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip) ≠ dip
    → update (rt (σ i)) sip (0, unk, val, Suc 0, sip)
    ⊏ dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip)))"
  proof clarify
    fix dip
    assume a2: "dip ∈ vD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip))"
    and a3: "dip ∈ vD (rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip))))"
    and a4: "the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip) ≠ dip"
    show "update (rt (σ i)) sip (0, unk, val, Suc 0, sip)
    ⊏ dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip)))"
  proof (cases "dip = sip")
    assume "dip = sip"
    with ⟨(the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip)) ≠ dip⟩
      have False by simp
    thus ?thesis ..
  next
    assume [simp]: "dip ≠ sip"
    from a2 have "dip ∈ vD(rt (σ i)) ∨ dip = sip"
      by (rule vD_update_val)
    with ⟨dip ≠ sip⟩ have "dip ∈ vD(rt (σ i))" by simp
    moreover from a3 have "dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))" by simp
    moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
    ultimately have "rt (σ i) ⊏ dip rt (σ' (the (nhop (rt (σ i)) dip)))"
      using a1 ov by - (drule(1) basic, simp)
    with ⟨dip ≠ sip⟩ show ?thesis
      by - (erule rt_strictly_fresher_update_other, simp)
  qed
  qed

317
\{ fix σ a σ' nhop \\
assume pre: "∀dip. dip ∈ vD (rt (σ i)) \land dip ∈ vD (rt (σ (nhop dip))) \land nhop dip \neq dip \\
→ rt (σ i) \sqsubseteq dip rt (σ (nhop dip))" \\
and ow: "?S i σ σ' a" \\
have "∀dip. dip ∈ vD (invalidate (rt (σ i)) (dests (σ i))) \\
\land dip ∈ vD (rt (σ' (nhop dip))) \land nhop dip \neq dip \\
→ rt (σ i) \sqsubseteq dip rt (σ' (nhop dip))" \\
proof clarify \\
fix dip \\
assume "dip ∈ vD (invalidate (rt (σ i)) (dests (σ i)))" \\
and "dip ∈ vD (rt (σ' (nhop dip)))" \\
and "nhop dip \neq dip" \\
from this(1) have "dip ∈ vD (rt (σ i))" \\
by (clarsimp dest!: vD.invalidate_vD_not_dests) \\
moreover from ow have "∀j. j \neq i → σ j = σ' j" by auto \\
ultimately have "rt (σ i) \sqsubseteq dip rt (σ' (nhop dip))" \\
using pre \dip ∈ vD (rt (σ' (nhop dip))) \land nhop dip \neq dip \\
by metis \\
with "∀j. j \neq i → σ j = σ' j" show "rt (σ i) \sqsubseteq dip rt (σ' (nhop dip))" \\
by (metis rt_strictly_fresher_irefl) \\
qed \\
\} note invalidate = this \\

\{ fix σ a σ' dip oip osn sip hops i \\
assume pre: "∀dip. dip ∈ vD (rt (σ i)) \\
\land dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip)))) \\
\land the (nhop (rt (σ i)) dip) \neq dip \\
→ rt (σ i) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip)))" \\
and ow: "?S i σ σ' a" \\
and "Suc 0 \leq osn" \\
and a6: "sip \neq oip → oip ∈ kD (rt (σ sip)) \\
\land osn \leq nsqn (rt (σ sip)) oip \\
\land (nsqn (rt (σ sip)) oip = osn \\
→ (the (dhops (rt (σ sip)) oip) \leq hops \\
\land the (flag (rt (σ sip)) oip) = inv))" \\
and after: "σ' i = σ i \sqsubseteq rt := update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)" \\
have "∀dip. dip ∈ vD (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) \\
\land dip ∈ vD (rt (σ' (the (nhop (update (rt (σ i)) oip \\
(osn, kno, val, Suc hops, sip)) dip)))) \\
\land the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip) \neq dip \\
→ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip) \\
\sqsubseteq dip \\
rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip)))" \\
proof clarify \\
fix dip \\
assume a2: "dip ∈ vD (update (rt (σ i)) oip (osn, kno, val, Suc (hops), sip))" \\
and a3: "dip ∈ vD (rt (σ' (the (nhop (update (rt (σ i)) oip \\
(osn, kno, val, Suc hops, sip)) dip))))" \\
and a4: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip) \neq dip" \\
from ow have a5: "∀j. j \neq i → σ j = σ' j" by auto \\
show "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip) \\
\sqsubseteq dip \\
rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip)))" \\
is "?rt1 \sqsubseteq dip ?rt2 dip" \\
proof (cases "?rt1 = rt (σ i)"") \\
assume nochange [simp]: \\
"update (rt (σ i)) oip (osn, kno, val, Suc hops, sip) = rt (σ i)" \\
from after have "σ' i = σ i" by simp \\
with a5 have "∀j. σ j = σ' j" by metis \\
from a2 have "dip ∈ vD (rt (σ i))" by simp
moreover from a3 have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))"
using nochange and ∨ j. σ j = σ' j by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) ⊑ dip rt (σ (the (nhop (rt (σ i)) dip)))"
using pre by simp

hence "rt (σ i) ⊑ dip rt (σ' (the (nhop (rt (σ i)) dip)))"
using (∨ j. σ j = σ' j) by simp
thus "?thesis" by simp

next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip ∈ kD(rt (σ i))" by auto
show ?thesis
proof (cases "dip = oip")
  assume "dip ≠ oip"

  with a2 have "dip ∈ vD (rt (σ i))" by auto
  moreover with a3 a5 after and (dip ≠ oip)
  have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))"
  by simp metis
  moreover from a4 and (dip ≠ oip) have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
  ultimately have "rt (σ i) ⊑ dip rt (σ (the (nhop (rt (σ i)) dip)))"
  using pre by simp

  with after and a5 and (dip ≠ oip) show ?thesis
  by simp (metis rt_strictly_fresher_update_other
            rt_strictly_fresher_irefl)

next
assume "dip = oip"

with a4 and change have "sip ≠ oip" by simp
with a6 have "oip ∈ kD(rt (σ sip))"
  and "osn ≤ nsqn (rt (σ sip)) oip" by auto

from a3 change (dip = oip) have "oip ∈ vD(rt (σ' sip))" by simp
hence "the (flag (rt (σ' sip)) oip) = val" by simp

from (oip ∈ kD(rt (σ sip)))
have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
  ∧ the (dhops (rt (σ' sip)) oip) ≤ hops)"

proof
  assume "oip ∈ vD(rt (σ sip))"
  hence "the (flag (rt (σ sip)) oip) = val" by simp
  with a6 (sip ≠ oip) have "nsqn (rt (σ sip)) oip = osn →
    the (dhops (rt (σ sip)) oip) ≤ hops"
    by simp
  show ?thesis
proof (cases "sip = i")
  assume "sip ≠ i"

  with a5 have "σ sip = σ' sip" by simp
  with osn ≤ nsqn (rt (σ sip)) oip
    and (nsqn (rt (σ sip)) oip = osn →
    the (dhops (rt (σ sip)) oip) ≤ hops)
  show ?thesis by auto

next
— alternative to using sip_not_ip
assume [simp]: "sip = i"
have "?rt1 = rt (σ i)"
proof (rule update_cases_kD, simp_all)
  from Suc 0 ≤ osn: show "0 < osn" by simp
next
  from oip ∈ kD(rt (σ sip)) and (sip = i) show "oip ∈ kD(rt (σ i))"
    by simp
next
  assume "sqn (rt (σ i)) oip < osn"
also from ⟨\text{osn} \leq \text{nsqn} (rt (\sigma \text{ sip} )) \circ \text{iop}⟩
also have "\ldots \leq \text{nsqn} (rt (\sigma \text{ i})) \circ \text{iop}" by simp
by (rule nsqn_sqn)
finally have "\text{sqn} (rt (\sigma \text{ i})) \circ \text{iop} < \text{sqn} (rt (\sigma \text{ i})) \circ \text{iop}".
hence False by simp
thus "(\lambda a. \text{if } a = \text{iop} \text{ then } \text{Some} (\text{osn}, \text{kno}, \text{val}, \text{Suc\ hops}, i) \text{ else } rt (\sigma \text{ i}) a) = rt (\sigma \text{ i})" ..

next
assume "\text{sqn} (rt (\sigma \text{ i})) \circ \text{iop} = \text{osn}"
and "\text{Suc\ hops} < \text{the} (\text{dhops} (rt (\sigma \text{ i})) \circ \text{iop})"
from this(1) and \circ \text{iop} \in vD (rt (\sigma \text{ sip})) have "\text{nsqn} (rt (\sigma \text{ i})) \circ \text{iop} = \text{osn}"
by simp
with \circ \text{nsqn} (rt (\sigma \text{ sip})) \circ \text{iop} = \text{osn} \longrightarrow \text{the} (\text{dhops} (rt (\sigma \text{ sip})) \circ \text{iop}) \leq \text{hops}:
have "\text{the} (\text{dhops} (rt (\sigma \text{ i})) \circ \text{iop}) \leq \text{hops}" by simp
with ⟨\text{Suc\ hops} < \text{the} (\text{dhops} (rt (\sigma \text{ i})) \circ \text{iop})⟩ have False by simp
thus "(\lambda a. \text{if } a = \text{iop} \text{ then } \text{Some} (\text{osn}, \text{kno}, \text{val}, \text{Suc\ hops}, i) \text{ else } rt (\sigma \text{ i}) a) = rt (\sigma \text{ i})" ..

next
assume "\text{the} (\text{flag} (rt (\sigma \text{ i})) \circ \text{iop}) = \text{inv}"
with ⟨\text{the} (\text{flag} (rt (\sigma \text{ sip})) \circ \text{iop}) = \text{val}⟩ have False by simp
thus "(\lambda a. \text{if } a = \text{iop} \text{ then } \text{Some} (\text{osn}, \text{kno}, \text{val}, \text{Suc\ hops}, i) \text{ else } rt (\sigma \text{ i}) a) = rt (\sigma \text{ i})" ..

next
from \circ \text{iop} \in kD(rt (\sigma \text{ sip}))
show "(\lambda a. \text{if } a = \text{iop} \text{ then } \text{Some} (\text{the} (rt (\sigma \text{ i}) \circ \text{iop})) \text{ else } rt (\sigma \text{ i}) a) = rt (\sigma \text{ i})"
by (auto dest!: kD_Some)
qed
with change have False ..
thus \text{?thesis} ..
q
d
thus \text{?thesis} proof
assume osnlt: "\text{osn} < \text{nsqn} (rt (\sigma \text{ sip})) \circ \text{iop}"
from \circ \text{dip} \in kD(rt (\sigma \text{ i})): and \circ \text{dip} = \text{iop} have "\text{dip} \in kD (?rt1)" by simp
moreover from a3 have "\text{dip} \in kD (?rt2 \text{ dip})" by simp
moreover have "\text{nsqn} ?rt1 \text{ dip} < \text{nsqn} (?rt2 \text{ dip}) \text{ dip}"
proof -
have "\text{nsqn} ?rt1 \circ \text{iop} = \text{osn}"
by (simp add: \circ \text{dip} = \text{iop} \circ \text{nsqn_update_changed_kno_val} [OF change [THEN not_sym]])
also have "\ldots < \text{nsqn} (rt (\sigma ' \text{ sip})) \circ \text{iop}" using osnlt .
also have "\ldots = \text{nsqn} (?rt2 \text{ dip}) \circ \text{iop}" by (simp add: change)
finally show \text{?thesis}
using \circ \text{dip} = \text{iop} by simp
qed
ultimately show \text{?thesis}
by (rule rt_strictly_fresher_ltI)
next
  assume osneq: "osn = nsqn (rt (σ' sip)) oip ∧ the (dhops (rt (σ' sip)) oip) ≤ hops"

  have "oip∈kD(?rt1)" by simp
  moreover from a3 ⟨dip = oip⟩ have "oip∈kD(?rt2 oip)" by simp

  moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"
  proof -
    from osneq have "osn = nsqn (rt (σ' sip)) oip" ..
    also have "osn = nsqn ?rt1 oip"
      by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
    also have "nsqn (rt (σ' sip)) oip = nsqn (?rt2 oip) oip"
      by (simp add: change)
    finally show ?thesis .
  qed

  moreover have "π5(there (?rt2 oip)) < π5(there (?rt1 oip))"
  proof -
    from osneq have "the (dhops (rt (σ' sip)) oip) ≤ hops" ..
    moreover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip∈kD(rt (σ' sip))" by auto
    ultimately have "π5(there (rt (σ' sip) oip)) ≤ hops"
      by (auto simp add: proj5_eq_dhops)
    also from change after have "hops < π5(there (σ' i) oip))"
      by (simp add: proj5_eq_dhops) (metis dhops_update_changed lessI)
    finally have "π5(there (σ' sip) oip)) < π5(there (σ' i) oip))" .
    with change after show ?thesis by simp
  qed

  ultimately have "?rt1 ⊏ ?rt2 oip"
  by (rule rt_strictly_fresher_eqI)
  with ⟨dip = oip⟩ show ?thesis by simp
  qed

} note rreq_rrep_update = this

have "opaodv i |= (otherwith ((=)) {i}) (orecvmsg (λσ. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →
      onl ΓAODV
(λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip)))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊏ rt (σ (the (nhop (rt (σ i)) dip))))"
proof (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_sip [THEN weaken]]
  onl_invariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
  onl_invariant_sterms [OF aodv_wf rerr_sip [THEN weaken]]
  onl_invariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]]
  onl_invariant_sterms [OF aodv_wf odsn_rrep [THEN weaken]]
 solve: basic update_0_unk invalidate rreq_rrep_update
 simp add: seqlsimp)

fix σ σ' p l
assume or: "(σ, p) ∈ oreachable (opaodv i) (?S i) (other quality_increases {i})"
and "other quality_increases {i} σ σ'"
and ll: "l ∈ labels ΓAODV p"
and pre: "∀dip. dip∈vD (rt (σ i))
    ∧ dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊏ rt (σ (the (nhop (rt (σ i)) dip))))"
from this(1-2)
have or': "(σ', p) ∈ oreachable (opaodv i) (?S i) (other quality_increases {i})" by - (rule oreachable_other')

from or and ll have next_hop: "∀dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ kD(σ i) ∧ nhip ≠ dip
⇒ dip ∈ kD(σ nhip)
∧ sqn (σ i) dip ≤ sqn (σ nhip) dip
by (auto dest!: onl_oinvariant_weakenD [OF seq_compare_next_hop'])

from or and 11 have unk_hops_one: "∀ dip ∈ kD (σ i). sqn (σ i) dip = 0
⇒ sqn (σ i) dip = unk
∧ the (dhops (σ i) dip) = 1
∧ the (nhop (σ i) dip) = dip"
by (auto dest!: ozero_seq_unk_hops_one[OF oadv_trans aodv_trans]
otherwith_actionD simp: seqlsimp)

other quality increases {i} σ j σ' j
by (erule otherE, metis singleton_iff)

have "∀ dip. dip ∈ vD (σ i)
∧ dip ∈ vD (σ j)
∧ the (nhop (σ i) dip) ≠ dip
⇒ rt (σ i) ⊏ dip rt (σ j)
∧ the (nhop (σ i) dip) ≠ dip"
proof clarify
fix dip
assume "dip ∈ vD(σ i)"
and "dip ∈ vD(σ j)"
and "the (nhop (σ i) dip) ≠ dip"
from this(1) and (σ i = σ j)
have "dip ∈ vD(σ i)"
and "dip ∈ vD(σ j)"
by auto simp: Let_def

have "0 < sqn (σ i) dip"
proof (rule neq0_conv [THEN iffD1, OF notI])
assume "sqn (σ i) dip = 0"
with ⟨dip ∈ kD(σ i)⟩ and unk_hops_one
have "?nhip = dip" by simp
with (?nhip ≠ dip) show False ..
qed
also have "... = nsqn (rt (σ i) dip)"
by (rule vD_nsqn_sqn [OF ⟨dip ∈ vD(σ i)⟩, THEN sym])
also have "... ≤ sqn (rt (σ ?nhip) dip)"
by (rule nsqns)
also have "... ≤ sqn (rt (σ ?nhip) dip)"
by (rule nsqns_sqn)
finally have "0 < sqn (rt (σ ?nhip) dip)".

have "rt (σ i) ⊏ dip rt (σ ?nhip)"
proof (cases "dip ∈ vD(σ ?nhip)")
assume "dip ∈ vD(σ ?nhip)"
with pre ⟨dip ∈ vD(σ i)⟩ and (?nhip ≠ dip)
have "rt (σ i) ⊏ dip rt (σ ?nhip)" by auto
moreover from ∀ j. quality increases (σ j) (σ' j)
have "quality increases (σ ?nhip) (σ' ?nhip)" ..
ultimately show ?thesis
using ⟨dip ∈ kD(σ ?nhip)⟩
by (rule strictly_fresher_quality_increases_right)
next
  assume "dip ∉ vD(rt (σ ?nhip))"
with ⟨dip ∉ kD(rt (σ ?nhip))⟩ have "dip ∈ iD(rt (σ ?nhip))" ..
hence "the (flag (rt (σ ?nhip)) dip) = inv"
  by auto
have "nsq (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
  by (rule nsqns)
also from ⟨dip ∈ iD(rt (σ ?nhip))⟩ have "... = sqn (rt (σ ?nhip)) dip - 1" ..
also have "... < sqn (rt (σ' ?nhip)) dip"
proof -
  from ⟨∀ j. quality_increases (σ j) (σ' j)⟩
    have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
hence "∀ ip. sqn (rt (σ ?nhip)) dip ≤ sqn (rt (σ' ?nhip)) dip"..
with ⟨0 < sqn (rt (σ ?nhip)) dip⟩ show ?thesis by auto
qed
finally have "nsqn (rt (σ i)) dip < nsqn (rt (σ' ?nhip)) dip".
moreover from ⟨dip ∈ vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
  have "dip ∈ kD(rt (σ' ?nhip))" by simp
ultimately show "rt (σ i) ⊏ dip rt (σ' ?nhip)"
  using ⟨dip ∈ kD(rt (σ i))⟩ by - (rule rt_strictly_fresher_ltI)
qed
thus ?thesis unfolding Let_def .
qed

lemma seq_compare_next_hop:
  fixes w
  shows "opaodv i |={ (otherwith ((=)) {i}) (orecvmsg msg_fresh),
    other quality_increases {i} \} \} \}
  global (λσ. ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →
    dip ∈ kD(rt (σ nhip))
    ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"
  by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onlD)

lemma seq_nhop_quality_increases:
  shows "opaodv i |={ (otherwith ((=)) {i})
    (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} \} \} \}
  global (λσ. ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊏ dip (rt (σ nhip)))"
  by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onlD)
end

3.10 Routing graphs and loop freedom

theory C_Loop_Freedom
imports C_Aodv_Predicates C_Fresher
begin

Define the central theorem that relates an invariant over network states to the absence of loops in the associate
routing graph.

definition
  \texttt{rt\_graph} :: "(ip ⇒ state) ⇒ ip ⇒ ip rel"
where
  "rt\_graph σ = (λdip. 
      \{(ip, ip') | ip ip' dsn dsk hops. 
        ip \neq dip ∧ rt (σ ip) dip = Some (dsn, dsk, val, hops, ip')\})"

Given the state of a network \(σ\), a routing graph for a given destination \(dip\) address abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

lemma \texttt{rt\_graphE [elim]}:
  fixes \(n dip ip ip'\)
  assumes "\((ip, ip')\) ∈ rt\_graph \(σ\) dip"
  shows "ip \neq dip ∧ (\exists r. rt (σ ip) = r 
    ∧ (\exists dsn dsk hops. r dip = Some (dsn, dsk, val, hops, ip')))"
  using \texttt{assms unfolding rt\_graph_def by auto}

lemma \texttt{rt\_graph_vD [dest]}:
  "\(\forall ip ip' σ dip. (ip, ip') ∈ rt\_graph σ dip \implies dip ∈ vD(rt (σ ip))\)"
  unfolding \texttt{rt\_graph_def vD_def by auto}

lemma \texttt{rt\_graph_vD_trans [dest]}:
  "\(\forall ip ip' σ dip. (ip, ip') ∈ (rt\_graph σ dip) \implies dip ∈ vD(rt (σ ip))\)"
  by (erule converse_tranclE) auto

lemma \texttt{rt\_graph_not_dip [dest]}:
  "\(\forall ip ip' σ dip. (ip, ip') ∈ rt\_graph σ dip \implies ip \neq dip\)"
  unfolding \texttt{rt\_graph_def by auto}

lemma \texttt{rt\_graph_not_dip_trans [dest]}:
  "\(\forall ip ip' σ dip. (ip, ip') ∈ (rt\_graph σ dip) \implies ip \neq dip\)"
  by (erule converse_tranclE) auto

NB: the property below cannot be lifted to the transitive closure

lemma \texttt{rt\_graph_nhip_is_nhop [dest]}:
  "\(\forall ip ip' σ dip. (ip, ip') ∈ rt\_graph σ dip \implies ip' = the (nhop (rt (σ ip)) dip)\)"
  unfolding \texttt{rt\_graph_def by auto}

theorem \texttt{inv\_to\_loop\_freedom}: assumes "\(∀ dip. let nhip = the (nhop (rt (σ i)) dip) 
    in dip ∈ vD (rt (σ i)) \cap vD (rt (σ nhip)) ∧ nhip \neq dip 
    \implies (rt (σ i)) \sqsubseteq dip (rt (σ nhip))\)"
  shows "\(∀ dip. irrefl ((rt\_graph σ dip)')\)"
  using \texttt{assms proof (intro allI)
    fix σ :: "ip ⇒ state" and dip
    assume inv: "∀ dip. let nhip = the (nhop (rt (σ ip)) dip) 
      in dip ∈ vD (rt (σ ip)) \cap vD (rt (σ nhip)) ∧ 
      nhip \neq dip \implies rt (σ ip) \sqsubseteq dip rt (σ nhip)" 
    { fix ip ip'
      assume "(ip, ip') ∈ (rt\_graph σ dip)'
      and "dip ∈ vD(rt (σ ip))"
      and "ip' \neq dip"
      hence "rt (σ ip) \sqsubseteq dip rt (σ ip')"
      proof induction
        fix nhip
        assume "(ip, nhip) ∈ rt\_graph σ dip"
        and "dip ∈ vD(rt (σ nhip))"
        and "nhip \neq dip"
        from "(ip, nhip) ∈ rt\_graph σ dip" have "dip ∈ vD(rt (σ ip))" 
          and "nhip = the (nhop (rt (σ ip)) dip)"
        by auto
        from "dip ∈ vD(rt (σ ip))" and "dip ∈ vD(rt (σ nhip))"
have "dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip))" ..
with 'nhip = the (nhop (rt (σ ip)) dip)
and 'nhip ≠ dip
and inv
show "rt (σ ip) △ dip rt (σ nhip)"
by (clarsimp simp: Let_def)

next
fix nhip nhip'
assume "((ip, nhip) ∈ (rt_graph σ dip)+)"
and "(nhip, nhip') ∈ rt_graph σ dip"
and IH: "[\[ dip ∈ vD(rt (σ nhip)); nhip ≠ dip \] ] =⇒ rt (σ ip) △ dip rt (σ nhip)"
and "dip ∈ vD(rt (σ nhip'))"
and "nhip' ≠ dip"
from ⟨(nhip, nhip') ∈ rt_graph σ dip⟩ have 1: "dip ∈ vD(rt (σ nhip))"
and 2: "nhip ≠ dip" by auto
from 1 2 have 3: "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
with ⟨'nhip' ≠ dip⟩

also have "rt (σ nhip) △ dip rt (σ nhip')"
proof -
from dip ∈ vD(rt (σ nhip)) and dip ∈ vD(rt (σ nhip'))
have "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
with 'nhip' ≠ dip
and 'nhip' = the (nhop (rt (σ nhip)) dip)
and inv
show "rt (σ nhip) △ dip rt (σ nhip')"
by (clarsimp simp: Let_def)
qued
finally show "rt (σ ip) △ dip rt (σ nhip')" ..
qued} note fresher = this

show "irrefl ((rt_graph σ dip)+)"
unfolding irrefl_def proof (intro allI notI)
fix ip
assume "((ip, ip) ∈ (rt_graph σ dip)+)"
moreover then have "dip ∈ vD(rt (σ ip))"
and "ip ≠ dip" by auto
ultimately have "rt (σ ip) △ dip rt (σ ip)" by (rule fresher)
thus False by simp
qed
qed

d3.11 Lift and transfer invariants to show loop freedom

3.11 Lift and transfer invariants to show loop freedom

theory C_Aodv_Loop_Freedom
imports AIN.OClosed_Transfer AIN.Qmsg_Lifting C_Global_Invariants C_Loop_Freedom
begin

3.11.1 Lift to parallel processes with queues

lemma par_step_no_change_on_send_or_receive:
fixes σ s a σ' s'
assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos Γ AODV i) (seqp_sos Γ QMSG)"
and "a ≠ τ"
shows "σ' i = σ i"
using assms by (rule qmsg_no_change_on_send_or_receive)

lemma par_nhop_quality_increases:
shows "opaodv i ((i qmsg |= (otherwith ((=)) {i}) (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)), other quality_increases {i} →)"
global (λσ. ∀dip. let nhip = the (nhop (rt (σ i))) dip 
  in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip 
  → (rt (σ i)) ⊑ dip (rt (σ nhip)))

proof (rule lift_into_qmsg [OF seq_nhop_quality_increases])

show "opaoi \( i \models_A \) (otherwith \((=)\) \{i\})
  (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other_quality_increases \{i\} \rightarrow
  globala (λ(σ, _, σ′). quality_increases (σ i) (σ′ i))"

proof (rule ostep_invariant_weakenE [OF olocal_quality_increases], simp_all)

fix t :: "(((nat ⇒ state) × (state, msg, pseq, pseq label) seqp), msg seq_action) transition"

assume "onll Γ\( AODV \) (λ(σ, _, _, (σ′, _)). ∀j. quality_increases (σ j) (σ′ j)) t"

thus "quality_increases (fst (fst t) i) (fst (snd (snd t) i))"

by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)

qed

proof

lemma par:req:q:seqn:quality:increases:

"opaoi \( i \models_{qmsg} \) (λσ ...; opaoi \( i \models_{qmsg} \) (λσ ... True) \{i\} →
  globala (λ(σ, _, σ′). quality_increases (σ i) (σ′ i))"

proof -

have "opaoi \( i \models_A \) (λσ ...; opaoi \( i \models_{qmsg} \) (λσ ... True) \{i\} →
  globala (λ(σ, _, σ′). quality_increases (σ i) (σ′ i))"

by (rule ostep_invariant_weakenE [OF olocal_quality_increases])

(auto dest!: onllD seqllD elim!: aodv_ex_labelE)

hence "opaoi \( i \models_{qmsg} \) (λσ ...; opaoi \( i \models_{qmsg} \) (λσ ... True) \{i\} →
  globala (λ(σ, _, σ′). quality_increases (σ i) (σ′ i))"

by (rule lift_step_into_qmsg_statelessassm) simp_all

thus ?thesis by rule auto

qed

lemma par:req:q:sn:quality:increases:

shows "opaoi \( i \models_{qmsg} \) (λσ ...; opaoi \( i \models_{qmsg} \) (λσ ... True) \{i\} →
  globala (λ(σ, a, σ′). anycast (msg_fresh σ) a)"

proof -

have "opaoi \( i \models_A \) (λσ ...; opaoi \( i \models_{qmsg} \) (λσ ... True) \{i\} →
  globala (λ(σ, a, σ′). anycast (msg_fresh σ) a)"


fix t

assume "onll Γ\( AODV \) (λ(σ, a, _). anycast (msg_fresh σ) a) t"

thus "globala (λ(σ, a, σ′). anycast (msg_fresh σ) a) t"

by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)

qed auto

hence "opaoi \( i \models_{qmsg} \) (λσ ...; opaoi \( i \models_{qmsg} \) (λσ ... True) \{i\} →
  globala (λ(σ, a, σ′). anycast (msg_fresh σ) a)"

by (rule lift_step_into_qmsg_statelessassm) simp_all

thus ?thesis by rule auto

qed

lemma par:anycast:msg:zhops:

shows "opaoi \( i \models_{qmsg} \) (λσ ...; opaoi \( i \models_{qmsg} \) (λσ ... True) \{i\} →
  globala (λ(σ, a, _). anycast msg_zhops a)"

proof -


have "opaoi \( i \models_A \) (act TT, other (λσ ... True) \{i\} →
  seqll i (onll Γ\( AODV \) (λ(σ, a, _). anycast msg_zhops a))"

by (rule open_seq:step:invariant)

hence "opaoi \( i \models_A \) (λσ ...; opaoi \( i \models_{qmsg} \) (λσ ... True) \{i\} →
  globala (λ(σ, a, _). anycast msg_zhops a))")
proof (rule ostep_invariant_weakenE)
  fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
  assume "seqll i (onll Γ AODV (λ(_, a, _). anycast msg_zhops a)) t"
  thus "globala (λ(_, a, _). anycast msg_zhops a) t"
  by (cases t) (clarsimp dest!: seqllD onllD, metis aodv_ex_label)
qed simp_all

hence "opaodv i (⟨⟨i qmsg = A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
  globala (λ(_, a, _). anycast msg_zhops a)⟩⟩)
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

3.11.2 Lift to nodes

lemma node_step_no_change_on_send_or_receive:
  assumes "((σ, NodeS i P R), a, (σ', NodeS i' P' R')) ∈ onode_sos
    (oparp_sos i (oseqp_sos Γ AODV i) (seqp_sos Γ QMSG))"
  and "a ≠ τ"
  shows "σ' = σ"
  using assms by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

lemma node_nhop_quality_increases:
  shows "⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
    (λσ . oarrivemsg (λ m. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases (i)⟩
  → globala (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
  in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
  → (rt (σ i)) ⊏ dip (rt (σ nhip)))" by (rule node_lift [OF par_nhop_quality_increases]) auto

lemma node_quality_increases:
  "⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
    (otherwith ((=)) {i} (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →)
  globala (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
  in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
  → (rt (σ i)) ⊏ dip (rt (σ nhip)))" by (rule node_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

lemma node_rreq_rrep_nsecn_fresh_any_step:
  shows "⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
    (λσ . oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
  globala (λ(_, a, _). castmsg (msg_fresh σ) a)⟩
  by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsecn_fresh_any_step])

lemma node_anycast_msg_zhops:
  shows "⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
    (λσ . oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
  globala (λ(_, a, _). anycast msg_zhops a)⟩
  by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

lemma node_silent_change_only:
  shows "⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
    (λσ . oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
  globala (λ(_, a, _). a ≠ τ → σ' = σ)⟩
  proof (rule ostep_invariantI, simp (no_asm), rule impl)
    fix σ :: a σ'.
    assume or: "⟨(σ, σ'), ∈ oreachable (⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
      (λσ . oarrivemsg (λ_. True) σ),
    other (λ_. True) {i} →)
  globala (λ(_, a, _). a ≠ τ → σ' = σ)⟩
  and tr: "((σ, σ'), a, (σ', σ')) ∈ trans (⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
  and "a ≠ τ" from or obtain p R where "ζ = NodeS i p R" by - (drule node_net_state, metis)
3.11.3 Lift to partial networks

**Lemma 328**

\[
\text{arrive\_rreq\_rrep\_nsqn\_fresh\_inc\_sn \ [simp]:}
\]

assumes "\(\sigma m. \text{msg\_fresh} \sigma m \land P \sigma m) \sigma m\"

shows "\(\sigma\) \(\text{arrive\_msg}(\lambda_. \text{rreq\_rrep\_sn}) \sigma\)"

using \(\text{assms}\) by \(\text{(cases m)}\ \text{auto}\)

**Proof**

\(\text{proof (rule ostep\_invariant\_D, simp (no\_asm))}\)

\(\text{fix } s a \sigma' s' \\)

assume or: "\((\sigma, s) \in \text{oreachable} \ (i : \text{opaodv i} \langle i \ qmsg : R \rangle)\)

\(\lambda\sigma. \text{arrivemsg}(\lambda_. \text{rreq\_rrep\_sn}) \sigma\)

\(\text{other (\lambda_. \text{True}) \{i\} \rightarrow} \text{globala} (\lambda(\sigma, a, \sigma').

\(\text{castmsg}(\lambda. \text{msg\_fresh} \sigma m \land \text{msg\_zhops m} a)\)"

\(\text{from or tr am have } "\text{castmsg}(\lambda. \text{msg\_fresh} \sigma a)"

\(\text{by } \text{(auto dest!: ostep\_invariant\_D [OF node\_rreq\_rrep\_nsqn\_fresh\_any\_step])}\)

moreover from or tr am have "\text{castmsg}(\lambda. \text{msg\_zhops}) a"

\(\text{by } \text{(auto dest!: ostep\_invariant\_D [OF node\_anycast\_msg\_zhops])}\)

ultimately show "\text{castmsg}(\lambda. \text{msg\_fresh} \sigma m \land \text{msg\_zhops m} a)"

\(\text{by } \text{(case_tac a) auto}\)

qed
3.11.4 Lift to closed networks

lemma onet_nhop_quality_increases:
shows "onet (\lambda. opaodv i \langle\langle i qmsg \rangle p)
           \mid (\lambda \_ \_ True, other quality_increases (net_tree_ips p) \rightarrow)
       global (\lambda \sigma. \forall i \in net_tree_ips p. \forall dip.
           let nhop = \text{the (nhop (rt (\sigma i)) dip)
                   in dip \in vD (rt (\sigma i)) \cap vD (rt (\sigma nhip)) \land nhip \neq dip
                   \rightarrow (rt (\sigma i)) \sqsubseteq_dip (rt (\sigma nhip)))"
(is "\_ \mid (_, ?U \rightarrow) ?inv")
proof (rule inclosed_closed)
from opnet_nhop_quality_increases/show "\text{\lambda \sigma. \forall i \in net_tree_ips p. \forall dip.
           let nhop = \text{the (nhop (rt (\sigma i)) dip)
                   in dip \in vD (rt (\sigma i)) \cap vD (rt (\sigma nhip)) \land nhip \neq dip
                   \rightarrow (rt (\sigma i)) \sqsubseteq_dip (rt (\sigma nhip)))""
proof (rule oinvariant_weakenE)
fix \sigma \sigma' :: "ip \Rightarrow state"
and a :: "msg node_action"
assume "otherwith ((=)) (net_tree_ips p) inoclosed \sigma \sigma' a"
thus "otherwith ((=)) (net_tree_ips p) (oarrivemsg (\lambda \sigma \_ m. msg_fresh \sigma m \land msg_zhops m)) \sigma \sigma' a"
proof (rule otherwithEI)
fix \sigma :: "ip \Rightarrow state"
and a :: "msg node_action"
assume "inoclosed \sigma a"
thus "oarrivemsg (\lambda \sigma \_ m. msg_fresh \sigma m \land msg_zhops m) \sigma a"
proof (cases a)
fix ii ni ms
assume "a = ii \land ni:arrive(ms)"
moreover with \text{\langle inoclosed \sigma a \rangle} obtain d di where "ms = newpkt(d, di)"
by (cases ms) auto
ultimately show ?thesis by simp
qed simp_all
qed simp_all
qed

3.11.5 Transfer into the standard model

interpretation aodv_openproc: openproc paodv opaodv id
rewrites "aodv_openproc.initmissing = initmissing"
proof -
show "openproc paodv opaodv id"
proof unfold_locales
fix i :: ip
have "\{(\sigma, q). (\sigma i, q) \in \sigma_{\text{AODV}} \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst } \sigma_{\text{AODV}} j)\} \subseteq \sigma_{\text{AODV}}," unfolding \sigma_{\text{AODV}}_def \sigma_{\text{AODV}}'_def
proof (rule equalityD1)
show "\forall \_ m. \text{\text{\{\_ i, q\}} \in \\{f i, p\}} \land (\forall j. j \neq i
\rightarrow \sigma j \in \text{fst } \{f i, p\}) = \{f, p\}" by (rule set_eqI) auto
qed
thus "\{ (\sigma, q) | \sigma, q, s \in \text{init (paodv i)}
\land (\sigma i, q) = \text{id s}
\land (\forall j. j \neq i \rightarrow \sigma j \in (\text{fst o id}) ' \text{init (paodv j))} \} \subseteq \text{init (opaodv i)}"
by simp
next
show "\forall j. \text{init (paodv j)} \neq {}"
unfolding \sigma_{\text{AODV}}_def by simp
next
fix i s a s' \sigma \sigma'
assume "\sigma i = \text{fst (id s)}"
and "\sigma' i = \text{fst (id s')}"
and "(s, a, s') \in \text{trans (paodv i)}"
then obtain q q' where "s = (\sigma i, q)"
and "s' = (\sigma' i, q')"
and "(\sigma i, q), a, (\sigma' i, q') \in \text{trans (paodv i)}"
by (cases s, cases s') auto
from this(3) have "((σ, q), a, (σ', q')) ∈ trans (opaodv i)"
   by simp (rule open_seqp_action [OF aodv_wf])

with ⟨s = (σ i, q)⟩ and ⟨s' = (σ' i, q')⟩
  show "(((σ, snd (id s)), a, (σ', snd (id s'))) ∈ trans (opaodv i)"
   by simp

qed

then interpret opn: openproc paodv opaodv id.

have (simp): "∀ i. (SOME x. x ∈ (fst o id) ` init (paodv i)) = aodv_init i"
  unfolding σAODV_def by simp

hence "∀ i. openproc.initmissing paodv id i = initmissing i"
  unfolding opn.initmissing_def opn.someinit_def initmissing_def
   by (auto split: option.split)

thus "openproc.initmissing paodv id = initmissing" ..

qed

interpretation aodv_openproc_par_qmsg: openproc_parq paodv opaodv id qmsg

rewrites "aodv_openproc_par_qmsg.netglobal = netglobal"
and "aodv_openproc_par_qmsg.initmissing = initmissing"

proof -

  show "openproc_parq paodv opaodv id qmsg"
   by (unfold_locales) simp

  then interpret opq: openproc_parq paodv opaodv id qmsg.

  have im: "∀ σ. openproc.initmissing (λi. paodv i ` ⟨⟨ qmsg ⟩ ` λ(p, q). (fst (id p), snd (id p), q)) = initmissing σ"
   unfolding opq.initmissing_def opq.someinit_def initmissing_def
   unfolding σAODV_def σQM SG_def
   by (clarsimp cong: option.case_cong simp del: One_nat_def simp add: fst_initmissing_netgmap_default_aodv_init_netlift [symmetric, unfolded initmissing_def])
  thus "openproc.initmissing (λi. paodv i ` ⟨⟨ qmsg ⟩ ` λ(p, q). (fst (id p), snd (id p), q)) = initmissing"
   by (rule ext)

  have "∀ P σ. openproc.netglobal (λi. paodv i ` ⟨⟨ qmsg ⟩ ` λ(p, q). (fst (id p), snd (id p), q)) P = netglobal P σ"
   unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
   unfolding σAODV_def σQM SG_def
   by (clarsimp cong: option.case_cong simp del: One_nat_def simp add: fst_initmissing_netgmap_default_aodv_init_netlift [symmetric, unfolded initmissing_def])
  thus "openproc.netglobal (λi. paodv i ` ⟨⟨ qmsg ⟩ ` λ(p, q). (fst (id p), snd (id p), q)) = netglobal"
   by auto

qed

lemma net_nhop_quality_increases:

  assumes "wf_net_tree n"
  shows "closed (pnet (λi. paodv i ` ⟨⟨ qmsg ⟩ ` λ(p, q). (fst (id p), snd (id p), q))) σ
   netglobal (λσ. ∀ i. ?inv σ i))"

proof -

  from (wf_net_tree n)

   have proto: "closed (pnet (λi. paodv i ` ⟨⟨ qmsg ⟩ ` λ(p, q). (fst (id p), snd (id p), q))) σ
   netglobal (λσ. ∀ i. ?inv σ i))"
     by (rule aodv_openproc_par_qmsg.close_opnet [OF _ onet_nhop_quality_increases])

show thesis

  unfolding invariant_def opnet_sos.opnet_tau1

  proof (rule, simp only: aodv_openproc_par_qmsg.netglobalsimp
    ∨ fix σ i
    assume sr: "σ ∈ reachable (closed (pnet (λi. paodv i ` ⟨⟨ qmsg ⟩ ` λ(p, q). (fst (id p), snd (id p), q)))) TT"
     hence "∀ i ∈ net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) σ i"
       by (drule invariantD [OF proto], 

330
simp only: aodv_openproc_par_qmsg.netglobalsimp
    fst_initmissing_netgmap_pair_fst)
thus "?inv (fst (initmissing (netgmap fst σ))) i"
proof (cases "i \in net_tree_ips n")
  assume "i \notin net_tree_ips n"
  from sr have "σ \in reachable (pnet (λi. paodv i ⟨⟨ qmsg ⟩⟩ n) TT)
    ..
  hence "net_ips σ = net_tree_ips n" ..
  with (i \notin net_tree_ips n) have "i \notin net_ips σ" by simp
  hence "(fst (initmissing (netgmap fst σ))) i = aodv_init i"
    by simp
  thus ?thesis by simp
qed metis
qed

3.11.6 Loop freedom of AODV

theorem aodv_loop_freedom:
  assumes "wf_net_tree n"
  shows "closed (pnet (λi. paodv i ⟨⟨ qmsg ⟩⟩ n) \models netglobal (λσ. ∀dip. irrefl ((rt_graph σ dip)⁺)))"
using assms by (rule aodv_openproc_par_qmsg.netglobal_weakenE
    [OF net_nhop_quality_increases inv_to_loop_freedom])
end
Chapter 4

Variant D: Forwarding the Route Request

Explanation [4, §10.5]: In AODV’s route discovery process, a destination node (or an intermediate node with an active route to the destination) will generate a RREP message in response to a received RREQ message. The RREQ message is then dropped and not forwarded. This termination of the route discovery process at the destination can lead to other nodes inadvertently creating non-optimal routes to the source node [5]. A possible modification to solve this problem is to allow the destination node to continue to forward the RREQ message. A route request is only stopped if it has been handled before. The forwarded RREQ message from the destination node needs to be modified to include a Boolean flag handled that indicates a RREP message has already been generated and sent in response to the former message. In case the flag is set to true, it prevents other nodes (with valid route to the destination) from sending a RREP message in response to their reception of the forwarded RREQ message.

4.1 Predicates and functions used in the AODV model

theory D_Aodv_Data
imports D_Fwdrreqs
begin

4.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

definition inc :: "sqn ⇒ sqn"
where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ inc x"
unfolding inc_def by simp

lemma inc_minus_suc_0 [simp]:
"inc x - Suc 0 = x"
unfolding inc_def by simp

lemma inc_never_one' [simp, intro]: "inc x ≠ Suc 0"
unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
by simp

4.1.2 Modelling Routes

A route is a 6-tuple, (dsn, dsk, flag, hops, nhip, pre) where dsn is the ‘destination sequence number’, dsk is the ‘destination-sequence-number status’, flag is the route status, hops is the number of hops to the destination, nhip is the next hop toward the destination, and pre is the set of ‘precursor nodes’—those interested in hearing about changes to the route.

type_synonym r = "sqn × k × f × nat × ip × ip set"
definition proj2 :: "r ⇒ sqn" ("π₂")
where \( \pi_2 \equiv \lambda (dsn, _, _, _, _, _). \ dsn \)

**definition**\( \) proj3 \( :: \) \( r \Rightarrow k \) \( (\pi_3) \)
where \( \pi_3 \equiv \lambda (_, dsk, _, _, _, _). \ dsk \)

**definition**\( \) proj4 \( :: \) \( r \Rightarrow f \) \( (\pi_4) \)
where \( \pi_4 \equiv \lambda (_, _, flag, _, _, _). \ flag \)

**definition**\( \) proj5 \( :: \) \( r \Rightarrow \text{nat} \) \( (\pi_5) \)
where \( \pi_5 \equiv \lambda (_, _, _, hops, _, _). \ hops \)

**definition**\( \) proj6 \( :: \) \( r \Rightarrow \text{ip} \) \( (\pi_6) \)
where \( \pi_6 \equiv \lambda (_, _, _, _, nhip, _). \ nhip \)

**definition**\( \) proj7 \( :: \) \( r \Rightarrow \text{ip set} \) \( (\pi_7) \)
where \( \pi_7 \equiv \lambda (_, _, _, _, _, pre). \ pre \)

**lemma** projs \[ simp \]:
\[
\pi_2 (dsn, dsk, flag, hops, nhip, pre) = dsn
\]
\[
\pi_3 (dsn, dsk, flag, hops, nhip, pre) = dsk
\]
\[
\pi_4 (dsn, dsk, flag, hops, nhip, pre) = flag
\]
\[
\pi_5 (dsn, dsk, flag, hops, nhip, pre) = hops
\]
\[
\pi_6 (dsn, dsk, flag, hops, nhip, pre) = nhip
\]
\[
\pi_7 (dsn, dsk, flag, hops, nhip, pre) = pre
\]
by (clarsimp simp: proj2_def proj3_def proj4_def proj5_def proj6_def proj7_def)+

**lemma** proj3_pred \[ intro \]: \( \Gamma \vdash (\pi_3 x) \)
by (rule k.induct)

**lemma** proj4_pred \[ intro \]: \( \Gamma \vdash (\pi_4 x) \)
by (rule f.induct)

**lemma** proj6_pair_snd \[ simp \]:
fixes dsn' \( r \)
shows \( \pi_6 (dsn', snd (r)) = \pi_6 (r) \)
by (cases r) simp

4.1.3 Routing Tables

Routing tables map ip addresses to route entries.

**type_synonym** rt = "ip \( \Rightarrow \) r"

**syntax**
\( _\Sigma_route \) :: "rt \Rightarrow ip \( \Rightarrow \) r" \( (\sigma_{route}(\_ \_ \_ \_ \_ \_)) \)

**translations**
\( \sigma_{route}(rt, dip) = \text{rt dip} \)

**definition** sqn :: "rt \Rightarrow ip \Rightarrow sqn"
where \( sqn \ rt \ dip \equiv \text{case } \sigma_{route}(rt, dip) \text{ of Some } r \Rightarrow \pi_2 (r) \mid \text{None } \Rightarrow 0 \)

**definition** sqnf :: "rt \Rightarrow ip \Rightarrow \text{k}"
where \( sqnf \ rt \ dip \equiv \text{case } \sigma_{route}(rt, dip) \text{ of Some } r \Rightarrow \pi_3 (r) \mid \text{None } \Rightarrow \text{unk} \)

**abbreviation** flag :: "rt \Rightarrow ip \Rightarrow f"
where \( \text{flag } rt \ dip \equiv \text{map_option } \pi_4 (\sigma_{route}(rt, dip)) \)

**abbreviation** dhops :: "rt \Rightarrow ip \Rightarrow \text{nat}"
where \( \text{dhops } rt \ dip \equiv \text{map_option } \pi_5 (\sigma_{route}(rt, dip)) \)

**abbreviation** nhop :: "rt \Rightarrow ip \Rightarrow ip"
where \( \text{nhop } rt \ dip \equiv \text{map_option } \pi_6 (\sigma_{route}(rt, dip)) \)
abbreviation prec :: "rt ⇒ ip ⇒ ip set"
where "precs rt dip ≡ map_option π₇ (σ_route(rt, dip))"

definition vD :: "rt ⇒ ip set"
where "vD rt ≡ {dip. flag rt dip = Some val}"

definition iD :: "rt ⇒ ip set"
where "iD rt ≡ {dip. flag rt dip = Some inv}"

definition kD :: "rt ⇒ ip set"
where "kD rt ≡ {dip. rt dip ≠ None}"

lemma kD_is_vD_and_iD: "kD rt = vD rt ∪ iD rt"
unfolding kD_def vD_def iD_def by auto

lemma vD_iD_gives_kD [simp]:
"∀ ip rt. ip ∈ vD rt =⇒ ip ∈ kD rt"
"∀ ip rt. ip ∈ iD rt =⇒ ip ∈ kD rt"
unfolding kD_is_vD_and_iD by simp_all

lemma kD_Some [dest]:
fixes dip rt
assumes "dip ∈ kD rt"
shows "∃dsn dsk flag hops nhip pre.
σ_route(rt, dip) = Some (dsn, dsk, flag, hops, nhip, pre)"
using assms unfolding kD_def by simp

lemma kD_None [dest]:
fixes dip rt
assumes "dip ∉ kD rt"
shows "σ_route(rt, dip) = None"
using assms unfolding kD_def
by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]:
fixes dip rt
assumes "dip ∈ vD rt"
shows "∃dsn dsk flag hops nhip pre.
σ_route(rt, dip) = Some (dsn, dsk, flag, hops, nhip, pre)"
using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
unfolding vD_def by simp

lemma iD_Some [dest]:
fixes dip rt
assumes "dip ∈ iD rt"
shows "∃dsn dsk flag hops nhip pre.
σ_route(rt, dip) = Some (dsn, dsk, inv, hops, nhip, pre)"
using assms unfolding iD_def by simp

lemma val_is_vD [elim]:
fixes ip rt
assumes "ip ∈ kD(rt)"
and "the (flag rt ip) = val"
shows "ip ∈ vD(rt)"
using assms unfolding vD_def by auto

lemma inv_is_iD [elim]:
fixes ip rt
assumes "ip ∈ kD(rt)"
and "the (flag rt ip) = inv"
shows "ip ∈ iD(rt)"
using assms unfolding iD_def by auto

334
lemma \textit{id\_flag\_is\_inv} [elim, simp]:
  \begin{verbatim}
  fixes ip rt
  assumes "ip \in iD(rt)"
  shows "the (flag rt ip) = inv"
  proof -
  from \{ip \in iD(rt)\} have "ip \in kD(rt)" by auto
  with assms show ?thesis unfolding iD_def by auto
  qed
  \end{verbatim}

lemma \textit{kD\_but\_not\_vD\_is\_iD} [elim]:
  \begin{verbatim}
  fixes ip rt
  assumes "ip \in kD(rt)"
  and "ip /\in vD(rt)"
  shows "ip \in iD(rt)"
  proof -
  from \{ip \in kD(rt)\} obtain dsn dsk f hops nhop pre
  where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop, pre)"
  by (metis kD_Some)
  from \{ip \in kD(rt)\} have "f \neq val"
  proof (rule contrapos_nn)
    assume "f = val"
    with rtip have "the (flag rt ip) = val" by simp
    with \{ip \in kD(rt)\} show "ip \in vD(rt)" ..
  qed
  with rtip have "the (flag rt ip) = inv" by simp
  with \{ip \in kD(rt)\} show "ip \in iD(rt)" ..
  qed
  \end{verbatim}

lemma \textit{vD\_or\_iD} [elim]:
  \begin{verbatim}
  fixes ip rt
  assumes "ip \in kD(rt)"
  and "ip \in vD(rt) = \Rightarrow P rt ip"
  and "ip \in iD(rt) = \Rightarrow P rt ip"
  shows "P rt ip"
  proof -
  from \{ip \in kD(rt)\} have "ip \in vD(rt) \cup iD(rt)"
  by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
  qed
  \end{verbatim}

lemma \textit{proj5\_eq\_dhops}:
  \begin{verbatim}
  \(\forall dip rt. dip \in kD(rt) = \Rightarrow \pi_5 (\text{the } (rt dip)) = \text{the } (dhops rt dip)\)
  unfolding sqn_def by (drule kD_Some) clarsimp
  \end{verbatim}

lemma \textit{proj4\_eq\_flag}:
  \begin{verbatim}
  \(\forall dip rt. dip \in kD(rt) = \Rightarrow \pi_4 (\text{the } (rt dip)) = \text{the } (flag rt dip)\)
  unfolding sqn_def by (drule kD_Some) clarsimp
  \end{verbatim}

lemma \textit{proj2\_eq\_sqn}:
  \begin{verbatim}
  \(\forall dip rt. dip \in kD(rt) = \Rightarrow \pi_2 (\text{the } (rt dip)) = \text{sqn rt dip}\)
  unfolding sqn_def by (drule kD_Some) clarsimp
  \end{verbatim}

lemma \textit{kD\_sqnf\_is\_proj3} [simp]:
  \begin{verbatim}
  \(\forall ip rt. ip \in kD(rt) = \Rightarrow \text{sqnf rt ip } = \pi_3 (\text{the } (rt ip))\)
  unfolding sqn_def by auto
  \end{verbatim}

lemma \textit{vD\_flag\_val} [simp]:
  \begin{verbatim}
  \(\forall dip rt. dip \in vD (rt) = \Rightarrow \text{the } (flag rt dip) = \text{val}\)
  unfolding vD_def by clarsimp
  \end{verbatim}

lemma \textit{kD\_update} [simp]:
  \begin{verbatim}
  \(\forall rt nip v. kD (rt(nip \mapsto v)) = insert nip (kD rt)\)
  unfolding kD_def by auto
  \end{verbatim}

lemma \textit{kD\_empty} [simp]: "kD Map.empty = {}"
unfolding kD_def by simp
lemma ip_equal_or_known [elim]:
    fixes rt ip ip'
    assumes "ip = ip' ∨ ip ∈ kD(rt)"
    and "ip = ip' ⟹ P rt ip ip'"
    and "[ ip ≠ ip'; ip ∈ kD(rt)] ⟹ P rt ip ip'"
    shows "P rt ip ip'"
using assms by auto

4.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

Updating Precursor Lists

definition addpre :: "r ⇒ ip set ⇒ r"
    where "addpre r npre ≡ let (dsn, dsk, flag, hops, nhip, pre) = r in
          (dsn, dsk, flag, hops, nhip, pre ∪ npre)"

lemma proj2_addpre:
    fixes v pre
    shows "π₂(addpre v pre) = π₂(v)"
    unfolding addpre_def by (cases v) simp

lemma proj3_addpre:
    fixes v pre
    shows "π₃(addpre v pre) = π₃(v)"
    unfolding addpre_def by (cases v) simp

lemma proj4_addpre:
    fixes v pre
    shows "π₄(addpre v pre) = π₄(v)"
    unfolding addpre_def by (cases v) simp

lemma proj5_addpre:
    fixes v pre
    shows "π₅(addpre v pre) = π₅(v)"
    unfolding addpre_def by (cases v) simp

lemma proj6_addpre:
    fixes dsn dsk flag hops nhip pre npre
    shows "π₆(addpre v npre) = π₆(v) ∪ npre"
    unfolding addpre_def by (cases v) simp

lemma proj7_addpre:
    fixes dsn dsk flag hops nhip pre npre
    shows "π₇(addpre v npre) = π₇(v) ∪ npre"
    unfolding addpre_def by (cases v) simp

lemma addpre_empty: "addpre r {} = r"
    unfolding addpre_def by simp

lemma addpre_r: "addpre (dsn, dsk, f1, hops, nhip, pre) npre = (dsn, dsk, f1, hops, nhip, pre ∪ npre)"
    unfolding addpre_def by simp

lemmas addpre_simps [simp] = proj2_addpre proj3_addpre proj4_addpre proj5_addpre proj6_addpre proj7_addpre addpre_empty addpre_r

definition addpreRT :: "rt ⇒ ip set ⇒ rt"
    where "addpreRT rt dip npre ≡ map_option (λs. rt (dip ↦ addpre s npre)) (σ_route(rt, dip))"
lemma snd_addpre [simp]:
"\(dsn dsn' v pre. (dsn, and(addpre (dsn', v) pre)) = addpre (dsn, v) pre\)"
unfolding addpre_def by clarsimp

lemma proj2_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip\in\kD rt"
  and "ip'\in\kD rt"
  shows "\(\pi_2(\text{the (the (addpreRT rt ip' npre) ip)) = \pi_2(\text{the (rt ip)}))\)"
  using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj3_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip\in\kD rt"
  and "ip'\in\kD rt"
  shows "\(\pi_3(\text{the (the (addpreRT rt ip' npre) ip)) = \pi_3(\text{the (rt ip)}))\)"
  using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj5_addpreRT [simp]:
"\(\pi_5(\text{the (the (addpreRT rt dip npre) ip)) = \pi_5(\text{the (rt ip)}))\)"
unfolding addpreRT_def by auto

lemma flag_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip\in\kD rt"
  shows "\(\text{flag (the (addpreRT rt dip pre)) ip = flag rt ip}\)"
  unfolding addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma kD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip\in\kD rt"
  shows "\(\kD (\text{the (addpreRT rt dip npre)}) = \kD rt\)"
unfolding kD_def addpreRT_def by clarsimp blast

lemma vD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip\in\kD rt"
  shows "\(\text{vD (the (addpreRT rt dip npre)) = vD rt}\)"
  unfolding vD_def addpreRT_def
  using assms [THEN kD_Some] by clarsimp auto

lemma iD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip\in\kD rt"
  shows "\(\text{iD (the (addpreRT rt dip npre)) = iD rt}\)"
  unfolding iD_def addpreRT_def by clarsimp auto

lemma nhop_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip\in\kD rt"
  shows "\(\text{nhop (the (addpreRT rt dip pre)) ip = nhop rt ip}\)"
  unfolding sqn_def addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma sqn_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip\in\kD rt"
  shows "\(\text{sqn (the (addpreRT rt dip pre)) ip = sqn rt ip}\)"
  unfolding sqn_def addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)
lemma dhops_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "dhops (the (addpreRT rt dip pre)) ip = dhops rt ip"
unfolding addpreRT_def
using assms [THEN kD_Some] by (clarsimp)

lemma sqnf_addpreRT [simp]:
"\ip dip. ip∈kD rt \implies sqnf (the (addpreRT rt dip npre)) dip = sqnf rt dip"
unfolding sqnf_def addpreRT_def by auto

Updating route entries

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD rt"
  shows "\case rt dip of None ⇒ en | Some r ⇒ es r = es (the (rt dip))"
using assms [THEN kD_Some] by auto

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip /∈ kD rt"
  shows "\case rt dip of None ⇒ en | Some r ⇒ es r = en"
using assms [THEN kD_None] by auto

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip pre
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "sqn rt ip = dsn"
unfolding sqn_def using assms by simp

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip /∈ kD rt"
  shows "sqn rt dip = 0"
using assms unfolding sqn_def by simp

definition update_arg_wf :: "r ⇒ bool"
where "update_arg_wf r ≡ π_4 (r) = val ∧ 
      (π_2 (r) = 0) = (π_3 (r) = unk) ∧ 
      (π_3 (r) = unk −→ π_5 (r) = 1)"

lemma update_arg_wf_gives_cases:
"\r. update_arg_wf r = ⇒ (π_2 (r) = 0) = (π_3 (r) = unk)"
unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples [simp]:
"\nhip pre. update_arg_wf (0, unk, val, Suc 0, nhip, pre)"
"\n hops nhip pre. update_arg_wf (Suc n, kno, val, hops, nhip, pre)"
unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples’ [elim]:
"\n hops nhip pre. Suc 0 ≤ n =⇒ update_arg_wf (n, kno, val, hops, nhip, pre)"
unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: "\nhip pre. P (0, unk, val, Suc 0, nhip, pre)"
  and c2: "\dsn hops nhip pre. dsn > 0 =⇒ P (dsn, kno, val, hops, nhip, pre)"
  shows "P r"
proof -

obtain dsn dsk flag hops nhip pre
where *: "r = (dsn, dsk, flag, hops, nhip, pre)" by (cases r)
with ⟨update_arg_wf r⟩ have wf1: "flag = val"
and wf2: "(dsn = 0) = (dsk = unk)"
and wf3: "dsk = unk →→ (hops = 1)"

unfolding update_arg_wf_def by auto
have "P (dsn, dsk, flag, hops, nhip, pre)"
proof (cases r)
  assume "dsk = unk"
  moreover with wf2 wf3
  have "dsn = 0" and "hops = Suc 0" by auto
  ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c1)
next
  assume "dsk = kno"
  moreover with wf2
  have "dsn > 0" by simp
  ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c2)
qed

with * show "P r" by simp
qed

definition update :: "rt ⇒ ip ⇒ r ⇒ rt"
where
  "update rt ip r ≡ case σ_route(rt, ip) of
    None ⇒ rt (ip ↦→ r)
  | Some s ⇒
    if π₂(s) < π₂(r) then rt (ip ↦→ addpre r (π₇(s)))
    else if π₂(s) = π₂(r) ∧ (π₅(s) > π₅(r) ∨ π₄(s) = inv)
      then rt (ip ↦→ addpre r (π₇(s)))
    else if π₃(r) = unk
      then rt (ip ↦→ (π₃(s), snd (addpre r (π₇(s)))))
    else rt (ip ↦→ addpre s (π₇(r)))"

lemma update_simps [simp]:
fixes r s nr nr' ns rt ip
defines "s ≡ the σ_route(rt, ip)"
and "nr ≡ addpre r (π₇(s))"
and "nr' ≡ (π₃(s), π₃(nr), π₄(nr), π₅(nr), π₆(nr), π₇(nr))"
and "ns ≡ addpre s (π₇(r))"
shows
  "[ip ∉ kD(rt)]" ⇒ update rt ip r = rt (ip ↦→ r)"
  "[ip ∈ kD(rt); sqn rt ip < π₂(r)]" ⇒ update rt ip r = rt (ip ↦→ nr)"
  "[ip ∈ kD(rt); sqn rt ip = π₂(r); the (dhops rt ip) > π₅(r)]" ⇒ update rt ip r = rt (ip ↦→ nr)"
  "[ip ∈ kD(rt); sqn rt ip = π₂(r); flag rt ip = Some inv]" ⇒ update rt ip r = rt (ip ↦→ nr)"
  "[ip ∈ kD(rt); π₃(r) = unk; (π₂(r) = 0) = (π₃(r) = unk)]" ⇒ update rt ip r = rt (ip ↦→ nr')"
  "[ip ∈ kD(rt); sqn rt ip ≥ π₂(r); π₃(r) = kno; sqn rt ip = π₂(r) ⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val ]" ⇒ update rt ip r = rt (ip ↦→ ns)"

proof -
  assume "ip ∉ kD(rt)"
  hence "σ_route(rt, ip) = None" ..
  thus "update rt ip r = rt (ip ↦→ r)"
  unfolding update_def by simp
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip < π₂(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
  with ⟨sqn rt ip < π₂(r)⟩ show "update rt ip r = rt (ip ↦→ nr)"
  unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"
and "sqn rt ip = π₂(r)"
and "the (dhops rt ip) > π₅(r)"
from this(1) obtain dsn dsk fl hops nhip pre
where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
with (sqn rt ip = π₂(r)) and (the (dhops rt ip) > π₅(r))
show "update rt ip r = rt (ip ↦ nr)"
unfolding update_def nr_def s_def by auto

next
assume "ip ∈ kD(rt)"
and "π₃(r) = unk"
and "(π₂(r) = 0) = (π₃(r) = unk)"
from this(1) obtain dsn dsk fl hops nhip pre
where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
with (π₂(r) = 0) = (π₃(r) = unk) and (π₃(r) = unk)
show "update rt ip r = rt (ip ↦ nr)"
unfolding update_def nr_def s_def by auto

next
assume "ip ∈ kD(rt)"
and otherassms: "sqn rt ip ≥ π₂(r)"
"π₃(r) = kno"
"sqn rt ip = π₂(r) ⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val"
from this(1) obtain dsn dsk fl hops nhip pre
where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
by (metis kD_Some)
with otherassms show "update rt ip r = rt (ip ↦ ns)"
unfolding update_def ns_def s_def by auto
qed

lemma update_cases [elim]:
assumes "(π₂(r) = 0) = (π₃(r) = unk)"
and c1: "[\[ \[ ip \notin kD(rt) \] \] \Rightarrow P (rt (ip ↦ r))"
and c2: "[\[ \[ ip \in kD(rt); sqn rt ip < π₂(r) \] \] \Rightarrow P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip))))))"
and c3: "[\[ \[ ip \in kD(rt); sqn rt ip = π₂(r); the (dhops rt ip) > π₅(r) \] \] \Rightarrow P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip))))))"
and c4: "[\[ \[ ip \in kD(rt); sqn rt ip = π₂(r); the (flag rt ip) = inv \] \] \Rightarrow P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip))))))"
and c5: "[\[ \[ ip \in kD(rt); π₃(r) = unk \] \] \Rightarrow P (rt (ip ↦ (π₇(the σroute(rt, ip)), π₃(r), π₄(r), π₅(r), π₆(r), π₇(addpre r (π₇(the σroute(rt, ip)))))))"
and c6: "[\[ \[ ip \in kD(rt); sqn rt ip ≥ π₂(r); π₃(r) = kno; sqn rt ip = π₂(r) ⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val \] \] \Rightarrow P (rt (ip ↦ addpre (the σroute(rt, ip)) (π₇(r)))))"
shows "(P (update rt ip r))"
proof (cases "ip ∈ kD(rt)")
assume "ip \notin kD(rt)"
with c1 show \?thesis
by simp
next
assume "ip ∈ kD(rt)"
moreover then obtain \(dsn\ dsk\ fl\ hops\ nhip\ pre\)
where \(rteq:\ \text{"rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"}\)
by (metis \text{kD\_Some})
moreover obtain \(dsn'\ dsk'\ fl'\ hops'\ nhip'\ pre'\)
where \(req:\ \text{"r = (dsn', dsk', fl', hops', nhip', pre')"}\)
by (cases \(r\)) metis
ultimately show \(\text{thesis}\)
using \(\langle (\pi_2(r) = 0) = (\pi_3(r) = \text{unk}) \rangle\)
c2 \(\{\text{OF \(ip\in\text{kD}(rt)\)}\}\)
c3 \(\{\text{OF \(ip\in\text{kD}(rt)\)}\}\)
c4 \(\{\text{OF \(ip\in\text{kD}(rt)\)}\}\)
c5 \(\{\text{OF \(ip\in\text{kD}(rt)\)}\}\)
c6 \(\{\text{OF \(ip\in\text{kD}(rt)\)}\}\)
unfolding update_def sqn_def by auto
qed

lemma update_cases_kD:
assumes \(\text{"(}\pi_2(r) = 0) = (\pi_3(r) = \text{unk})\)\)
and \(\text{"ip \in \text{kD}(rt)\"}\)
and \(c2: \text{"sqn rt ip < \(\pi_2(r)\) \(\Rightarrow\) P (rt (ip \mapsto \text{addpre r (}\pi_7\text{(the }\sigma\text{route(r, ip))}))\)\"}\)
and \(c3: \text{"[\[ sqn rt ip = \(\pi_2(r)\); the (dhops rt ip) > \(\pi_5(r)\]\]} \(\Rightarrow\) P (rt (ip \mapsto \text{addpre r (}\pi_7\text{(the }\sigma\text{route(r, ip))})\)\"}\)
and \(c4: \text{"[\[ sqn rt ip = \(\pi_2(r)\); the (flag rt ip) = inv\]} \(\Rightarrow\) P (rt (ip \mapsto \text{addpre r (}\pi_7\text{(the }\sigma\text{route(r, ip))})\)\"}\)
and \(c5: \text{"\(\pi_3(r) = \text{unk}\) \(\Rightarrow\) P (rt (ip \mapsto (\pi_2\text{(the }\sigma\text{route(r, ip)), }\pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r), \pi_7\text{(addpre r (}\pi_7\text{(the }\sigma\text{route(r, ip))))}))\")\"}\)
and \(c6: \text{"[\[ sqn rt ip \geq \(\pi_2(r)\); \(\pi_3(r) = \text{kno}\); sqn rt ip = \(\pi_2(r)\) \(\Rightarrow\) the (dhops rt ip) \leq \(\pi_5(r)\) \(\wedge\) the (flag rt ip) = val\]} \(\Rightarrow\) P (rt (ip \mapsto \text{addpre (the }\sigma\text{route(r, rt)) (}\pi_7(r)\)\)\)\"}\)
shows \("(P (update rt ip r))"\)
using assms(1) proof (rule update_cases)
assume \("sqn rt ip < \(\pi_2(r)\)\")
thus \("P (rt(ip \mapsto \text{addpre r (}\pi_7\text{(the (rt ip))}))\)" by (rule c2)
next
assume \("sqn rt ip = \(\pi_2(r)\)\")
and \("the (dhops rt ip) > \(\pi_5(r)\)\")
thus \("P (rt(ip \mapsto \text{addpre r (}\pi_7\text{(the (rt ip))}))\)" by (rule c3)
next
assume \("sqn rt ip = \(\pi_2(r)\)\")
and \("the (flag rt ip) = inv\")
thus \("P (rt(ip \mapsto \text{addpre r (}\pi_7\text{(the (rt ip))}))\)" by (rule c4)
next
assume \("\pi_3(r) = \text{unk}\")
thus \("P (rt (ip \mapsto (\pi_7\text{(the }\sigma\text{route(r, rt, ip)), }\pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r), \pi_7\text{(addpre r (}\pi_7\text{(the }\sigma\text{route(r, rt, ip))))}))\")\" by (rule c5)
next
assume \("sqn rt ip \geq \(\pi_2(r)\)\")
and \("\pi_3(r) = \text{kno}\")
and \("sqn rt ip = \(\pi_2(r)\) \(\Rightarrow\) the (dhops rt ip) \leq \(\pi_5(r)\) \(\wedge\) the (flag rt ip) = val\")
thus \("P (rt (ip \mapsto \text{addpre (the }\sigma\text{route(rt, rt)) (}\pi_7(r)\)\)\)" by (rule c6)
qed (simp add: \(\langle \text{ip \in kD(rt)\rangle}\)\)

lemma in\_kD\_after\_update [simp]:
fixes \(rt\ nip\ dsn\ dsk\ flag\ hops\ nhip\ pre\)
shows \("kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) = insert nip (kD rt)\")
unfolding update_def
by (cases "rt nip") auto

lemma nhop\_of\_update [simp]:
fixes rt dip dsn dsk flag hops nhip
assumes "rt ≠ update rt dip (dsn, dsk, flag, hops, nhip, {})"
shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip, {})) dip) = nhip"
proof -
  from assms
  have update_neq: "∀ v. rt dip = Some v ⟹
    update rt dip (dsn, dsk, flag, hops, nhip, {})
      ≠ rt(dip → addpre (the (rt dip)) (π₇ (dsn, dsk, flag, hops, nhip, {})))"
    by auto
  show ?thesis
    proof (cases "rt dip = None")
      assume "rt dip = None"
      thus ?thesis unfolding update_def by clarsimp
    next
      assume "rt dip ≠ None"
      then obtain v where "rt dip = Some v" by (metis not_None_eq)
      with update_neq [OF this] show ?thesis
        unfolding update_def by auto
    qed
  qed

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn (λ x. if x = rip then Some v else rt x) ip
      = (if ip = rip then π₂(v) else sqn rt ip)"
  unfolding sqn_def by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip pre
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip, pre)) dip"
  proof (rule update_cases)
    show "(π₂ (dsn, dsk, val, hops, nhip, pre) = 0) = (π₃ (dsn, dsk, val, hops, nhip, pre) = unk)"
      by simp (rule assms)
  qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip pre
  assumes "1 ≤ hops"
  shows "sqn rt ip ≤ sqn (update rt ip' (dsn, dsk, flag, hops, nhip, pre)) ip"
  using assms unfolding update_def sqn_def
  by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops ip rip nhip pre
  assumes ex: "∀ ip∈kD rt. the (dhops rt ip) ≥ 1"
    and ip: "(ip = rip ∧ Suc 0 ≤ hops) ∨ (ip ≠ rip ∧ ip∈kD rt)"
  shows "Suc 0 ≤ the (dhops (update rt rip (dsn, dsk, flag, hops, nhip, pre)) ip)"
  using ip proof
    assume "ip = rip ∧ Suc 0 ≤ hops" thus ?thesis
    unfolding update_def using ex
    by (cases "rip ∈ kD rt") (drule(1) bspec, auto)
  next
    assume "ip ≠ rip ∧ ip∈kD rt" thus ?thesis
    using ex unfolding update_def
    by (cases "rip∈kD rt") auto
  qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)
lemma nhop_update_another [simp]:
  fixes dip ip rt dan dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dan dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  "∀ rt ip dsn dsk flag hops nhip pre. sqn (rt(ip ↦ v)) ip = π₂(v)"
  unfolding sqn_def
  by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip, {})) dip) = hops"
  using assms unfolding update_def
  by (clarsimp split: option.splits_asm option.split if_split_asm) auto

lemma nhop_update_unk_val [simp]:
  "∀ rt dip dsn dsk flag hops sip. the (nhop (update rt dip (dsn, unk, val, hops, ip, npre)) dip) = ip"
  unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip, {}) ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.split if_split_asm) auto

lemma update_rt_split_asm:
  "∀ rt ip dsn dsk flag hops sip.
   P (update rt ip (dsn, dsk, flag, hops, sip, {})) =
      (¬ (rt = update rt ip (dsn, dsk, flag, hops, sip, {})) ∧ ¬P rt
       ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip, {})
       ∧ ¬P (update rt ip (dsn, dsk, flag, hops, sip, {}))))"
  by auto

lemma sqn_update [simp]: "∀ rt dip dsn flg hops sip.
 rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
 ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsn"
  unfolding update_def by (clarsimp split: option.splits if_split_asm) auto

lemma sqnf_update [simp]: "∀ rt dip dsn flg hops sip.
 rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
 ⇒ sqnf (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits if_split_asm) auto

lemma update_kno_dsn_greater_zero:
  "∀ rt dip dsn hops npre. 1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip, npre)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∀ rt dip dsn dsk flag hops sip.
  343
\text{rt} \not= \text{update rt dip (dsn, dsk, flg, hops, sip, \{\})}
\Rightarrow \pi_3(\text{the (update rt dip (dsn, dsk, flg, hops, sip, \{\}) dip})) = \text{dsk}"

unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma nhop_update_changed_kno_val [simp]: 
"\forall rt \ ip \ dsn \ dsk \ hops \ nhip. \nrt \not= \text{update rt ip (dsn, kno, val, hops, nhip, \{\})} \n\Rightarrow \text{the (nhop (update rt ip (dsn, kno, val, hops, nhip, \{\}) ip)) = nhip}"

unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma flag_update [simp]: 
"\forall rt \ dip \ dsn \ flg \ hops \ sip. \nrt \not= \text{update rt dip (dsn, kno, flg, hops, sip, \{\})} \n\Rightarrow \text{the (flag (update rt dip (dsn, kno, flg, hops, sip, \{\}) dip)) = flg}"

unfolding update_def
by (clarsimp split: option.split if_split_asm) auto

lemma the_flag_Some [dest!]:
fixes ip rt
assumes "the (flag rt ip) = x"
and "ip \in kD rt"
shows "flag rt ip = Some x"
using assms by auto

lemma kD_update_unchanged [dest]:
fixes rt dip dsn dsk flag hops nhip pre
assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip, pre)"
shows "dip \in\ kD(rt)"
proof -
  have "dip \in kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp
  with assms show thesis by simp
qed

lemma nhop_update [simp]: 
"\forall rt \ dip \ dsn \ dsk \ flg \ hops \ sip. \nrt \not= \text{update rt dip (dsn, dsk, flg, hops, sip, \{\})} \n\Rightarrow \text{the (nhop (update rt dip (dsn, dsk, flg, hops, sip, \{\}) dip)) = sip}"

unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma sqn_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip \not= dip"
shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip \not= dip"
shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:
"\forall dip rt dip’ dsn dsk hops nhip pre. \ndip \in vD(update rt dip’ (dsn, dsk, val, hops, nhip, pre)) \Rightarrow (dip \not\in vD(rt) \vee dip=dip’)"

unfolding update_def vD_def by (clarsimp split: option.split_asm if_split_asm)

Invalidating route entries

definition invalidate :: "rt \Rightarrow (ip \Rightarrow \text{sqn}) \Rightarrow rt"
where "invalidate rt dests \equiv \lambda ip. \text{case (rt ip, dests ip) of}
  (None, _) \Rightarrow None"
lemma proj3_invalidate [simp]:
"\(\forall dip. \pi_3(\text{the (invalidate rt dests dip)}) = \pi_3(\text{the (rt dip)})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj5_invalidate [simp]:
"\(\forall dip. \pi_5(\text{the (invalidate rt dests dip)}) = \pi_5(\text{the (rt dip)})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
"\(\forall dip. \pi_6(\text{the (invalidate rt dests dip)}) = \pi_6(\text{the (rt dip)})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj7_invalidate [simp]:
"\(\forall dip. \pi_7(\text{the (invalidate rt dests dip)}) = \pi_7(\text{the (rt dip)})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_kD_inv [simp]:
"\(\forall rt dests. kD(\text{invalidate rt dests}) = kD rt\)"
unfolding invalidate_def kD_def
by (simp split: option.split)

lemma invalidate_sqn:
fixes rt dip dests
assumes "\(\forall rsn. \text{dests dip = Some rsn} \Rightarrow \text{sqn rt dip} \leq rsn\)"
shows "\(\text{sqn rt dip} \leq \text{sqn (invalidate rt dests dip)}\)"
proof (cases "dip \notin kD(rt)"
  assume "\(\neg dip \in kD(rt)\)"
  hence "dip \notin kD(rt)" by simp
  then obtain dsn dsk flag hops nhip pre where "\(rt dip = \text{Some (dsn, dsk, flag, hops, nhip, pre)}\)"
  by (metis kD_Some)
  with assms show "\(\text{sqn rt dip} \leq \text{sqn (invalidate rt dests dip)}\)"
  by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
qed simp

lemma sqn_invalidate_in_dests [simp]:
fixes dests ipa rsn rt
assumes "\(\text{dests ipa = Some rsn}\)"
  and "\(ipa \in kD(rt)\)"
shows "\(\text{sqn (invalidate rt dests) ipa = rsn}\)"
unfolding invalidate_def sqn_def
using assms(1) assms(2) [THEN kD_Some]
by clarsimp

lemma dhops_invalidate [simp]:
"\(\forall dip. \text{the (dhops (invalidate rt dests dip)) = the (dhops rt dip)}\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
"\(\forall dip. \text{sqnf (invalidate (rt \xi \\ (dests \xi)) dip) = sqnf (rt \xi dip)}\)"
unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
"\(\forall dip. \text{the (nhop (invalidate (rt \xi \\ (dests \xi)) dip) = the (nhop (rt \xi dip))}\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
fixes rt dests dip
assumes "\(dip \notin \text{dom(dests)}\)"
shows "\(\text{invalidate rt dests dip} = rt dip\)"
using assms unfolding invalidate_def
by (clarsimp split: option.split_asm)

**lemma invalidate_none [simp]:**

- **fixes rt dests dip**
- **assumes "dip\(\notin kD(rt)\)"
- **shows "invalidate rt dests dip = None"**
- **using assms unfolding invalidate_def by clarsimp**

**lemma vD_invalidate_vD_not_dests:**

\[
\forall \text{ dip rt dests. dip} \in vD(\text{invalidate rt dests}) \implies \text{ dip} \in vD(\text{rt}) \land \text{ dests dip} = \text{ None}
\]

- **unfolding invalidate_def vD_def**
- **by (clarsimp split: option.split_asm)**

**lemma sqn_invalidate_not_in_dests [simp]:**

- **fixes dests dip rt**
- **assumes "dip \(\notin \text{ dom(dests)}\)"
- **shows "sqn (\text{invalidate rt dests}) dip = sqn \text{ rt dip}"**
- **using assms unfolding sqn_def by simp**

**lemma invalidate_changes:**

- **fixes rt dests dip dsn dsk flag hops nhip pre**
- **assumes "\text{invalidate rt dests dip} = \text{ Some (dsn, dsk, flag, hops, nhip, pre)}"**
- **shows "dsn = (\text{ case dests dip of None } \Rightarrow \pi_2(\text{the (rt dip)}) | \text{ Some rsn } \Rightarrow \text{ rsn})"**
- **and "dsk = \pi_3(\text{the (rt dip)})"**
- **and "flag = (\text{ if dests dip = None then } \pi_4(\text{the (rt dip)}) \text{ else inv})"**
- **and "hops = \pi_5(\text{the (rt dip)})"**
- **and "nhip = \pi_6(\text{the (rt dip)})"**
- **and "pre = \pi_7(\text{the (rt dip)})""**
- **using assms unfolding invalidate_def**
- **by (cases "\text{rt dip}\", clarsimp, cases "\text{dests dip}\") auto**

**lemma proj3_inv:**

\[
\forall \text{ dip rt dests. dip} \in kD(\text{rt}) \implies \pi_3(\text{the (invalidate rt dests dip)}) = \pi_3(\text{the (rt dip)})
\]

- **by (clarsimp simp: invalidate_def kD_def split: option.split)**

**lemma dests_iD_invalidate [simp]:**

- **assumes "\text{dests ip} = \text{ Some rsn}"**
- **and "\text{ip} \in kD(\text{rt})"**
- **shows "\text{ip} \in iD(\text{invalidate rt dests})"**
- **using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def**
- **by (clarsimp split: option.split)**

### 4.1.5 Route Requests

Generate a fresh route request identifier.

**definition nrreqid :: "(\text{ip} \times \text{rreqid}) \text{ set} \Rightarrow \text{ip} \Rightarrow \text{rreqid}"**

where "nrreqid \text{ rreqs ip} = \text{ Max (\{n. (ip, n) \in \text{rreqs}\} \cup \{0\}) + 1}"

### 4.1.6 Queued Packets

Functions for sending data packets.

- **type_synonym store = "\text{ip} \to (\text{p} \times \text{data list})"**

- **definition sigma_queue :: "\text{store} \Rightarrow \text{ip} \Rightarrow \text{data list}"**

where "\sigma_{\text{queue}}(\text{store, dip}) \equiv \text{ case store dip of None } \Rightarrow [] | \text{ Some (p, q) } \Rightarrow q"

- **definition qD :: "\text{store} \Rightarrow \text{ip set}"**

where "qD \equiv \text{dom}"

- **definition add :: "\text{data} \Rightarrow \text{ip} \Rightarrow \text{store} \Rightarrow \text{store}"**

where "add \text{ d dip store} \equiv \text{ case store dip of}"

\[
\text{None } \Rightarrow \text{ store (dip } \Rightarrow (\text{req, [d]}))
\]
Some \((p, q) \Rightarrow store (dip \mapsto (p, q @ [d])))"
The \textit{msg} type models the different messages used within AODV. The instantiation as a \textit{msg} is a technicality due to the special treatment of \textit{newpkt} messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

\textbf{definition} \textit{rreq} :: \texttt{"nat \times rreqid \times ip \times sqn \times k \times ip \times sqn \times ip \times bool \Rightarrow msg"}
where \texttt{"rreq \equiv \lambda(hops, rreqid, dip, dsn, dsk, oip, osn, sip, handled). Rreq hops rreqid dip dsn dsk oip osn sip handled"}

\textbf{lemma} \textit{rreq_simp} [simp]:
\texttt{"rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip, handled) = Rreq hops rreqid dip dsn dsk oip osn sip handled"}
unfolding \textit{rreq_def} by simp

\textbf{definition} \textit{rrep} :: \texttt{"nat \times ip \times sqn \times ip \Rightarrow msg"}
where \texttt{"rrep \equiv \lambda(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"}

\textbf{lemma} \textit{rrep_simp} [simp]:
\texttt{"rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"}
unfolding \textit{rrep_def} by simp

\textbf{definition} \textit{rerr} :: \texttt{"(ip \Rightarrow sqn) \times ip \Rightarrow msg"}
where \texttt{"rerr \equiv \lambda(dests, sip). Rerr dests sip"}

\textbf{lemma} \textit{rerr_simp} [simp]:
\texttt{"rerr(dests, sip) = Rerr dests sip"}
unfolding \textit{rerr_def} by simp

\textbf{lemma} \textit{not_eq_newpkt_rreq} [simp]:
\texttt{"\neg eq_newpkt (Rreq hops rreqid dip dsn dsk oip osn sip handled)"
unfolding \textit{eq_newpkt_def} by simp

\textbf{lemma} \textit{not_eq_newpkt_rrep} [simp]:
\texttt{"\neg eq_newpkt (Rrep hops dip dsn oip sip)"
unfolding \textit{eq_newpkt_def} by simp

\textbf{lemma} \textit{not_eq_newpkt_rerr} [simp]:
\texttt{"\neg eq_newpkt (Rerr dests sip)"
unfolding \textit{eq_newpkt_def} by simp

\textbf{lemma} \textit{not_eq_newpkt_pkt} [simp]:
\texttt{"\neg eq_newpkt (Pkt d dip sip)"
unfolding \textit{eq_newpkt_def} by simp

\textbf{definition} \textit{pkt} :: \texttt{"data \times ip \times ip \Rightarrow msg"}
where \texttt{"pkt \equiv \lambda(d, dip, sip). Pkt d dip sip"}

\textbf{lemma} \textit{pkt_simp} [simp]:
\texttt{"pkt(d, dip, sip) = Pkt d dip sip"}
unfolding \textit{pkt_def} by simp

end

4.3 The AODV protocol

\textbf{theory} \textit{D_Aodv}
\textbf{imports} \textit{D_Aodv_Data D_Aodv_Message}
\texttt{AWN.AWN_SOS_Labels AWN.AWN_Invariants}
\textbf{begin}

4.3.1 Data state

\textbf{record} \textit{state} =
\texttt{ip :: "ip"}
\texttt{sn :: "sqn"}
\texttt{rt :: "rt"}
\texttt{rreqs :: "(ip \times rreqid) set"}

348
abbreviation aodv_init :: "ip ⇒ state"
where "aodv_init i ≡ (
  ip = i,
  sn = 1,
  rt = Map.empty,
  rreqs = {},
  store = Map.empty,
  msg = (SOME x. True),
  data = (SOME x. True),
  dests = (SOME x. True),
  pre = (SOME x. True),
  rreqid = (SOME x. True),
  dip = (SOME x. True),
  oip = (SOME x. True),
  hops = (SOME x. True),
  dsn = (SOME x. True),
  dsk = (SOME x. True),
  osn = (SOME x. True),
  sip = (SOME x. x ≠ i),
  handled = (SOME x. True)
)"

lemma some_neq_not_eq [simp]: "¬((SOME x :: nat. x ≠ i) = i)"
by (subst some_eq_ex) (metis zero_neq_numeral)

definition clear_locals :: "state ⇒ state"
where "clear_locals ξ = ξ (!)
  msg := (SOME x. True),
  data := (SOME x. True),
  dests := (SOME x. True),
  pre := (SOME x. True),
  rreqid := (SOME x. True),
  dip := (SOME x. True),
  oip := (SOME x. True),
  hops := (SOME x. True),
  dsn := (SOME x. True),
  dsk := (SOME x. True),
  osn := (SOME x. True),
  sip := (SOME x. x ≠ ip ξ),
  handled := (SOME x. True)
)"

lemma clear_locals_sip_not_ip [simp]: "¬(sip (clear_locals ξ) = ip ξ)"
unfolding clear_locals_def by simp

lemma clear_locals_but_not_globals [simp]:
  "ip (clear_locals ξ) = ip ξ"
4.3.2 Auxilliary message handling definitions

definition is_newpkt
where "is_newpkt ξ ≡ case msg ξ of
                Newpkt data' dip' ⇒ { ξ[|data := data', dip := dip'|] }
                   _ ⇒ {}"

definition is_pkt
where "is_pkt ξ ≡ case msg ξ of
       Pkt data' dip' oip' ⇒ { ξ[|data := data', dip := dip', oip := oip'|] }
                      _ ⇒ {}"

definition is_rreq
where "is_rreq ξ ≡ case msg ξ of
        Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' handled' ⇒
        { ξ[|hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
                       dsk := dsk', oip := oip', osn := osn', sip := sip',
                       handled := handled'|] }
                   _ ⇒ {}"

lemma is_rreq_asm [dest!]:
  assumes "ξ' ∈ is_rreq ξ"
  shows "(∃hops' rreqid' dip' dsn' dsk' oip' osn' sip' handled'.
        msg ξ = Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' handled' ∧
                   dsk := dsk', oip := oip', osn := osn', sip := sip',
                   handled := handled' |])"
  using assms unfolding is_rreq_def
  by (cases "msg ξ") simp_all

definition is_rrep
where "is_rrep ξ ≡ case msg ξ of
       Rrep hops' dip' dsn' oip' sip' ⇒
         _ ⇒ {}"

lemma is_rrep_asm [dest!]:
  assumes "ξ' ∈ is_rrep ξ"
  shows "(∃hops' dip' dsn' oip' sip'.
        msg ξ = Rrep hops' dip' dsn' oip' sip' ∧
  using assms unfolding is_rrep_def
  by (cases "msg ξ") simp_all

definition is_rerr
where "is_rerr ξ ≡ case msg ξ of
       Rerr dests' sip' ⇒ { ξ[|dests := dests', sip := sip'|] }
         _ ⇒ {}"

lemma is_rerr_asm [dest!]:
  assumes "ξ' ∈ is_rerr ξ"
  shows "(∃dests' sip'.
        msg ξ = Rerr dests' sip' ∧
        ξ' = ξ[|dests := dests', sip := sip'|])"
  using assms unfolding is_rerr_def
  by (cases "msg ξ") simp_all

lemmas is_msg_defs =
lemma is_msg_inv_ip [simp]:
    "ξ' ∈ is_rerr ξ → ip ξ' = ip ξ"
    "ξ' ∈ is_rrep ξ → ip ξ' = ip ξ"
    "ξ' ∈ is_rreq ξ → ip ξ' = ip ξ"
    "ξ' ∈ is_pkt ξ → ip ξ' = ip ξ"
    "ξ' ∈ is_newpkt ξ → ip ξ' = ip ξ"
    unfolding is_msg_defs
    by (cases "msg ξ", clarsimp+) +

lemma is_msg_inv_sn [simp]:
    "ξ' ∈ is_rerr ξ → sn ξ' = sn ξ"
    "ξ' ∈ is_rrep ξ → sn ξ' = sn ξ"
    "ξ' ∈ is_rreq ξ → sn ξ' = sn ξ"
    "ξ' ∈ is_pkt ξ → sn ξ' = sn ξ"
    "ξ' ∈ is_newpkt ξ → sn ξ' = sn ξ"
    unfolding is_msg_defs
    by (cases "msg ξ", clarsimp+) +

lemma is_msg_inv_rt [simp]:
    "ξ' ∈ is_rerr ξ → rt ξ' = rt ξ"
    "ξ' ∈ is_rrep ξ → rt ξ' = rt ξ"
    "ξ' ∈ is_rreq ξ → rt ξ' = rt ξ"
    "ξ' ∈ is_pkt ξ → rt ξ' = rt ξ"
    "ξ' ∈ is_newpkt ξ → rt ξ' = rt ξ"
    unfolding is_msg_defs
    by (cases "msg ξ", clarsimp+) +

lemma is_msg_inv_rreqs [simp]:
    "ξ' ∈ is_rerr ξ → rreqs ξ' = rreqs ξ"
    "ξ' ∈ is_rrep ξ → rreqs ξ' = rreqs ξ"
    "ξ' ∈ is_rreq ξ → rreqs ξ' = rreqs ξ"
    "ξ' ∈ is_pkt ξ → rreqs ξ' = rreqs ξ"
    "ξ' ∈ is_newpkt ξ → rreqs ξ' = rreqs ξ"
    unfolding is_msg_defs
    by (cases "msg ξ", clarsimp+) +

lemma is_msg_inv_store [simp]:
    "ξ' ∈ is_rerr ξ → store ξ' = store ξ"
    "ξ' ∈ is_rrep ξ → store ξ' = store ξ"
    "ξ' ∈ is_rreq ξ → store ξ' = store ξ"
    "ξ' ∈ is_pkt ξ → store ξ' = store ξ"
    "ξ' ∈ is_newpkt ξ → store ξ' = store ξ"
    unfolding is_msg_defs
    by (cases "msg ξ", clarsimp+) +

lemma is_msg_inv_sip [simp]:
    "ξ' ∈ is_pkt ξ → sip ξ' = sip ξ"
    "ξ' ∈ is_newpkt ξ → sip ξ' = sip ξ"
    unfolding is_msg_defs
    by (cases "msg ξ", clarsimp+) +

4.3.3 The protocol process

datatype pseqp =
  PAodv
| PNewPkt
| PPkt
| PRreq
| PRrep
| PRerr

fun nat_of_seqp :: "pseqp ⇒ nat"
where

"nat_of_seqp PAodv = 1"
| "nat_of_seqp PPkt = 2"
| "nat_of_seqp PNewPkt = 3"
| "nat_of_seqp PRreq = 4"
| "nat_of_seqp PRrep = 5"
| "nat_of_seqp PRerr = 6"

instantiation "pseqp" :: ord
begin
  definition less_eq_seqp [iff]: "l1 \leq l2 = (nat_of_seqp l1 \leq nat_of_seqp l2)"
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
instance ..
end

abbreviation AODV
where
  "AODV \equiv \lambda_. [clear_locals] call(PAodv)"
abbreviation PKT
where
  "PKT args \equiv
  \[
  \xi. let (data, dip, oip) = args \xi in
  (clear_locals \xi) () data := data, dip := dip, oip := oip []
  call(PPkt)"
abbreviation NEWPKT
where
  "NEWPKT args \equiv
  \[
  \xi. let (data, dip) = args \xi in
  (clear_locals \xi) () data := data, dip := dip []
  call(PNewPkt)"

abbreviation RREQ
where
  "RREQ args \equiv
  \[
  \xi. let (hops, rreqid, dip, dsn, dsk, oip, osn, sip, handled) = args \xi in
  (clear_locals \xi) () hops := hops, rreqid := rreqid, dip := dip,
  call(PRreq)"

abbreviation RREP
where
  "RREP args \equiv
  \[
  \xi. let (hops, dip, dsn, oip, sip) = args \xi in
  (clear_locals \xi) () hops := hops, dip := dip, dsn := dsn,
  oip := oip, sip := sip []
  call(PRrep)"

abbreviation RERR
where
  "RERR args \equiv
  \[
  \xi. let (dests, sip) = args \xi in
  (clear_locals \xi) () dests := dests, sip := sip []
  call(PRerr)"

fun \Gamma_AODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
  "\Gamma_AODV PAodv = labelled PAodv (receive(\lambda msg' \xi. \xi () msg := msg' [])).
  (\is_newpkt) NEWPKT(\lambda \xi. (data \xi, ip \xi))
  \oplus (\is_pkt) PKT(\lambda \xi. (data \xi, dip \xi, oip \xi))
  \oplus (\is_rreq)"
\[
\begin{align*}
\text{λ} \xi \rightarrow & \text{update (rt } \xi \text{) } \langle \text{sip } \xi, \text{unq, val, } 0, \text{unq, val, } 1, \text{sip } \xi, \text{, {}, } \rangle \text{; } \\
\text{RREQ}(\lambda \xi . \langle \text{hops } \xi, \text{rreqid } \xi, \text{dip } \xi, \text{dsn } \xi, \text{dip } \xi, \text{osn } \xi, \text{, sip } \xi, \text{, handled } \xi \rangle) \\
\text{is_rerr} & \\
\begin{align*}
\text{λ} \xi \rightarrow & \text{update (rt } \xi \text{) } \langle \text{sip } \xi, \text{unq, val, } 0, \text{unq, val, } 1, \text{sip } \xi, \text{, {}, } \rangle \text{; } \\
\text{RREP}(\lambda \xi . \langle \text{hops } \xi, \text{dip } \xi, \text{dsn } \xi, \text{, oip } \xi, \text{, sip } \xi \rangle) \\
\text{is_rerr} & \\
\begin{align*}
\text{λ} \xi \rightarrow & \text{update (rt } \xi \text{) } \langle \text{sip } \xi, \text{unq, val, } 0, \text{unq, val, } 1, \text{sip } \xi, \text{, {}, } \rangle \text{; } \\
\text{RREP}(\lambda \xi . \langle \text{hops } \xi, \text{dip } \xi, \text{dsn } \xi, \text{, oip } \xi, \text{, sip } \xi \rangle) \\
\end{align*}
\end{align*}
\end{align*}
\]
Γ_{AODV} PRreq = labelled PRreq

\( \langle \xi, (\text{oip}_\xi, \text{rreqid}_\xi) \in \text{rreqs} \rangle \)

\( \text{AODV}() \)

\( \oplus \langle \xi, (\text{oip}_\xi, \text{rreqid}_\xi) \notin \text{rreqs} \rangle \)

\( \langle \xi, \emptyset \rangle \)

\( \begin{align*}
\xi & : \text{rt} := \text{update} (\text{rt} \xi) (\text{oip}_\xi) (\text{osn}_\xi, \text{kno}, \text{val}, \text{hops} \xi + 1, \text{sip}_\xi, \{\}) \}\] \\
\xi & : \text{rreqs} := \text{rreqs} \xi \cup \{(\text{oip}_\xi, \text{rreqid}_\xi)\} \]\]

( \( \langle \xi, \text{handled} \xi = \text{False} \rangle \) \\
( \( \langle \xi, \text{dip} \xi = \text{ip} \xi \rangle \)

\( \begin{align*}
\xi & : \text{sn} := \text{max} (\text{sn} \xi, (\text{dsn} \xi)) \\
\text{unicast}(\lambda \xi. \text{the} (\text{nhop} (\text{rt} \xi) (\text{oip} \xi)), \lambda \xi. \text{rrep}(0, \text{dip} \xi, \text{sn} \xi, \text{oip} \xi, \text{ip} \xi)). \text{AODV}() \\
\end{align*} \)

\( \begin{align*}
\xi & : \text{dests} := (\lambda \text{rip}. \text{if} (\text{rip} \in \text{vD} (\text{rt} \xi) \land \text{nhop} (\text{rt} \xi) \text{rip} = \text{nhop} (\text{rt} \xi) (\text{oip} \xi)) \\
& \quad \text{then Some (inc (\text{sqn} (\text{rt} \xi) \text{rip})) else None}) \]\]

\( \begin{align*}
\xi & : \text{rt} := \text{invalidate} (\text{rt} \xi) (\text{dests} \xi) \]\]

\( \begin{align*}
\xi & : \text{store} := \text{setRFF} (\text{store} \xi) (\text{dests} \xi) \]\]

\( \begin{align*}
\xi & : \text{pre} := \bigcup \{ \text{the} (\text{precs} (\text{rt} \xi) \text{rip}) \mid \text{rip} \in \text{dom} (\text{dests} \xi) \} \]\]

\( \begin{align*}
\xi & : \text{dests} := (\lambda \text{rip}. \text{if} ((\text{dests} \xi) \text{rip} \neq \text{None} \land \text{the} (\text{precs} (\text{rt} \xi) \text{rip}) \neq \{\}) \\
& \quad \text{then (dests} \xi \text{rip} \text{else None}) \}\]\]

\( \text{groupcast}(\lambda \xi. \text{pre} \xi, \lambda \xi. \text{rerr}(\text{dests} \xi, \text{ip} \xi)). \text{AODV}() \)

\( \oplus \langle \xi, \text{dip} \xi \neq \text{ip} \xi \rangle \)

\( \begin{align*}
\xi & : \text{dip} \xi \in \text{vD} (\text{rt} \xi) \land \text{dsn} \xi \leq \text{sqn} (\text{rt} \xi) (\text{dip} \xi) \wedge \text{sqnf} (\text{rt} \xi) (\text{dip} \xi) = \text{kno} \\
\xi & : \text{rt} := \text{the} (\text{addpreRT} (\text{rt} \xi) (\text{dip} \xi) (\text{osn} \xi)) \]\]

\( \begin{align*}
\xi & : \text{store} := \text{setRFF} (\text{store} \xi) (\text{dests} \xi) \]\]

\( \begin{align*}
\xi & : \text{pre} := \bigcup \{ \text{the} (\text{precs} (\text{rt} \xi) \text{rip}) \mid \text{rip} \in \text{dom} (\text{dests} \xi) \} \]\]

\( \begin{align*}
\xi & : \text{dests} := (\lambda \text{rip}. \text{if} ((\text{dests} \xi) \text{rip} \neq \text{None} \land \text{the} (\text{precs} (\text{rt} \xi) \text{rip}) \neq \{)) \\
& \quad \text{then (dests} \xi \text{rip} \text{else None}) \}\]\]

\( \text{groupcast}(\lambda \xi. \text{pre} \xi, \lambda \xi. \text{rerr}(\text{dests} \xi, \text{ip} \xi)). \text{AODV}() \)

\( \oplus \langle \xi, \text{dip} \xi \notin \text{vD} (\text{rt} \xi) \lor \text{sqn} (\text{rt} \xi) (\text{dip} \xi) < \text{dsn} \xi \lor \text{sqnf} (\text{rt} \xi) (\text{dip} \xi) = \text{unk} \rangle \\
\text{broadcast}(\lambda \xi. \text{rreq}(\text{hops} \xi + 1, \text{rreqid} \xi, \text{dip} \xi, \text{dsn} \xi, \text{dsk} \xi, \text{oip} \xi, \text{osn} \xi, \text{ip} \xi, \text{False})). \text{AODV}() \)

\( \oplus \langle \xi, \text{handled} \xi = \text{True} \rangle \\
\text{broadcast}(\lambda \xi. \text{rreq}(\text{hops} \xi + 1, \text{rreqid} \xi, \text{dip} \xi, \text{dsn} \xi, \text{dsk} \xi, \text{oip} \xi, \text{osn} \xi, \text{ip} \xi, \text{True})). \text{AODV}() \)

\( \)"
lemma \( \Gamma \cdot \xi \cdot \| \ \text{rt} := \text{the (addpreRT (rt \xi) (\text{nhop (rt \xi) (dip \xi)}))} \newline \{ \text{the (nhop (rt \xi) (oip \xi))} \} \) \] unicast (\( \lambda \xi \cdot \{ \text{the (nhop (rt \xi) (oip \xi)), \lambda \xi \cdot \text{rrep(hops \xi + 1, dip \xi, dsn \xi, oip \xi, ip \xi)} \}) \) . \ AODV() \)

\[ \begin{align*}
\xi \cdot \xi \cdot & \ \text{dests} ::= (\lambda \text{rip}. \text{if (rip} \in \text{vD (rt \xi) \land \text{nhop (rt \xi) rip} = \text{nhop (rt \xi) (oip \xi)})) \\
& \quad \text{then Some (inc (sqn (rt \xi) rip)) else None}) \] 
\end{align*} \]

lemma \( \Gamma \cdot \xi \cdot \| \ \text{store} := \text{setRRF (store \xi) (dests \xi)} \) \] 

\[ \begin{align*}
\xi \cdot \xi \cdot & \ \text{pre} ::= \bigcup \{ \text{the (precs (rt \xi) rip) \land rip. rip} \in \text{dom (dests \xi)} \} \] 
\end{align*} \]

lemma \( \Gamma \cdot \xi \cdot \| \ \text{dests} ::= (\lambda \text{rip}. \text{if ((dests \xi) rip} \neq \text{None} \land \text{the (precs (rt \xi) rip)} \neq \{\}) \\
& \quad \text{then (dests \xi) rip else None}) \] \]

\[ \begin{align*}
groupcast (\lambda \xi \cdot \text{pre} \xi, \lambda \xi \cdot \text{rerr(dests \xi, ip \xi)}). \ AODV() \\
\end{align*} \]

lemma \( \Gamma \cdot \xi \cdot \| \ \text{rt} = \text{update (rt \xi) (dip \xi) (dsn \xi, kno, val, hops \xi + 1, sip \xi, \{\})} \) \] \[
\ AODV() \]

quote "\( \Gamma_{AODV} \) \ PErr = labelled \ PErr \ {
\begin{align*}
\xi \cdot \xi \cdot & \ \text{dests} ::= (\lambda \text{rip}. \text{case (dests \xi) rip of None} \Rightarrow \text{None}) \\
& \quad \text{Some rsn} \Rightarrow \text{if rip} \in \text{vD (rt \xi) \land the (nhop (rt \xi) rip) = sip \xi} \\
& \quad \quad \land \text{sqn (rt \xi) rip} < \text{rsn then Some rsn else None}) \] 
\end{align*} \]

lemma \( \Gamma_{AODV} \cdot \text{simps simp del, code del} \)

fun \( \Gamma_{AODV} \cdot \text{simps simp, code} = \Gamma_{AODV} \cdot \text{simps simplified} \)

where "\( \Gamma_{AODV} \cdot \text{skeleton PAdv} = \text{seqp_skeleton (\( \Gamma_{AODV} \) PAdv)} \) " \[
/ "\( \Gamma_{AODV} \cdot \text{skeleton PNewPkt} = \text{seqp_skeleton (\( \Gamma_{AODV} \) PNewPkt)} \) " \[
/ "\( \Gamma_{AODV} \cdot \text{skeleton PPkt} = \text{seqp_skeleton (\( \Gamma_{AODV} \) PPkt)} \) " \[
/ "\( \Gamma_{AODV} \cdot \text{skeleton PRreq} = \text{seqp_skeleton (\( \Gamma_{AODV} \) PRreq)} \) " \[
/ "\( \Gamma_{AODV} \cdot \text{skeleton PRrep} = \text{seqp_skeleton (\( \Gamma_{AODV} \) PRrep)} \) " \[
/ "\( \Gamma_{AODV} \cdot \text{skeleton PRerr} = \text{seqp_skeleton (\( \Gamma_{AODV} \) PRerr)} \) " \]

lemma \( \Gamma_{AODV} \cdot \text{skeleton_wf simp]: \)

"\( \text{wellformed (\( \Gamma_{AODV} \) skeleton)} \) "

proof (rule, intro allI)

fix pn pn'

show "\( \text{call(pn') \notin ctermsl (\( \Gamma_{AODV} \) skeleton pn)} \) "
      by (cases pn) simp_all

qed

declare \( \Gamma_{AODV} \cdot \text{skeleton.simps simp del, code del} \)

lemma \( \Gamma_{AODV} \cdot \text{skeleton.simps simp, code} = \Gamma_{AODV} \cdot \text{skeleton.simps simplified \( \Gamma_{AODV} \) simps seqp_skeleton.simps} \)

lemma aodv_proc_cases [dest]:

fixes p pn

shows "\( p \in ctermsl (\( \Gamma_{AODV} \) pn) \implies \)"
\[
(p \in ctermsl (\( \Gamma_{AODV} \) PAdv) \lor p \in ctermsl (\( \Gamma_{AODV} \) PNewPkt) \lor p \in ctermsl (\( \Gamma_{AODV} \) PPkt) \lor p \in ctermsl (\( \Gamma_{AODV} \) PRreq) \lor
\]
\[ p \in \text{ctermsl } \left( \Gamma_{AODV} \text{PRrep} \right) \lor \]
\[ p \in \text{ctermsl } \left( \Gamma_{AODV} \text{PRerr} \right) \]

by (cases \( pn \)) simp_all

definition \( \sigma_{AODV} \) :: \( \text{ip} \Rightarrow \left( \text{state} \times \left( \text{state}, \text{msg}, \text{pseqp}, \text{pseqp label} \right) \text{ seqp} \right) \text{ set} \)

where \( \sigma_{AODV} i \equiv \{(\text{aodv_init } i, \Gamma_{AODV} \text{PAodv})\} \)

abbreviation \( \text{paodv} \) :: \( \text{ip} \Rightarrow \left( \text{state} \times \left( \text{state}, \text{msg}, \text{pseqp}, \text{pseqp label} \right) \text{ seqp}, \text{msg seq_action} \right) \text{ automaton} \)

where \( \text{paodv } i \equiv (| \text{init } = \sigma_{AODV} i, \text{trans } = \text{seqp_sos } \Gamma_{AODV} |) \)

lemma \( \text{aodv_trans} \): \( \text{trans } (\text{paodv } i) = \text{seqp_sos } \Gamma_{AODV} \)

by simp

lemma \( \text{aodv_control_within} \) [simp]: \( \text{control_within } \Gamma_{AODV} (\text{init } (\text{paodv } i)) \)

unfolding \( \sigma_{AODV} \text{def} \) by (rule control_within I) (auto simp del: \( \Gamma_{AODV} \text{simps} \))

lemma \( \text{aodv_wf} \) [simp]: \( \text{wellformed } \Gamma_{AODV} \)

proof (rule, intro allI)

fix \( pn pn' \)

show \( \text{call}(pn') \notin \text{ctermsl } \left( \Gamma_{AODV} pn \right) \)

by (cases \( pn \)) simp_all

qed

lemmas \( \text{aodv_labels_not_empty} \) [simp] = labels_not_empty [OF \( \text{aodv_wf} \)]

lemma \( \text{aodv_ex_label} \) [intro]: \( \exists \, l \in \text{labels } \Gamma_{AODV} \ p \)

by (metis aodv_labels_not_empty all_not_in_conv)

lemma \( \text{aodv_ex_labelE} \) [elim]:

assumes \( \forall l \in \text{labels } \Gamma_{AODV} \ p. P \ l \ p \)

and \( \exists \, l. P \ l \ p \Longrightarrow Q \)

shows \( Q \)

using assms by (metis aodv_ex_label)

lemma \( \text{aodv_simple_labels} \) [simp]: \( \text{simple_labels } \Gamma_{AODV} \)

proof

fix \( pn \ p \)

assume \( \text{p\in\subterms}(\Gamma_{AODV} \ pn) \)

thus \( \exists \, l. \text{labels } \Gamma_{AODV} \ p = \{l\} \)

by (cases \( pn \)) (simp_all cong: seqp_congs | elim disjE)+

qed

lemma \( \sigma_{AODV} \text{labels} \) [simp]: \( (\xi, p) \in \sigma_{AODV} i \Longrightarrow \text{labels } \Gamma_{AODV} \ p = \{\text{PAodv}:-0\} \)

unfolding \( \sigma_{AODV} \text{def} \) by simp

lemma \( \text{aodv_init_kD_empty} \) [simp]:

\( (\xi, p) \in \sigma_{AODV} i \Longrightarrow \text{kD } (\text{rt } \xi) = \{\} \)

unfolding \( \sigma_{AODV} \text{def} \) \text{kD def by simp}

lemma \( \text{aodv_init_sip_not_ip} \) [simp]: \( \neg(\text{sip } (\text{aodv_init } i) = i) \) by simp

lemma \( \text{aodv_init_sip_not_ip'} \) [simp]:

assumes \( (\xi, p) \in \sigma_{AODV} i \)

shows \( \text{sip } \xi \neq \text{ip } \xi \)

using assms unfolding \( \sigma_{AODV} \text{def} \) by simp

lemma \( \text{aodv_init_sip_not_i} \) [simp]:

assumes \( (\xi, p) \in \sigma_{AODV} i \)

shows \( \text{sip } \xi \neq i \)

using assms unfolding \( \sigma_{AODV} \text{def} \) by simp

356
lemma clear_locals_sip_not_ip':
  assumes "ip \( \xi \) = i"
  shows "\( \neg (\text{sip (clear_locals } \xi \text{) = i}) \)"
  using assms by auto

Stop the simplifier from descending into process terms.

declare seq_congs [cong]

Configure the main invariant tactic for AODV.

declare 
\[\Gamma_{AODV} \text{_simps [cterms\_env]}
\]
aodv_proc_cases [cterms\_cases]

seq_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms\_intros]

seq_step_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms\_intros]

end

4.4 Invariant assumptions and properties

theory D_Aodv_Predicates
imports D_Aodv
begin

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

abbreviation not_Pkt :: "msg \\Rightarrow\ bool"
where "not_Pkt m \equiv \text{ case } m \text{ of } Pkt \_ \_ \_ \Rightarrow False \mid \_ \Rightarrow True"

definition msg_sender :: "msg \\Rightarrow\ ip"
where "msg_sender m \equiv \text{ case } m \text{ of } Rreq \_ \_ \_ \_ \_ \_ \_ \_ icp \_ \Rightarrow icp 
\mid Rrep \_ \_ \_ icp \Rightarrow icp 
\mid Rerr \_ icp \Rightarrow icp 
\mid Pkt \_ \_ icp \Rightarrow icp"

lemma msg_sender_simps [simp]:
  \("\forall hops rreqid dip dsn dsk oip osn sip handled. 
msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip handled) = sip"
  "\forall hops dip dsn icp. msg_sender (Rreq hops dip dsn icp) = sip"
  "\forall dests icp. msg_sender (Rreq dests icp) = sip"
  "\forall d dip icp. msg_sender (Pkt d dip icp) = sip"
unfolding msg_sender_def by simp_all

definition msg_zhops :: "msg \\Rightarrow\ bool"
where "msg_zhops m \equiv \text{ case } m \text{ of } 
Rreq hopsc \_ dipc \_ oipc \_ sipc \_ \Rightarrow hopsc = 0 \rightarrow oipc = sipc 
\mid Rrep hopsc dipc \_ sipc \Rightarrow hopsc = 0 \rightarrow dipc = sipc 
\mid _ \Rightarrow True"

lemma msg_zhops_simps [simp]:
  "\forall hops rreqid dip dsn dsk oip osn sip handled. 
msg_zhops (Rreq hops rreqid dip dsn dsk oip osn sip handled) = (hops = 0 \rightarrow oip = sip)"
  "\forall hops dsn icp. msg_zhops (Rreq hops dip dsn icp) = (hops = 0 \rightarrow dip = sip)"
  "\forall dests icp. msg_zhops (Rreq dests icp) = True"
  "\forall d dip. msg_zhops (Newpkt d dip) = True"
unfolding msg_zhops_def by simp_all

definition rreq_rrep_sn :: "msg \\Rightarrow\ bool"
where "rreq_rrep_sn m \equiv \text{ case } m \text{ of } 
Rreq hopsdnc \_ \_ oipsc \_ \_ \_ \_ osnc \_ \_ \_ \_ \Rightarrow osnc \geq 1 
\mid Rrep \_ \_ dscn \_ \_ \_ \_ \_ \_ \_ \_ \_ \Rightarrow dscn \geq 1 
\mid _ \Rightarrow True"
lemma rreq_rrep_sn_simps [simp]:
"∀hops rreqid dip dsn oip osn sip handled.
  rreq_rrep_sn (Rreq hops rreqid dip dsn oip osn sip handled) = (osn ≥ 1)"
"∀hops dip dsn oip sip.
  rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn ≥ 1)"
"∀dests sip.
  rreq_rrep_sn (Rerr dests sip) = True"
"∀d dip.
  rreq_rrep_sn (Newpkt d dip) = True"
"∀d dip sip.
  rreq_rrep_sn (Pkt d dip sip) = True"
unfolding rreq_rrep_sn_def by simp_all

definition rreq_rrep_fresh :: "rt ⇒ msg ⇒ bool"
where "rreq_rrep_fresh crt m ≡
  case m of
    Rreq hopsc _ _ _ _ oipc osnc ipcc _ ⇒
      (∀ripc ∈ kD(crt).
        (∀ripc ∈ iD(crt).
          the (destsc ripc) = sqn crt ripc))
    Rrep hopsc dipc dsnc _ ipcc ⇒
      (∀dipc ∈ kD(crt).
        (∀dipc ∈ iD(crt).
          the (dipc ripc) = sqn crt dipc))
    _ ⇒ True"

lemma rreq_rrep_fresh [simp]:
"∀hops rreqid dip dsn oip osn sip handled.
  rreq_rrep_fresh crt (Rreq hops rreqid dip dsn oip osn sip handled) =
    (sip ≠ oip → oip ∈ kD(crt))
    (∀dipc ∈ kD(crt).
      (∀dipc ∈ iD(crt).
        the (dipc ripc) = sqn crt dipc))
    (∀ripc ∈ iD(crt).
      the (destsc ripc) = sqn crt ripc))
    (∀ripc ∈ iD(crt).
      the (destsc ripc) = sqn crt ripc))
"∀hops dip dsn oip sip.
  rreq_rrep_fresh crt (Rrep hops dip dsn oip sip) =
    (∀dipc ∈ kD(crt).
      (∀dipc ∈ iD(crt).
        the (dipc ripc) = sqn crt dipc))
    (∀dipc ∈ iD(crt).
      the (dipc ripc) = sqn crt dipc))
    (∀ripc ∈ iD(crt).
      the (destsc ripc) = sqn crt ripc))
"∀dests sip.
  rreq_rrep_fresh crt (Rerr dests sip) = True"
"∀d dip.
  rreq_rrep_fresh crt (Newpkt d dip) = True"
"∀d dip sip.
  rreq_rrep_fresh crt (Pkt d dip sip) = True"
unfolding rreq_rrep_fresh_def by simp_all

definition rerr_invalid :: "rt ⇒ msg ⇒ bool"
where "rerr_invalid crt m ≡
  case m of
    Rerr destsc _ ⇒
      (∀ripc ∈ dom(destsc).
        (∀ripc ∈ dom(destsc).
          the (destsc ripc) = sqn crt ripc))
    _ ⇒ True"

lemma rerr_invalid [simp]:
"∀hops rreqid dip dsn dsk oip osn sip handled.
  rerr_invalid crt (Rreq hops rreqid dip dsn dsk oip osn sip handled) = True"
"∀hops dip dsn oip sip.
  rerr_invalid crt (Rrep hops dip dsn oip sip) = True"
"∀dests sip.
  rerr_invalid crt (Rerr dests sip) = (∀ripc ∈ dom(dests).
    ripc ∈ iD(crt))
"∀d dip.
  rerr_invalid crt (Newpkt d dip) = True"
"∀d dip sip.
  rerr_invalid crt (Pkt d dip sip) = True"
unfolding rerr_invalid_def by simp_all

definition initmissing :: "(nat ⇒ state option) × 'a ⇒ (nat ⇒ state) × 'a"
where "initmissing = (∀i. case (fst σ) i of None ⇒ aodv_init i | Some s ⇒ s, snd σ)"

lemma not_in_net_ips_fst_init_missing [simp]:
assumes "i /∈ net_ips σ"
shows "fst (initmissing (netgmap fst σ)) i = aodv_init i"
using assms unfolding initmissing_def by simp

358
lemma \texttt{fst_initmissing_netgmap_pair_fst [simp]}:
"\texttt{fst (initmissing (netgmap (\lambda(p, q). (fst (id p), snd (id p), q)) s))} \\
= \texttt{fst (initmissing (netgmap fst s))}"
\texttt{unfolding initmissing_def by auto}

We introduce a streamlined alternative to \texttt{initmissing} with \texttt{netgmap} to simplify invariant statements and thus facilitate their comprehension and presentation.

lemma \texttt{fst_initmissing_netgmap_default_aodv_init_netlift}:
"\texttt{fst (initmissing (netgmap fst s))} = \texttt{default aodv_init (netlift fst s)}"
\texttt{unfolding initmissing_def default_def \\
by (simp add: \texttt{fst_netgmap_netlift del: One_nat_def)}

definition
\texttt{netglobal :: "((nat ⇒ state) ⇒ bool) ⇒ ((state × 'b) × 'c) net_state ⇒ bool"}
where
\texttt{"netglobal P ≡ (λs. P (default aodv_init (netlift fst s)))"}

end

4.5 Quality relations between routes

theory \texttt{D_Fresher}
imports \texttt{D_Aodv_Data}
begin

4.5.1 Net sequence numbers

On individual routes
definition
\texttt{nsqn_r :: "r ⇒ sqn"}
where
\texttt{"nsqn, r ≡ if π₄(r) = val ∨ π₂(r) = 0 then π₂(r) else (π₂(r) - 1)"

lemma \texttt{nsqnr_def'}:
"\texttt{nsqn, r} = (if π₄(r) = inv then π₂(r) - 1 else π₂(r))"
\texttt{unfolding nsqn_r_def by simp}

lemma \texttt{nsqn_zero [simp]}:
"∀dsn dsk flag hops nhip pre. \texttt{nsqn, (0, dsk, flag, hops, nhip, pre)} = 0"
\texttt{unfolding nsqn_r_def by clarsimp}

lemma \texttt{nsqn_val [simp]}:
"∀dsn dsk hops nhip pre. \texttt{nsqn, (dsn, dsk, val, hops, nhip, pre)} = dsn"
\texttt{unfolding nsqn_r_def by clarsimp}

lemma \texttt{nsqn_inv [simp]}:
"∀dsn dsk hops nhip pre. \texttt{nsqn, (dsn, dsk, inv, hops, nhip, pre)} = dsn - 1"
\texttt{unfolding nsqn_r_def by clarsimp}

lemma \texttt{nsqn_lte_dsn [simp]}:
"∀dsn dsk flag hops nhip pre. \texttt{nsqn, (dsn, dsk, flag, hops, nhip, pre)} ≤ dsn"
\texttt{unfolding nsqn_r_def by clarsimp}

On routes in routing tables

definition
\texttt{nsqn :: "rt ⇒ ip ⇒ sqn"}
where
\texttt{"nsqn ≡ λrt dip. case \texttt{route}(rt, dip) of None ⇒ 0 | Some r ⇒ nsqn_r(r)"

lemma \texttt{nsqn_sqn_def}:
"∀rt dip. \texttt{nsqn rt dip} = (if flag rt dip = Some val ∨ sqn rt dip = 0
then sqn rt dip else sqn rt dip - 1)

unfolding nsqn_def sqn_def by (clarsimp split: option.split)

lemma not_in_kD_nsqn [simp]:
assumes "dip /∈ kD(rt)
shows "nsqn rt dip = 0"
using assms unfolding nsqn_def by simp

lemma kD_nsqn:
assumes "dip ∈ kD(rt)"
shows "nsqn rt dip = nsqn rt (the (σroute(rt, dip)))"
using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

lemma nsqr_r_flag_pred [simp, intro]:
fixes dsn dsk flag hops nhip pre
assumes "P (nsqn rt (dsn, dsk, val, hops, nhip, pre))"
and "P (nsqn rt (dsn, dsk, inv, hops, nhip, pre))"
shows "P (nsqn rt (dsn, dsk, flag, hops, nhip, pre))"
using assms by (cases flag) auto

lemma nsqn_r_addpreRT_inv [simp]:
"∀ rt dip npre dip'. dip ∈ kD(rt) =⇒ nsqn rt (the (the (addpreRT rt dip npre) dip')) = nsqn rt (the (rt dip'))"

unfolding addpreRT_def nsqn_def
by (frule kD_Some) (clarsimp split: option.split)

lemma sqn_nsqn:
"∀ rt dip. sqn rt dip - 1 ≤ nsqn rt dip"

unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma nsqn_sqn: "nsqn rt dip ≤ sqn rt dip"

unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma val_nsqn_sqn [elim, simp]:
assumes "ip ∈ kD(rt)"
and "the (flag rt ip) = val"
shows "nsqn rt ip = sqn rt ip"
using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
assumes "ip ∈ vD(rt)"
shows "nsqn rt ip = sqn rt ip - 1"
proof -
from ⟨ip ∈ vD(rt)⟩ have "ip ∈ kD(rt)"
and "the (flag rt ip) = val" by auto
thus ?thesis ..
qed

lemma inv_nsqn_sqn [elim, simp]:
assumes "ip ∈ kD(rt)"
and "the (flag rt ip) = inv"
shows "nsqn rt ip = sqn rt ip - 1"
using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
assumes "ip ∈ kD(rt)"
shows "nsqn rt ip = sqn rt ip - 1"
proof -
from ⟨ip ∈ kD(rt)⟩ have "ip ∈ kD(rt)"
and "the (flag rt ip) = inv" by auto
thus ?thesis ..
qed

lemma nsqn_update_changed_kno_val [simp]: "∀ rt ip dsn dsk hops nhip.
rt ≠ update rt ip (dsn, kno, val, hops, nhip, {}) 
⇒ nsqn (update rt ip (dsn, kno, val, hops, nhip, {})) ip = dsn

unfolding nsqn_def update_def
by (clarsimp simp: kD_nsqn split: option.split_asm option.split if_split_asm)
(metis fun_upd_triv)

lemma nsqn_addpreRT_inv [simp]:
"∀rt dip npre dip'. dip ∈ kD(rt) ⇒
nsqn (the (addpreRT rt dip npre)) dip' = nsqn rt dip'"
unfolding addpreRT_def nsqn_def nsqn_.def
by (frule kD_Some) (clarsimp split: option.split)

lemma nsqn_update_other [simp]:
fixes dsn dsk flag hops dip nhip pre rt ip
assumes "dip ≠ ip"
shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip, pre)) dip = nsqn rt dip"
using assms unfolding nsqn_def
by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
assumes "dip ∈ kD(rt)"
and "dests dip = Some rsn"
shows "nsqn (invalidate rt dests) dip = rsn - 1"
using assms proof -
from assms obtain dsk hops nhip pre
where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip, pre)"
unfolding invalidate_def
by auto
moreover from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp
ultimately show ?thesis
using ⟨dests dip = Some rsn⟩ by simp
qed

lemma nsqn_invalidate_other [simp]:
assumes "dip∈kD(rt)"
and "dip/∈dom dests"
shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
using assms by (clarsimp simp add: kD_nsqn)

4.5.2 Comparing routes

definition
dresher :: "r ⇒ r ⇒ bool" ("(_/ ⊑ _)"
where
"fresher r r' ≡ ((nsqn, r < nsqn, r') ∨ (nsqn, r = nsqn, r' ∧ π₅(r) ≥ π₅(r')))"

lemma fresherI1 [intro]:
assumes "nsqn, r < nsqn, r'"
shows "r ⊑ r'"
unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
assumes "nsqn, r = nsqn, r'"
and "π₅(r) ≥ π₅(r')"
shows "r ⊑ r'"
unfolding fresher_def using assms by simp

lemma fresherI [intro]:
assumes "(nsqn, r < nsqn, r') ∨ (nsqn, r = nsqn, r' ∧ π₅(r) ≥ π₅(r'))"
shows "r ⊑ r'"
unfolding fresher_def using assms.

lemma fresherE [elim]:
assumes "r ⊑ r'"
and "nsqn r < nsqn r' ⟹ P r r'"
and "nsqn r = nsqn r' ∧ π₅ r ≥ π₅ r' ⟹ P r r'"
shows "P r r'"
using assms unfolding fresher_def by auto

lemma fresher_refl [simp]: "r ⊑ r"
unfolding fresher_def by simp

lemma fresher_trans [elim, trans]:
"[ x ⊑ y; y ⊑ z ] ⟹ x ⊑ z"
unfolding fresher_def by auto

lemma not_fresher_trans [elim, trans]:
"[ ¬(x ⊑ y); ¬(z ⊑ x) ] ⟹ ¬(z ⊑ y)"
unfolding fresher_def by auto

lemma fresher_dsn_flag_hops_const [simp]:
fixes dsn dsk dsk' flag hops nhip nhip' pre pre'
shows "(dsn, dsk, flag, hops, nhip, pre) ⊑ (dsn, dsk', flag, hops, nhip', pre')"
unfolding fresher_def by (cases flag) simp_all

lemma addpre_fresher [simp]: "∀ r npre. r ⊑ (addpre r npre)"
by clarsimp

4.5.3 Comparing routing tables

definition rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where "rt_fresher ≡ λ dip rt rt'. (the (σ_route (rt, dip))) ⊑ (the (σ_route (rt', dip)))"

abbreviation rt_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊑ / _)") [51, 999, 51] 50
where "rt1 ⊑ ip rt2 ≡ rt_fresher ip rt1 rt2"

lemma rt_fresher_I1 [intro]:
assumes "dip ∈ kD rt1"
and "dip ∈ kD rt2"
and "nsqn rt1 dip < nsqn rt2 dip"
shows "(rt1 ⊑ dip rt2) = (nsqn (the (rt1 dip)) < nsqn (the (rt2 dip)) ∨ nsqn (the (rt1 dip)) = nsqn (the (rt2 dip)) ∧ π₅ (the (rt2 dip)) ≥ π₅ (the (rt1 dip)))"
unfolding rt_fresher_def by (simp add: kD_nsqn proj5_eq_dhops)

lemma single_rt_fresher [intro]:
assumes "the (rt1 ip) ⊑ the (rt2 ip)"
shows "rt1 ⊑ ip rt2"
using assms unfolding rt_fresher_def .

lemma rt_fresher_single [intro]:
assumes "rt1 ⊑ ip rt2"
shows "the (rt1 ip) ⊑ the (rt2 ip)"
using assms unfolding rt_fresher_def .

lemma rt_fresher_def2:
assumes "dip ∈ kD rt1"
and "dip ∈ kD rt2"
shows "(rt1 ⊑ dip rt2) = (nsqn rt1 dip < nsqn rt2 dip ∧ the (dhops rt1 dip) ≥ the (dhops rt2 dip))"
using assms unfolding rt_fresher_def by (simp add: kD_nsqn proj5_eq_dhops)

lemma rt_fresherI11 [intro]:
assumes "dip ∈ kD rt1"
and "dip ∈ kD rt2"
and "nsqn rt1 dip < nsqn rt2 dip"

lemma rt_fresherI2 [intro]:
assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "nsqn rt1 dip = nsqn rt2 dip"
  and "the (dhops rt1 dip) ≥ the (dhops rt2 dip)"
shows "rt1 ⊑ dip rt2"
unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherE [elim]:
assumes "rt1 ⊑ dip rt2"
  and "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "[nsqn rt1 dip < nsqn rt2 dip] ∈ P rt1 rt2 dip"
  and "[nsqn rt1 dip = nsqn rt2 dip; the (dhops rt1 dip) ≥ the (dhops rt2 dip)] ∈ P rt1 rt2 dip"
shows "P rt1 rt2 dip"
using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "rt ⊑ dip rt"
unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt3"
shows "rt1 ⊑ dip rt3"
using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
assumes "the (rt dip) ⊑ r"
shows "rt ⊑ dip (λip. if ip = dip then Some r else rt ip)"
using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
"rt_fresh_as ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ (rt2 ⊑ dip rt1)"

abbreviation rt_fresh_as_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/≈_/_") [51, 999, 51] 50)
where
"rt1 ≈i rt2 ≡ rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_ref1 [simp]: "∀rt dip. rt ≈ dip rt"
unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
"∀rt1 rt2 rt3 dip. [ rt1 ≈ dip rt2; rt2 ≈ dip rt3 ] ⇒ rt1 ≈ dip rt3"
unfolding rt_fresh_as_def rt_fresher_def
by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt1"
shows "rt1 ≈ dip rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "the (rt1 dip) ⊑ the (rt2 dip)"
  and "the (rt2 dip) ⊑ the (rt1 dip)"

shows \("rt1 \approx_{\text{dip}} rt2\)\\
using \text{assms unfolding rt\_fresh\_as\_def}\\
by (\text{clarsimp dest!: single\_rt\_fresher})\\

\text{lemma nsqn\_rt\_fresh\_asI:}\\
\text{assumes} \ "\text{dip} \in kD(\text{rt})"\\
and \ "\text{dip} \in kD(\text{rt}')"\\
and \ "\text{nsqn } \text{rt dip} = \text{nsqn rt' dip}"\\
and \ "\pi_5(\text{the (rt dip)}) = \pi_5(\text{the (rt' dip)})"\\
\text{shows} \ "rt \approx_{\text{dip}} rt'"\\
\text{proof}\\
\text{from} \ \text{assms(1-2,4) have dhops'}: \ "(\text{dhops } rt' dip) \leq (\text{dhops } rt dip)"\\
\text{by (simp add: proj5_eq_dhops)}\\
\text{with} \ \text{assms(1-3) show} \ "rt \sqsubseteq_{\text{dip}} rt'"\\
\text{by (rule rt\_fresherI2)}\\
\text{next}\\
\text{from} \ \text{assms(1-2,4) have dhops}: \ "(\text{dhops } rt dip) \leq (\text{dhops } rt' dip)"\\
\text{by (simp add: proj5_eq_dhops)}\\
\text{with} \ \text{assms(2,1) assms(3) [symmetric] show} \ "rt' \sqsubseteq_{\text{dip}} rt"\\
\text{by (rule rt\_fresherI2)}\\
\text{qed}\\

\text{lemma rt\_fresh\_asE [elim]:}\\
\text{assumes} \ "rt1 \approx_{\text{dip}} rt2"\\
and \ "[ rt1 \sqsubseteq_{\text{dip}} rt2; rt2 \sqsubseteq_{\text{dip}} rt1 ] \implies P rt1 rt2 dip"\\
\text{shows} \ "P rt1 rt2 dip"\\
\text{using} \ \text{assms unfolding rt\_fresh\_as\_def by simp}\\

\text{lemma rt\_fresh\_asD1 [dest]:}\\
\text{assumes} \ "rt1 \approx_{\text{dip}} rt2"\\
\text{shows} \ "rt1 \sqsubseteq_{\text{dip}} rt2"\\
\text{using} \ \text{assms unfolding rt\_fresh\_as\_def by simp}\\

\text{lemma rt\_fresh\_asD2 [dest]:}\\
\text{assumes} \ "rt1 \approx_{\text{dip}} rt2"\\
\text{shows} \ "rt2 \sqsubseteq_{\text{dip}} rt1"\\
\text{using} \ \text{assms unfolding rt\_fresh\_as\_def by simp}\\

\text{lemma rt\_fresh\_as_sym:}\\
\text{assumes} \ "rt1 \approx_{\text{dip}} rt2"\\
\text{shows} \ "rt2 \approx_{\text{dip}} rt1"\\
\text{using} \ \text{assms unfolding rt\_fresh\_as\_def by simp}\\

\text{lemma not\_rt\_fresh\_asI1 [intro]:}\\
\text{assumes} \ "\neg (rt1 \sqsubseteq_{\text{dip}} rt2)"\\
\text{shows} \ "\neg (rt1 \approx_{\text{dip}} rt2)"\\
\text{proof}\\
\text{assume} \ "rt1 \approx_{\text{dip}} rt2"\\
\text{hence} \ "rt1 \sqsubseteq_{\text{dip}} rt2" ..\\
\text{with} \ "\neg (rt1 \sqsubseteq_{\text{dip}} rt2)" \text{ show False} ..\\
\text{qed}\\

\text{lemma not\_rt\_fresh\_asI2 [intro]:}\\
\text{assumes} \ "\neg (rt2 \sqsubseteq_{\text{dip}} rt1)"\\
\text{shows} \ "\neg (rt1 \approx_{\text{dip}} rt2)"\\
\text{proof}\\
\text{assume} \ "rt1 \approx_{\text{dip}} rt2"\\
\text{hence} \ "rt2 \sqsubseteq_{\text{dip}} rt1" ..\\
\text{with} \ "\neg (rt2 \sqsubseteq_{\text{dip}} rt1)" \text{ show False} ..\\
\text{qed}\\

\text{lemma not\_single\_rt\_fresher [elim]:}\\
\text{assumes} \ "\neg (\text{the (rt1 ip)} \subseteq \text{the (rt2 ip)})"\\
\text{shows} \ "\neg (rt1 \sqsubseteq_{\text{ip}} rt2)"
proof
  assume "rt1 ⊑ rt2"
  hence "the (rt1 ip) ⊑ the (rt2 ip)"
  with \(\neg\) \(\langle\text{the (rt1 ip) ⊑ the (rt2 ip)}\rangle\) show False ..
qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
  assumes "\(\neg\) (rt1 ⊑ rt2)"
  shows "\(\neg\) (the (rt1 ip) ⊑ the (rt2 ip))"
proof
  assume "the (rt1 ip) ⊑ the (rt2 ip)"
  hence "rt1 ⊑ ip rt2" ..
  with \(\neg\) (rt1 ⊑ ip rt2) show False ..
qed

lemma rt_fresh_as_nsqnr::
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "rt1 ≈ dip rt2"
  shows "nsqn (the (rt2 dip)) = nsqn (the (rt1 dip))"
using assms(3)
unfolding rt_fresh_as_def
by (auto simp: rt_fresher_def2 [OF \(\langle\text{dip ∈ kD(rt1)\rangle}\)]
    rt_fresher_def2 [OF \(\langle\text{dip ∈ kD(rt2)\rangle}\)]
    kD_nsqn [OF \(\langle\text{dip ∈ kD(rt1)\rangle}\)]
    kD_nsqn [OF \(\langle\text{dip ∈ kD(rt2)\rangle}\)]

lemma rt_fresher_mapupd [intro!]:
  assumes "dip ∈ kD(rt)"
  and "dip ≠ ip"
  shows "rt ⊑ dip rt (dip ↦ r)"
using assms unfolding rt_fresher_def
by simp

lemma rt_fresher_map_update_other [intro!]:
  assumes "dip ∈ kD(rt)"
  and "dip ≠ ip"
  shows "rt ⊑ dip update rt ip r"
using assms unfolding rt_fresher_def
by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip ∈ kD(rt)"
  and "dip ≠ ip"
  shows "rt ⊑ dip update rt ip r"
using assms unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip ∈ kD(rt)"
  and "the (dhops rt dip) ≥ 1"
  and "update_arg_wf r"
  shows "rt ⊑ dip update rt ip r"
proof (cases "dip = ip")
  assume "dip ≠ ip" with dip∈kD(rt) show ?thesis
  by (rule rt_fresher_update_other)
next
  assume "dip = ip"

  from dip∈kD(rt) obtain dsn, dsk, fn, hops, nhip, pre
    where rtn [simp]: "the (rt dip) = (dsn, dsk, fn, hops, nhip, pre)"
    by (metis prod_cases6)
  with \(\langle\text{the (dhops rt dip) ≥ 1}\rangle\) and \(\langle\text{dip ∈ kD(rt)\rangle}\) have "hops ≥ 1"
  by (metis proj5_eq_dhops projs(4))

365
from \(\langle \text{dip}\in kD(rt) \rangle\) \(\text{rtn} \) have \([\text{simp}]\): "sqn rt dip = dsn_n"
  and \([\text{simp}]\): "the (dhops rt dip) = hops_n"
  and \([\text{simp}]\): "the (flag rt dip) = f_n"
by \((\text{simp add: sqn_def proj5_eq_dhops [symmetric] proj4_eq_flag [symmetric]})\)+

from \(\langle \text{update_arg_wf r} \rangle\) have "(dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n) \sqsubseteq \text{the } ((\text{update rt dip r}) \text{ dip})"
proof \((\text{rule wf_r_cases})\)
  fix nhip pre
  from \(\langle \text{hops}_n \geq 1 \rangle\) have "\(\forall \text{pre}' . (dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n) \sqsubseteq ((\text{update rt dip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{nhip}, \text{pre}'))\)"
  unfolding fresher_def sqn_def by \((\text{cases f_n})\) auto
  thus "(dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n) \sqsubseteq \text{the } ((\text{update rt dip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{nhip}, \text{pre}')) \text{ dip})"
  using \(\langle \text{dip}\in kD(rt) \rangle\) by - \((\text{rule update_cases_kD, simp_all})\)

next
fix dsn :: sqn and hops nhip pre
assume "0 < dsn"
show "(dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n) \sqsubseteq \text{the } ((\text{update rt dip } (dsn, \text{kn}o, \text{val}, \text{hops}, \text{nhip}, \text{pre} \cup \text{pre}_n))\)"
  unfolding fresher_def by simp
next
assume "dsn_n = dsn"
  and "hops < hops_n"
  thus "(dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n) \sqsubseteq \text{the } ((\text{update rt dip } (dsn, \text{kn}o, \text{val}, \text{hops}, \text{nhip}, \text{pre} \cup \text{pre}_n))\)"
  unfolding fresher_def nsqn_r_def by simp

qed

hence "rt \sqsubseteq_{\text{dip}} \text{update rt dip r}"
  by \((\text{rule single_rt_fresher, simp})\)
  with \(\langle \text{dip}\in \text{ip} \rangle\) show ?thesis by simp
qed

theorem rt_fresher_invalidate \([\text{simp}]\):
assumes "\(\langle \text{dip}\in kD(rt) \rangle\)"
  and \(\text{indests: } \forall \text{rip} \in \text{dom(dests)} . \text{rip}\in vD(rt) \wedge \text{sqn rt rip} < \text{the } (\text{dests rip})\)"
shows "\(\text{rt} \sqsubseteq_{\text{dip}} \text{invalidate rt dests}\)"
proof \((\text{cases } \langle \text{dip}\in \text{dom(dests)} \rangle)\)
  assume "\(\text{dip}\in \text{dom(dests)}\)"
  thus ?thesis using \(\langle \text{dip}\in kD(rt) \rangle\) by \((\text{rule single_rt_fresher, simp})\)
next
assume "\(\text{dip}\in \text{dom(dests)}\)"
  moreover with \(\text{indests} \) have "\(\text{dip}\in vD(rt)\)"
    and "sqn rt dip < \text{the } (\text{dests dip})"
    by auto
  ultimately show ?thesis
  unfolding invalidate_def sqn_def
    by \((\text{rule single_rt_fresher, auto simp: fresher_def})\)
qed

lemma nsqn_r_invalidate \([\text{simp}]\):
assumes "dip ∈ kD(rt)"
and "dip ∈ dom(dests)"
shows "nsqn, (the (invalidate rt dests dip)) = the (dests dip) - 1"
using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
assumes "dip ∈ kD(rt)"
and "∀ rip ∈ dom(dests). rip ∈ vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
shows "rt ≈ dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)"
  assume "dip /∈ dom(dests)"
  with ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)"
    by simp
  with ⟨dip ∈ kD(rt)⟩ show ?thesis
    by rule (simp_all add: ⟨dip /∈ dom(dests)⟩)
next
  assume "dip ∈ dom(dests)"
  with assms(2)
  have "dip ∈ vD(rt)" and "the (dests dip) = inc (sqn rt dip)"
    by (rule nsqnファッション invalidate)
  from ⟨dip ∈ vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
  also have "sqn rt dip = nsqn rt (the (invalidate rt dests dip))"
    using ⟨dip ∈ kD(rt)⟩ by (rule nsqnファッション invalidate)
  with ⟨the (dests dip) = inc (sqn rt dip)⟩
  show "sqn rt dip = nsqn rt (the (invalidate rt dests dip))" by simp
  qed
  also from ⟨dip ∈ kD(invalidate rt dests)⟩
  have "nsqn rt (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
    by (simp add: kD_nsqn)
  finally show "nsqn rt dip = nsqn (invalidate rt dests) dip".
  qed simp
qed

lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]

lemma rt_fresh_as_addpreRT [simp]:
assumes "ip ∈ kD(rt)"
shows "rt ≈ dip the (addpreRT rt ip npre)"
using assms [THEN kD_Some] by (auto simp: addpreRT_def)

lemmas rt_fresher_addpreRT [simp] = rt_fresh_as_addpreRT [THEN rt_fresh_asD1]

4.5.4 Strictly comparing routing tables

definition rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
  "rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ ¬(rt1 ≈ dip rt2)"
abbreviation
  rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ∩ _ _)" [51, 999, 51] 50)
where
  "rt1 ∩ rt2 ≡ rt_strictly_fresher i rt1 rt2"

lemma rt_strictly_fresher_def'':
  "rt1 ∩ rt2 = ((rt1 ∩ rt2) ∧ ¬(rt2 ⊑ i rt1))"
unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

lemma rt_strictly_fresherI' [intro]:
assumes "rt1 ⊑ rt2"
and "¬(rt2 ⊑ rt1)"
shows "rt1 ⊏ rt2"
using assms unfolding rt_strictly_fresher_def’’ by simp

lemma rt_strictly_fresherE' [elim]:
assumes "rt1 ⊏ rt2"
and "[ rt1 ⊑ rt2; ¬(rt2 ⊑ rt1) ] ⇒ P rt1 rt2 i"
shows "P rt1 rt2 i"
using assms unfolding rt_strictly_fresher_def’’ by simp

lemma rt_strictly_fresherI [intro]:
assumes "rt1 ⊏ rt2"
and "¬(rt1 ≈ rt2)"
unfolding rt_strictly_fresher_def using assms ..

lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]

lemma rt_strictly_fresherE [elim]:
assumes "rt1 ⊏ rt2"
and "[ rt1 ⊑ rt2; ¬(rt2 ⊑ rt1) ] ⇒ P rt1 rt2 i"
shows "P rt1 rt2 i"
using assms(1) unfolding rt_strictly_fresher_def by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
"rt1 ⊏ rt2 =
(nsqr r (the (rt1 i)) < nsqr r (the (rt2 i))
∨ (nsqr r (the (rt1 i)) = nsqr r (the (rt2 i)) ∧ π_5(the (rt1 i)) > π_5(the (rt2 i))))"
unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
assumes "rt1 ⊏ rt2"
shows "the (rt1 dip) ⊑ the (rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
assumes "rt1 ≈ dip rt2"
shows "¬ rt1 ≈ dip rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1 ⊏ dip rt2"
and "rt2 ⊑ dip rt3"
shows "rt1 ⊑ dip rt3"
using assms proof -
from rt1 ⊏ dip rt2; obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
also from rt2 ⊑ dip rt3; obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
finally have "the (rt1 dip) ⊑ the (rt3 dip)".

moreover have "¬ (rt1 ≈ dip rt3)"
proof -
from rt1 ⊏ dip rt2; obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
also from rt2 ⊑ dip rt3; obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))".
thus ?thesis ..
qed ultimately show "rt1 ⊏ dip rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏ dip rt)"
unfolding rt_strictly_fresher_def by clarsimp
lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1 ⊑ dip rt2"
and "rt2 ⊑ dip rt3"
shows "rt1 ⊏ dip rt3"
proof -
  from rt1 ⊑ dip rt2: have "rt1 ⊑ dip rt2"
  and "¬ (rt2 ⊑ dip rt1)"
  unfolding rt_strictly_fresher_def' by auto
  from this(1) and (rt2 ⊑ dip rt3) have "rt1 ⊑ dip rt3" ..
moreover from (¬ (rt2 ⊑ dip rt1)) have "¬ (rt3 ⊑ dip rt1)"
  proof (rule contrapos_nn)
    assume "rt3 ⊑ dip rt1"
    with (rt2 ⊑ dip rt3) show "rt2 ⊑ dip rt1" ..
  qed
ultimately show "rt1 ⊑ dip rt3"
  unfolding rt_strictly_fresher_def' by auto
qed

lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
assumes "rt1 ⊑ dip rt2"
and "rt2 ⊏ dip rt3"
shows "rt1 ⊏ dip rt3"
proof -
  from rt2 ⊏ dip rt3: have "rt2 ⊏ dip rt3"
  and "¬ (rt3 ⊑ dip rt2)"
  unfolding rt_strictly_fresher_def' by auto
  from rt1 ⊑ dip rt2 and this(1) have "rt1 ⊏ dip rt3" ..
moreover from (¬ (rt3 ⊑ dip rt2)) have "¬ (rt3 ⊑ dip rt1)"
  proof (rule contrapos_nn)
    assume "rt3 ⊑ dip rt1"
    thus "rt3 ⊏ dip rt2" using rt1 ⊏ dip rt2 ..
  qed
ultimately show "rt1 ⊏ dip rt3"
  unfolding rt_strictly_fresher_def' by auto
qed

lemma rt_fresher_imp_nsqn_le:
assumes "rt1 ⊑ ip rt2"
and "ip ∈ kD rt1"
and "ip ∈ kD rt2"
shows "nsqn rt1 ip ≤ nsqn rt2 ip"
using assms(1)
by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

lemma rt_strictly_fresher_ltI [intro]:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "nsqn rt1 dip < nsqn rt2 dip"
shows "rt1 ⊏ dip rt2"
proof
from assms show "rt1 ⊑ dip rt2" ..
next
  show "¬ (rt1 ≈ dip rt2)"
  proof
    assume "rt1 ≈ dip rt2"
    hence "rt2 ⊑ dip rt1" ..
    hence "nsqn rt2 dip ≤ nsqn rt1 dip"
      using dip ∈ kD(rt2) dip ∈ kD(rt1) by (rule rt_fresher_imp_nsqn_le)
    with nsqn rt1 dip < nsqn rt2 dip show "False"

369
lemma rt_strictly_fresher_eqI [intro]:
assumes "i ∈ kD(rt1)"
and "i ∈ kD(rt2)"
and "nsqn rt1 i = nsqn rt2 i"
and "π5(the (rt2 i)) < π5(the (rt1 i))"
shows "rt1 ◁_j rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]:
"\dests dip rt rt'. \dests dip = None \implies (invalidate rt \dests ⊏ dip rt') = (rt ⊏ dip rt')"
unfolding invalidate_def rt_strictly_fresher_def'
by (rule iffI) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
assumes "dip ∈ vD(invalidate rt1 \dests)"
shows "(invalidate rt1 \dests ⊏ dip rt2) = (rt1 ⊏ dip rt2)"
proof (cases "dip ∈ dom(\dests)")
assume "dip ∉ dom(\dests)"
hence "dip ⊏ vD(invalidate rt1 \dests)"
unfolding invalidate_def vD_def
by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
ultimately show ?thesis
unfolding invalidate_def rt_strictly_fresher_def'
by auto
qed

lemma rt_strictly_fresher_update_other [elim!]:
"∀ dip ip rt rt'. dip ≠ ip; rt ⊏ dip rt' \implies update rt dip (osn, kno, val, hops, nhip, {}) ⊏ dip rt2"
unfolding rt_strictly_fresher_def' by clarsimp

lemma addpreRT_strictly_fresher [simp]:
assumes "dip ∈ kD(rt)"
shows "(the (addpreRT rt dip npre) ⊏ dip rt2) = (rt ⊏ dip rt2)"
using assms unfolding rt_strictly_fresher_def'
by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
assumes "dip ∈ vD (rt2 nhip)"
and *: "osn < sqn (rt2 nhip) dip"
and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
shows "update rt dip (osn, kno, val, hops, nhip, {}) ⊏ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip, {})) dip = osn"
  by (rule nsqn_update_changed_kno_val)
with \ dip ∈ vD (rt2 nhip)
have "nsqn, (the (update rt dip (osn, kno, val, hops, nhip, {})) dip) = osn"
  by (simp add: kD_nsqn)
also have "osn < sqn (rt2 nhip) dip" by (rule *)
also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
  unfolding nsqn_def using dip ∈ vD (rt2 nhip)
by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "nsqn, (the (update rt dip (osn, kno, val, hops, nhip, {})) dip)
  < nsqn, (the (rt2 nhip dip))".
qed
lemma dhops_le_hops_imp_update_strictly_fresher:
assumes "dip ∈ vD(rt2 nhip)"
and sqn: "sqn (rt2 nhip) dip = osn"
and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip, {})"
shows "update rt dip (osn, kno, val, Suc hops, nhip, {}) ⊏ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip = osn"
by (rule nsqn_update_changed_kno_val)
with ⟨dip ∈ vD(rt2 nhip)⟩
have "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip) = osn"
by (simp add: kD_nsqn)
also have "osn = sqn "(rt2 nhip) dip" by (rule sqn [symmetric])
also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
unfolding nsqn_def using ⟨dip ∈ vD(rt2 nhip)⟩
by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip) = nsqn, (the (rt2 nhip dip))".
next
have "the (dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
also have "hops < hops + 1" by simp
also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)"
using ** by simp
finally have "the (dhops (rt2 nhip) dip) < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)".
thus "π₅ (the (rt2 nhip dip)) < π₅ (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)"
using ⟨dip ∈ vD(rt2 nhip)⟩ by (simp add: proj5_eq_dhops)
qed

lemma nsqn_invalidate:
assumes "dip ∈ kD(rt)"
and "∀ ip∈dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sqn rt ip)"
shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
proof -
from assms have "rt ≈dip invalidate rt dests"
by (rule rt_fresh_as_inc_invalidate)
with ⟨dip ∈ kD(rt)⟩ ⟨dip ∈ kD(invalidate rt dests)⟩ show ?thesis
by (simp add: kD_nsqn del: invalidate_kD_inv)
(erule(2) rt_fresh_as_nsqnr)
qed

4.6 Invariant proofs on individual processes

theory D_Seq_Invariants
importsAWN.Invariants D_Aodv D_Aodv_Data D_Aodv_Predicates D_Fresher
begin
The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

lemma sequence_number_increases:
"paodv i |∈= onll Γ_{AODV} (λ(((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ'))"
by inv_cterms

lemma sequence_number_one_or_bigger:
"paodv i |∈= onll Γ_{AODV} (λ(ξ, _). 1 ≤ sn ξ)"
by (rule onll_step_to_invariantI [OF sequence_number_increases])
(auto simp: σ_{AODV_def})
We can get rid of the onl/onll if desired...

```
lemma sequence_number_increases'
  "paodv i |= (λ(ξ, _, l). sn ξ ≤ sn ξ'). sn ξ ≤ sn ξ'")
  by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)
```

```
lemma sequence_number_one_or_bigger'
  "paodv i ||= (λ(ξ, _). 1 ≤ sn ξ)" by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto
```

```
lemma sip_in_kD:
  "paodv i ||= onl Γ AODV (λ(ξ, l). l ∈ (\{PAodv:-7\} ∪ \{PAodv-:5\} ∪ \{PRrep-:0..PRrep-:1\}
  ∪ \{PRreq-:0..PRreq-:3\}) -> sip ξ ∈ kD (rt ξ))"
  by inv_cterms
```

```
lemma rrep_1_update_changes:
  "paodv i ||= onl Γ AODV (λ(ξ, l). (l = PRrep-:1 -> rt ξ ≠ update (rt ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})))"
  by inv_cterms
```

```
lemma addpreRT_partly_welldefined:
  "paodv i ||= onl Γ AODV (λ(ξ, l). (l ∈ {PRreq-:18..PRreq-:20} ∪ {PRrep-:2..PRrep-:6} -> dip ξ ∈ kD (rt ξ))
  ∧ (l ∈ {PRreq-:3..PRreq-:19} -> oip ξ ∈ kD (rt ξ)))"
  by inv_cterms
```

```
Proposition 7.38
lemma includes_nhip:
  "paodv i ||= onl Γ AODV (λ(ξ, l). ∀ dip ∈ kD(rt ξ). the (nhop (rt ξ) dip) ∈ kD(rt ξ))"
proof -
  { fix ip and ξ ξ' :: state
    assume "∀ dip ∈ kD (rt ξ). the (nhop (rt ξ) dip) ∈ kD (rt ξ)"
    and "ξ' = [rt := update (rt ξ) ip (0, unk, val, Suc 0, ip, {})]""  
    hence "∀ dip ∈ kD (rt ξ).
      the (nhop (update (rt ξ) ip (0, unk, val, Suc 0, ip, {})) dip) = ip
      ∨ the (nhop (update (rt ξ) ip (0, unk, val, Suc 0, ip, {})) dip) ∈ kD (rt ξ)"
    by clarsimp (metis nhop_update_unk_val update_another)
  } note one_hop = this
  { fix ip sip sn hops and ξ ξ' :: state
    assume "∀ dip ∈ kD (rt ξ). the (nhop (rt ξ) dip) ∈ kD (rt ξ)"
    and "ξ' = [rt := update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})]"
    and "sip ∈ kD (rt ξ)"
    hence "(the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) ip) = ip
      ∨ the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) ip) ∈ kD (rt ξ))
    ∧ (∀ dip ∈ kD (rt ξ).
      the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) dip) = ip
      ∨ the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) dip) ∈ kD (rt ξ))"
    by (metis kD_update_unchanged nhop_update_changed update_another)
  } note nhip_is_sip = this
show ?thesis
  by (inv_cterms inv adds: onl_invariant_sterms [OF aodv_wf sip_in_kD]
    onl_invariant_sterms [OF aodv_wf addpreRT_partly_welldefined]
    solve: one_hop nhip_is_sip)
qed
```

```
Proposition 7.22: needed in Proposition 7.4
lemma addpreRT_welldefined:
  "paodv i |= onl Γ AODV (λ(ξ, l). (l ∈ {PRreq-:18..PRreq-:20} ∪ {PRrep-:2..PRrep-:6} -> dip ξ ∈ kD (rt ξ))
  ∧ (l ∈ {PRreq-:3..PRreq-:19} -> oip ξ ∈ kD (rt ξ)))" (is "_ ||= onl Γ AODV ?P")
  unfolding invariant_def
proof
```

372
fix $s$

assume $"s \in \text{reachable (paodv i) TT}"$

then obtain $\xi$ $p$ where $"s = (\xi, p)"

and $"(\xi, p) \in \text{reachable (paodv i) TT}"$

by (metis prod.exhaust)

have "onl $\Gamma_{AODV}$ $?P$ $(\xi, p)$"

proof (rule onlI)

fix $l$

assume $"l \in \text{labels } \Gamma_{AODV} p"$

with $\langle(\xi, p) \in \text{reachable (paodv i) TT}\rangle$

have I1: "$l \in \{\text{PRreq-:18..PRreq-:20}\} \rightarrow \text{dip } \xi \in \text{kD(rt } \xi)"

and I2: "$l = \text{PRreq-:19} \rightarrow \text{oip } \xi \in \text{kD(rt } \xi)"

by (auto dest!: invariantD [OF addpreRT_partly_welldefined])

moreover from $\langle(\xi, p) \in \text{reachable (paodv i) TT}\rangle \langle l \in \text{labels } \Gamma_{AODV} p\rangle$

and I3

ultimately show "$?P (\xi, l)"

by simp

qed

with $\langle s = (\xi, p) \rangle$

show "onl $\Gamma_{AODV} \ ?P$ s"

by simp

qed

Proposition 7.4

lemma known_destinations_increase:

\[ \text{paodv i} \models \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{kD (rt } \xi) \subseteq \text{kD (rt } \xi')) \]

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]

simp add: subset_insertI)

Proposition 7.5

lemma rreqs_increase:

\[ \text{paodv i} \models \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{rreqs } \xi \subseteq \text{rreqs } \xi') \]

by (inv_cterms simp add: subset_insertI)

lemma dests_bigger_than_sqn:

\[ \text{paodv i} \models \text{onll } \Gamma_{AODV} (\lambda(\xi, l). l \in \{\text{PAodv-:15..PAodv-:19}\}

\union \{\text{PPkt-:7..PPkt-:11}\}

\union \{\text{PRreq-:11..PRreq-:15}\}

\union \{\text{PRreq-:24..PRreq-:28}\}

\union \{\text{PRrep-:10..PRrep-:14}\}

\union \{\text{PRerr-:1..PRerr-:5}\}

\rightarrow (\forall ip \in \text{dom(dests } \xi). \text{ip} \in \text{kD(rt } \xi) \wedge \text{sqn (rt } \xi) \text{ip} \leq (\text{the (dests } \xi \text{ip))))"\]

proof -

have sqninv:

"\forall ip \in \text{dom(dests } \xi). \text{ip} \in \text{kD(rt } \xi) \wedge \text{sqn rt } \xi \text{ip} \leq (\text{the (dests } \xi \text{ip)); dests ip = Some rsn ]}

\rightarrow \text{sqn (invalidate rt dests } \xi \text{ip} \leq \text{rsn}"

by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto

have indects:

"\forall ip \in \text{dom(dests } \xi). \text{ip} \in \text{kD(rt } \xi) \wedge \text{sqn rt } \xi \text{ip} \leq (\text{the (dests } \xi \text{ip)); dests ip = Some rsn ]}

\rightarrow \text{ip} \in \text{kD(rt } \xi) \wedge \text{sqn rt } \xi \text{ip} \leq \text{rsn}"

by (metis domI option.sel)

show ?thesis

by inv_cterms

(clarsimp split: if_split_asm option.split_asm elim!: sqninv indects)+

qed

Proposition 7.6

lemma sqns_increase:

"\text{paodv i} \models \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall ip. \text{sqn (rt } \xi) \text{ip} \leq \text{sqn (rt } \xi') \text{ip})"
\{ \text{fix } \xi :: \text{state} \\
\text{assume } *: \ "\forall \text{ip} \in \text{dom(dests } \xi \text{)}. \text{ip} \in kD (\text{rt } \xi) \land \text{sqn} (\text{rt } \xi) \text{ ip} \leq \text{the} (\text{dests } \xi \text{ ip})" \\
\text{have } "\forall \text{ip}. \text{sqn} (\text{rt } \xi) \text{ ip} \leq \text{sqn} (\text{invalidate} (\text{rt } \xi) (\text{dests } \xi)) \text{ ip}" \\
\text{proof} \\
\text{fix ip} \\
\text{from } * \text{ have } "\text{ip} \notin \text{dom(dests } \xi \text{)} \lor \text{sqn} (\text{rt } \xi) \text{ ip} \leq \text{the} (\text{dests } \xi \text{ ip})" \text{ by simp} \\
\text{thus } "\text{sqn} (\text{rt } \xi) \text{ ip} \leq \text{sqn} (\text{invalidate} (\text{rt } \xi) (\text{dests } \xi)) \text{ ip}" \\
\text{by (metis domI invalidate_sqn option.sel)} \\
\text{qed} \}
\text{note solve_invalidate = this} \\
\text{show } ?\text{thesis} \\
\text{by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined] onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn] simp add: solve_invalidate)} \\
\text{qed}

Proposition 7.7

\text{lemma ip_constant:} \\
"paodv i \models onl \Gamma_{AODV} (\lambda(\xi, \_). \text{ip } \xi = i)" \\
\text{by (inv_cterms simp add: } \sigma_{AODV} _\text{def})

Proposition 7.8

\text{lemma sender_ip_valid':} \\
"paodv i \models A onll \Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{anycast } (\lambda m. \text{not_Pkt } m \rightarrow \text{msg_sender } m = \text{ip } \xi) a)" \\
\text{by inv_cterms}

\text{lemma sender_ip_valid:} \\
"paodv i \models A onll \Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{anycast } (\lambda m. \text{not_Pkt } m \rightarrow \text{msg_sender } m = i) a)" \\
\text{by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])} \\
\text{(auto dest!: onlD onllD)}

\text{lemma received_msg_inv:} \\
"paodv i \models (\text{recvmsg } P \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, l). l \in \{\text{PAodv-:1}\} \rightarrow P (\text{msg } \xi))" \\
\text{by inv_cterms}

Proposition 7.9

\text{lemma sip_not_ip':} \\
"paodv i \models (\text{recvmsg } (\lambda m. \text{not_Pkt } m \rightarrow \text{msg_sender } m \neq i) \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, \_). \text{sip } \xi \neq \text{ip } \xi)" \\
\text{by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv] onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]] simp add: clear_locals_sip_not_ip')} \\
\text{clarsimp+}

\text{lemma sip_not_ip:} \\
"paodv i \models (\text{recvmsg } (\lambda m. \text{not_Pkt } m \rightarrow \text{msg_sender } m \neq i) \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, \_). \text{sip } \xi \neq i)" \\
\text{by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv] onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]] simp add: clear_locals_sip_not_ip')} \\
\text{clarsimp+}

Neither sip_not_ip' nor sip_not_ip is needed to show loop freedom.

Proposition 7.10

\text{lemma hop_count_positive:} \\
"paodv i \models onl \Gamma_{AODV} (\lambda(\xi, \_). \forall \text{ip} \in kD (\text{rt } \xi). \text{the} (\text{dhops } (\text{rt } \xi) \text{ ip}) \geq 1)" \\
\text{by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]) auto}

\text{lemma rreq_dip_in_vD_dip_eq_ip:} \\
"paodv i \models onl \Gamma_{AODV} (\lambda(\xi, 1). (1 \in \{\text{PRreq-:18..PRreq-:21}\} \rightarrow \text{dip } \xi \in \text{vD}(\text{rt } \xi)) \\
\land (1 \in \{\text{PRreq-:6, PRreq-:7}\} \rightarrow \text{dip } \xi = \text{ip } \xi) \\
\land (1 \in \{\text{PRreq-:17..PRreq-:21}\} \rightarrow \text{dip } \xi \neq \text{ip } \xi))" \\
\text{proof (inv_cterms, elim conjE)} \\
\text{fix } l \xi pp p' \\
\text{assume } "(\xi, pp) \in \text{reachable (paodv i) TT}" \\
\text{and } "\{\text{PRreq-:19}\}[\lambda \xi. \text{rt := the} (\text{addpreRT } (\text{rt } \xi) (\text{oip } \xi) \{\text{the} (\text{nhop} (\text{rt } \xi) (\text{dip } \xi)))\}] p'"
Lemma Proposition 7.11

Lemma ancast_msg_zhops:
"\forall rreqid dip dsn dsr oip sip. paodv i \models A \ oAODV (\lambda (a, _. a). ancast msg_zhops a)"

Proof (inv_cterms inv add:
onl_invariant_sterms [OF aodv_wf rreq_in_vD dip_eq_ip [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]],
elim conjE)

Fix \( \xi \) \( \in \) reachable (paodv i) TT

And "\{PReq-:20\}unicast(\lambda \xi. \the (nhop (rt \xi) (oip \xi)),
\lambda \xi. \Rep (the (dhops (rt \xi) (dip \xi))) (dip \xi) (sqn (rt \xi) (dip \xi)) (oip \xi) (ip \xi)).
p' \triangleright p' \in \Gamma_{AODV} pp"

And "1 = PReq-:20"

And "a = unicast (the (nhop (rt \xi) (oip \xi)))
(Rep (the (dhops (rt \xi) (dip \xi))) (dip \xi) (sqn (rt \xi) (dip \xi)) (oip \xi) (ip \xi))"

And *: "\forall ip \in kD (rt \xi). Suc 0 \leq \the (dhops (rt \xi) ip)"

And "dip \xi \in \nu (rt \xi)"

From * have "dip \xi \in kD (rt \xi)"

By (rule vD_id_gives_kD(1))

With * have "Suc 0 \leq \the (dhops (rt \xi) (dip \xi))"..

Thus "0 \leq \the (dhops (rt \xi) (dip \xi))" by simp

Qed

Lemma hop_count_zero_oip_dip_sip:
"paodv i \models (recvmsg msg_zhops \rightarrow) \nu \Gamma_{AODV} (\lambda (\xi, 1).
(\xi \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n\|n. True\} \rightarrow
\the (dhops \xi = 0 \rightarrow oip \xi = \nu sip \xi))
\wedge
((\xi \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-:n\|n. True\} \rightarrow
\the (dhops \xi = 0 \rightarrow dip \xi = \nu sip \xi))))"

By (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

Lemma osn_rreq:
"paodv i \models (recvmsg req_rrep_sn \rightarrow) \nu \Gamma_{AODV} (\lambda (\xi, 1).
1 \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n\|n. True\} \rightarrow 1 \leq osn \xi)"

By (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

Lemma osn_rreq' :
"paodv i \models (recvmsg (\lambda m. req_rrep_sn m \land msg_zhops m) \rightarrow) \nu \Gamma_{AODV} (\lambda (\xi, 1).
1 \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n\|n. True\} \rightarrow 1 \leq osn \xi)"

Proof (rule invariant_weakenE [OF osn_rreq])

Fix a
Assume "recvmsg (\lambda m. req_rrep_sn m \land msg_zhops m) a"
Thus "recvmsg req_rrep_sn m a"

By (cases a) simp_all

Qed

Lemma dsn_rrep:
"paodv i \models (recvmsg req_rrep_sn \rightarrow) \nu \Gamma_{AODV} (\lambda (\xi, 1).
1 \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-:n\|n. True\} \rightarrow 1 \leq dsn \xi)"

By (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

Lemma dsn_rrep' :
proof (rule invariant_weakenE [OF dsn_rrep])
  fix a
  assume "recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
  thus "recvmsg rreq_rrep_sn a"
  by (cases a) simp_all
qed

lemma hop_count_zero_oip_dip_sip':
  "paodv i ||= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a)
  by (cases a) simp_all
qed

Proposition 7.12
lemma zero_seq_unk_hops_one':
  "paodv i ||= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a)
  by (cases a) simp_all
qed
by (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
} note prreq_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "∀ dip∈kD(rt).
          (sqn rt dip = 0 −→ π₃(the (rt dip)) = unk) ∧
         (π₃(the (rt dip)) = unk −→ the (dhops rt dip) = Suc 0) ∧
         (the (dhops rt dip) = Suc 0 −→ the (nhop rt dip) = dip)"
  and "Suc 0 ≤ dsn"
  and "ip ≠ dip −→ ip∈kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip = 0 −→
           π₃ (the (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip)) = unk"
} note prreq_ok3 [simp] = this

{ fix rt sip
  assume "∀ dip∈kD rt.
          (sqn rt dip = 0 −→ π₃(the (rt dip)) = unk) ∧
         (π₃(the (rt dip)) = unk −→ the (dhops rt dip) = Suc 0) ∧
         (the (dhops rt dip) = Suc 0 −→ the (nhop rt dip) = dip)"
  hence "∀ dip∈kD rt.
          (sqn (update rt sip (0, unk, val, Suc 0, sip, {})) dip = 0 −→
           π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {})) dip)) = unk) ∧
            (the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0) ∧
            (the (nhop (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = dip)"
} note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "∀ sip rt.
π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {})) sip)) = unk −→
the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = Suc 0"
by (rule update_cases) simp_all

have prreq_ok6 [simp]: "∀ sip rt.
 sqn (update rt sip (0, unk, val, Suc 0, sip, {})) sip = 0 −→
π₃ (the (update rt sip (0, unk, val, Suc 0, sip, {})) sip)) = unk"
by (rule update_cases) simp_all

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
      onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip']
      seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans]
      onl_invariant_sterms [OF aodv_wf osn_rreq']
      onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+
qed

lemma zero_seq_unk_hops_one:
"paodv i |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) −→ onl Γₐₒᵈᵥ (λξ, _).
          ∀ dip∈kD(rt ξ). (sqn (rt ξ) dip = 0 −→ (sqnf (rt ξ) dip = unk
          ∧ the (dhops (rt ξ) dip) = 1
          ∧ the (nhop (rt ξ) dip) = dip))))"
by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
"paodv i |= (recvmsg rreq_rrep_sn −→ onl Γₐₒᵈᵥ (λξ, 1).
          ∀ dip∈kD(rt ξ). π₃(the (rt ξ) dip) = unk ∨ 1 ≤ π₂(the (rt ξ) dip)))"
proof -
{ fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
assume "dsk1 = unk ∨ Suc 0 ≤ dsn2"
  hence "π₃(the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1, pre1) sip)) = unk
       ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) sip"
  unfolding update_def by (cases "dsk1 = unk") (clarsimp split: option.split)+
}
Proposition 7.13
**lemma rreq_rrep_sn_any_step_invariant:**

\[ \text{"paodv i } \vdash \text{A (recvmsg rreq_rrep_sn }\rightarrow\text{onll }\Gamma_{AODV}(\lambda(\_\_\_\_, a, \_\_\_). \text{anycast rreq_rrep_sn a}"

**proof**

- have sqnf_kno: "paodv i } \vdash onl \Gamma_{AODV}(\lambda(\xi, 1).

\[ (l \in \{PRreq:-18..PRreq:-20\} \rightarrow \text{sqnf (rt }\xi) \text{ (dip }\xi) = \text{kno})"

by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined])

**show ?thesis**

by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf preRT_welldefined]

**proof**

inv_cterms

qed

**Proposition 7.14**

**lemma rreq_rrep_fresh_any_step_invariant:**

\[ \text{"paodv i } \vdash \text{A onll }\Gamma_{AODV}(\lambda((\xi, \_\_\_), a, \_\_\_). \text{anycast (rreq_rrep_fresh (rt }\xi) a"

**proof**

- have rreq_oip: "paodv i } \vdash onl \Gamma_{AODV}(\lambda(\xi, 1).

\[ (l \in \{PRreq:-13..PRreq:-9\} \cup \{PRreq:-17, PRreq:-30, PRreq:-32\} \rightarrow \text{oip }\xi \in \text{kD(rt }\xi)

\[ \wedge (\text{sqn (rt }\xi) \text{ (oip }\xi) > \text{ (osn }\xi)

\[ \vee (\text{sqn (rt }\xi) \text{ (oip }\xi) = \text{ (osn }\xi

\[ \wedge \text{the (dhops (rt }\xi) \text{ (oip }\xi) \leq \text{ Suc (hops }\xi)

\[ \wedge \text{the (flag (rt }\xi) \text{ (oip }\xi) = \text{ val})"

**proof inv_cterms**

**fix 1 l l' pp p'**

**assume "(\xi, pp) } \in \text{reachable (paodv i) TT"**

and "\{PRreq:-2\}A\xi. \{rt :=

\[ \text{update (rt }\xi) \text{ (oip }\xi) \text{ (osn }\xi, \text{kno, val, Suc (hops }\xi), \text{ sip }\xi, \{\}\} p' } \in \text{sterms }\Gamma_{AODV} pp"

and "l' = PRreq:-3"

**show "osn }\xi < \text{sqn (update (rt }\xi) \text{ (oip }\xi) \text{ (osn }\xi, \text{kno, val, Suc (hops }\xi), \text{ sip }\xi, \{\}) (oip }\xi

\[ \wedge \text{the (dhops (update (rt }\xi) \text{ (oip }\xi) \text{ (osn }\xi, \text{kno, val, Suc (hops }\xi), \text{ sip }\xi, \{\}) (oip }\xi

\[ \wedge \text{the (flag (update (rt }\xi) \text{ (oip }\xi) \text{ (osn }\xi, \text{kno, val, Suc (hops }\xi), \text{ sip }\xi, \{\}) (oip }\xi

\[ \leq \text{ Suc (hops }\xi)

\[ \wedge \text{the (flag (update (rt }\xi) \text{ (oip }\xi) \text{ (osn }\xi, \text{kno, val, Suc (hops }\xi), \text{ sip }\xi, \{\}) (oip }\xi

\[ = \text{ val)"

**unfolding update_def by (clarsimp split: option.split)

\[ (\text{metis linorder_neqE_nat not_less)}

**qed**

**have rreq_prrep: "paodv i } \vdash onl \Gamma_{AODV}(\lambda(\xi, 1).

\[ (l \in \{PRreq:-2..PRreq:-7\} \rightarrow \text{(dip }\xi \in \text{kD(rt }\xi)

\[ \wedge \text{sqn (rt }\xi) \text{ (dip }\xi) = \text{dsn }\xi

\[ \wedge \text{the (dhops (rt }\xi) \text{ (dip }\xi) \leq \text{Suc (hops }\xi)

\[ \wedge \text{the (flag (rt }\xi) \text{ (dip }\xi) = \text{val})"

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf preRT_update_changes]

onl_invariant_sterms [OF aodv_wf sip_in_kD])

**have rreq_oip_kD: "paodv i } \vdash onl \Gamma_{AODV}(\lambda(\xi, 1). (l \in \{PRreq:-3..PRreq:-28\} \rightarrow \text{oip }\xi \in \text{kD(rt }\xi)"

by (inv_cterms inv add: onl_invariant_sterms TT [OF aodv_wf preRT_welldefined])

**have rreq_dip_kD_oip_sdn: "paodv i } \vdash onl \Gamma_{AODV}(\lambda(\xi, 1).

\[ (l \in \{PRreq:-18..PRreq:-21\} \rightarrow \text{(dip }\xi \in \text{kD(rt }\xi)

\[ \wedge \text{sqn (rt }\xi) \text{ (oip }\xi) > \text{(osn }\xi

\[ \vee (\text{sqn (rt }\xi) \text{ (oip }\xi) = \text{(osn }\xi

\[ \wedge \text{the (dhops (rt }\xi) \text{ (oip }\xi \leq \text{ Suc (hops }\xi)

\[ \wedge \text{the (flag (rt }\xi) \text{ (oip }\xi) = \text{val})"

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip]

379
onl_invariant_sterms [OF aodv_wf addpreRT_welldefined])

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip]
    onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip]
    onl_invariant_sterms [OF aodv_wf rrep_prrep]
    onl_invariant_sterms [OF aodv_wf rreq_oip_kD]
    onl_invariant_sterms [OF aodv_wf rreq_dip_kD_oip_sqn])
qed

Proposition 7.15

lemma rerr_invalid_any_step_invariant:
  "paodv i \lhd A onll \Gamma AODV (\lambda((\xi, _), a, _). anycast (rerr_invalid (rt \xi)) a)"
proof -
  have dests_inv: "paodv i \lhd A onll \Gamma AODV (\lambda((\xi, l), a, _). anycast (rerr_invalid (rt \xi)) a)
    \lhd (\forall ip \in (\{\text{PAodv-:15, PPkt-:7, PRreq-:11, PRreq-:24, PRrep-:10, PRerr-:1}\} -\rightarrow (\forall ip \in \text{dom}(\text{dests} \xi). ip \in \text{vD}(rt \xi)))
      \land (1 \in \{\text{PAodv-:16..PAodv-:19}\}
          \cup \{\text{PPkt-:8..PPkt-:11}\}
          \cup \{\text{PRreq-:12..PRreq-:15}\}
          \cup \{\text{PRreq-:25..PRreq-:28}\}
          \cup \{\text{PRrep-:11..PRrep-:14}\}
          \cup \{\text{PRerr-:2..PRerr-:5}\} -\rightarrow (\forall ip \in \text{dom}(\text{dests} \xi). ip \in \text{iD}(rt \xi)
            \land \text{the} (\text{dests} \xi ip) = \text{inc} (\text{sqn} (rt \xi) ip)))
      \land (1 = \text{PPkt-:14} -\rightarrow \text{dip} \xi \in \text{iD}(rt \xi)))"
    by inv_cterms (clarsimp split: if_split_asm option.split_asm simp: domIff)+
  show ?thesis
    by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_inv])
qed

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs \((rip, rsn)\) is assigned to the variable \(\text{dests}\) of type \(ip \rightarrow \text{sqn}\), or to the first argument of the function \(\text{rerr}\), this set is a partial function, i.e., there is at most one entry \((rip, rsn)\) for each destination \(rip\): guaranteed by type.

lemma dests_vD_inc_sqn:
  "paodv i \lhd A onll \Gamma AODV (\lambda((\xi, l), (1 \in \{\text{PAodv-:15, PPkt-:7, PRreq-:11, PRreq-:24, PRrep-:10}\}
    \rightarrow (\forall ip \in \text{dom}(\text{dests} \xi). ip \in \text{vD}(rt \xi)
      \land (\text{the} (\text{dests} \xi ip) = \text{inc} (\text{sqn} (rt \xi) ip)))
    \land (1 = \text{PRerr-:1} -\rightarrow (\forall ip \in \text{dom}(\text{dests} \xi). ip \in \text{vD}(rt \xi)
      \land (\text{the} (\text{dests} \xi ip) > \text{sqn} (rt \xi) ip)))"
  by inv_cterms (clarsimp split: if_split_asm option.split_asm simp: domIff)+

Proposition 7.27

lemma route_tables_fresher:
  "paodv i \lhd A (\text{recvmsg rreq_rrep_sn} \rightarrow) onll \Gamma AODV (\lambda((\xi, _), (1 \in \{\text{PAodv-:15, PPkt-:7, PRreq-:11, PRreq-:24, PRrep-:10}\}
    \rightarrow (\forall dip \in \text{iD}(rt \xi). \text{rt} \xi \subseteq \text{dip} \text{rt} \xi))"
proof (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aodv_wf osn_rreq]
    onl_invariant_sterms [OF aodv_wf dsn_rrep]
    onl_invariant_sterms [OF aodv_wf addpreRT_welldefined [THEN invariant_restrict_inD]])

fix \(\xi \, pp \, p'\)
4.7 The quality increases predicate

theory D_Quality_Increases
imports D_Aodv_Predicates D_Fresher

begin

definition quality_increases :: "state ⇒ state ⇒ bool"
where "quality_increases ξ ξ' ≡ (∀ dip∈ kD( rt ξ). dip ∈ kD( rt ξ') ∧ rt ξ ⊑ dip rt ξ') ∧ (∀ dip. sqn ( rt ξ) dip ≤ sqn ( rt ξ') dip)"

lemma quality_increasesI [intro!]:
  assumes "∀ dip. dip ∈ kD( rt ξ) ⇒ dip ∈ kD( rt ξ')"
  and "∀ dip. [ dip ∈ kD( rt ξ); dip ∈ kD( rt ξ') ] ⇒ rt ξ ⊑ dip rt ξ'" 
  and "∀ dip. sqn ( rt ξ) dip ≤ sqn ( rt ξ') dip"
  shows "quality_increases ξ ξ'"

unfolding quality_increases_def using assms by clarsimp

lemma quality_increasesE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
  and "dip∈ kD( rt ξ)"
  and "[ dip ∈ kD( rt ξ'); rt ξ ⊑ dip rt ξ'; sqn ( rt ξ) dip ≤ sqn ( rt ξ') dip ] ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"

using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rt_fresherD [dest]:
  fixes ip
  assumes "quality_increases ξ ξ'"

end
lemma quality_increases_sqnE [elim]:
fixes dip
assumes "quality_increases $\xi \xi'$"
and "sqn (rt $\xi$) dip $\leq$ sqn (rt $\xi'$) dip $\implies$ R dip $\xi$ $\xi'$"
shows "R dip $\xi$ $\xi'$"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases $\xi \xi'$" by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
fixes $\sigma \sigma'$ dip
assumes "rt ($\sigma$ i) $\sqsubseteq dip$ rt ($\sigma'$ nhip)"
and qinc: "quality_increases ($\sigma$ nhip) ($\sigma'$ nhip)"
and "dip $\in$ kD(rt ($\sigma$ nhip))"
shows "rt ($\sigma$ i) $\sqsubseteq dip$ rt ($\sigma'$ nhip)"
proof -
  from qinc have "rt ($\sigma$ nhip) $\sqsubseteq dip$ rt ($\sigma'$ nhip)" using ⟨dip $\in$ kD(rt ($\sigma$ nhip))⟩ by auto
  with ⟨rt ($\sigma$ i) $\sqsubseteq dip$ rt ($\sigma$ nhip)⟩ show ?thesis ..
qed

lemma kD_quality_increases [elim]:
assumes "i $\in$ kD(rt $\xi$)"
and "quality_increases $\xi \xi'$"
shows "i $\in$ kD(rt $\xi'$)"
using assms by auto

lemma kD_nsqn_quality_increases [elim]:
assumes "i $\in$ kD(rt $\xi$)"
and "quality_increases $\xi \xi'$"
shows "i $\in$ kD(rt $\xi'$) $\land$ nsqn (rt $\xi$) i $\leq$ nsqn (rt $\xi'$) i"
proof -
  from assms have "i $\in$ kD(rt $\xi'$)" ..
  moreover with assms have "rt $\xi$ $\sqsubseteq_1$ rt $\xi'$" by auto
  ultimately have "nsqn (rt $\xi$) i $\leq$ nsqn (rt $\xi'$) i"
  using ⟨i $\in$ kD(rt $\xi$)⟩ by (erule(2) rt_fresher_imp_nsqn_le)
  with ⟨i $\in$ kD(rt $\xi'$)⟩ show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
assumes "i $\in$ kD(rt $\xi$)"
and "quality_increases $\xi \xi'$"
shows "nsqn (rt $\xi$) i $\leq$ nsqn (rt $\xi'$) i"
using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
assumes "i $\in$ kD(rt $\xi$)"
and "s $\leq$ nsqn (rt $\xi$) i"
and "quality_increases $\xi \xi'$"
shows "i $\in$ kD(rt $\xi'$) $\land$ s $\leq$ nsqn (rt $\xi'$) i"
proof
  from ⟨i $\in$ kD(rt $\xi$)⟩ and ⟨quality_increases $\xi \xi'$⟩ show "i $\in$ kD(rt $\xi'$)" ..
  next
  from ⟨i $\in$ kD(rt $\xi$)⟩ and ⟨quality_increases $\xi \xi'$⟩ have "nsqn (rt $\xi$) i $\leq$ nsqn (rt $\xi'$) i" ..
  with ⟨s $\leq$ nsqn (rt $\xi$) i⟩ show "s $\leq$ nsqn (rt $\xi'$) i" by (rule le_trans)
qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
assumes "i $\in$ kD(rt $\xi$)"
and "quality_increases ξ ξ'"
and "s < nsqn (rt ξ) i"
shows "s < nsqn (rt ξ') i"
proof -
from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
with "s < nsqn (rt ξ) i": show "s < nsqn (rt ξ') i" by simp
qed

lemma nsqn_quality_increases_dhops [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality_increases ξ ξ'"
and "nsqn (rt ξ) i = nsqn (rt ξ') i"
shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
using assms unfolding quality_increases_def
by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality_increases ξ ξ'"
and "s = nsqn (rt ξ) i"
shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

lemma quality_increases_rreq_rrep_props [elim]:
fixes sn ip hops sip
assumes qinc: "quality_increases (σ sip) (σ' sip)"
and "1 ≤ sn"
and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
∧ (nsqn (rt (σ sip)) ip = sn
→ (the (dhops (rt (σ sip)) ip) ≤ hops
∨ the (flag (rt (σ sip)) ip) = inv))"
shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
∧ (nsqn (rt (σ' sip)) ip = sn
→ (the (dhops (rt (σ' sip)) ip) ≤ hops
∨ the (flag (rt (σ' sip)) ip) = inv))"
(is "_ ∧ ?nsqnafter")
proof -
from * obtain "ip ∈ kD(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto
from quality_increases (σ sip) (σ' sip)
have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
from quality_increases (σ sip) (σ' sip) and ip ∈ kD (rt (σ sip))
have "ip ∈ kD (rt (σ' sip))" ..

from sn ≤ nsqn (rt (σ sip)) ip have ?nsqnafter
proof
assume "sn < nsqn (rt (σ sip)) ip"
also from ip ∈ kD(rt (σ sip)) and quality_increases (σ sip) (σ' sip)
have "... ≤ nsqn (rt (σ' sip)) ip" ..
finally have "sn < nsqn (rt (σ' sip)) ip" .
thus ?thesis by simp
next
assume "sn = nsqn (rt (σ sip)) ip"
with ip ∈ kD(rt (σ sip)) and quality_increases (σ sip) (σ' sip)
have "sn < nsqn (rt (σ' sip)) ip
∨ (sn = nsqn (rt (σ' sip)) ip
∧ the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip))" ..
hence "sn < nsqn (rt (σ' sip)) ip
∨ (nsqn (rt (σ' sip)) ip = sn ∧ the (dhops (rt (σ' sip)) ip) ≤ hops
∨ the (flag (rt (σ' sip)) ip) = inv))"
proof
assume "sn < nsqn (rt (σ' sip)) ip" thus ?thesis ..
next
assume "sn = nsqn (rt (σ' sip)) ip
∧ the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)

hence "sn = nsqn (rt (σ sip)) ip"
and "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)" by auto

from * and ⟨sn = nsqn (rt (σ sip)) ip⟩ have "the (dhops (rt (σ sip)) ip) ≤ hops
∨ the (flag (rt (σ sip)) ip) = inv"

by simp
thus ?thesis

next

assume "the (flag (rt (σ sip)) ip) = inv"

with ⟨ip ∈ kD(rt (σ sip))⟩ have "nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1" ..

from ⟨ip ∈ kD(rt (σ sip))⟩ show ?thesis

proof
(rule vD_or_iD)
assume "ip ∈ vD(rt (σ sip))"

hence "nsqn (rt (σ sip)) ip ≤ sqn (rt (σ sip)) ip" ..

with ⟨sn ≥ 1⟩ and ⟨sn = nsqn (rt (σ sip)) ip⟩ have "sqn (rt (σ sip)) ip > 1" by simp

thus ?thesis ..

qed

lemma quality_increases_rreq_rrep_props':

fixes sn ip hops sip
assumes "∀ j. quality_increases (σ j) (σ' j)"
and "1 ≤ sn"
and *
⟨ip ∈ kD(rt (σ sip))⟩ ∧ sn ≤ nsqn (rt (σ sip)) ip
∧ (nsqn (rt (σ sip)) ip = sn
    → (the (dhops (rt (σ sip)) ip) ≤ hops
        ∨ the (flag (rt (σ sip)) ip) = inv))" shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
∧ (nsqn (rt (σ' sip)) ip = sn
    → (the (dhops (rt (σ' sip)) ip) ≤ hops
        ∨ the (flag (rt (σ' sip)) ip) = inv))"

proof -
from assms(1) have "quality_increases (σ sip) (σ' sip)" ..
thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props)

qed
lemma rteq_quality_increases:
assumes "∀ j. j ≠ i → quality_increases (σ j) (σ' j)"
and "rt (σ' i) = rt (σ i)"
shows "∀ j. quality_increases (σ j) (σ' j)"
using assms by clarsimp (metis order_refl quality_increasesI rt_fresher_refl)

definition msg_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"
where "msg_fresh σ m ≡
  case m of
  Rreq hops rreqid dip dsn dsk oip osn sip handled.
    msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip handled) =
      (dsn ≥ 1 ∧ (sip ≠ oip → oip∈kD(rt (σ sip)) ∧ nsqn (rt (σ sip)) oip ≥ osn ∧ (nsqn (rt (σ sip)) oip = osn → (hops ≥ the (dhops (rt (σ sip)) oip) ∨ the (flag (rt (σ sip)) oip) = inv))))"
  | Rrep hops dip dsn oip osn sip.
    msg_fresh σ (Rrep hops dip dsn oip osn sip) =
      (dsn ≥ 1 ∧ (sip ≠ dip → dip∈kD(rt (σ sip)) ∧ nsqn (rt (σ sip)) dip ≥ dsn ∧ (nsqn (rt (σ sip)) dip = dsn → (hops ≥ the (dhops (rt (σ sip)) dip) ∨ the (flag (rt (σ sip)) dip) = inv))))"
  | Rerr destsc sipc.
    msg_fresh σ (Rerr destsc sipc) =
      (∀ ripc∈dom(destsc). (ripc∈kD(rt (σ sipc)) ∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))
  | _ ⇒ True"

lemma msg_fresh [simp]:
"∀ hops rreqid dip dsn dsk oip osn sip handled.
  msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip handled) =
  (dsn ≥ 1 ∧ (sip ≠ oip → oip∈kD(rt (σ sip)) ∧ nsqn (rt (σ sip)) oip ≥ osn ∧ (nsqn (rt (σ sip)) oip = osn → (hops ≥ the (dhops (rt (σ sip)) oip) ∨ the (flag (rt (σ sip)) oip) = inv))))"
"∀ hops dip dsn oip sip.
  msg_fresh σ (Rrep hops dip dsn oip sip) =
  (dsn ≥ 1 ∧ (sip ≠ dip → dip∈kD(rt (σ sip)) ∧ nsqn (rt (σ sip)) dip ≥ dsn ∧ (nsqn (rt (σ sip)) dip = dsn → (hops ≥ the (dhops (rt (σ sip)) dip) ∨ the (flag (rt (σ sip)) dip) = inv))))"
"∀ dests sip.
  msg_fresh σ (Rerr dests sip) =
  (∀ ripc∈dom(dests). (ripc∈kD(rt (σ sip)) ∧ the (dests ripc) - 1 ≤ nsqn (rt (σ sip)) ripc))"
"∀ d dip.
  msg_fresh σ (Newpkt d dip) = True"
"∀ d dip sip.
  msg_fresh σ (Pkt d dip sip) = True"

unfolding msg_fresh_def by simp_all

lemma msg_fresh_inc_sn [simp, elim]:
"msg_fresh σ m ⇒ rreq_rrep_sn m"
by (cases m) simp_all

lemma recv_msg_fresh_inc_sn [simp, elim]:
"orecvmsg (msg_fresh) σ m ⇒ recvmsg rreq_rrep_sn m"
by (cases m) simp_all

lemma rreqmsgs nsnq_is_fresh [simp]:
fixes σ msg hops rreqid dip dsn dsk oip osn sip handled
assumes "rreq_rrep_fresh (rt (σ sip)) (Rreq hops rreqid dip dsn dsk oip osn sip handled)"
and "rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip handled)"
shows "msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip handled)"
is "msg_fresh σ ?msg"

proof -
  let ?rt = "rt (σ sip)"
from assms(2) have "1 ≤ osn" by simp
thus ?thesis
unfolding msg_fresh_def
proof (simp only: msg.case, intro conjI impI)
assume "sip ≠ oip"
with assms(1) show "oip ∈ kD(?rt)" by simp
next
assume "sip ≠ oip"
and "nsqn ?rt oip = osn"
show "the (dhops ?rt oip) ≤ hops ∨ the (flag ?rt oip) = inv" proof (cases "oip∈vD(?rt)")
  assume "oip∈vD(?rt)"
  hence "nsqn ?rt oip = sqn ?rt oip" ..
  with ⟨nsqn ?rt oip = osn⟩ have "sqn ?rt oip = osn" by simp
  with assms(1) and ⟨sip ≠ oip⟩ have "the (dhops ?rt oip) ≤ hops" by simp
  thus ?thesis ..
next
assume "oip∉vD(?rt)"
moreover from assms(1) and ⟨sip ≠ oip⟩ have "oip∈kD(?rt)" by simp
ultimately have "oip ∈ iD(?rt)" by auto
hence "the (flag ?rt oip) = inv" ..
thus ?thesis ..
qed
next
assume "sip ≠ oip" with assms(1) have "osn ≤ sqn ?rt oip" by auto
thus "osn ≤ nsqn (rt (σ sip)) oip" proof (rule nat_le_eq_or_lt)
  assume "osn < sqn ?rt oip"
  hence "osn ≤ sqn ?rt oip - 1" by simp
  also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
  finally show "osn ≤ nsqn ?rt oip" .
next
assume "osn = sqn ?rt oip"
with assms(1) and ⟨sip ≠ oip⟩ have "oip∈kD(?rt)"
and "the (flag ?rt oip) = val"
by auto
hence "nsqn ?rt oip = sqn ?rt oip" ..
with ⟨osn = sqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
thus "osn ≤ nsqn ?rt oip" by simp
qed
qed simp

lemma rrep_nsqn_is_fresh [simp]:
fixes σ msg hops dip dsn oip sip
assumes "rrep_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
and "rrep_rrep_sn (Rrep hops dip dsn oip sip)"
shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
(is "msg_fresh σ ?msg")
proof -
  let ?rt = "rt (σ sip)"
  from assms have "sip ≠ dip → dip∈kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
    by simp
  hence "sip ≠ dip → dip∈kD(?rt) ∧ nsqn ?rt dip ≥ dsn"
    by clarsimp
  with assms show "msg_fresh σ ?msg"
    by clarsimp
qed

lemma rerr_nsqn_is_fresh [simp]:
fixes σ dests sip
assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
shows "msg_fresh σ (Rerr dests sip)"
(is "msg_fresh σ ?msg")
proof -
  let ?rt = "rt (σ sip)"

from assms have *: "(\forall rip \in \text{dom}(dests). (rip \in \text{iD}\,(rt\,(\sigma\,sip)))
\land \text{the}\,(\text{dests}\,\text{rip}) = \text{sqn}\,(\text{rt}\,(\sigma\,sip))\,\text{rip}))"
by clarsimp
have "(\forall rip \in \text{dom}(dests). (rip \in \text{kD}\,(rt\,(\sigma\,sip)))
\land \text{the}\,(\text{dests}\,\text{rip}) - 1 \leq \text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,\text{rip})"
proof
fix rip
assume "rip \in \text{dom}\,\text{dests}"
with * have "rip \in \text{iD}\,(\text{rt}\,(\sigma\,sip))" and "the\,(\text{dests}\,\text{rip}) = \text{sqn}\,(\text{rt}\,(\sigma\,sip))\,\text{rip})"
by auto
from this(2) have "the\,(\text{dests}\,\text{rip}) - 1 = \text{sqn}\,(\text{rt}\,(\sigma\,sip))\,\text{rip} - 1"
by simp
also have "... \leq \text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,\text{rip}" by (rule \text{sqn_nsqn})
finally have "the\,(\text{dests}\,\text{rip}) - 1 \leq \text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,\text{rip}".
with \(rip \in \text{iD}\,(rt\,(\sigma\,sip))\):
show "rip \in \text{kD}\,(rt\,(\sigma\,sip)) \land \text{the}\,(\text{dests}\,\text{rip}) - 1 \leq \text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,\text{rip}" by clarsimp
qed
thus "msg\,fresh\,\sigma\,?msg"
by simp
qed
lemma quality\,increases\_msg\,fresh [elim]:
assumes qinc: "\(\forall j. \text{quality\,increases}\,(\sigma\,j)\,(\sigma'\,j)\)"
and "msg\,fresh\,\sigma\,m"
shows "msg\,fresh\,\sigma'\,m"
using assms(2)
proof (cases m)
fix hops rreqid dip dsn dsk oip osn sip handled
assume [simp]: "m = Rreq\,hops\,rreqid\,dip\,dsn\,dsk\,oip\,osn\,sip\,handled"
and "msg\,fresh\,\sigma\,m"
then have "osn \geq 1" and "sip = oip \lor (oip \in \text{kD}\,(rt\,(\sigma\,sip)) \land osn \leq \text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,oip
\land (\text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,oip = osn
\lor (\text{the}\,(\text{dhops}\,(\text{rt}\,(\sigma\,sip))\,oip) \leq hops
\lor \text{the}\,(\text{flag}\,(\text{rt}\,(\sigma\,sip))\,oip) = inv)))"
by auto
from this(2) show ?thesis
proof
assume "sip = oip" with \(osn \geq 1\) show ?thesis by simp
next
assume "oip\in\text{kD}(rt\,(\sigma\,sip)) \land osn \leq \text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,oip
\land (\text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,oip = osn
\lor (\text{the}\,(\text{dhops}\,(\text{rt}\,(\sigma\,sip))\,oip) \leq hops
\lor \text{the}\,(\text{flag}\,(\text{rt}\,(\sigma\,sip))\,oip) = inv)))"
moreover from qinc have "quality\,increases\,(\sigma\,sip)\,(\sigma'\,sip)"..
ultimately have "oip\in\text{kD}(rt\,(\sigma'\,sip)) \land osn \leq \text{nsqn}\,(\text{rt}\,(\sigma'\,sip))\,oip
\land (\text{nsqn}\,(\text{rt}\,(\sigma'\,sip))\,oip = osn
\lor (\text{the}\,(\text{dhops}\,(\text{rt}\,(\sigma'\,sip))\,oip) \leq hops
\lor \text{the}\,(\text{flag}\,(\text{rt}\,(\sigma'\,sip))\,oip) = inv)))"
using \(osn \geq 1\) by (rule quality\,increases\_rreq\_rrep\_props [rotated 2])
with \(osn \geq 1\) show "msg\,fresh\,\sigma'\,m"
by (clarsimp)
qed
next
fix hops dip dsn oip sip
assume [simp]: "m = Rrep\,hops\,dip\,dsn\,oip\,sip"
and "msg\,fresh\,\sigma\,m"
then have "dsn \geq 1" and "sip = dip \lor (dip\in\text{kD}(rt\,(\sigma\,sip)) \land dsn \leq \text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,dip
\land (\text{nsqn}\,(\text{rt}\,(\sigma\,sip))\,dip = dsn
\lor (\text{the}\,(\text{dhops}\,(\text{rt}\,(\sigma\,sip))\,dip) \leq hops
\lor \text{the}\,(\text{flag}\,(\text{rt}\,(\sigma\,sip))\,dip) = inv)))"
by auto
from this(2) show "?thesis"
proof
  assume "sip = dip" with \( dsn \geq 1 \) show \(?thesis\) by simp
next
  assume \( \text{"dip} \in kD(rt (\sigma \text{ sip}) \land dsn \leq \text{nsqn } (rt (\sigma \text{ sip})) \text{ dip} \land (\text{nsqn } (rt (\sigma \text{ sip})) \text{ dip} = dsn \rightarrow (\text{the } (dhops (rt (\sigma \text{ sip})) \text{ dip}) \leq \text{hops} \lor \text{the } (flag (rt (\sigma \text{ sip})) \text{ dip}) = \text{inv})")\)
moreover from qinc have "quality_increases (\sigma \text{ sip}) (\sigma' \text{ sip})" ..
ultimately have "\( \text{dip} \in kD(rt (\sigma' \text{ sip}) \land dsn \leq \text{nsqn } (rt (\sigma' \text{ sip})) \text{ dip} \land (\text{nsqn } (rt (\sigma' \text{ sip})) \text{ dip} = dsn \rightarrow (\text{the } (dhops (rt (\sigma' \text{ sip}) \text{ dip}) \leq \text{hops} \lor \text{the } (flag (rt (\sigma' \text{ sip})) \text{ dip}) = \text{inv})")\)

using \( dsn \geq 1 \) by \( \text{rule quality_increases_rreq_rrep_props [rotated 2]} \)

next
  fix dests sip
  assume \[\text{[simp]}\]: "m = Rerr dests sip"
  and "\text{msg_fresh } \sigma \text{ m}"
  then have "\( \forall \text{rip} \in \text{dom(dests)}. \text{rip} \in kD(rt (\sigma \text{ sip})) \land \text{the } (\text{dests rip}) - 1 \leq \text{nsqn } (rt (\sigma \text{ sip})) \text{ rip}\)"
  proof
    fix rip
    assume "\text{rip} \in \text{dom(dests)}"
    with \( * \) have "\text{rip} \in kD(rt (\sigma \text{ sip})) \land \text{the } (\text{dests rip}) - 1 \leq \text{nsqn } (rt (\sigma \text{ sip})) \text{ rip}\)"
    by simp

    moreover from qinc have "quality_increases (\sigma \text{ sip}) (\sigma' \text{ sip})" by simp
    ultimately show "\text{rip} \in kD(rt (\sigma' \text{ sip}) \land \text{the } (\text{dests rip}) - 1 \leq \text{nsqn } (rt (\sigma' \text{ sip})) \text{ rip}\) ..
    qed
  thus \(?thesis\) by simp
  qed simp_all
end

4.8 The ‘open’ AODV model

theory \(D_\text{OAodv}\)
imports \(D_\text{Aodv} \text{ AWN.OAWN_SOS_Labels AWN.OAWN_Convert}\)
begin
Definitions for stating and proving global network properties over individual processes.
definition \(\sigma_{\text{AODV}}' :: ((ip \Rightarrow \text{state}) \times ((\text{state}, msg, pseqp, pseqp label) seqp)) \text{ set}\)
where \("\sigma_{\text{AODV}}' \equiv \{ (\lambda \text{ i. aodv_init i, } \Gamma_{\text{AODV}} \text{ PAodv}) \}\)"
abbreviation opaodv
:: "ip \Rightarrow ((ip \Rightarrow \text{state}) \times (\text{state}, msg, pseqp, pseqp label) seqp, msg seq_action) \text{ automaton}\"
where \("\text{opaodv } i \equiv \{ \text{ init } = \sigma_{\text{AODV}}', \text{ trans } = \text{ oseqp_sos } \Gamma_{\text{AODV}} \text{ i } \}\)"
lemma initiali_aodv \[\text{intro!, simp}\]: "initiali i (\text{init (opaodv i)}) \text{ (init (paodv i)})"
unfolding \(\sigma_{\text{AODV}}\_\text{def} \text{ \sigma_{AODV}'_\text{def} by rule simp_all}\)
lemma oadv_control_within \[\text{simp}\]: "\text{control_within } \Gamma_{\text{AODV}} \text{ (init (opaodv i))}\"
unfolding \(\sigma_{\text{AODV}}'_\text{def} \text{ by rule control_withinI } \text{ (auto simp del: } \Gamma_{\text{AODV}}\_\text{simp}\)"
lemma \(\sigma_{\text{AODV}}'_\text{labels} \[\text{simp}\]: "\( (\sigma, p) \in \sigma_{\text{AODV}}' \implies \text{labels } \Gamma_{\text{AODV}} \text{ p } = \{ \text{PAodv - 0}\}\)"
unfolding \(\sigma_{\text{AODV}}'\_\text{def} \text{ by simp}\)
lemma oadv_init_kD_empty \[\text{simp}\]: 
"\( (\sigma, p) \in \sigma_{\text{AODV}}' \implies kD (rt (\sigma \text{ i})) = \{\}\)"

388
unfolding $\sigma_{AODV}'_{-def}$ kD_def by simp

lemma oadv_init_vD_empty [simp]:
"$(\sigma, p) \in \sigma_{AODV}' \Longrightarrow vD (rt (\sigma \ i)) = {}$"
unfolding $\sigma_{AODV}'_{-def}$ vD_def by simp

lemma oadv_trans: "trans (opaodv i) = oseqp_sos $\Gamma_{AODV}$ i"
by simp

declare oseq_invariant_ctermsI [OF oadv_wf oadv_control_within oadv_simple_labels oadv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF oadv_wf oadv_control_within oadv_simple_labels oadv_trans, cterms_intros]

end

4.9 Global invariant proofs over sequential processes

theory D_Global_Invariants
imports D_Seq_Invariants
  D_Aodv_Predicates
  D_Fresher
  D_Quality_Increases
 AWN.OAWN_Convert
  D_OAodv
begin

lemma other_quality_increases [elim]:
assumes "other quality_increases I $\sigma \sigma'$"
shows "\(\forall j. quality_increases (\sigma j) (\sigma' j)\)"
using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q) $\sigma \sigma'$ a" 
  and weakenP: "\(\forall \sigma m. P \sigma m \Longrightarrow P' \sigma m\)" 
  and weakenQ: "\(\forall \sigma m. Q \sigma m \Longrightarrow Q' \sigma m\)"
  shows "otherwith P' I (orecvmsg Q') $\sigma \sigma'$ a"
proof
  fix j
  assume "\(j \notin I\)"
  with * have "P (\sigma j) (\sigma' j)" by auto
  thus "P' (\sigma j) (\sigma' j)" by (rule weakenP)
next
  from * have "orecvmsg Q \sigma a" by auto
  thus "orecvmsg Q' \sigma a" by rule (erule weakenQ)
qed

lemma oreceived_msg_inv:
  assumes other: "\(\forall \sigma \sigma' m. \left[ P \sigma m; other Q \{i\} \sigma \sigma' \right] \Longrightarrow P \sigma' m\)"
  and local: "\(\forall \sigma m. P \sigma m \Longrightarrow P (\sigma (i := \sigma i (msg := m))) m\)"
  shows "\(\forall \sigma' \sigma a. orecvmsg Q \sigma a\)"
proof (inv_cterms, intro impI)
  fix \(\sigma \sigma' l\)
  assume "l = PAodv -:1 \Longrightarrow P \sigma (msg (\sigma i))"
  and "l = PAodv -:1"
  and "other Q \{i\} \sigma a"
  from this(1-2) have "P \sigma (msg (\sigma i))" ..
  hence "P \sigma' (msg (\sigma i))" using \(\forall \sigma Q \{i\} \sigma \sigma'\) 
  by (rule other)
  moreover from \(\forall \sigma Q \{i\} \sigma \sigma'\) have "\sigma' i = \sigma i" ..
  ultimately show "P \sigma' (msg (\sigma i))" by simp
next
\begin{proof}

\begin{align*}
\text{fix } \sigma &\, \sigma' \, \text{msg} \\
\text{assume } \text{"otherwith } Q \{i\} (\text{orevmsg } P) \sigma \sigma' \text{ (receive msg)"} \\
&\quad \text{and } \text{"} \sigma' i = \sigma i \{\text{msg} := \text{msg}\} \text{"}
\end{align*}

from this(1) have \text{"} P \sigma \text{msg} \text{"}

and \text{"} \forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j) \text{"} \text{by auto}

from this(1) have \text{"} P (\sigma (i := \sigma i \{\text{msg} := \text{msg}\})) \text{msg} \text{"} \text{by (rule local)}

\text{thus } \text{"} P \sigma' \text{msg} \text{"}

\text{proof (rule other)}

\text{from } \langle \sigma' i = \sigma i \{\text{msg} := \text{msg}\} \rangle \text{ and } \langle \forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j) \rangle

\text{show } \text{"} \text{other } Q \{i\} (\sigma (i := \sigma i \{\text{msg} := \text{msg}\})) \text{"}

\text{by } - \text{ (rule otherI, auto)}

\text{qed}

\text{qed (Equivalent to) Proposition 7.27}

\end{proof}

\begin{lemmas}

\text{lemma local_quality_increases:}

\text{"opaodv i } \models A \text{ (recvmsg req_rrep_sn } \rightarrow \text{ onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _,)). \text{quality_increases } \xi \xi')"

\text{proof (rule step_invariantI)}

\text{fix } s a s'

\text{assume sr: } \text{"} s \in \text{reachable (opaodv i) (recvmsg req_rrep_sn)"}

\text{and tr: } \text{"} (s, a, s') \in \text{trans (opaodv i)"}

\text{and rm: } \text{"} \text{recvmsg req_rrep_sn } a \text{"}

\text{from sr have srTT: } \text{"} s \in \text{reachable (opaodv i) TT"}

\text{from route_tables_fresher sr tr rm}

\text{have } \text{"} \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{dip } kD (\text{rt } \xi) \text{rt } \xi' ) (s, a, s') \text{"}

\text{by (rule step_invariantD)}

\text{moreover from known_destinations_increase srTT tr TT_True}

\text{have } \text{"} \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{ip. } \text{sqn (rt } \xi) \text{ip } \leq \text{sqn (rt } \xi') \text{ip) (s, a, s')"}

\text{by (rule step_invariantD)}

\text{moreover from sqns_increase srTT tr TT_True}

\text{have } \text{"} \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{ip. } \text{sqn (rt } \xi) \text{ip } \leq \text{sqn (rt } \xi') \text{ip) (s, a, s')"}

\text{by (rule step_invariantD)}

\text{ultimately show } \text{"} \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{quality_increases } \xi \xi') (s, a, s') \text{"}

\text{unfolding onll_def by auto}

\text{qed}

\text{lemmas olocal_quality_increases =}

open_seq_step_invariant \[OF local_quality_increases initiali_aodv oaodv_trans aodv_trans, simplified seqll_onll_swap]\n
\text{lemma equality_increases:}

\text{"opaodv i } \models A \text{ (otherwith quality_increases } \{i\} (\text{orevmsg } \lambda_. \text{req_rrep_sn}),

\text{other quality_increases } \{i\} \rightarrow )

\text{onll } \Gamma_{AODV} (\lambda((\sigma, _,), _, (\sigma', _,)). \forall \text{j. quality_increases } (\sigma j) (\sigma' j))" \text{ (is } \text{"} \models A \text{ (?)S, } \rightarrow ) \text{"}

\text{proof (rule onll_ostep_invariantI, simp)}

\text{fix } \sigma p a \sigma' p' l'

\text{assume or: } \text{"} (\sigma, p) \in \text{oreachable (opaodv i) } ?S \text{ (other quality_increases } \{i\})" 

\text{and ll: } \text{"} l \in \text{labels } \Gamma_{AODV} p \text{"}

\text{and } ?S \sigma \sigma' a

\text{and tr: } \text{"} (\{(\sigma, p), a, (\sigma', p')\} \in \text{oseqp_sos } \Gamma_{AODV} i \text{"}

\text{and ll': } \text{"} l' \in \text{labels } \Gamma_{AODV} p' \text{"}

\text{from this(1-3) have } \text{"} \text{orevmsg } (\lambda_. \text{req_rrep_sn} ) \sigma a \text{"}

\text{by (auto dest!: oreachable_weakenE \[where QS="act (recvmsg req_rrep_sn)" 

\text{and QU="other quality_increases } \{i\} \text{"]}

\text{otherwith_actionD) with or have orv: } \text{"} (\sigma, p) \in \text{oreachable (opaodv i) (act (recvmsg req_rrep_sn))}

\text{other quality_increases } \{i\}\text{"}

\text{by } -\text{ (erule oreachable_weakenE, auto) with tr ll ll' and } \text{orevmsg } (\lambda_. \text{req_rrep_sn}) \sigma a \text{ have } \text{"} \text{quality_increases } (\sigma i) (\sigma' i) \text{"}

\end{lemmas}
by \((\text{drule oll\_ostep\_invariantD \[OF olocal\_quality\_increases\]}, \text{auto simp: seqll\_def})\)
with \(?S \sigma \sigma' a \) show \("\forall j. \text{quality\_increases} (\sigma j) (\sigma' j)"
by \((\text{auto dest!: otherwith\_syncD})\)
qed

lemma \(\text{rreq\_rrep\_nsqf\_any\_step\_invariant}:\)
"opaodv i \(\vdash A \) (act (recvmsg rreq\_rrep\_sn), other A \(\{i\} \rightarrow\)
\(\text{onll} \Gamma_{\text{AODV}} (\lambda((\sigma, _), a, _). \text{anycast} (\text{msg\_fresh} \sigma) a)\)"
proof \((\text{rule ostep\_invariantI}, \text{simp del: act\_simp})\)
fix \(\sigma p a \sigma' p'\)
assume or: "\((\sigma, p) \in \text{oreachable} (opaodv i) \) (act (recvmsg rreq\_rrep\_sn)) \(\) (other A \(\{i\})"
and "\((\sigma, p), a, (\sigma', p') \)) \in \text{oseqp\_sos} \Gamma_{\text{AODV}} i" 
and recv: "act (recvmsg rreq\_rrep\_sn) \sigma \sigma' a"
obtain \(l l'\) where "\(l \in \text{labels} \Gamma_{\text{AODV}} p\)" and 
"\(l' \in \text{labels} \Gamma_{\text{AODV}} p'\)
by \((\text{metis aodv\_ex\_label})\)
from \("((\sigma, p), a, (\sigma', p') \)) \in \text{oseqp\_sos} \Gamma_{\text{AODV}} i\)"
have \("((\sigma, p), a, (\sigma', p') \)) \in \text{trans} (opaodv i)"
by simp
have "anycast (rreq\_rrep\_fresh (rt (\sigma i))) a"
proof -
have "opaodv i \(\vdash A \) (act (recvmsg rreq\_rrep\_sn), other A \(\{i\} \rightarrow\)
\(\text{onll} \Gamma_{\text{AODV}} (\text{seqll} i \(\lambda((\xi, _), a, _). \text{anycast} (\text{rreq\_rrep\_fresh} (\text{rt} \xi)) a)\))"
by \((\text{rule ostep\_invariant\_weakenE \[OF open_seq\_step\_invariant \[OF rreq\_rrep\_fresh\_any\_step\_invariant initiali\_aodv, simplified seqll\_onll\_swap\]\])\) auto
hence "\(\text{onll} \Gamma_{\text{AODV}} (\text{seqll} i \(\lambda((\xi, _), a, _). \text{anycast} (\text{rreq\_rrep\_fresh} (\text{rt} \xi)) a)\))
\((\sigma, p), a, (\sigma', p'))"
using or tr recv by - \((\text{erule(4) ostep\_invariant\_E})\)
thus \(?\text{thesis}\)
using \("l \in \text{labels} \Gamma_{\text{AODV}} p\) and \("l' \in \text{labels} \Gamma_{\text{AODV}} p'\)\) by auto
qed

moreover have "anycast (rerr\_invalid (rt (\sigma i))) a"
proof -
have "opaodv i \(\vdash A \) (act (recvmsg rreq\_rrep\_sn), other A \(\{i\} \rightarrow\)
\(\text{onll} \Gamma_{\text{AODV}} (\text{seqll} i \(\lambda((\xi, _), a, _). \text{anycast} (\text{rerr\_invalid} (\text{rt} \xi)) a)\))"
by \((\text{rule ostep\_invariant\_weakenE \[OF open_seq\_step\_invariant \[OF rerr\_invalid\_any\_step\_invariant initiali\_aodv, simplified seqll\_onll\_swap\]\])\) auto
hence "\(\text{onll} \Gamma_{\text{AODV}} (\text{seqll} i \(\lambda((\xi, _), a, _). \text{anycast} (\text{rerr\_invalid} (\text{rt} \xi)) a)\))
\((\sigma, p), a, (\sigma', p'))"
using or tr recv by - \((\text{erule(4) ostep\_invariant\_E})\)
thus \(?\text{thesis}\)
using \("l \in \text{labels} \Gamma_{\text{AODV}} p\) and \("l' \in \text{labels} \Gamma_{\text{AODV}} p'\)\) by auto
qed

moreover have "anycast rreq\_rrep\_sn a"
proof -
from or tr recv
have "\(\text{onll} \Gamma_{\text{AODV}} (\text{seqll} i \(\lambda(\_, a, _). \text{anycast} \text{rreq\_rrep\_sn} a)) \((\sigma, p), a, (\sigma', p')\)"
by \((\text{rule ostep\_invariant\_E \[OF open_seq\_step\_invariant \[OF rreq\_rrep\_sn\_any\_step\_invariant initiali\_aodv, oadv\_trans oadv\_trans, simplified seqll\_onll\_swap\]\])\) auto
thus \(?\text{thesis}\)
using \("l \in \text{labels} \Gamma_{\text{AODV}} p\) and \("l' \in \text{labels} \Gamma_{\text{AODV}} p'\)\) by auto
qed

moreover have "anycast (\lambda m. \text{not\_Pkt} m \rightarrow \text{msg\_sender} m = i) a"
proof -
have "opaodv i \(\vdash A \) (act (recvmsg rreq\_rrep\_sn), other A \(\{i\} \rightarrow\)
\(\text{onll} \Gamma_{\text{AODV}} (\text{seqll} i \(\lambda((\xi, _), a, _). \text{anycast} (\lambda m. \text{not\_Pkt} m \rightarrow \text{msg\_sender} m = i) a))\))"
by \((\text{rule ostep\_invariant\_weakenE \[OF open_seq\_step\_invariant \[OF sender\_ip\_valid initiali\_aodv, simplified seqll\_onll\_swap\]\])\) auto

ultimately have "anycast (msg_fresh σ) a"
by (simp_all add: anycast_def
del: msg_fresh
split: seq_action.split_asm msg.split_asm) simp_all
thus "onll Γ_AODV (λ((σ, _, a, _). anycast (msg_fresh σ) a) ((σ, p), a, (σ', p'))"
by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaodv i |?= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onl Γ_AODV (λ(σ, l). l ∈ {PAodv-:1} −→ msg_fresh σ (msg (σ i)))"
proof (rule oreceived_msg_inv)
fix σ σ' m
assume *: "msg_fresh σ m" and "other quality_increases {i} σ σ'"
from this(2) have "∀ j. quality_increases (σ j) (σ' j)" ..
thus "msg_fresh σ' m" using *
next
fix σ m
assume "msg_fresh σ m"
thus "msg_fresh (σ(i := σ i |msg := m)) m" proof (cases m)
fix dests sip
assume "m = Rerr dests sip"
with ⟨msg_fresh σ m⟩ show ?thesis by auto
qed auto

lemma oquality_increases_nsqn_fresh:
"opaodv i |?= A (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onl Γ_AODV (λ((σ, _, a, _). anycast (msg_fresh σ) a) ((σ, p), a, (σ', p')))
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
"opaodv i |?= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onl Γ_AODV (λ(σ, l). (l ∈ {PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2} ∧ sip (σ i) ≠ oip (σ i)) →
oip (σ i) ∈ kD(rt (σ (sip (σ i)))) ∧ nsqn (rt (σ (sip (σ i)))) (oip (σ i)) ≥ osn (σ i)
∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i)
→ (hops (σ i) ≥ (dhops (rt (σ (sip (σ i)))) (oip (σ i)))
∧ the (flag (rt (σ (sip (σ i)))) (oip (σ i))) = inv))))
(is "_ |?= (?S, ?U →)_")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh
aodv_wf oaodv_trans]
onl_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq]
simp add: seqlsimp
simp del: One_nat_def, rule impI)
fix \( \sigma \sigma' \) p l

assume "\((\sigma, p) \in \text{oreachable} \ (\text{opaodv} \ i) ?S ?U" 
and "1 \in \text{labels} \Gamma_{AODV} \ p" 
and pre:

\[
(l = \text{PAodv}:-4 \lor l = \text{PAodv}:-5 \lor l = \text{PRreq}:-0 \lor l = \text{PRreq}:-2) \land \text{sip} (\sigma i) \neq \text{oip} (\sigma i) \\
\implies \text{osp} (\sigma i) \in kD (rt (\sigma (\text{sip} (\sigma i)))) \\
\land \text{osn} (\sigma i) \leq \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i)) \\
\land \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i)) = \text{osn} (\sigma i) \\
\implies \text{the} (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) \leq \text{hops} (\sigma i) \\
\lor \text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) = \text{inv}\)
\]

and "other_quality_increases \{i\} \sigma \sigma'" 
and hyp: "\((l = \text{PAodv}:-4 \lor l = \text{PAodv}:-5 \lor l = \text{PRreq}:-0 \lor l = \text{PRreq}:-2) \land \text{sip} (\sigma' i) \neq \text{oip} (\sigma' i)" 
(is "?labels \land \text{sip} (\sigma' i) \neq \text{oip} (\sigma' i)"

from (4) have "\(\sigma' i = \sigma i \)" ..

with hyp have hyp': "?labels \land \text{sip} (\sigma i) \neq \text{oip} (\sigma i)" by simp

show "\(\text{osp} (\sigma i) \in kD (rt (\sigma (\text{sip} (\sigma i)))) \land \text{osn} (\sigma i) \leq \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i)) \land \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i)) = \text{osn} (\sigma i) \implies \text{the} (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) \leq \text{hops} (\sigma i) \lor \text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) = \text{inv}\)"

proof (cases "\text{sip} (\sigma i) = i")

assume "\(\text{osp} (\sigma i) \neq i\)"

from (other_quality_increases \{i\} \sigma \sigma')
have "\(\text{quality_increases} (\sigma (\text{sip} (\sigma i))) \land (\sigma (\text{sip} (\sigma i))) = (\sigma (\sigma i))\)"

by (rule otherE) (clarsimp simp: seqlsimp)

moreover from \(\sigma, p \in \text{oreachable} \ (\text{opaodv} \ i) ?S ?U\) \(1 \in \text{labels} \Gamma_{AODV} \ p\) and hyp
have "1 \leq \text{osn} (\sigma i)"

by (auto dest!: onl_oninvariant_weakenD [OF oosn_rreq]

simp add: sqelnsimp \(\sigma' i = \sigma i\))

moreover from \(\text{sip} (\sigma i) \neq i\) hyp' and pre
have "\(\text{osp} (\sigma i) \in kD (rt (\sigma (\text{sip} (\sigma i)))) \land \text{osn} (\sigma i) \leq \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i)) \land \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i)) = \text{osn} (\sigma i) \implies \text{the} (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) \leq \text{hops} (\sigma i) \lor \text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) = \text{inv}\)"

by (auto simp: \(\sigma' i = \sigma i\))

ultimately show ?thesis

by (rule quality_increases_rreq_rrep_props)

next

assume "\text{sip} (\sigma i) = i" thus ?thesis

using \(\sigma' i = \sigma i\) hyp and pre by auto

qed

qd (auto elim!: quality_increases_rreq_rrep_props)

lemma odsn_rrep:

\text{opaodv} \ i \models (\text{other_with_quality_increases} \{i\} \ (\text{orecvmsg} \ \text{msg_fresh}), \ \text{other_quality_increases} \{i\} \rightarrow)

\text{onl} \ \Gamma_{AODV} \ (\text{seq1} (\text{\lambda}(\xi, 1). \ l \in \{\text{PAodv}:-6, \text{PAodv}:-7\} \cup \{\text{PRrep}:-n|n. \ True\} \rightarrow \ 1 \leq \text{dsn} \ (\xi)))"

by (rule oninvariant_weakenE [OF open_seq_invariant [OF dsn_rrep_initiali_aodv]])

(auto simp: seq1_onl_swap)

lemma rrep_sip:

\text{opaodv} \ i \models (\text{other_with_quality_increases} \{i\} \ (\text{orecvmsg} \ \text{msg_fresh}), \ \text{other_quality_increases} \{i\} \rightarrow)

\text{onl} \ \Gamma_{AODV} \ (\text{\lambda}(\sigma, 1).

\(1 \in \{\text{PAodv}:-6, \text{PAodv}:-7, \text{PRrep}:-0, \text{PRrep}:-1\} \land \text{sip} (\sigma i) \neq \text{dip} (\sigma i) \rightarrow \ \text{dip} (\sigma i) \in kD (rt (\sigma (\text{sip} (\sigma i)))) \land \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i)) \geq \text{dsn} (\sigma i) \land \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i)) = \text{dsn} (\sigma i) \implies \text{hops} (\sigma i) \geq \text{the} (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i))) \lor \text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i))) = \text{inv}))"

(is "\_ \models (?S, ?U \rightarrow \_)"

proof (inv_cterms inv add: oseq_step_invariant_sterms [OF quality_increases_nsqn_fresh aodv_wf aodv_trans]
lemma rerr_sip:
"oapadv i |¬ (otherwith quality_increases (i) (orecvmsg msg_fresh),
other quality_increases (i) →)
onl Γ_{AODV} (λ(sip, i).
1 ∈ {PAodv::8, PAodv::9, PReq::0, PReq::1} 
→ (∀rip∈dom(dests (sip i)). rip∈kd(rt (σ (sip i)))) ∧
the (dests (sip i) ripc) - 1 ≤ nsqn (rt (σ (sip i)))) ripc)"
(is "_ |¬ (?S, ?U → _) _")

proof -
{ fix dests rip sip rsn and σ σ' :: "ip ⇒ state"
assume γinc: "∀j. quality_increases (σ j) (σ' j)"
and *: "∀rip∈dom dests. rip ∈ kd (rt (σ sip))
∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
and "dests rip = Some rsn"
from this(3) have "rip∈dom dests" by auto
with * and (dests rip = Some rsn) have "rip∈kd(rt (σ sip))"
and "rsn - 1 ≤ nsqn (rt (σ sip)) rip"

by (auto dest!: bspec)
from qinc have "quality_increases (σ sip) (σ' sip)" ..
have "rip ∈ kD(rt (σ sip)) ∧ rsn - 1 ≤ nsqn (rt (σ sip)) rip"
proof
  from rip∈kD(rt (σ sip)) and quality_increases (σ sip) (σ' sip)
  show "rip ∈ kD(rt (σ sip))" ..
next
  from rip∈kD(rt (σ sip)) and quality_increases (σ sip) (σ' sip)
  have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ sip)) rip" ..
with rsn - 1 ≤ nsqn (rt (σ sip)) rip show "rsn - 1 ≤ nsqn (rt (σ sip)) rip"
  by (rule le_trans)
qed

{ note partial = this

show ?thesis
by (inv_cterms inv add: oseq_step_invariant_sterms [OF quality_increases_nsqn_fresh aodv_wf oadv_trans]
onl_o invariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
other_qu 1, intro conjl)
(clarsimp simp del: One_nat_def split: if_split_asm option.split_asm, erule(2) partial)+
qed

lemma prerr_guard: "paodv i \|= onl Γ\ AODV (λ(ξ , _). (1 = PRem:-1
\rightarrow (∀ip∈dom(dests ξ). ip∈vD(rt ξ)
∧ the (nhop (rt ξ) ip) = sip ξ ∧ sqn (rt ξ) ip < the (dests ξ ip))))"
by (inv_cterms) (clarsimp split: option.split_asm if_split_asm)

lemmas oaddpreRT_welldefined =
open_seq_invariant [OF addpreRT_welldefined initiali_aodv oadv_trans aodv_trans,
simplified seql_onl_swap,
THEN oinvariant_anyact]

lemmas odests_vD_inc_sqn =
open_seq_invariant [OF dests_vD_inc_sqn initiali_aodv oadv_trans aodv_trans,
simplified seql_onl_swap,
THEN oinvariant_anyact]

lemmas oprerr_guard =
open_seq_invariant [OF prerr_guard initiali_aodv oadv_trans aodv_trans,
simplified seql_onl_swap,
THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':
  "opaodv i \|= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} \rightarrow onl Γ\ AODV (λ(σ , _).
\forall dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ kD(rt (σ i)) ∧ nhip \neq dip \rightarrow
dip ∈ kD(rt (σ nhip)) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"
(is "_ \|= (?S, ?U \rightarrow _)")
proof -
{ fix nhop and σ σ' :: "ip ⇒ state"
  assume pre: "\forall dip∈kD(rt (σ i)). nhop dip \neq dip \rightarrow
dip∈kD(rt (σ (nhop dip))) \land nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
and qinc: "\forall j. quality_increases (σ j) (σ' j)"
  have "\forall dip∈kD(rt (σ i)). nhop dip \neq dip \rightarrow
dip∈kD(rt (σ' (nhop dip))) \land nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
proof (intro ballI impI)
  fix dip
  assume "dip∈kD(rt (σ i))"
and "nhop dip ≠ dip"

with pre have "dip∈kD(rt (σ (nhop dip)))"
and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"

by auto

from qinc have qinc_nhop: "quality_increases (σ (nhop dip)) (σ' (nhop dip))" ..
with ⟨dip∈kD(rt (σ (nhop dip)))⟩ have "dip∈kD (rt (σ' (nhop dip)))" ..

moreover have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
proof -
from ⟨dip∈kD(rt (σ (nhop dip)))⟩ qinc_nhop
have "nsqn (rt (σ (nhop dip))) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
with nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip show ?thesis
by simp
qed

ultimately show "dip∈kD(rt (σ' (nhop dip)))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
qed

} note basic = this

{ fix nhop and σ σ' :: "ip ⇒ state"
assume pre: "∀ dip∈kD (rt (σ i)). nhop dip ≠ dip → dip∈kD (rt (σ (nhop dip)))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"

and ndest: "∀ ripc∈dom (dests (σ i)). ripc ∈ kD (rt (σ (sip (σ i))))
∧ the (dests (σ i) ripc) - 1 ≤ nsqn (rt (σ (sip (σ i)))) ripc"

and issip: "∀ ip∈dom (dests (σ i)). nhop ip = sip (σ i)"

and qinc: "∀ j. quality_increases (σ j) (σ' j)"

have "∀ dip∈kD (rt (σ i)). nhop dip ≠ dip → dip ∈ kD (rt (σ' (nhop dip)))
∧ nsqn (invalidate (rt (σ i))) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"

proof (intro ballI impI)

fix dip
assume "dip∈kD (rt (σ i))"
and "nhop dip ≠ dip"

with pre and qinc have "dip∈kD (rt (σ' (nhop dip)))"
and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"

by (auto dest!: basic)

have "nsqn (invalidate (rt (σ i))) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"

proof (cases "dip∈dom (dests (σ i))")

assume "dip∈dom (dests (σ i))"

with ⟨dip∈kD(rt (σ i)))⟩ obtain dsn where "dests (σ i) dip = Some dsn"

by auto

with ⟨dip∈kD(rt (σ i)))⟩ have "nsqn (invalidate (rt (σ i))) (dests (σ i))) dip = dsn - 1"

by (rule nsqn.invalidate_eq)

moreover have "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"

proof -
from ⟨dests (σ i) dip = Some dsn⟩ have "the (dests (σ i) dip) = dsn" by simp

with ndest and ⟨dip∈dom (dests (σ i))⟩ have "dip ∈ kD (rt (σ (sip (σ i))))"

"dsn - 1 ≤ nsqn (rt (σ (sip (σ i)))) dip"

by auto

moreover from issip and ⟨dip∈dom (dests (σ i))⟩ have "nhop dip = sip (σ i)" ..

ultimately have "dip ∈ kD (rt (σ (nhop dip)))"

and "dsn - 1 ≤ nsqn (rt (σ (nhop dip))) dip" by auto

with qinc show "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"

by simp (metis kD(nsqn.quality_increases_trans)

qed

ultimately show ?thesis by simp

next
assume "dip ∉ dom (dests (σ i))"

with ⟨dip∈kD(rt (σ i)))⟩ have "nsqn (invalidate (rt (σ i))) (dests (σ i))) dip = nsqn (rt (σ i)) dip"

by (rule nsqn.invalidate_other)

with nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip show ?thesis by simp

qed
with \( \text{dip} \in \text{kd}(rt (\sigma' (\text{nhop} dip))) \)

\[
\text{show } "\text{dip} \in \text{kd}(rt (\sigma' (\text{nhop} dip)))
\land \text{nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip } \leq \text{nsqn (rt (\sigma' (\text{nhop} dip))) dip}"
.. \]

qed
}

note basic_prerr = this

\{ fix \( \sigma' \cdot \sigma :: "ip \Rightarrow \text{state}"
assume a1: "\forall \text{dip} \in \text{kd}(rt (\sigma i)). \ \text{the (nhop (rt (\sigma i)) dip) } \neq \text{dip}
\rightarrow \text{dip} \in \text{kd}(rt (\sigma (\text{the (nhop (rt (\sigma i)) dip))))
\land \text{nsqn (rt (\sigma i)) dip } \leq \text{nsqn (rt (\sigma (\text{the (nhop (rt (\sigma i)) dip)))) dip}"
and a2: "\forall j. \text{quality_increases (\sigma j) (\sigma' j)}"
have "\( \forall j. \text{quality_increases (\sigma j) (\sigma' j)} \)"
\}

\{ proof

fix dip
assume "\( \text{dip} \in \text{kd}(rt (\sigma i)) \)"
with a1 and a2
have "\( \text{the (nhop (rt (\sigma i)) dip) } \neq \text{dip} \rightarrow \text{dip} \in \text{kd}(rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip))))
\land \text{nsqn (rt (\sigma i)) dip } \leq \text{nsqn (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))) dip}" by - (drule(1) basic, auto)
thus "\( \text{?P dip} \) by (cases "\text{dip} = \text{sip (\sigma i)}") auto
qed
\}

note nhop_update_sip = this

\{ fix \( \sigma' \cdot \sigma :: \text{oip \ \\ sip sosn hops} \)
assume pre: "\( \forall \text{dip} \in \text{kd}(rt (\sigma i)). \ \text{the (nhop (rt (\sigma i)) dip) } \neq \text{dip}
\rightarrow \text{dip} \in \text{kd}(rt (\sigma (\text{the (nhop (rt (\sigma i)) dip))))
\land \text{nsqn (rt (\sigma i)) dip } \leq \text{nsqn (rt (\sigma (\text{the (nhop (rt (\sigma i)) dip)))) dip}"
and qinc: "\( \forall j. \text{quality_increases (\sigma j) (\sigma' j)} \)"
and *: "\text{sip } \neq \text{oip} \rightarrow \text{oip} \in \text{kd}(rt (\sigma \text{sip}))
\land \text{osn } \leq \text{nsqn (rt (\sigma \text{sip})) oip}
\land \text{nsqn (rt (\sigma \text{sip})) oip } = \text{osn}
\rightarrow \text{the (dhaps (rt (\sigma \text{sip})) oip) } \leq \text{hops}
\lor \text{the (flag (rt (\sigma \text{sip})) oip) } = \text{inv})" from pre and qinc

have pre': "\( \forall \text{dip} \in \text{kd}(rt (\sigma i)). \ \text{the (nhop (rt (\sigma i)) dip) } \neq \text{dip}
\rightarrow \text{dip} \in \text{kd}(rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip))))
\land \text{nsqn (rt (\sigma i)) dip } \leq \text{nsqn (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))) dip}" by (rule basic)

have "\( \text{the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc \text{hops}, sip, {})) oip) } \neq \text{oip}
\rightarrow \text{oip} \in \text{kd}(rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) oip)
(osn, kno, val, Suc \text{hops}, sip, {})) oip}))
\land \text{nsqn (update (rt (\sigma i)) oip (osn, kno, val, Suc \text{hops}, sip, {})) oip}
\leq \text{nsqn (rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) oip)
(osn, kno, val, Suc \text{hops}, sip, {})) oip))}) oip})" (is "\( \text{?nhop_not_oip } \rightarrow \text{?oip_in_kd} \land \text{?nsqn_le_nsqn} \)"

proof (rule, split update rt_split_asm)
assume "\text{rt (\sigma i) } = \text{update (rt (\sigma i)) oip (osn, kno, val, Suc \text{hops}, sip, {})}"
and "\text{the (nhop (rt (\sigma i)) oip) } \neq \text{oip}"
with pre' show "\( \text{?oip_in_kd} \land \text{?nsqn_le_nsqn} \)" by auto
next
assume rtnot: "\text{rt (\sigma i) } \neq \text{update (rt (\sigma i)) oip (osn, kno, val, Suc \text{hops}, sip, {})}"
and notoip: "\text{?nhop_not_oip}"
with * qinc have ?oip_in_kd
by (clarsimp elim!: kD_quality_increases)
moreover with * pre qinc rtnot notoip have ?nsqn_le_nsqn
by simp (metis kD\_nsqn\_quality\_increases\_trans)
ultimately show "\(?oip\_in\_kD \land \?nsqn\_le\_nsqn" ..
qed
}

\} note update1 = this

\{ fix \(\sigma \) \(\sigma'\) oip sip osn hops
assume pre: "\(\forall dip: kD (rt (\sigma i)). \) the (nhop (rt (\sigma i)) dip) \(\neq\) dip
\(\rightarrow dip: kD(rt (\sigma (the (nhop (rt (\sigma i)) dip))))\)
\(\wedge nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (the (nhop (rt (\sigma i)) dip)))) dip"\)
and qinc: "\(\forall j. \) quality\_increases (\(\sigma j\)) (\(\sigma' j\))"
and \*: "sip \(\neq\) oip \(\rightarrow\) oip \(\in\) kD(rt (\(\sigma sip\)))
\(\wedge \) osn \(\leq\) nsqn (rt (\(\sigma sip\))) oip
\(\rightarrow\) the (dhops (rt (\(\sigma sip\))) oip) \(\leq\) hops
\(\lor\) the (flag (rt (\(\sigma sip\))) oip) = inv)"
from pre and qinc
have pre': "\(\forall dip: kD (rt (\sigma i)). \) the (nhop (rt (\sigma i)) dip) \(\neq\) dip
\(\rightarrow dip: kD(rt (\sigma' (the (nhop (rt (\sigma i)) dip))))\)
\(\wedge nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (the (nhop (rt (\sigma i)) dip)))) dip"\)
by (rule basic)
have "\(\forall dip: kD(rt (\sigma i)). \) the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip) \(\neq\) dip
\(\rightarrow dip: kD(rt (\sigma' (the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip))))\)
\(\wedge nsqn (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip \leq nsqn (rt (\sigma' (the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))) dip" (is "\(\forall dip: kD(rt (\sigma i)). \) _ \(\rightarrow\) ?dip\_in\_kD dip \(\land\) ?nsqn\_le\_nsqn dip")
proof (intro ballI impI, split update\_rt\_split\_asm)
fix dip
assume "dip: kD(rt (\sigma i))"
and "the (nhop (rt (\sigma i)) dip) \(\neq\) dip"
and "rt (\sigma i) = update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})"
with pre' show "\(?dip\_in\_kD dip \land \?nsqn\_le\_nsqn dip" by simp
next
fix dip
assume "dip: kD(rt (\sigma i))"
and notdip: "the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip) \(\neq\) dip"
and rtnot: "rt (\sigma i) \(\neq\) update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})"
show "\(?dip\_in\_kD dip \land \?nsqn\_le\_nsqn dip" proof (cases "dip = oip")
assume "dip \(\neq\) oip"
with pre' (dip: kD(rt (\sigma i))) notdip
show ?thesis by clarsimp
next
assume "dip = oip"
with rtnot qinc (dip: kD(rt (\sigma i))) notdip *
have "\(?dip\_in\_kD dip" by simp (metis kD\_quality\_increases)
moreover from (dip = oip) rtnot qinc (dip: kD(rt (\sigma i))) notdip *
have "\(?nsqn\_le\_nsqn dip" by simp (metis kD\_nsqn\_quality\_increases\_trans)
ultimately show ?thesis ..
qed
defined

\} note update2 = this

have "opaodv i \models (?S, ?U \rightarrow) onl \Gamma\_AODV (\lambda(\sigma, _). \(\forall dip: kD(rt (\sigma i)). \) the (nhop (rt (\sigma i)) dip) \(\neq\) dip
\(\rightarrow dip: kD(rt (\sigma (the (nhop (rt (\sigma i)) dip))))\)
\(\wedge nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (the (nhop (rt (\sigma i)) dip)))) dip)"
by (inv\_cters inv add: oseq\_step\_invariant\_sterns [OF quality\_increases\_nsqn\_fresh aodv\_wf
oaodv\_trans]
onl\_oinvariant\_sterns [OF aodv\_wf oaddpreRT\_welldefined]

onl_invariant_terms [OF aodv_wf odests_vD_inc_sqn]
onl_invariant_terms [OF aodv_wf oprerr_guard]
onl_invariant_terms [OF aodv_wf rreq_sip]
onl_invariant_terms [OF aodv_wf rrep_sip]
onl_invariant_terms [OF aodv_wf rerr_sip]
other_quality_increases
other_localD

solve: basic basic_prerr
simp add: seqlsimp nsqn_invalidate nhop_update_sip
simp del: One_nat_def)

thus ?thesis unfolding Let_def by auto

qed

Proposition 7.30

lemmas okD_unk_or_atleast_one =
open_seq_invariant [OF kD_unk_or_atleast_one initiali_aodv,
simplified seql_onl_swap]

lemmas ozero_seq_unk_hops_one =
open_seq_invariant [OF zero_seq_unk_hops_one initiali_aodv,
simplified seql_onl_swap]

lemma oreachable_fresh_okD_unk_or_atleast_one:
  fixes dip
  assumes "((σ, p) ∈ oreachable (opaodv i))
     (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
     ∧ msg_zhops m)))
     (other_quality_increases {i})"
  and "dip∈kD(rt (σ i))"
  shows "π3(the (rt (σ i) dip)) = unk ∨ 1 ≤ π2(the (rt (σ i) dip))"
(is "?P dip")
proof -
  have "∃ l∈labels ΓAODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip∈kD (rt (σ i)). ?P dip"
    by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaodv_trans aodv_trans]],
        auto dest!: otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip∈kD(rt (σ i))⟩ show ?thesis by simp
qed

lemma oreachable_fresh_ozero_seq_unk_hops_one:
  fixes dip
  assumes "((σ, p) ∈ oreachable (opaodv i))
     (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
     ∧ msg_zhops m)))
     (other_quality_increases {i})"
  and "dip∈kD(rt (σ i))"
  shows "sqn (rt (σ i)) dip = 0 → sqnf (rt (σ i)) dip = unk
    ∧ the (dhops (rt (σ i)) dip) = 1
    ∧ the (nhop (rt (σ i)) dip) = dip"
(is "?P dip")
proof -
  have "∃ l∈labels ΓAODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip∈kD (rt (σ i)). ?P dip"
    by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]],
        auto dest!: onlD otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip∈kD(rt (σ i))⟩ show ?thesis by simp
qed

lemma seq_nhop_quality_increases':
  shows "opaodv i |= (otherwith ((=)) {i})
     (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),

proof -

have weaken:
"∀ P Q R. P, p  (otherwith quality_increases I (orecmsg Q), other quality_increases I →) P
⇒ p  (otherwith ((σ)) I (orecmsg (λσ. Q σ. m → R σ. m)), other quality_increases I →) P"
by auto

{ fix i a and σ σ' :: "ip ⇒ state"
assume a1: "∀ dip. dip ∈ VD rt (σ i))
∧ dip ∈ VD (rt (σ (the (nhop (rt (σ i)) dip)))))
∧ (the (nhop (rt (σ i)) dip)) ≠ dip
→ rt (σ i) ⊑ dip rt (σ (the (nhop (rt (σ i)) dip))))"
and ow: "∀ dip. dip ∈ VD rt (σ i))
∧ dip ∈ VD (rt (σ' (the (nhop (rt (σ i)) dip))))
∧ (the (nhop (rt (σ i)) dip)) ≠ dip
→ rt (σ i) ⊑ dip rt (σ' (the (nhop (rt (σ i)) dip))))"
proof clarify
fix dip
assume a2: "∀ dip. dip ∈ VD rt (σ i))"
and a3: "∀ dip ∈ VD rt (σ' (the (nhop (rt (σ i)) dip))))"
and a4: "∀ (the (nhop (rt (σ i)) dip)) ≠ dip"
from ow have "∀ j. j ≠ i → σ j = σ' j" by auto
show "∀ (σ i) ⊑ dip rt (σ' (the (nhop (rt (σ i)) dip))))"
proof (cases "∀ (the (nhop (rt (σ i)) dip)) = i")
assume "∀ (the (nhop (rt (σ i)) dip)) ≠ i"
with (dip ∈ VD rt (σ i)) have "∀ dip ∈ VD rt (σ (the (nhop (rt (σ i)) dip))))" by simp
with a1 a2 a4 have "∀ (the (σ i) ⊑ dip rt (σ (the (nhop (rt (σ i)) dip))))" by simp
with (the (nhop (rt (σ i)) dip)) = i have "∀ (σ i) ⊑ dip rt (σ (the (nhop (rt (σ i)) dip))))" by simp
hence False by simp
thus ?thesis ..
next
assume "∀ (the (nhop (rt (σ i)) dip)) ≠ i"
with ∀ j. j ≠ i → σ j = σ' j
have "∀ (σ (the (nhop (rt (σ i)) dip))) = σ' (the (nhop (rt (σ i)) dip)))" by simp
with (the (nhop (rt (σ i)) dip)) = i have "∀ (σ i) ⊑ dip rt (σ (the (nhop (rt (σ i)) dip))))" by simp
with * show ?thesis by simp
qed
qed

{ fix σ σ' a dip sip i
assume a1: "∀ dip. dip ∈ VD rt (σ i))
∧ dip ∈ VD (rt (σ (the (nhop (rt (σ i)) dip))))
∧ (the (nhop (rt (σ i)) dip)) ≠ dip
→ rt (σ i) ⊑ dip rt (σ (the (nhop (rt (σ i)) dip))))"
and ow: "∀ dip. dip ∈ VD rt (σ i))
∧ dip ∈ VD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {}))
∧ (the (nhop (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip) ≠ dip
→ update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})
 resil rt (σ' (the (nhop (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip))))"
proof clarify
fix dip
assume a2: "∀ dip ∈ VD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {}))"
and a3: "∀ dip ∈ VD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {}))"
and a4: the (nhop (update (rt (\sigma i))) sip (0, unk, val, Suc 0, sip, {})) dip \neq dip

show "update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})
    \sqsubseteq_dip rt (\sigma' (the (nhop (update (rt (\sigma i))) sip (0, unk, val, Suc 0, sip, {}))) dip)"

proof (cases "dip = sip")

assume "dip = sip"

with \{ the (nhop (update (rt (\sigma i))) sip (0, unk, val, Suc 0, sip, {})) dip \neq dip \}

have False by simp

thus ?thesis ..

next

assume [simp]: "dip \neq sip"

from a2 have "dip \in\ VD (rt (\sigma i)) \lor dip = sip"

by (rule VD_update_val)

with \{ dip \neq sip \} have "dip \in\ VD (rt (\sigma i))" by simp

moreover from a3 have "dip \in\ VD (rt (\sigma' (the (nhop (rt (\sigma i)) dip))))" by simp

moreover from a4 have "the (nhop (rt (\sigma i)) dip) \neq dip" by simp

ultimately have "rt (\sigma i) \sqsubseteq_dip rt (\sigma' (the (nhop (rt (\sigma i)) dip)))"

using a1 ow by \{- (drule(1) basic, simp)\}

with \{ dip \neq sip \} show ?thesis

by \{- (erule rt_strictly_fresher_update_other, simp)\}

qed

qed

\}
ote update_0_unk = this

\{ fix \sigma a \sigma' nhop

assume pre: "\forall dip. dip \in\ VD (rt (\sigma i)) \land dip \in\ VD (rt (\sigma (nhop dip))) \land nhop dip \neq dip

\longrightarrow rt (\sigma i) \sqsubseteq_dip rt (\sigma (nhop dip))"

and ow: "?S i \sigma \sigma' a"

have "\forall dip. dip \in\ VD (invalidate (rt (\sigma i)) (dests (\sigma i)))

\land dip \in\ VD (rt (\sigma' (nhop dip))) \land nhop dip \neq dip

\longrightarrow rt (\sigma i) \sqsubseteq_dip rt (\sigma' (nhop dip))"

proof clarify

fix dip

assume "dip \in\ VD (invalidate (rt (\sigma i)) (dests (\sigma i)))"

and "dip \in\ VD (rt (\sigma' (nhop dip)))"

and "nhop dip \neq dip"

from this(1) have "dip \in\ VD (rt (\sigma i))"

by (clarsimp dest!: VD_invalidate_vD_not_dests)

moreover from ow have "\forall j. j \neq i \longrightarrow \sigma j = \sigma' j" by auto

ultimately have "rt (\sigma i) \sqsubseteq_dip rt (\sigma (nhop dip))"

using pre \{ dip \in\ VD (rt (\sigma' (nhop dip))); nhop dip \neq dip \}

by metis

with \{ dip \neq sip \} show "rt (\sigma i) \sqsubseteq_dip rt (\sigma' (nhop dip))"

by (metis rt_strictly_fresher_irefl)

qed

\}
ote invalidate = this

\{ fix \sigma a \sigma' dip oip osn sip hops i

assume pre: "\forall dip. dip \in\ VD (rt (\sigma i))

\land dip \in\ VD (rt (\sigma (the (nhop (rt (\sigma i)) dip))))

\land the (nhop (rt (\sigma i)) dip) \neq dip

\longrightarrow rt (\sigma i) \sqsubseteq_dip rt (\sigma (the (nhop (rt (\sigma i)) dip)))"

and ow: "?S i \sigma \sigma' a"

and "Suc 0 \leq osn"

and a6: "sip \neq oip \longrightarrow oip \in kD (rt (\sigma sip))

\land dip \in\ VD (rt (\sigma (the (nhop (rt (\sigma i)) dip))))

\land the (nhop (rt (\sigma i)) dip) \neq dip

\longrightarrow rt (\sigma i) \sqsubseteq_dip rt (\sigma (the (nhop (rt (\sigma i)) dip)))"

and after: "\sigma' i = \sigma i (rt := update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}))"

have "\forall dip. dip \in\ VD (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}))

\land dip \in\ VD (rt (\sigma' (the (nhop (update (rt (\sigma i)) oip

(osn, kno, val, Suc hops, sip, {})) dip))))

\land the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip) \neq dip

\longrightarrow update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})"

and after: "\sigma' i = \sigma i (rt := update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}))"

have "\forall dip. dip \in\ VD (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}))

\land dip \in\ VD (rt (\sigma' (the (nhop (update (rt (\sigma i)) oip

(osn, kno, val, Suc hops, sip, {})) dip))))

\land the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip) \neq dip

\longrightarrow update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})"

and after: "\sigma' i = \sigma i (rt := update (rt (\sigma i)) oip (osn, kno, val,Suc hops, sip, {}))"

have "\forall dip. dip \in\ VD (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}))

\land dip \in\ VD (rt (\sigma' (the (nhop (update (rt (\sigma i)) oip

(osn, kno, val, Suc hops, sip, {})) dip))))

\land the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip) \neq dip

\longrightarrow update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})

and after: "\sigma' i = \sigma i (rt := update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}))"


proof clarify
fix dip
assume a2: "dip ∈ vD(update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {}))"
and a3: "dip ∈ vD(rt (σ' (the (nhop (update (rt (σ i)) oip
(osn, kno, val, Suc hops, sip, {})) dip)))"
and a4: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"
from ow have a5: "∀ j. j ≠ i → σ j = σ' j" by auto
show "update (rt (σ i)) oip (osn, kno, Suc hops, sip, {})
\sqsubseteq dip (σ' (the (nhop (update (rt (σ i)) oip
(osn, kno, val, Suc hops, sip, {})) dip)))"
(is "?rt1 \sqsubseteq dip ?rt2 dip")
proof (cases "?rt1 = rt (σ i)"

assume nochange [simp]:
"update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {}) = rt (σ i)"

from after have "σ' i = σ i" by simp
with a5 have "∀ j. σ j = σ' j" by metis

from a2 have "dip ∈ vD (rt (σ i))" by simp
moreover from a3 have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))"
using nochange and "∀ j. σ j = σ' j" by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip)))"
using pre by simp

hence "rt (σ i) \sqsubseteq dip rt (σ' (the (nhop (rt (σ i)) dip)))"
using "∀ j. σ j = σ' j" by simp
thus "?thesis" by simp

next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip ∈ kD(rt (σ' i))" by auto
show ?thesis
proof (cases "dip = oip"

assume "dip ≠ oip"

with a2 have "dip ∈ vD (rt (σ i))" by auto
moreover with a3 a5 after and "dip ≠ oip"
have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))"
by simp metis
moreover from a4 and "dip ≠ oip" have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip)))"
using pre by simp

with after and a5 and "dip ≠ oip" show ?thesis
by simp (metis rt_strictly_fresher_update_other
rt_strictly_fresher_irefl)

next
assume "dip = oip"

with a4 and change have "sip ≠ oip" by simp
with a6 have "oip ∈ kD(rt (σ sip))"
and "osn ≤ nsqn (rt (σ sip)) oip" by auto

from a3 change (dip = oip) have "oip ∈ vD(rt (σ' sip))" by simp
hence "the (flag (rt (σ' sip)) oip) = val" by simp

from (oip ∈ kD(rt (σ sip))) have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
∧ the (dhops (rt (σ' sip)) oip) ≤ hops)"
proof
assume "oip ∈ vD(rt (σ sip))"

hence "the (flag (rt (σ sip)) oip) = val" by simp
with $a_6 \ (sip \neq oip)$ have 

```plaintext
"nsqn (rt (σ sip)) oip = osn \rightarrow
the (dhops (rt (σ sip)) oip) \leq hops"
```

by simp

show ?thesis

proof (cases "sip = i")

assume "sip ≠ i"

with $a_5$ have "σ sip = σ' sip" by simp

with $osn \leq nsqn (rt (σ sip)) oip$

and $\langle nsqn (rt (σ sip)) oip = osn \rightarrow the (dhops (rt (σ sip)) oip) \leq hops \rangle$

show ?thesis by auto

next

— alternative to using sip_not_ip

assume [simp]: "sip = i"

have "?rt1 = rt (σ i)"

proof (rule update_cases_kD, simp_all)

from $\langle Suc 0 \leq osn \rangle$

show "0 < osn" by simp

next

from $\langle oip \in vD (rt (σ sip)) \rangle$

and $\langle sip = i \rangle$

show "oip \in kD(rt (σ i))"

by simp

next

assume "sqn (rt (σ i)) oip < osn"

also from $\langle osn \leq nsqn (rt (σ sip)) oip \rangle$

have "... \leq nsqn (rt (σ i)) oip" by simp

also have "... \leq sqn (rt (σ i)) oip"

by (rule nsqn_sqn)

finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip".

hence False by simp

thus "(λa. if a = oip then Some (osn, kno, val, Suc hops, i, π_7 (the (rt (σ i) oip)))
else rt (σ i) a) = rt (σ i)" ..

next

assume "sqn (rt (σ i)) oip = osn"

and "Suc hops < the (dhops (rt (σ i)) oip)"

from this(1) and $\langle oip \in vD (rt (σ sip)) \rangle$

have "nsqn (rt (σ i)) oip = osn"

by simp

with $\langle nsqn (rt (σ sip)) oip = osn \rightarrow the (dhops (rt (σ sip)) oip) \leq hops \rangle$

by simp

with $\langle Suc hops < the (dhops (rt (σ i)) oip) \rangle$

have False by simp

thus "(λa. if a = oip then Some (osn, kno, val, Suc hops, i, π_7 (the (rt (σ i) oip)))
else rt (σ i) a) = rt (σ i)" ..

next

assume "the (flag (rt (σ i)) oip) = inv"

with $\langle the (flag (rt (σ sip)) oip) = val \rangle$

have False by simp

thus "(λa. if a = oip then Some (osn, kno, val, Suc hops, i, π_7 (the (rt (σ i) oip)))
else rt (σ i) a) = rt (σ i)" ..

next

from $\langle oip \in kD(rt (σ sip)) \rangle$

show "(λa. if a = oip then Some (the (rt (σ i) oip)) else rt (σ i) a) = rt (σ i)"

by (auto dest!: kD_Some)

qed

with change have False ..

thus ?thesis ..

qed
(auto simp: sqn_def)
with \osn \leq \nsqn (rt (\s' \sip)) \oip have "osn < nsqn (rt (\s' \sip)) \oip"
by simp
thus \thesis ..
qed
thus \thesis
proof
assume osnlt: "osn < nsqn (rt (\s' \sip)) \oip"
from \dip \in \KD(rt (\s' \i)) and \dip = \oip have "\dip \in \KD(?rt1)" by simp
moreover from a3 have "\dip \in \KD(?rt2 \dip)" by simp
moreover have "\nsqn ?rt1 \dip < \nsqn (?rt2 \dip) \dip"
  proof 
  have "\nsqn ?rt1 \oip = \osn"
    by (simp add: \dip = \oip nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "... < \nsqn (rt (\s' \sip)) \oip" using osnlt .
  also have "... = \nsqn (?rt2 \oip) \oip" by (simp add: change)
  finally show \thesis 
    using \dip = \oip by simp
  qed
ultimately show \thesis
  by (rule rt_strictly_fresher_ltI)
next
assume osneq: "osn = nsqn (rt (\s' \sip)) \oip \land (the (dhops (rt (\s' \sip)) \oip) \leq \hops"
have "\oip \in \KD(?rt1)" by simp
moreover from a3 \dip = \oip have "\oip \in \KD(?rt2 \oip)" by simp
moreover have "\nsqn ?rt1 \oip = \nsqn (?rt2 \oip) \oip"
  proof 
  from osneq have "osn = nsqn (rt (\s' \sip)) \oip" ..
  also have "osn = \nsqn ?rt1 \oip"
    by (simp add: \dip = \oip nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "\nsqn (rt (\s' \sip)) \oip = \nsqn (?rt2 \oip) \oip"
    by (simp add: change)
  finally show \thesis 
    by (simp add: change)
  qed
moreover have "\pi5(\the (\?rt2 \oip) \oip) < \pi5(\the (\?rt1 \oip))"
  proof 
  from osneq have "the (dhops (rt (\s' \sip)) \oip) \leq \hops" ..
  moreover from \oip \in \VD (rt (\s' \sip)) have "\oip \in \KD(rt (\s' \sip))" by auto
  ultimately have "\pi5(\the (rt (\s' \sip) \oip)) \leq \hops"
    by (auto simp add: proj5_eq_dhops)
  also from change after have "\hops < \pi5(\the (rt (\s' \i) \oip))"
    by (simp add: proj5_eq_dhops) (metis dhops_update_changed_lessI)
  finally have "\pi5(\the (rt (\s' \sip) \oip)) < \pi5(\the (rt (\s' \i) \oip))" .
  with change after show \thesis by simp
  qed
ultimately have "?rt1 \sqsubseteq \oip ?rt2 \oip"
  by (rule rt_strictly_fresher_eqI)
with \dip = \oip show \thesis by simp
qed
qed

\text{note} \ rreq_rrep_update = \text{this}

have "\opaodv \i \equiv (otherwith ((\_)) \i) (orecmsg (\lambda \s m. \msg_fresh \s m \land \msg_zhops m)),
  other quality Increases \i \rightarrow"
\onl \Gamma_{AODV}
(\lambda (\s, \_). \forall \dip. \dip \in \VD (rt (\s \i)) \cap \VD (rt (\s (the (nhop (rt (\s \i) \dip))))
  \land (the (nhop (rt (\s \i)) \dip) \neq \dip)
\[ \Rightarrow rt\ (\sigma\ i) \sqsubseteq dip\ rt\ (\sigma\ (the\ (nhop\ (rt\ (\sigma\ i))\ dip))) \]

\textbf{proof (inv\_cterms}\ inv\ add: onl\_oinvariant\_sterms [OF aodv\_wf\ rreq\_sip [THEN weaken]]
\text{onl}\_oinvariant\_sterms [OF aodv\_wf\ rrep\_sip [THEN weaken]]
\text{onl}\_oinvariant\_sterms [OF aodv\_wf\ rerr\_sip [THEN weaken]]
\text{onl\_oinvariant\_sterms [OF aodv\_wf\ oosn\_rreq [THEN weaken]]}
\text{onl\_oinvariant\_sterms [OF aodv\_wf\ odsn\_rrep [THEN weaken]]}
\text{onl\_oinvariant\_sterms [OF aodv\_wf\ oaddpreRT\_welldefined]}
\text{solve: basic\ update\_0\_unk\ invalidate\ rreq\_rrep\_update}
\text{simp\ add: seq\_simp}

\text{fix} \ \sigma\ \sigma'\ p\ l
\text{assume or:} \ "(\sigma, p) \in\ oreachable\ (opaodv\ i)\ (?S\ i)\ (other\ quality\_increases\ \{i\})"
\text{and } "other\ quality\_increases\ \{i\} \sigma\ \sigma'"
\text{and ll: } "l \in\ labels\ \Gamma_{AODV}\ p"
\text{and pre: } "\forall dip. dip \in vD (rt\ (\sigma\ i))
\wedge \text{the (nhop (rt\ (\sigma\ i))\ dip))} \neq dip
\Rightarrow rt\ (\sigma\ i) \sqsubseteq dip\ rt\ (\sigma\ (the\ (nhop\ (rt\ (\sigma\ i))\ dip)))"

\text{from this(1-2)}
\text{have or': } "(\sigma', p) \in\ oreachable\ (opaodv\ i)\ (?S\ i)\ (other\ quality\_increases\ \{i\})"
\text{by } (rule\ oreachable\_other')

\text{from or and ll have next\_hop: } "\forall dip. let\ nhip = the\ (nhop\ (rt\ (\sigma\ i))\ dip))
\in dip \in kD(r (\sigma\ i)) \wedge nhip \neq dip
\Rightarrow dip \in kD(r (\sigma\ nhip))
\wedge \text{nsqn (rt\ (\sigma\ i))\ dip} \leq \text{nsqn (rt\ (\sigma\ nhip))\ dip}"
\text{by } (auto\ dest!: onl\_oinvariant\_weakenD [OF seq\_compare\_next\_hop'])

\text{from or and ll have unk\_hops\_one: } "\forall dip. sqn (rt\ (\sigma\ i)) dip = 0
\Rightarrow sqnf (rt\ (\sigma\ i)) dip = unk
\wedge \text{the (dhops (rt\ (\sigma\ i))\ dip))} = 1
\wedge \text{the (nhop (rt\ (\sigma\ i))\ dip))} = dip"
\text{by } (auto\ dest!: onl\_oinvariant\_weakenD [OF ozero\_seq\_unk\_hops\_one [OF oaodv\_trans\ aodv\_trans]])
\text{otherwith\_actionD}
\text{simp: seq\_simp)

\text{from } "other\ quality\_increases\ \{i\} \sigma\ \sigma'\ have\ "\sigma'\ i = \sigma\ i"\ by\ auto
\text{hence } "quality\_increases\ (\sigma\ i)\ (\sigma'\ i)"\ by\ auto
\text{with } "other\ quality\_increases\ \{i\} \sigma\ \sigma'\ have\ "\forall j.\ quality\_increases\ (\sigma\ j)\ (\sigma'\ j)"
\text{by } -(erule\ otherE,\ metis\ singleton\_iff)

\text{show } "\forall dip. dip \in vD (rt\ (\sigma'\ i))
\wedge dip \in vD (rt\ (\sigma'\ (the\ (nhop\ (rt\ (\sigma'\ i))\ dip)))
\wedge the (nhop (rt\ (\sigma'\ i)) dip) \neq dip
\Rightarrow rt\ (\sigma'\ i) \sqsubseteq dip\ rt\ (\sigma'\ (the\ (nhop\ (rt\ (\sigma'\ i))\ dip)))"

\text{proof clarify}
\text{fix dip}
\text{assume } "dip \in vD\ (rt\ (\sigma'\ i))"
\text{and } "dip \in vD\ (rt\ (\sigma'\\ (the\ (nhop\ (rt\ (\sigma'\ i))\ dip))))"
\text{and } "the\ (nhop\ (rt\ (\sigma'\ i))\ dip) \neq dip"
\text{from this(1) and } \sigma'\ i = \sigma\ i\ have "dip \in vD\ (rt\ (\sigma\ i))"
\text{and } "dip \in kD\ (rt\ (\sigma\ i))"
\text{by auto}

\text{from } the (nhop (rt (\sigma' i)) dip) \neq dip\ and\ \sigma' i = \sigma i\ have "the\ (nhop\ (rt (\sigma\ i))\ dip) \neq dip"\ (is "?nhip \neq _")\ by\ simp
\text{with } dip \in kD\ (rt\ (\sigma\ i))\ and\ next\_hop
\text{have } "dip \in kD\ (rt\ (\sigma\ ?nhip))"
\text{and } nsqns: "nsqn\ (rt\ (\sigma\ i))\ dip \leq nsqn\ (rt\ (\sigma\ ?nhip))\ dip"
\text{by } (auto\ simp: Let\_def)

\text{have } "0 < sqn\ (rt\ (\sigma\ i))\ dip"
\text{proof (rule\ neq\_conv [THEN\ iffD1,\ OF\ notI])}
\text{assume } "sqn\ (rt\ (\sigma\ i))\ dip = 0"

405
with \( \langle \text{ dip} \in kD(rt (\sigma i)) \rangle \) and unk_hops_one

have \( ?nhip = \text{ dip} \) by simp
with \( ?nhip \neq \text{ dip} \) show False ..

qed
also have \( \ldots = nsqn (rt (\sigma i)) \text{ dip} \)
  by (rule vD_nsqn_sqn [OF \( \langle \text{ dip} \in vD(rt (\sigma i)) \rangle \), THEN sym])
also have \( \ldots \leq nsqn (rt (\sigma ?nhip)) \text{ dip} \)
  by (rule nsqns)
also have \( \ldots \leq sqn (rt (\sigma ?nhip)) \text{ dip} \)
  by (rule nsqn_sqn)
finally have \( 0 < sqn (rt (\sigma ?nhip)) \text{ dip} \).

have \( rt (\sigma i) \sqsubseteq dip \text{ rt (\sigma' ?nhip)} \)
proof (cases \( \langle \text{ dip} \in kD(rt (\sigma' ?nhip)) \rangle \))
  assume \( \langle \text{ dip} \in kD(rt (\sigma' ?nhip)) \rangle \)
  with \( \langle \text{ dip} \in vD(rt (\sigma i)) \rangle \) and \( ?nhip \neq \text{ dip} \)
  have \( rt (\sigma i) \sqsubseteq dip \text{ rt (\sigma' ?nhip)} \) by auto
moreover from \( \forall j. \text{ quality_increases} (\sigma j) (\sigma' j) \)
  have \( \text{ quality_increases} (\sigma ?nhip) (\sigma' ?nhip) \) ..
ultimately show ?thesis
  using \( \langle \text{ dip} \in kD(rt (\sigma ?nhip)) \rangle \)
  by (rule strictly_fresher_quality_increases_right)
next
assume \( \langle \text{ dip} \in vD(rt (\sigma ?nhip)) \rangle \)
with \( \langle \text{ dip} \in kD(rt (\sigma ?nhip)) \rangle \) have \( \langle \text{ dip} \in vD(rt (\sigma ?nhip)) \rangle \) ..
hence \( \langle \text{ flag (rt (\sigma ?nhip)) dip = inv} \rangle \)
  by auto
have \( nsqn (rt (\sigma i)) \text{ dip} \leq nsqn (rt (\sigma ?nhip)) \text{ dip} \)
  by (rule nsqns)
also from \( \langle \text{ dip} \in iD(rt (\sigma ?nhip)) \rangle \)
  have \( \ldots = sqn (rt (\sigma ?nhip)) \text{ dip} - 1 \) ..
also have \( \ldots < sqn (rt (\sigma' ?nhip)) \text{ dip} \)
  proof -
  from \( \forall j. \text{ quality_increases} (\sigma j) (\sigma' j) \)
  have \( \text{ quality_increases} (\sigma ?nhip) (\sigma' ?nhip) \) ..
  hence \( \langle \forall ip. \text{ sqn (rt (\sigma ?nhip)) ip} \leq \text{ sqn (rt (\sigma' ?nhip)) ip} \rangle \) by auto
  hence \( \langle \text{ sqn (rt (\sigma ?nhip)) dip} \leq \text{ sqn (rt (\sigma' ?nhip)) dip} \rangle \) ..
  with \( 0 < \text{ sqn (rt (\sigma ?nhip)) dip} \) show ?thesis by auto
qed
also have \( \ldots = nsqn (rt (\sigma ?nhip)) \text{ dip} \)
proof (rule vD_nsqn_sqn [THEN sym])
  from \( \langle \text{ dip} \in vD(rt (\sigma' ?nhip)) \rangle \) and \( \langle \sigma' i = \sigma i \) show \( \langle \text{ dip} \in vD(rt (\sigma' ?nhip)) \rangle \) by simp
qed
finally have \( nsqn (rt (\sigma i)) \text{ dip} < nsqn (rt (\sigma' ?nhip)) \text{ dip} \) .

moreover from \( \langle \text{ dip} \in vD(rt (\sigma' (the (nhop (rt (\sigma' i)) dip))))) \) and \( \langle \sigma' i = \sigma i \) have \( \langle \text{ dip} \in kD(rt (\sigma' ?nhip)) \rangle \) by auto
ultimately show \( rt (\sigma i) \sqsubseteq dip \text{ rt (\sigma' ?nhip)} \)
  using \( \langle \text{ dip} \in kD(rt (\sigma i)) \rangle \) by - (rule rt_strictly_fresher_ltI)
  with \( \langle \sigma' i = \sigma i \) show \( rt (\sigma' i) \sqsubseteq dip \text{ rt (\sigma' (the (nhop (rt (\sigma' i)) dip)))} \)
    by simp
  qed
  thus ?thesis unfolding Let_def .
  qed

lemma seq_compare_next_hop:
  fixes w
  shows "\text{ opaodv i \|= (otherwith (\(\{i\}\)) \{orecvmsg msg_fresh\)}, other quality_increases \(\{i\} \rightarrow\) global (\(\lambda \sigma. \forall dip. \text{ let nhip = the (nhop (rt (\sigma i)) dip) \in dip \in kD(rt (\sigma i)) \wedge nhip \neq dip \rightarrow\) \) \) \)"
\[ \text{dip} \in kD(\text{rt} (\sigma nhip)) \]
\[ \land \quad \text{nsqn}(\text{rt} (\sigma i)) \text{ dip} \leq \text{nsqn}(\text{rt} (\sigma nhip)) \text{ dip} \]

by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onlD)

**Lemma seq_nhop_quality_increases:**
shows "\( o\text{aodv} i \mid (=) \{\text{otherwith} ((=)) \{i\}, \text{orecvmsg} (\lambda \sigma m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m))\),
other\ quality\_increases \{i\} \rightarrow\)
global (\( \lambda \sigma. \forall \text{dip}. \text{let nhip = the (nhop (rt (\sigma i)) dip) in dip} \in vD(\text{rt} (\sigma i)) \cap vD(\text{rt} (\sigma nhip)) \land nhip \neq dip\)
\rightarrow (\text{rt} (\sigma i)) \sqsubseteq \text{dip} (\text{rt} (\sigma nhip)))"

by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onlD)

**4.10 Routing graphs and loop freedom**

theory D_Loop_Freedom
imports D_Aodv_Predicates D_Fresher
begin

Define the central theorem that relates an invariant over network states to the absence of loops in the associate routing graph.

definition
\[ \text{rt\_graph} :: (ip \Rightarrow \text{state}) \Rightarrow \text{ip} \Rightarrow \text{ip rel} \]
where
\[ "\text{rt\_graph} \sigma = (\lambda \text{dip.}
\{(\text{ip}, \text{ip'}) \mid \text{ip ip'} dsn dsk hops pre.
\text{ip} \neq \text{dip} \land \text{rt} (\sigma \text{ip}) \text{ dip} = \text{Some} (dsn, dsk, val, hops, ip', pre))\)"

Given the state of a network \( \sigma \), a routing graph for a given destination \( \text{dip} \) abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

**Lemma rt_graphE [elim]:**
fixes \( \text{n dip ip ip'} \)
assumes "\( (\text{ip, ip'}) \in \text{rt\_graph} \sigma \text{ dip} \)"
shows "\( \text{ip} \neq \text{dip} \land (\exists r. \text{rt} (\sigma \text{ip}) = r
\land (\exists dsn dsk hops pre. r \text{ dip} = \text{Some} (dsn, dsk, val, hops, ip', pre))\)"

using asms unfolding rt_graph_def by auto

**Lemma rt_graph_vD [dest]:**
\[ "\lambda \text{ip ip'} \sigma \text{ dip}. (\text{ip, ip'}) \in \text{rt\_graph} \sigma \text{ dip} \Rightarrow \text{dip} \in vD(\text{rt} (\sigma \text{ip}))" \]
unfolding rt_graph_def vD_def by auto

**Lemma rt_graph_vD_trans [dest]:**
\[ "\lambda \text{ip ip'} \sigma \text{ dip}. (\text{ip, ip'}) \in (\text{rt\_graph} \sigma \text{ dip})^+ \Rightarrow \text{dip} \in vD(\text{rt} (\sigma \text{ip}))" \]
by (erule converse_tranclE) auto

**Lemma rt_graph_not_dip [dest]:**
\[ "\lambda \text{ip ip'} \sigma \text{ dip}. (\text{ip, ip'}) \in \text{rt\_graph} \sigma \text{ dip} \Rightarrow \text{ip} \neq \text{dip}" \]
unfolding rt_graph_def by auto

**Lemma rt_graph_not_dip_trans [dest]:**
\[ "\lambda \text{ip ip'} \sigma \text{ dip}. (\text{ip, ip'}) \in (\text{rt\_graph} \sigma \text{ dip})^+ \Rightarrow \text{ip} \neq \text{dip}" \]
by (erule converse_tranclE) auto

NB: the property below cannot be lifted to the transitive closure

**Lemma rt_graph_nhip_is_nhop [dest]:**
\[ "\lambda \text{ip ip'} \sigma \text{ dip}. (\text{ip, ip'}) \in \text{rt\_graph} \sigma \text{ dip} \Rightarrow \text{ip'} = \text{the (nhop (rt (\sigma \text{ip})) dip)}" \]
unfolding rt_graph_def by auto

**Theorem inv_to_loop_freedom:**
assumes "\( \forall i \text{ dip}. \text{let nhip = the (nhop (rt (\sigma i)) dip) in dip} \in vD(\text{rt} (\sigma i)) \cap vD(\text{rt} (\sigma nhip)) \land nhip \neq dip\)"
\[
\rightarrow (\text{rt } (\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma \text{nhip}))
\]

shows "\(\forall \text{dip}. \text{irrefl } ((\text{rt_graph } \sigma \text{ dip})^+)\)"

using assms proof (intro allI)

fix \(\sigma:: \text{ip} \Rightarrow \text{state}\) and dip

assume inv: "\(\forall \text{ip dip}. \text{let nhip = the (nhop (rt (\sigma \text{ip})) dip) in dip} \in \text{vD(r}\text{t (\sigma \text{ip})} \cap \text{vD(r}\text{t (\sigma \text{nhip})}) \land \text{nhip} \neq \text{dip} \rightarrow \text{rt (\sigma \text{ip})} \sqsubseteq_{\text{dip}} \text{rt (\sigma \text{nhip})}\)"

{ fix ip ip'
  assume "(ip, ip') \in (\text{rt_graph } \sigma \text{ dip})^+"
  and "\text{dip} \in \text{vD(r}\text{t (\sigma \text{ip}')})"
  and "\text{ip'} \neq \text{dip}"
  hence "\text{rt (\sigma \text{ip})} \sqsubseteq_{\text{dip}} \text{rt (\sigma \text{ip}')}\"
  proof induction
    fix nhip
    assume "(ip, nhip) \in \text{rt_graph } \sigma \text{ dip}"
    and "\text{dip} \in \text{vD(r}\text{t (\sigma \text{nhip})})"
    and "nhip \neq \text{dip}"
    from "(ip, nhip) \in \text{rt_graph } \sigma \text{ dip}"
    have 1: "\text{dip} \in \text{vD(r}\text{t (\sigma \text{nhip})})" and 2: "nhip \neq \text{dip}"
    by auto
    from 1 2 have "\text{rt (\sigma \text{ip})} \sqsubseteq_{\text{dip}} \text{rt (\sigma \text{nhip})}" by (rule fresher)
    thus False by simp
  qed

next
  fix nhip nhip'
  assume "(ip, nhip) \in (\text{rt_graph } \sigma \text{ dip})^+"
  and "(nhip, nhip') \in \text{rt_graph } \sigma \text{ dip}"
  and IH: "\[ \text{dip} \in \text{vD(r}\text{t (\sigma \text{nhip})}); \text{nhip} \neq \text{dip} \implies \text{rt (\sigma \text{ip})} \sqsubseteq_{\text{dip}} \text{rt (\sigma \text{nhip})}\]"
  and "\text{dip} \in \text{vD(r}\text{t (\sigma \text{nhip}')})"
  and "nhip' \neq \text{dip}"
  from "(nhip, nhip') \in \text{rt_graph } \sigma \text{ dip}" have 1: "\text{dip} \in \text{vD(r}\text{t (\sigma \text{nhip})})" and 2: "nhip' \neq \text{dip}"
  by auto
  from 1 2 have "\text{rt (\sigma \text{nhip})} \sqsubseteq_{\text{dip}} \text{rt (\sigma \text{nhip}')}" by (rule IH)
  thus False by simp
  qed
}

note fresher = this

show "\text{irrefl } ((\text{rt_graph } \sigma \text{ dip})^+)"

unfolding irrefl_def proof (intro allI notI)

fix ip

assume "(ip, ip) \in (\text{rt_graph } \sigma \text{ dip})^+"

moreover then have "\text{dip} \in \text{vD(r}\text{t (\sigma \text{ip})})" and "ip \neq dip"

by auto

ultimately have "\text{rt (\sigma \text{ip})} \sqsubseteq_{\text{dip}} \text{rt (\sigma \text{ip})}" by (rule fresher)

thus False by simp

qed

qed
4.11 Lift and transfer invariants to show loop freedom

theory D_Aodv_Loop_Freedom
importsAWN.Qmsg_Lifting D_Global_Invariants D_Loop_Freedom
begin

4.11.1 Lift to parallel processes with queues

lemma par_step_no_change_on_send_or_receive:
fixes σ s a σ' s'
assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG)
and a ≠ τ"
shows "σ' i = σ i"
using assms by (rule qmsg_no_change_on_send_or_receive)

lemma par_nhop_quality_increases:
shows "opaodv i ⟨⟨ i qmsg ⟩= A (orecvmsg (λσ. msg_fresh σ m ∧ msg_zhops m)),
other quality_increases {i} →
global (λσ. ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
→ (rt (σ i) ⊏ dip (rt (σ nhip))))"
proof (rule lift_into_qmsg_statelessassm) simp_all

next
fix σ σ' a
assume "otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
other quality_increases {i} →
globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"

proof (rule ostep_invariant_weakenE [OF olocal_quality_increases], simp_all)
fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
assume "onll Γ_AODV (λ((σ, _), _, (σ', _)). quality_increases (σ j) (σ' j)) t"
thus "quality_increases (fst (fst t) i) (fst (snd (snd t)) i)"
by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)

next
fix σ σ' a
assume "otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
other quality_increases {i} →
globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"

proof (rule ostep_invariant_weakenE [OF olocal_quality_increases], simp_all)
fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
assume "onll Γ_AODV (λ((σ, _), _, (σ', _)). quality_increases (σ j) (σ' j)) t"
thus "quality_increases (fst (fst t) i) (fst (snd (snd t)) i)"
by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)

qed

lemma par_rreq_rrep_sn_quality_increases:
shows "opaodv i ⟨⟨ i qmsg ⟩= A (λσ _. orecvmsg (λ_. orecvmsg (λ_. rreq_rrep_sn) σ), other (λ_. True) {i} →
globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
proof -
have "opaodv i ⟨⟨ i qmsg ⟩= A (λσ _. orecevmsg (λ_. rreq_rrep_sn) σ), other (λ_. True) {i} →
globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
by (rule ostep_invariant_weakenE [OF olocal_quality_increases], simp_all)

hence "opaodv i ⟨⟨ i qmsg ⟩= A (λσ _. orecevmsg (λ_. rreq_rrep_sn) σ), other (λ_. True) {i} →
globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

lemma par_rreq_rrep_nsqn_fresh_any_step:
shows "opaodv i ⟨⟨ i qmsg ⟩= A (λσ _. orecevmsg (λ_. rreq_rrep_sn) σ), other (λ_. True) {i} →
globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
proof -
have "opaodv i ⟨⟨ i qmsg ⟩= A (λσ _. orecevmsg (λ_. rreq_rrep_sn) σ), other (λ_. True) {i} →
globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"

globala \((\lambda (\sigma, a, \sigma'). \text{anycast} \ (\text{msg\_fresh} \ \sigma) \ a)\)

proof (rule ostep\_invariant\_weakenE [OF rreq\_rrep\_nsqn\_fresh\_any\_step\_invariant])

fix t

assume "onll \Gamma_{AODV} \ (\lambda ((\sigma, _), a, _). \text{anycast} \ (\text{msg\_fresh} \ \sigma) \ a) \ t"

thus "globala \((\lambda (\sigma, a, \sigma'). \text{anycast} \ (\text{msg\_fresh} \ \sigma) \ a) \ t\)"

by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)

qed auto

hence "opaodv i \((\langle i \ qmsg \ =_A (\lambda _\ _ \ . \text{orecvmsg} \ (\lambda _\ _ \ . \text{rreq\_rrep\_sn}) \ \sigma, \text{other} \ (\lambda _\ _ \ . \ True) \ \{i\} \to) \text{globala} \ (\lambda (\_, a, \_). \text{anycast} \ (\text{msg\_fresh} \ \sigma) \ a)\)"

by (rule lift\_step\_into\_qmsg\_statelessassm) simp all

thus ?thesis by rule auto

qed

lemma par\_anycast\_msg\_zhops:

shows "opaodv i \((\langle i \ qmsg \ =_A (\lambda _\ _ \ . \text{orecvmsg} \ (\lambda _\ _ \ . \text{rreq\_rrep\_sn}) \ \sigma, \text{other} \ (\lambda _\ _ \ . \ True) \ \{i\} \to) \text{globala} \ (\lambda (_, a, _). \text{anycast} \ \text{msg\_zhops} \ a)\)"

proof -

from anycast\_msg\_zhops \ initiali\_aodv \ oaodv\_trans \ aodv\_trans

have "opaodv i \|=A \ (act \ TT, \text{other} \ (\lambda _\ _ \ _. \ True) \ {i} \to) \text{seqll} \ i \ \text{onll} \ Γ_{AODV} \ (\lambda (_, a, _). \text{anycast} \ \text{msg\_zhops} \ a))"

by (rule open\_seq\_step\_invariant)

hence "opaodv i \|=A \ (\lambda _\ _ \ . \text{orecvmsg} \ (\lambda _\ _ \ . \text{rreq\_rrep\_sn}) \ \sigma, \text{other} \ (\lambda _\ _ \ . \ True) \ \{i\} \to) \text{globala} \ (\lambda (_, a, _). \text{anycast} \ \text{msg\_zhops} \ a)"

proof (rule ostep\_invariant\_weakenE)

fix t :: "(((nat \Rightarrow \text{state}) \times \text{state}, \text{msg}, \text{pseqp}, \text{pseqp \ label} \ \text{seqp}), \text{msg \ seq\_action}) \ \text{transition}"

assume "seqll i \ \text{onll} \ Γ_{AODV} \ (\lambda (_, a, _). \text{anycast} \ \text{msg\_zhops} \ a)) \ t"

thus "globala \ (\lambda (_, a, _). \text{anycast} \ \text{msg\_zhops} \ a)"

by (cases t) (clarsimp dest!: seqllD onllD, metis aodv_ex_label)

qed simp_all

hence "opaodv i \((\langle i \ qmsg \ =_A (\lambda _\ _ \ . \text{orecvmsg} \ (\lambda _\ _ \ . \text{rreq\_rrep\_sn}) \ \sigma, \text{other} \ (\lambda _\ _ \ . \ True) \ \{i\} \to) \text{globala} \ (\lambda (_, a, _). \text{anycast} \ \text{msg\_zhops} \ a)\)"

by (rule lift\_step\_into\_qmsg\_statelessassm) simp all

thus ?thesis by rule auto

qed

4.11.2 Lift to nodes

lemma node\_step\_no\_change\_on\_send\_or\_receive:

assumes "((\sigma, \text{NodeS \ i \ P \ R}), \ \text{a}, \ (\sigma', \text{NodeS \ i' \ P' \ R'})) \ \in \ \text{onode\_sos} \ \text{par\_arp\_sos \ i \ (oseqp\_sos \ Γ_{AODV}) \ (seqp\_sos \ Γ_{QM \ SG})}"

and "\text{a \neq \tau}"

shows "\sigma' \ i = \sigma \ i"

using \assms

by (cases \text{a}) (auto elim!: par\_step\_no\_change\_on\_send\_or\_receive)

lemma node\_nhop\_quality\_increases:

shows "\langle i : \text{opaodv} \ i \ \langle i \ qmsg \ : \ R \rangle_o \ = \ (\text{otherwith} \ (\sigma)) \ \{i\} \to \text{globala} \ (\lambda \sigma. \ \text{other_quality\_increases} \ (i) \to) \text{globala} \ (\lambda \sigma. \ \forall \text{dip}. \ \text{let nhp} = \text{the \ (nhop} \ (\text{rt} \ (\sigma \ i)) \ \text{dip}) \ \text{in dip} \in \ \text{vd} \ \text{rt} \ (\sigma \ i) \ \cap \ \text{vd} \ (\text{rt} \ (\sigma \ \text{nhp}) \ \cap \ \text{nhp} \neq \text{dip} \to \text{rt} \ (\sigma \ i) \ \text{\wedge dip} \ (\text{rt} \ (\sigma \ \text{nhp})))\)"

by (rule node\_lift [OF \text{par\_nhop\_quality\_increases}) simp\_all]

lemma node\_quality\_increases:

shows "\langle i : \text{opaodv} \ i \ \langle i \ qmsg \ : \ R \rangle_o \ = \ \langle i \ : \text{opaodv} \ i \ \langle i \ qmsg \ : \ R \rangle_o \ = \ (\text{otherwith} \ (\sigma)) \ \{i\} \to \text{globala} \ (\lambda \sigma. \ \text{other_quality\_increases} \ (i) \to) \text{globala} \ (\lambda (\sigma, a, _'). \text{quality\_increases} \ (\sigma \ i) \ (\sigma') \ i)"

by (rule node\_lift\_step\_statelessassm [OF \text{par\_rreq\_rrep\_sn\_quality\_increases]) simp\_all]

lemma node\_rreq\_rrep\_nsqn\_fresh\_any\_step:

shows "\langle i : \text{opaodv} \ i \ \langle i \ qmsg \ : \ R \rangle_o \ =_A \ (\lambda \sigma. \ \text{other\_quality\_increases} \ (\sigma) \ (\sigma') \ i) \to "

by (rule node\_lift\_step\_statelessassm [OF \text{par\_rreq\_rrep\_sn\_quality\_increases]) simp\_all]
lemma node_anycast_msg_zhops:
shows "(i : opaodvi (i qmsg : R) o = A (\(\sigma\). oarrivemsg (\(\lambda\)_.. rreq_rrep_sn) \(\sigma\), other (\(\lambda\). True) {i} \(\rightarrow\))
globala (\(\lambda\)(\(\sigma\), a, \(\sigma\')). castmsg msg_zhops a)"
by (rule node_lift_anycast_statelessasm [OF par_rreq_rrep_nsqn_fresh_any_step])

lemma node_silent_change_only:
shows "(i : opaodv i (i qmsg : R) o = A (\(\sigma\). oarrivemsg (\(\lambda\)_.. rreq_rrep_sn) \(\sigma\), other (\(\lambda\). True) {i} \(\rightarrow\))
globala (\(\lambda\)(\(\sigma\), a, \(\sigma\')). a \(\neq\) \(\tau\) \(\rightarrow\) \(\sigma\)' i = \(\sigma\) i)"
proof (rule ostep_invariantI, simp (no_asm), rule impI)
  have "fix i R
      assume or: "(\(\sigma\), \(\zeta\)) \(\in\) oreachable ((i : opaodv i (i qmsg : R) o)
          (\(\lambda\)_.. oarrivemsg (\(\lambda\)_.. True) \(\sigma\))
          (other (\(\lambda\). True) {i})"
      and tr: "((\(\sigma\), \(\zeta\)), a, (\(\sigma\)', \(\zeta\)') \(\in\) trans ((i : opaodv i (i qmsg : R) o)"
      and "a \(\neq\) \(\tau\)_n"
from or obtain p R where "\(\zeta\) = NodeS i P R"
by (drule node_net_state, metis)
with tr have "((\(\sigma\), NodeS i P R), a, (\(\sigma\)', \(\zeta\)'))
\(\in\) onode_sos (oparp_sos i (\(\tau\)_ step) (\(\tau\)_ step))" 
  by simp
thus "\(\sigma\)' i = \(\sigma\) i" using (a \(\neq\) \(\tau\)_n)
by (cases rule: onode_sos_cases)
(auto elim: qmsg_no_change_on_send_or_receive)
qed

4.11.3 Lift to partial networks

lemma arrive_rreq_rrep_nsqn_fresh_inc_sn [simp]:
assumes "oarrivemsg (\(\lambda\)_.. m. msg_fresh \(\sigma\) m \(\land\) P \(\sigma\) m) \(\sigma\) m"
shows "oarrivemsg (\(\lambda\)_.. rreq_rrep_sn) \(\sigma\) m"
using assms (cases m) auto

lemma opnet_nhop_quality_increases:
shows "opnet (\(\lambda\)_.. opaodv i (i qmsg : R) p =
  (otherwith ((\(\sigma\)))) (net_tree_ips p)
    (oarrivemsg (\(\lambda\)_.. msg_fresh \(\sigma\) m \(\land\) msg_zhops m)),
    global (\(\lambda\)_.. \(\forall\)i\(\in\)net_tree_ips p. \(\forall\)dip.
      let nhp = the (nhop (rt (\(\sigma\))) dip)
      in dip \(\in\) \(\forall\)\(d\) (rt (\(\sigma\))) \(\cap\) \(\forall\)\(d\) (rt (\(\sigma\)) nhp) \(\cap\) nhp \(\neq\) dip
      \(\rightarrow\) (rt (\(\sigma\))) \(\subseteq\) dip (rt (\(\sigma\)) nhp))"
proof (rule pnet_lift [OF node_nhop_quality_increases])
  fix i R
  have "(i : opaodv i (i qmsg : R) o = A (\(\lambda\)_.. oarrivemsg (\(\lambda\)_.. rreq_rrep_sn) \(\sigma\),
    other (\(\lambda\). True) {i} \(\rightarrow\))
globala (\(\lambda\)(\(\sigma\)_.. a, \(\sigma\')). castmsg (\(\lambda\)_.. msg_fresh \(\sigma\) m \(\land\) msg_zhops m) a)"
proof (rule ostep_invariantI, simp (no_asm))
  fix \(\sigma\)_s a \(\sigma\)' s'
  assume or: "(\(\sigma\), \(\sigma\)) \(\in\) oreachable ((i : opaodv i (i qmsg : R) o)
      (\(\lambda\)_.. oarrivemsg (\(\lambda\)_.. rreq_rrep_sn) \(\sigma\))
      (other (\(\lambda\). True) {i})"
and tr: "((\(\sigma\), \(\sigma\), \(\sigma\)', \(\sigma\)') \(\in\) trans ((i : opaodv i (i qmsg : R) o)"
and am: "oarrivemsg (\(\lambda\)_.. rreq_rrep_sn) \(\sigma\) a"
from or tr am have "castmsg (msg_fresh \(\sigma\)) a"
  by (auto dest!: ostep_invariantD [OF node_rreq_rrep_nsqn_fresh_step])
moreover from or tr am have "castmsg (msg_zhops) a"
  by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
ultimately show "castmsg (\(\lambda\)_.. msg_fresh \(\sigma\) m \(\land\) msg_zhops m) a"
  by (case_tac a) auto
thus "\(\{i : \text{opaodv} i \mid (i \text{ qmsg} : R)\} \models_A \)
\(\langle \lambda \sigma . \text{oarrivemsg} (\lambda m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) \sigma, \)
other quality\_increases \{i\} \rightarrow \text{globala} (\lambda (\sigma, a, _). \)
castmsg (\lambda m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) a)"
by rule auto

next
fix \(i R\)
show "\(\{i : \text{opaodv} i \mid (i \text{ qmsg} : R)\} \models_A \)
\(\langle \lambda \sigma . \text{oarrivemsg} (\lambda m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) \sigma, \)
other quality\_increases \{i\} \rightarrow \text{globala} (\lambda (\sigma, a, _). \)
a \neq \tau \land (\forall d. a \neq i : \text{deliver}(d)) \rightarrow \sigma i = \sigma' i)"
by (rule ostep\_invariant\_weakenE [OF node\_silent\_change\_only]) auto

qed simp_all

4.11.4 Lift to closed networks

lemma onet\_nhop\_quality\_increases:
shows "\(\text{oclosed} (\text{opnet} (\lambda i. \text{opaodv} i \langle \langle i \text{ qmsg} \rangle \rangle p) \mid (\lambda \_ \_ . \text{True}, \text{other quality\_increases} (\text{net\_tree\_ips} p) \rightarrow) \)
\(\text{global} (\lambda \sigma . \forall i \in \text{net\_tree\_ips} p. \forall \text{dip}. \)
let nhip = the (nhop (\text{rt} (\sigma i)) dip) \in \text{dip} \in \text{vD} (\text{rt} (\sigma i)) \cap \text{vD} (\text{rt} (\sigma nhip)) \land nhip \neq \text{dip} \rightarrow (\text{rt} (\sigma i)) \sqsubseteq \text{dip} (\text{rt} (\sigma nhip)))\)
(is "\(\_ \models (\_, ?U \rightarrow) ?\text{inv}\))
proof (rule inclosed\_closed)
from onet\_nhop\_quality\_increases
show "\(\text{opnet} (\lambda i. \text{opaodv} i \langle \langle i \text{ qmsg} \rangle \rangle p)\)
\(\models (\lambda \_ \_ . \text{True}, \text{other quality\_increases} (\text{net\_tree\_ips} p) \rightarrow) \)
\(\text{global} (\lambda \sigma . \forall i \in \text{net\_tree\_ips} p. \forall \text{dip}. \)
\text{let nhip} = \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip}) \in \text{dip} \in \text{vD} (\text{rt} (\sigma i)) \cap \text{vD} (\text{rt} (\sigma nhip)) \land nhip \neq \text{dip} \rightarrow (\text{rt} (\sigma i)) \sqsubseteq \text{dip} (\text{rt} (\sigma nhip)))\)"
proof (rule oinvariant\_weakenE)
fix \(\sigma \sigma' :: \text{ip} \Rightarrow \text{state}\) and \(a :: \text{msg node\_action}\)
assume "\text{otherwith} ((\Rightarrow)) (\text{net\_tree\_ips} p) \text{inoclosed} \sigma \sigma' a"
thus "\text{otherwith} ((\Rightarrow)) (\text{net\_tree\_ips} p) \text{oarrivemsg} (\lambda m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) \sigma \sigma' a"
proof (rule ootherwith\_EI)
fix \(\sigma :: \text{ip} \Rightarrow \text{state}\) and \(a :: \text{msg node\_action}\)
assume "\text{inoclosed} \sigma a"
thus "\text{oarrivemsg} (\lambda m. \text{msg\_fresh} \sigma m \land \text{msg\_zhops} m) \sigma a"
proof (cases a)
fix \(\text{ii ni ms}\)
assume "\text{a = ii\_ni\_arrive(ms)}"
moreover with :\text{inoclosed} \sigma a obtain d dip where "\text{ms} = \text{newpkt}(d, dip)"
by (cases ms) auto
ultimately show \(?\text{thesis}\) by simp
qed simp\_all
qed

4.11.5 Transfer into the standard model

interpretation aodv\_openproc: openproc paodv opaodv id
rewrites "aodv\_openproc\_init\_missing = init\_missing"
proof -
show "openproc paodv opaodv id"
proof unfold\_locales
fix \(i :: \text{ip}\)
have "{(σ, ζ). (σ i, ζ) ∈ σ_{AODV i} ∧ (∀ j. j ≠ i → σ j ∈ fst ' σ_{AODV j})} ⊆ σ_{AODV' i}"
proof (rule equalityD1)
  show "∀ p. {(σ, ζ). (σ i, ζ) ∈ {(f i, p)} ∧ (∀ j. j ≠ i → σ j ∈ fst ' {(f j, p)})} = {(f, p)}"
    by (rule set_eqI) auto
qed

thus "{ (σ, ζ) | σ ζ s. s ∈ init (paodv i)
      ∧ (σ i, ζ) = id s
      ∧ (∀ j. j ≠ i −→ σ j ∈ (fst o id) ' init (paodv j)) } ⊆ init (opaodv i)"
by simp

interpretation aodv_openproc_par_qmsg: openproc_parq paodv opaodv id qmsg
rewrites "aodv_openproc_par_qmsg.netglobal = netglobal"
and "aodv_openproc_par_qmsg.initmissing = initmissing"
proof -
  show "openproc_parq paodv opaodv id qmsg"
    by (unfold_locales) simp
then interpret opq: openproc_parq paodv opaodv id qmsg .

have im: "∀ i. (SOME x. x ∈ (fst o id) ' init (paodv i)) = aodv_init i"
proof (unfold_locales)
  have simp: "∀ i. (SOME x. x ∈ (fst o id) ' init (paodv i)) = aodv_init i"
    unfolding aodv_openproc_par_qmsg_def by simp
  hence "∀ i. openproc.initmissing paodv id i = initmissing i"
    unfolding opq.initmissing_def opq.someinit_def initmissing_def
    by (auto split: option.split)
  thus "openproc.initmissing paodv id = initmissing" ..
qed

interpretation aodv_openproc_par_qmsg: openproc_parq paodv opaodv id qmsg
rewrites "aodv_openproc_par_qmsg.netglobal = netglobal"
and "aodv_openproc_par_qmsg.initmissing = initmissing"
proof -
  show "openproc_parq paodv opaodv id qmsg"
    by (unfold_locales) simp
then interpret opq: openproc_parq paodv opaodv id qmsg .

have im: "∀ i. openproc.initmissing (λi. paodv i (⟨⟨ qmsg ⟩> (λ(p, q). (fst (id p), snd (id p), q)))) σ = initmissing σ"
unfolding opq.initmissing_def opq.someinit_def initmissing_def
unfolding aodv_openproc_par_qmsg_def by (clarsimp cong: option.case_cong)
thus "openproc.initmissing (λi. paodv i (⟨⟨ qmsg ⟩> (λ(p, q). (fst (id p), snd (id p), q)))) = initmissing σ"
by (rule ext)
have P im: "∀ P σ. openproc.netglobal (λi. paodv i (⟨⟨ qmsg ⟩> (λ(p, q). (fst (id p), snd (id p), q)))) P σ = netglobal P σ"
unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
unfolding aodv_openproc_par_qmsg_def by (clarsimp cong: option.case_cong)
  simp del: One_nat_def
  simp add: fst_initmissing_netgmap_default_aodv_init_netlift [symmetric, unfolded initmissing_def]
thus "openproc.netglobal (λi. paodv i (⟨⟨ qmsg ⟩> (λ(p, q). (fst (id p), snd (id p), q)))) = netglobal" 
    by auto
lemma net_nhop_quality_increases:
assumes "wf_net_tree n"
shows "closed (pnet (λi. paodv i ⟨⟨ qmsg ⟩⟩ n) ⩵ netglobal
(λσ. ∀ i dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) ⊆ vD (rt (σ nhip)) ∧ nhip ≠ dip
→ (rt (σ i)) ⪯ dip (rt (σ nhip)))"
(is "_ ⩵ netglobal (λσ. ∀ i. ?inv σ i)"")
proof -
from (wf_net_tree n)
  have proto: "closed (pnet (λi. paodv i ⟨⟨ qmsg ⟩⟩ n) ⩵ netglobal (λσ. ∀ i ∈ net_tree_ips n. ∀ dip.
  let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) ⊆ vD (rt (σ nhip)) ∧ nhip ≠ dip
→ (rt (σ i)) ⪯ dip (rt (σ nhip)))"
  by (rule aodv_openproc_par_qmsg.close_opnet [OF _ onet_nhop_quality_increases])
show ?thesis
unfolding invariant_def opnet_sos.opnet_tau1
proof (rule, simp only: aodv_openproc_par_qmsg.netglobalsimp
  fst_initmissing_netgmap_pair_fst, rule allI)
  fix σ i
  assume sr: "σ ∈ reachable (closed (pnet (λi. paodv i ⟨⟨ qmsg ⟩⟩ n)) TT"
  hence "∀ i ∈ net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"
    by - (drule invariantD [OF proto],
      simp only: aodv_openproc_par_qmsg.netglobalsimp
      fst_initmissing_netgmap_pair_fst)
  thus "?inv (fst (initmissing (netgmap fst σ))) i"
proof (cases "i ∈ net_tree_ips n")
  assume "i ∉ net_tree_ips n"
  from sr have "σ ∈ reachable (pnet (λi. paodv i ⟨⟨ qmsg ⟩⟩ n) TT" ..
    hence "net_ips σ = net_tree_ips n" ..
    with (i ∉ net_tree_ips n) have "i ∉ net_ips σ" by simp
    hence "(fst (initmissing (netgmap fst σ))) i = aodv_init i"
      by simp
    thus ?thesis by simp
  qed metis
qed

4.11.6 Loop freedom of AODV

theorem aodv_loop_freedom:
assumes "wf_net_tree n"
shows "closed (pnet (λi. paodv i ⟨⟨ qmsg ⟩⟩ n) ⩵ netglobal (λσ. ∀ dip. irrefl ((rt_graph σ dip) +))"
using assms by (rule aodv_openproc_par_qmsg.netglobal_weakenE
  [OF net_nhop_quality_increases inv_to_loop_freedom])
end
Chapter 5

Variants A–D: All proposed modifications

This model combines the changes proposed in each of the individual variant models.

5.1 Predicates and functions used in the AODV model

theory E_Aodv_Data
imports E_All_ABCD
begin

5.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

definition inc :: "sqn ⇒ sqn"
where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ inc x"
unfolding inc_def by simp

lemma inc_minus_suc_0 [simp]:
"inc x - Suc 0 = x"
unfolding inc_def by simp

lemma inc_never_one' [simp, intro]: "inc x ≠ Suc 0"
unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
by simp

5.1.2 Modelling Routes

A route is a t-tuple, \((dsn, dsk, flag, hops, nhip)\) where \(dsn\) is the ‘destination sequence number’, \(dsk\) is the ‘destination-sequence-number status’, \(flag\) is the route status, \(hops\) is the number of hops to the destination, and \(nhip\) is the next hop toward the destination.

type_synonym r = "sqn × k × f × nat × ip"

definition proj2 :: "r ⇒ sqn" ("\(\pi_2\)"
where "\(\pi_2 ≡ λ(dsn, _, _, _, _). dsn\"

definition proj3 :: "r ⇒ k" ("\(\pi_3\)"
where "\(\pi_3 ≡ λ(_, dsk, _, _, _). dsk\"

definition proj4 :: "r ⇒ f" ("\(\pi_4\)"
where "\(\pi_4 ≡ λ(_, _, flag, _, _). flag\"

definition proj5 :: "r ⇒ nat" ("\(\pi_5\)"
where "\(\pi_5 ≡ λ(_, _, _, hops, _). hops\"

definition proj6 :: "r ⇒ ip" ("\(\pi_6\)"

415
where \( \pi_6 \equiv \lambda(\_, \_, \_, \_, \_, nhip). \text{nhip} \)

**Lemma: projs [simp]:**

\[
\begin{align*}
\pi_2(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) & = \text{dsn} \\
\pi_3(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) & = \text{dsk} \\
\pi_4(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) & = \text{flag} \\
\pi_5(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) & = \text{hops} \\
\pi_6(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) & = \text{nhip}
\end{align*}
\]

by (clarsimp simp: proj2_def proj3_def proj4_def proj5_def proj6_def)+

**Lemma: proj3_pred [intro]:**

\[
\left[ \begin{array}{l} P \text{kno}; P \text{ unk} \end{array} \right] \Rightarrow P (\pi_3 x)
\]

by (rule k.induct)

**Lemma: proj4_pred [intro]:**

\[
\left[ \begin{array}{l} P \text{val}; P \text{ inv} \end{array} \right] \Rightarrow P (\pi_4 x)
\]

by (rule f.induct)

**Lemma: proj6_pair_snd [simp]:**

fixes \( \text{dsn}' \ r \)

shows \( \pi_6 (\text{dsn}', \text{snd} (r)) = \pi_6 (r) \)

by (cases r) simp

5.1.3 Routing Tables

Routing tables map ip addresses to route entries.

type synonym \( \text{rt} = \text{ip} \rightarrow r \)

syntax

"\_\Sigma_route" :: "rt \Rightarrow \text{ip} \rightarrow r" ("\sigma_{\text{route}}'(\_, \_, '\_')")

translations

"\sigma_{\text{route}}(rt, dip)" => "rt dip"

definition sqn :: "rt \Rightarrow \text{ip} \Rightarrow \text{sqn}"

where "sqn rt dip \equiv \text{case } \sigma_{\text{route}}(rt, dip) \text{ of } \text{Some } r \Rightarrow \pi_2(r) \mid \text{None } \Rightarrow 0"

definition sqnf :: "rt \Rightarrow \text{ip} \Rightarrow \text{k}"

where "sqnf rt dip \equiv \text{case } \sigma_{\text{route}}(rt, dip) \text{ of } \text{Some } r \Rightarrow \pi_3(r) \mid \text{None } \Rightarrow \text{unk}"

abbreviation flag :: "rt \Rightarrow \text{ip} \Rightarrow f"

where "flag rt dip \equiv \text{map_option } \pi_4 (\sigma_{\text{route}}(rt, dip))"

abbreviation dhops :: "rt \Rightarrow \text{ip} \Rightarrow \text{nat}"

where "dhops rt dip \equiv \text{map_option } \pi_5 (\sigma_{\text{route}}(rt, dip))"

abbreviation nhop :: "rt \Rightarrow \text{ip} \Rightarrow \text{ip}"

where "nhop rt dip \equiv \text{map_option } \pi_6 (\sigma_{\text{route}}(rt, dip))"

definition vD :: "rt \Rightarrow \text{ip set}"

where "vD rt \equiv \{\text{dip. flag rt dip } = \text{Some val}\}"

definition iD :: "rt \Rightarrow \text{ip set}"

where "iD rt \equiv \{\text{dip. flag rt dip } = \text{Some inv}\}"

definition kD :: "rt \Rightarrow \text{ip set}"

where "kD rt \equiv \{\text{dip. rt dip } \neq \text{None}\}"

lemma kD_is_vD_and_iD: "kD rt = vD rt \cup iD rt"

unfolding kD_def vD_def iD_def by auto

lemma vD_iD_gives_kD [simp]:

\[
\begin{align*}
\forall \text{rt. } \text{ip } \in \text{vD rt} & \Rightarrow \text{ip } \in \text{kD rt} \\
\forall \text{rt. } \text{ip } \in \text{iD rt} & \Rightarrow \text{ip } \in \text{kD rt}
\end{align*}
\]

unfolding kD_is_vD_and_iD by simp_all

416
lemma kD_Some [dest]
  fixes dip rt
assumes "dip \in kD rt"
  shows "\exists dsn dsk flag hops nhip. 
\sigma_{route}(rt, dip) = Some (dsn, dsk, flag, hops, nhip)"
  using assms unfolding kD_def by simp

lemma kD_None [dest]
  fixes dip rt
  assumes "dip /\in kD rt"
  shows "\sigma_{route}(rt, dip) = None"
  using assms unfolding kD_def
by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]
  fixes dip rt
assumes "dip \in vD rt"
  shows "\exists dsn dsk hops nhip. 
\sigma_{route}(rt, dip) = Some (dsn, dsk, val, hops, nhip)"
  using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
  unfolding vD_def by simp

lemma iD_Some [dest]
  fixes dip rt
assumes "dip \in iD rt"
  shows "\exists dsn dsk hops nhip. 
\sigma_{route}(rt, dip) = Some (dsn, dsk, inv, hops, nhip)"
  using assms unfolding iD_def by simp

lemma val_is_vD [elim]
  fixes ip rt
assumes "ip \in kD(rt)"
and "the (flag rt ip) = val"
  shows "ip \in vD(rt)"
  using assms unfolding vD_def by auto

lemma inv_is_iD [elim]
  fixes ip rt
assumes "ip \in kD(rt)"
and "the (flag rt ip) = inv"
  shows "ip \in iD(rt)"
  using assms unfolding iD_def by auto

lemma iD_flag_is_inv [elim, simp]
  fixes ip rt
assumes "ip \in iD(rt)"
  shows "the (flag rt ip) = inv"
  proof -
  from ⟨ip \in iD(rt)⟩ have "ip \in kD(rt)" by auto
  with assms show ?thesis unfolding iD_def by auto
  qed

lemma kD_but_not_vD_is_iD [elim]
  fixes ip rt
assumes "ip \in kD(rt)"
and "ip /\in vD(rt)"
  shows "ip \in iD(rt)"
  proof -
  from ⟨ip \in kD(rt)⟩ obtain dsn dsk f hops nhop
  where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop)"
  by (metis kD_Some)
from \( \langle \text{ip} \notin vD(rt) \rangle \) have "\( f \neq \text{val} \)"
proof (rule contrapos_nn)
  assume "\( f = \text{val} \)"
  with \( \text{rtip} \) have "the (flag rt ip) = \text{val}" by simp
  with \( \text{ip} \in kD(rt) \) show "\( \text{ip} \in vD(rt) \)" ..
qed

with \( \text{rtip} \) have "the (flag rt ip) = \text{inv}" by simp
with \( \text{ip} \in kD(rt) \) show "\( \text{ip} \in iD(rt) \)" ..
qed

lemma vD_or_iD [elim]:
  fixes \( \text{ip} \ rt \)
  assumes "\( \text{ip} \in kD(rt) \)"
  and "\( \text{ip} \in vD(rt) \implies \text{P rt ip} \)"
  and "\( \text{ip} \in iD(rt) \implies \text{P rt ip} \)"
  shows "\( \text{P rt ip} \)"
proof -
  from \( \langle \text{ip} \in kD(rt) \rangle \) have "\( \text{ip} \in vD(rt) \cup iD(rt) \)"
  by (simp add: kD_is_vD_and_iD)
  thus \( \text{?thesis} \) by (auto elim: assms(2-3))
qed

5.1.4 Updating Routing Tables
Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

Updating route entries

lemma in_kD_case [simp]:
  fixes \( \text{dip} \ rt \)
  assumes "\( \text{dip} \in kD(rt) \)"
  shows "\( \text{sqn rt dip} = \pi_2(\text{the (rt dip)}) \)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj5_eq_dhops: "\( \forall \text{dip rt}. \text{dip} \in kD(rt) = \Rightarrow \pi_5(\text{the (rt dip)}) = \text{the (dhops rt dip)} \)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "\( \forall \text{dip rt}. \text{dip} \in kD(rt) = \Rightarrow \pi_4(\text{the (rt dip)}) = \text{the (flag rt dip)} \)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "\( \forall \text{dip rt}. \text{dip} \in kD(rt) = \Rightarrow \pi_2(\text{the (rt dip)}) = sqn rt dip \)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "\( \forall \text{ip rt}. \text{ip} \in kD(rt) = \Rightarrow \text{sqnf rt ip} = \pi_3(\text{the (rt ip)}) \)"
unfolding sqnf_def by auto

lemma vD_flag_val [simp]:
  "\( \forall \text{dip rt}. \text{dip} \in vD (rt) = \Rightarrow \text{the (flag rt dip)} = \text{val} \)"
unfolding vD_def by clarsimp

lemma kD_update [simp]:
  "\( \forall \text{rt nip v. kD (rt\langle nip \mapsto v \rangle) = insert nip (kD rt)} \)"
unfolding kD_def by auto

lemma kD_empty [simp]: "kD Map.empty = {}"
unfolding kD_def by simp

lemma ip_equal_or_known [elim):
  fixes \( \text{rt} \ \text{ip} \ \text{ip}' \)
  assumes "\( \text{ip} = \text{ip}' \lor \text{ip} \in kD(rt) \)"
  and "\( \text{ip} = \text{ip}' = \Rightarrow \text{P rt ip ip}' \)"
  and "\( \text{[if ip \neq ip'; ip \in kD(rt)] = \Rightarrow \text{P rt ip ip}' \)"
  shows "\( \text{P rt ip ip}' \)"
  using assms by auto

418
shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = es (the (rt dip))"
using assms [THEN kD_Some] by auto

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip /∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = es (the (rt dip))"
using assms [THEN kD_None] by auto

lemma rt_Some_sqn [dest]:
  fixes rt ip dsn dsk flag hops nhip
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip)"
  shows "sqn rt ip = dsn"
unfolding sqn_def using assms by simp

definition update_arg_wf :: "r ⇒ bool"
where "update_arg_wf r ≡ π₄(r) = val ∧
  (π₂(r) = 0) = (π₃(r) = unk) ∧
  (π₃(r) = unk −→ π₅(r) = 1)"

lemma update_arg_wf_gives_cases:
"∀r. update_arg_wf r =⇒ (π₂(r) = 0) = (π₃(r) = unk)"
unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples [simp]:
"∀nhip. update_arg_wf (0, unk, val, Suc 0, nhip)"
"∀dsn hops nhip. update_arg_wf (Suc n, kno, val, hops, nhip)"
unfolding update_arg_wf_def by auto

lemma update_arg_wf_tuples' [elim]:
"∀n hops nhip. Suc 0 ≤ n =⇒ update_arg_wf (n, kno, val, hops, nhip)"
unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: "∀nhip. P (0, unk, val, Suc 0, nhip)"
  and c2: "∀dsn hops nhip. dsn > 0 =⇒ P (dsn, kno, val, hops, nhip)"
  shows "P r"
proof -
  obtain dsn dsk flag hops nhip
  where *: "r = (dsn, dsk, flag, hops, nhip)" by (cases r)
  with update_arg_wf r have wf1: "flag = val"
  and wf2: "dsn = 0" = (dsk = unk)"
  and wf3: "dsk = unk =⇒ (hops = 1)"
  unfolding update_arg_wf_def by auto
  have "P (dsn, dsk, flag, hops, nhip)"
  proof (cases dsk)
    assume "dsk = unk"
    moreover with wf2 wf3 have "dsn = 0" and "hops = Suc 0" by auto
    ultimately show ?thesis using (flag = val) by simp (rule c1)
  next
    assume "dsk = kno"
    moreover with wf2 have "dsn > 0" by simp
    ultimately show ?thesis using (flag = val) by simp (rule c2)
  qed
  with * show "P r" by simp

419
definition update :: "rt ⇒ ip ⇒ r ⇒ rt"

where
"update rt ip r ≡
  case σ_route(rt, ip) of
    None ⇒ rt (ip ↦→ r)
  | Some s ⇒
    if π_2(s) < π_2(r) then rt (ip ↦→ r)
    else if π_2(s) = π_2(r) ∧ (π_5(s) > π_5(r) ∨ π_4(s) = inv)
      then rt (ip ↦→ r)
    else if π_3(r) = unk
      then rt (ip ↦→ (π_3(s), snd (r)))
    else rt (ip ↦→ s)"

lemma update_simps [simp]:
  fixes r s nrt nr' ns rt ip
  defines "s ≡ the σ_route(rt, ip)"
  and "nr' ≡ (π_2(s), π_3(r), π_4(r), π_5(r), π_6(r))"
  shows
  "[ ip /∈ kD(rt) ] =⇒ update rt ip r = rt (ip ↦→ r)"
  "[ ip ∈ kD(rt); sqn rt ip < π_2(r) ] =⇒ update rt ip r = rt (ip ↦→ r)"
  "[ ip ∈ kD(rt); sqn rt ip = π_2(r);
    the (dhops rt ip) > π_5(r) ] =⇒ update rt ip r = rt (ip ↦→ r)"
  "[ ip ∈ kD(rt); π_3(r) = unk; (π_2(r) = 0) = (π_3(r) = unk) ] =⇒ update rt ip r = rt (ip ↦→ nr')"
  "[ ip ∈ kD(rt); sqn rt ip ≥ π_2(r); π_3(r) = kno;
    sqn rt ip = π_2(r) =⇒ the (dhops rt ip) ≤ π_5(r) ∧ the (flag rt ip) = val ]
    =⇒ update rt ip r = rt (ip ↦→ s)"

proof -
  assume "ip /∈ kD(rt)"
  hence "σ_route(rt, ip) = None" ..
  thus "update rt ip r = rt (ip ↦→ r)"
    unfolding update_def by simp

next
  assume "ip ∈ kD(rt)"
    and "sqn rt ip < π_2(r)"
  from this(1) obtain dsn dsk fl hops nhip
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
    by (metis kD_Some)
  with ⟨sqn rt ip < π_2(r)⟩ show "update rt ip r = rt (ip ↦→ r)"
    unfolding update_def s_def by auto

next
  assume "ip ∈ kD(rt)"
    and "sqn rt ip = π_2(r)"
    and "the (dhops rt ip) > π_5(r)"
  from this(1) obtain dsn dsk fl hops nhip
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
    by (metis kD_Some)
  with ⟨sqn rt ip = π_2(r)⟩ and ⟨the (dhops rt ip) > π_5(r)⟩
    show "update rt ip r = rt (ip ↦→ r)"
    unfolding update_def s_def by auto

next
  assume "ip ∈ kD(rt)"
    and "sqn rt ip = π_2(r)"
    and "flag rt ip = Some inv"
  from this(1) obtain dsn dsk fl hops nhip
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
    by (metis kD_Some)
  with ⟨sqn rt ip = π_2(r)⟩ and ⟨flag rt ip = Some inv⟩
    show "update rt ip r = rt (ip ↦→ r)"
    unfolding update_def s_def by auto

next
assume "ip ∈ kD(rt)"
and "π3(r) = unk"
and "(π2(r) = 0) = (π3(r) = unk)"

from this(1) obtain dsn dsk fl hops nhip
where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
by (metis kD_Some)

with (π2(r) = 0) = (π3(r) = unk) and (π3(r) = unk)
show "update rt ip r = rt (ip ↦ nr)"

unfolding update_def nr'_def s_def
by (cases r) simp

next
assume "ip ∈ kD(rt)"
and otherassms: "sqn rt ip ≥ π2(r)"
"π3(r) = kno"
"sqn rt ip = π2(r) =⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"

from this(1) obtain dsn dsk fl hops nhip
where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
by (metis kD_Some)

with otherassms show "update rt ip r = rt (ip ↦ s)"

unfolding update_def s_def by auto

qed

lemma update_cases [elim]:
assumes "("π3(r) = 0) = (π3(r) = unk)"
and c1: "[ip /∈ kD(rt)] =⇒ P (rt (ip ↦ r))"
and c2: "[ip ∈ kD(rt); sqn rt ip < π2(r)]
       =⇒ P (rt (ip ↦ r ))"
and c3: "[ip ∈ kD(rt); sqn rt ip = π2(r); the (dhops rt ip) > π5(r)]
       =⇒ P (rt (ip ↦ r ))"
and c4: "[ip ∈ kD(rt); sqn rt ip = π2(r); the (flag rt ip) = inv]
       =⇒ P (rt (ip ↦ r ))"
and c5: "[ip ∈ kD(rt); π3(r) = unk]
       =⇒ P (rt (ip ↦ (π3(the σroute(rt, ip)), π3(r),
π4(r), π5(r), π6(r))))"
and c6: "[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
       sqn rt ip = π2(r) =⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val]
       =⇒ P (rt (ip ↦ the σroute(rt, ip)))"
shows "(P (update rt ip r))"

proof (cases "ip ∈ kD(rt)")
assume "ip /∈ kD(rt)"
with c1 show ?thesis
  by simp

next
assume "ip ∈ kD(rt)"

moreover then obtain dsn dsk fl hops nhip
where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip)"
by (metis kD_Some)

moreover obtain dsn' dsk' fl' hops' nhip'
where req: "r = (dsn', dsk', fl', hops', nhip')"
by (cases r) metis

ultimately show ?thesis
using (π2(r) = 0) = (π3(r) = unk)
c2 [OF (ip∈kD(rt))]
c3 [OF (ip∈kD(rt))]
c4 [OF (ip∈kD(rt))]
c5 [OF (ip∈kD(rt))]
c6 [OF (ip∈kD(rt))]

unfolding update_def sqn_def by auto

qed

lemma update_cases_kD:
assumes "("π3(r) = 0) = (π3(r) = unk)"
and "ip ∈ kD(rt)"
3.1.2.2 and c2: "sqn rt ip < π2(r) \Rightarrow P (rt (ip \mapsto r))"
4.1.2.2 and c3: 

\[
\begin{align*}
[sqn rt ip = π2(r); & \text{ the (dhops rt ip) > π5(r)}] \\
\Rightarrow P (rt (ip \mapsto r))
\end{align*}
\]

4.1.2.2 and c4: 

\[
\begin{align*}
[sqn rt ip = π2(r); & \text{ the (flag rt ip) = inv}] \\
\Rightarrow P (rt (ip \mapsto r))
\end{align*}
\]

4.1.2.2 and c5: 

\[
\begin{align*}
π3(r) = unk = & \Rightarrow P (rt (ip \mapsto (π2(\text{the σroute(rt, ip)}, π3(r), π4(r), π5(r), π6(r)))))
\end{align*}
\]

4.1.2.2 and c6: 

\[
\begin{align*}
[sqn rt ip ≥ π2(r); \text{ the (dhops rt ip) ≤ π5(r) \& the (flag rt ip) = val}] \\
\Rightarrow P (rt (ip \mapsto \text{the σroute(rt, ip)}))
\end{align*}
\]

shows "(P (update rt ip r))" using assms(1) proof (rule update_cases)

assume "sqn rt ip < π2(r)"
thus "P (rt(ip \mapsto r))" by (rule c2)

next

assume "sqn rt ip = π2(r)"
and "the (dhops rt ip) > π5(r)"
thus "P (rt(ip \mapsto r))" by (rule c3)

next

assume "sqn rt ip = π2(r)"
and "the (flag rt ip) = inv"
thus "P (rt(ip \mapsto r))" by (rule c4)

next

assume "π3(r) = unk"
thus "P (rt (ip \mapsto (π2(\text{the σroute(rt, ip)}, π3(r), π4(r), π5(r), π6(r)))))
by (rule c5)

next

assume "sqn rt ip ≥ π2(r)"
and "π3(r) = kno"
and "sqn rt ip = π2(r) \Rightarrow \text{ the (dhops rt ip) ≤ π5(r) \& the (flag rt ip) = val}"
thus "P (rt (ip \mapsto (\text{the rt ip})))" by (rule c6)

qed (simp add: {ip ∈ kD(rt)})

lemma in_kD_after_update [simp]:
 fixes rt nip dsn dsk flag hops nhip pre
 shows "kD (update rt nip (dsn, dsk, flag, hops, nhip)) = insert nip (kD rt)"
 unfolding update_def
 by (cases "rt nip") auto

lemma nhop_of_update [simp]:
 fixes rt dip dsn dsk flag hops nhip
 assumes "rt \neq update rt dip (dsn, dsk, flag, hops, nhip)"
 shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip)) dip) = nhip"
 proof -
 from assms have update_neq: "\\forall v. rt dip = Some v \Rightarrow \\
 update rt dip (dsn, dsk, flag, hops, nhip) \\
 \neq rt(dip \mapsto (\text{the rt dip}))"
 by auto
 show ?thesis
 proof (cases "rt dip = None")
 assume "rt dip = None"
 thus "?thesis" unfolding update_def by clarsimp
 next
 assume "rt dip \neq None"
 then obtain v where "rt dip = Some v" by (metis not_None_eq)
 with update_neq [OF this] show ?thesis
 unfolding update_def by auto
 qed

qed
lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn (λx. if x = rip then Some v else rt x) ip
         = (if ip = rip then π2(v) else sqn rt ip)"
  unfolding sqn_def by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip)) dip"
  proof (rule update_cases)
    show "(π2 (dsn, dsk, val, hops, nhip) = 0) = (π3 (dsn, dsk, val, hops, nhip) = unk)"
      by simp (rule assms)
  qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip
  assumes "1 ≤ hops" 
  shows "sqn rt ip ≤ sqn (update rt ip' (dsn, dsk, flag, hops, nhip)) ip"
  using assms unfolding update_def sqn_def 
  by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops ip rip nhip
  assumes ex: "∀ ip∈kD rt. the (dhops rt ip) ≥ 1"
  and ip: "(ip = rip ∧ Suc 0 ≤ hops) ∨ (ip ≠ rip ∧ ip∈kD rt)"
  shows "Suc 0 ≤ the (dhops (update rt rip (dsn, dsk, flag, hops, nhip)) ip)"
  using ip proof
    assume "ip = rip ∧ Suc 0 ≤ hops" thus ?thesis
    unfolding update_def using ex
    by (cases "rip ∈ kD rt") (drule(1) bspec, auto)
  next
    assume "ip ≠ rip ∧ ip∈kD rt" thus ?thesis
    using ex unfolding update_def
    by (cases "rip∈kD rt") auto
  qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip)) ip = rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  "∀ rt ip dsn dsk flag hops nhip. sqn (rt(ip ↦→ v)) ip = π2(v)"
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip)"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip)) dip) = hops"
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma nhop_update_unk_val [simp]:
  "∀rt dip ip dsn hops.
  the (nhop (update rt dip (dsn, unk, val, hops, ip)) dip) = ip"
unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
assumes "update rt dip (dsn, dsk, flg, hops, sip) ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip)) dip) = sip"
using assms unfolding update_def
by (clarsimp split: option.splits if_split_asm) auto

lemma update_rt_split_asm:
  "∀rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip)) = 
  (¬(rt = update rt ip (dsn, dsk, flag, hops, sip) ∧ ¬P rt 
  ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip) 
  ∧ ¬P (update rt ip (dsn, dsk, flag, hops, sip))))"
by auto

lemma sqn_update [simp]: "∀rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip) 
  ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip)) dip = dsn"
unfolding update_def by (clarsimp split: option.split if_split_asm) auto

lemma sqnf_update [simp]: "∀rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip) 
  ⇒ sqnf (update rt dip (dsn, dsk, flg, hops, sip)) dip = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma update_kno_dsn_greater_zero:
  "∀rt ip dsn hops. 1 ≤ dsn =⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip)) dip)"
unfolding update_def
by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∀rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip) 
  ⇒ π₃ (the (update rt dip (dsn, dsk, flg, hops, sip)) dip) = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma nhop_update_changed_kno_val [simp]: "∀rt ip dsn dsk hops nhip.
  rt ≠ update rt ip (dsn, kno, val, hops, nhip) 
  ⇒ the (nhop (update rt ip (dsn, kno, val, hops, nhip)) ip) = nhip"
unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma flag_update [simp]: "∀rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip) 
  ⇒ the (flag (update rt dip (dsn, kno, flg, hops, sip)) dip) = flg"
unfolding update_def
by (clarsimp split: option.split_asm option.split if_split_asm) auto

lemma the_flag_Some [dest!]:
  fixes ip rt
assumes "the (flag rt ip) = x" 
  and "ip ∈ kD rt"


424
shows "flag rt ip = Some x"
using assms by auto

lemma kD_update_unchanged [dest]:
  fixes rt dip dsn dsk flag hops nhip
  assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip)"
  shows "dip ∈ kD(rt)"
proof -
  have "dip ∈ kD(update rt dip (dsn, dsk, flag, hops, nhip))" by simp
  with assms show ?thesis by simp
qed

lemma nhop_update [simp]: "∀ rt dip dsn dsk flg hops sip. rt ≠ update rt dip (dsn, dsk, flg, hops, sip)
  ⇒ the (nhop (update rt dip (dsn, dsk, flg, hops, sip)) dip) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits if_split_asm) auto

lemma sqn_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:
  "∀ dip rt dip' dsn dsk hops nhip. dip ∈ vD(update rt dip' (dsn, dsk, val, hops, nhip))
  ⇒ (dip ∈ vD(rt) ∨ dip = dip')"
unfolding update_def vD_def by (clarsimp split: option.split_asm if_split_asm)

Invalidating route entries

definition invalidate :: "rt ⇒ (ip ⇒ sqn) ⇒ rt"
where "invalidate rt dests ≡ λip. case (rt ip, dests ip) of
  (None, _) ⇒ None
  | (Some s, None) ⇒ Some s
  | (Some (_, dsk, _, hops, nhip), Some rsn) ⇒ Some (rsn, dsk, inv, hops, nhip)"

lemma proj3_invalidate [simp]:
  "∀ dip. π₃ ((invalidate rt dests) dip) = π₃ (the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj5_invalidate [simp]:
  "∀ dip. π₅ ((invalidate rt dests) dip) = π₅ (the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
  "∀ dip. π₆ ((invalidate rt dests) dip) = π₆ (the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

5.1.5 Route Requests

lemma invalidate_kD_inv [simp]:
  "∀ rt dests. kD (invalidate rt dests) = kD rt"
unfolding invalidate_def kD_def
by (simp split: option.split)
lemma invalidate_sqn:
  fixes rt dip dests
  assumes "∀ rsn. dests dip = Some rsn ⟹ sqn rt dip ≤ rsn"
  shows "sqn rt dip ≤ sqn (invalidate rt dests) dip"
proof (cases "dip ∉ kD(rt)"
  assume "¬ dip ∉ kD(rt)"
  hence "dip ∈ kD(rt)" by simp
  then obtain dsn dsk flag hops nhip where "rt dip = Some (dsn, dsk, flag, hops, nhip)"
    by (metis kD_Some)
  with assms show "sqn rt dip ≤ sqn (invalidate rt dests) dip"
    by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
qed simp

lemma sqn_invalidate_in_dests [simp]:
  fixes dests ipa rsn rt
  assumes "dests ipa = Some rsn"
    and "ipa ∈ kD(rt)"
  shows "sqn (invalidate rt dests) ipa = rsn"
unfolding invalidate_def sqn_def
using assms(1) assms(2) [THEN kD_Some]
by clarsimp

lemma dhops_invalidate [simp]:
  "∀ dip. dhops rt dip = dhops (invalidate rt dests) dip"
unfolding invalidate_def
by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
  "∀ dip. sqnf rt dip = sqnf (invalidate rt dests) dip"
unfolding sqnf_def invalidate_def
by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
  "∀ dip. nhop rt dip = nhop (invalidate rt dests) dip"
unfolding invalidate_def
by (clarsimp split: option.split)

lemma invalidate_other [simp]:
  fixes rt dests dip
  assumes "dip ∉ dom(dests)"
  shows "invalidate rt dests dip = rt dip"
using assms unfolding invalidate_def
by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes rt dests dip
  assumes "dip ∉ kD(rt)"
  shows "invalidate rt dests dip = None"
using assms unfolding invalidate_def
by clarsimp

lemma vD_invalidate_vD_not_dests:
  "∀ dip rt dests. dip ∈ vD(invalidate rt dests) ⟷ dip ∈ vD(rt) ∧ dests dip = None"
unfolding invalidate_def vD_def
by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
  fixes dests dip rt
  assumes "dip ∉ dom(dests)"
  shows "sqn rt dip = sqn (invalidate rt dests) dip"
unfolding sqn_def
by simp

lemma invalidate_changes:
  fixes rt dests dip dsn dsk flag hops nhip pre
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip)"
  shows "dsn = (case dests dip of None ⇒ π2(the (rt dip)) | Some rsn ⇒ rsn) ∧ dsk = π3(the (rt dip))"
∧ flag = (if dests dip = None then π_4(the (rt dip)) else inv)
∧ hops = π_5(the (rt dip))
∧ nhop = π_6(the (rt dip))"
using assms unfolding invalidate_def
by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "\ dip rt dests. dip∈kD (rt)
⇒ π_3(the (invalidate rt dests dip)) = π_3(the (rt dip))"
by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
assumes "dests ip = Some rsn"
and "ip∈kD(rt)"
shows "ip∈iD(invalidate rt dests)"
using assms(1) assms(2) [THEN kD_Some]
unfolding invalidate_def iD_def
by (clarsimp split: option.split)

5.1.6 Queued Packets

Functions for sending data packets.

type synonym store = "ip ⇒ (p × data list)"

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("σ_queue'(_, _)")
where "σ_queue'(store, dip) ≡ case store dip of None ⇒ [] | Some (p, q) ⇒ q"

definition qD :: "store ⇒ ip set"
where "qD ≡ dom"

definition add :: "data ⇒ ip ⇒ store ⇒ store"
where "add d dip store ≡ case store dip of
None ⇒ store (dip := None)
| Some (p, q) ⇒ store (dip := (p, q @ [d]))"

lemma qD_add [simp]:
fixes d dip store
shows "qD(add d dip store) = insert dip (qD store)"
unfolding add_def Let_def qD_def
by (clarsimp split: option.split)

definition drop :: "ip ⇒ store ⇒ store"
where "drop dip store ≡ map_option (λ(p, q). if tl q = [] then store (dip := None)
else store (dip := (p, tl q))) (store dip)"

definition sigma_p_flag :: "store ⇒ ip ⇒ p" ("σ_p-flag'(_, _)")
where "σ_p-flag'(store, dip) ≡ map_option fst (store dip)"

definition unsetRRF :: "store ⇒ ip ⇒ store"
where "unsetRRF store dip ≡ map_option (λ_, q). if dests dip = None then store dip
else map_option (λ(_, q). (req, q)) (store dip)"

5.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. nhop is partial, thus a ‘the’ is needed, similarly for dhops and addpreRT.
2. precs is partial.
3. $\sigma_{p\text{-flag}}(\text{store}, \text{dip})$ is partial.

4. The routing table ($rt$) is modelled as a map ($ip \Rightarrow r$ option) rather than a set of 7-tuples, likewise, the $r$ is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address ($dip$) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.

5. Similar remarks hold for the dests mapping passed to $\text{invalidate}$, and $\text{store}$.

end

5.2 AODV protocol messages

theory $E_{\text{Aodv\_Message}}$
imports $E_{\text{All\_ABCD}}$
begin

datatype msg =
   "Rreq nat ip sqn k ip sqn ip bool"
| "Rrep nat ip sqn ip ip"
| "Rerr "ip \map sqn" ip"
| "Newpkt data ip"
| "Pkt data ip ip"

instantiation msg :: msg
begin
   definition newpkt_def [simp]: "newpkt ≡ λ(d, dip). Newpkt d dip"
   definition eq_newpkt_def: "eq_newpkt m ≡ case m of Newpkt d dip ⇒ True | _ ⇒ False"

   instance by intro_classes (simp add: eq_newpkt_def)
end

The msg type models the different messages used within AODV. The instantiation as a msg is a technicality due to the special treatment of newpkt messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

definition rreq :: "nat × ip × sqn × k × ip × sqn × ip × bool ⇒ msg"
   where "rreq ≡ λ(hops, dip, dsn, dsk, oip, osn, sip, handled). Rreq hops dip dsn dsk oip osn sip handled"

   lemma rreq_simp [simp]:
      "rreq(hops, dip, dsn, dsk, oip, osn, sip, handled) = Rreq hops dip dsn dsk oip osn sip handled"
   unfolding rreq_def by simp

definition rrep :: "nat × ip × sqn × ip × ip ⇒ msg"
   where "rrep ≡ λ(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"

   lemma rrep_simp [simp]:
      "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
   unfolding rrep_def by simp

definition rerr :: "(ip ⇒ sqn) × ip ⇒ msg"
   where "rerr ≡ λ(dests, sip). Rerr dests sip"

   lemma rerr_simp [simp]:
      "rerr(dests, sip) = Rerr dests sip"
   unfolding rerr_def by simp

   lemma not_eq_newpkt_rreq [simp]: "¬eq_newpkt (Rreq hops dip dsn dsk oip osn sip handled)"
   unfolding eq_newpkt_def by simp

   lemma not_eq_newpkt_rrep [simp]: "¬eq_newpkt (Rrep hops dip dsn oip sip)"
   unfolding eq_newpkt_def by simp
lemma not_eq_newpkt_rerr [simp]: "¬ eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_pkt [simp]: "¬ eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp

definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"

lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp

end

5.3 The AODV protocol

theory E_Aodv
imports E_Aodv_Data E_Aodv_Message
AWN.AWN_SOS_Labels AWN.AWN_Invariants
begin

5.3.1 Data state

record state =
  ip :: "ip"
  sn :: "sqn"
  rt :: "rt"
  rreqs :: "(ip × sqn) set"
  store :: "store"
  msg :: "msg"
  data :: "data"
  dests :: "ip ⇒ sqn"

  dip :: "ip"
  oip :: "ip"
  hops :: "nat"
  dsn :: "sqn"
  dsk :: "k"
  osn :: "sqn"
  sip :: "ip"
  handled:: "bool"

abbreviation aodv_init :: "ip ⇒ state"
  where "aodv_init i ≡ (|
    ip = i,
    sn = 1,
    rt = Map.empty,
    rreqs = {},
    store = Map.empty,
    msg = (SOME x. True),
    data = (SOME x. True),
    dests = (SOME x. True),
    dip = (SOME x. True),
    oip = (SOME x. True),
    hops = (SOME x. True),
    dsn = (SOME x. True),
    dsk = (SOME x. True),
    osn = (SOME x. True),
    sip = (SOME x. x ≠ i),
    handled= (SOME x. True)
  )
lemma some_neq_not_eq [simp]: "¬((SOME x :: nat. x ≠ i) = i)"
  by (subst some_eq_ex) (metis zero_neq_numeral)

definition clearlocals :: "state ⇒ state"
where "clearlocals ξ = ξ (msg := (SOME x. True),
  data := (SOME x. True),
  dests := (SOME x. True),
  dip := (SOME x. True),
  oip := (SOME x. True),
  dsn := (SOME x. True),
  dsk := (SOME x. True),
  osn := (SOME x. True),
  sip := (SOME x. x ≠ ip ξ),
  handled := (SOME x. True))"

lemma clearlocals_sip_not_ip [simp]: "¬(sip (clearlocals ξ) = ip ξ)"
  unfolding clearlocals_def by simp

lemma clearlocals_but_notGlobals [simp]:
  "ip (clearlocals ξ) = ip ξ"
  "sn (clearlocals ξ) = sn ξ"
  "rt (clearlocals ξ) = rt ξ"
  "rreqs (clearlocals ξ) = rreqs ξ"
  "store (clearlocals ξ) = store ξ"
  unfolding clearlocals_def by auto

5.3.2 Auxilliary message handling definitions

definition is_newpkt
  where "is_newpkt ξ ≡ case msg ξ of
    Newpkt data’ dip’ ⇒ (ξ (msg := data’, dip := dip’))
    | _ ⇒ {}"

definition is_pkt
  where "is_pkt ξ ≡ case msg ξ of
    Pkt data’ dip’ oip’ ⇒ (ξ (msg := data’, dip := dip’, oip := oip’))
    | _ ⇒ {}"

definition is_rreq
  where "is_rreq ξ ≡ case msg ξ of
    Rreq hops’ dsn’ dsk’ oip’ osn’ sip’ handled’ ⇒
      (ξ (msg := Rreq hops’ dsn’ dsk’ oip’ osn’ sip’ handled’))
    | _ ⇒ {}"

lemma is_rreq_asm [dest!]:
  assumes "ξ ∈ is_rreq ξ"
  shows "(∃ hops’ dip’ dsn’ dsk’ oip’ osn’ sip’ handled’.
    msg ξ = Rreq hops’ dsn’ dsk’ oip’ osn’ sip’ handled’ ∧
    ξ’ = ξ (msg := Rreq hops’ dsn’ dsk’ oip’ osn’ sip’ handled’))"
  using assms unfolding is_rreq_def
  by (cases "msg ξ") simp_all

definition is_rrep
  where "is_rrep ξ ≡ case msg ξ of
Rrep hops' dip' dsn' oip' sip' ⇒

lemma is_rrep_asm [dest!]:
assumes "ξ' ∈ is_rrep ξ"
shows "(∃ hops' dip' dsn' oip' sip'.
msg ξ = Rrep hops' dip' dsn' oip' sip' ∧
using assms unfolding is_rrep_def
by (cases "msg ξ") simp_all

definition is_rerr
where "is_rerr ξ ≡ case msg ξ of
Rerr dests' sip' ⇒ \{ ξ(\{ dests := dests', sip := sip' \}) \}
| _ ⇒ {\}"

lemma is_rerr_asm [dest!]:
assumes "ξ' ∈ is_rerr ξ"
shows "(∃ dests' sip'.
msg ξ = Rerr dests' sip' ∧
ξ' = ξ(\{ dests := dests', sip := sip' \}))"
using assms unfolding is_rerr_def
by (cases "msg ξ") simp_all

lemmas is_msg_defs =
is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def

lemma is_msg_inv_ip [simp]:
"ξ' ∈ is_rerr ξ " "ξ' ∈ is_rrep ξ " "ξ' ∈ is_rreq ξ " "ξ' ∈ is_pkt ξ " "ξ' ∈ is_newpkt ξ "
⇒ ip ξ' = ip ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp)+

lemma is_msg_inv_sn [simp]:
"ξ' ∈ is_rerr ξ " "ξ' ∈ is_rrep ξ " "ξ' ∈ is_rreq ξ " "ξ' ∈ is_pkt ξ " "ξ' ∈ is_newpkt ξ "
⇒ sn ξ' = sn ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp)+

lemma is_msg_inv_rt [simp]:
"ξ' ∈ is_rerr ξ " "ξ' ∈ is_rrep ξ " "ξ' ∈ is_rreq ξ " "ξ' ∈ is_pkt ξ " "ξ' ∈ is_newpkt ξ "
⇒ rt ξ' = rt ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp)+

lemma is_msg_inv_rreqs [simp]:
"ξ' ∈ is_rerr ξ " "ξ' ∈ is_rrep ξ " "ξ' ∈ is_rreq ξ " "ξ' ∈ is_pkt ξ " "ξ' ∈ is_newpkt ξ "
⇒ rreqs ξ' = rreqs ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp)+
lemma is_msg_inv_store [simp]:

"ξ' ∈ is_rerr ξ \implies store ξ' = store ξ"
"ξ' ∈ is_rrep ξ \implies store ξ' = store ξ"
"ξ' ∈ is_rreq ξ \implies store ξ' = store ξ"
"ξ' ∈ is_pkt ξ \implies store ξ' = store ξ"
"ξ' ∈ is_newpkt ξ \implies store ξ' = store ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

lemma is_msg_inv_sip [simp]:

"ξ' ∈ is_pkt ξ \implies sip ξ' = sip ξ"
"ξ' ∈ is_newpkt ξ \implies sip ξ' = sip ξ"

unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

5.3.3 The protocol process

datatype pseqp =
  PAodv
| PNewPkt
| PPkt
| PRreq
| PRrep
| PRerr

fun nat_of_seqp :: "pseqp \Rightarrow nat"
where
  "nat_of_seqp PAodv = 1"
| "nat_of_seqp PPkt = 2"
| "nat_of_seqp PNewPkt = 3"
| "nat_of_seqp PRreq = 4"
| "nat_of_seqp PRrep = 5"
| "nat_of_seqp PRerr = 6"

instantiation "pseqp" :: ord
begin
definition less_eq_seqp [iff]: "l1 \leq l2 = (nat_of_seqp l1 \leq nat_of_seqp l2)"
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
instance ..

end

abbreviation AODV
where
  "AODV ≡ \_, [clear_locals] call(PAodv)"

abbreviation PKT
where
  "PKT args ≡
    \ξ. let (data, dip, oip) = args ξ in
    (clear_locals ξ) () data := data, dip := dip, oip := oip []
    call(PPkt)"

abbreviation NEWPKT
where
  "NEWPKT args ≡
    \ξ. let (data, dip) = args ξ in
    (clear_locals ξ) () data := data, dip := dip []
    call(PNewPkt)"

abbreviation RREQ
where
  "RREQ args ≡
    \ξ. let (hops, dip, dsn, dsk, oip, osn, sip, handled) = args ξ in
    (clear_locals ξ) () hops := hops, dip := dip,
abbreviation RREP
where
"RREP args ≡
\[ \xi. \ let \ (hops, dip, dns, oip, sip) = args \xi \ in
(clear_locals \xi) () hops := hops, dip := dip, dns := dns,
oip := oip, sip := sip ]
call(PRrep)"

abbreviation RERR
where
"RERR args ≡
\[ \xi. \ let \ (dests, sip) = args \xi \ in
(clear_locals \xi) () dests := dests, sip := sip ]
call(PRerr)"

fun AODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
"AODV PAdv = labelled PAdv (λ msg.\xi. \xi \ (msg := msg' \)).
 (is_newpkt) NEWPKT(λ ξ. (data ξ, ip ξ))
( is_pkt ) PKT(λ ξ. (data ξ, dip ξ, oip ξ))
( is_rreq )
\[ \xi. \ let \ (hops, dip, dns, oip, sip) = args \xi \ in
\begin{align*}
 & (clear_locals \xi) () \ rt := update (rt \xi) (sip \xi) (0, unk, val, 1, sip \xi) \\
 & RREQ(λ ξ. (hops ξ, dip ξ, dns ξ, oip ξ, osn ξ, sip ξ, handled ξ))
\end{align*}
)
( is_rerr )
\[ \xi. \ let \ (hops, dip, dns, oip, sip) = args \xi \ in
\begin{align*}
 & (clear_locals \xi) () \ rt := update (rt \xi) (sip \xi) (0, unk, val, 1, sip \xi) \\
 & RREP(λ ξ. (hops ξ, dip ξ, dns ξ, oip ξ, sip ξ))
\end{align*}
)
( λ ξ. { \xi | \ dip := dip \}) \{ \xi | \ dip ∈ qD(store ξ) \} \cap vD(rt ξ)
\begin{align*}
 & \xi. \ let \ (dests, sip) = args \xi \ in
\begin{align*}
 & (clear_locals \xi) () \ rt := update (rt \xi) (sip \xi) (0, unk, val, 1, sip \xi) \\
 & RREQ(λ ξ. (hops ξ, dip ξ, dns ξ, oip ξ, sip ξ))
\end{align*}
\end{align*}
)
( λ ξ. \{ \xi | \ dip := dip \}
\begin{align*}
 & \xi. \ let \ (dests, sip) = args \xi \ in
\begin{align*}
 & (clear_locals \xi) () \ rt := update (rt \xi) (sip \xi) (0, unk, val, 1, sip \xi) \\
 & RREQ(λ ξ. (hops ξ, dip ξ, dns ξ, oip ξ, sip ξ))
\end{align*}
\end{align*}
)
\[
\begin{align*}
\{ \xi. \text{ dip } \xi & \in v D (rt \xi) \} \\
& \text{unicast}(\lambda \xi. \text{ the (nhop (rt \xi) (dip \xi)), } \lambda \xi. \text{ pkt(data } \xi, \text{ dip } \xi, \text{ oip } \xi)). \text{AODV(}) \\
> & \text{\phantom{\{ \xi. \text dip } \xi \in v D (rt \xi)}) \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ handled } \xi = \text{ False} \} \\
\{ \xi. \text{ dip } \xi \in i D (rt \xi) \} \\
& \text{\phantom{\{ \xi. \text{ dip } \xi \notin i D (rt \xi)})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ dip } \xi, \text oip } \xi, \text osn } \xi \notin \text rreqs } \xi \} \\
& \text{\phantom{\{ \xi. \text{ dip } \xi, \text osn } \xi \notin \text rreqs } \xi \} \\
\{ \xi. \text{ handled } \xi = \text{ False} \} \\
\{ \xi. \text{ dip } \xi = \text{ ip } \xi \} \\
\{ \xi. \text{ dip } \xi \notin \text i D (rt \xi) \} \\
& \text{\phantom{\{ \xi. \text{ dip } \xi \notin \text i D (rt \xi)})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ dip } \xi \notin v D (rt \xi) \} \\
\{ \xi. \text{ dip } \xi \notin v D (rt \xi) \} \\
& \text{broadcast}(\lambda \xi. \text{ rreq(hops } \xi + 1, \text dip } \xi, \text dsn } \xi, \text dsk } \xi, \text oip } \xi, \text osn } \xi, \text ip } \xi, \text True)). \text{AODV(}) \\
> & \text{\phantom{\{ \xi. \text{ dip } \xi \notin v D (rt \xi)})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ handled } \xi = \text{ True} \} \\
& \text{\phantom{\{ \xi. \text{ handled } \xi = \text{ True} \} \})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
& \text{broadcast}(\lambda \xi. \text{ rreq(hops } \xi + 1, \text dip } \xi, \text dsn } \xi, \text dsk } \xi, \text oip } \xi, \text osn } \xi, \text ip } \xi, \text False)). \text{AODV(}) \\
> & \text{\phantom{\{ \xi. \text{ dip } \xi \in v D (rt \xi)})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
& \text{broadcast}(\lambda \xi. \text{ rreq(hops } \xi + 1, \text dip } \xi, \text dsn } \xi, \text dsk } \xi, \text oip } \xi, \text osn } \xi, \text ip } \xi, \text True)). \text{AODV(}) \\
> & \text{\phantom{\{ \xi. \text{ dip } \xi \in v D (rt \xi)})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
& \text{broadcast}(\lambda \xi. \text{ rreq(hops } \xi + 1, \text dip } \xi, \text max (sqn (rt \xi) (dip \xi)) (dsn } \xi), \\
& \text{dsk } \xi, \text oip } \xi, \text osn } \xi, \text ip } \xi, \text False)). \text{AODV(}) \\
> & \text{\phantom{\{ \xi. \text{ dip } \xi \in v D (rt \xi)})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
& \text{broadcast}(\lambda \xi. \text{ rreq(hops } \xi + 1, \text dip } \xi, \text dsn } \xi, \text dsk } \xi, \text oip } \xi, \text osn } \xi, \text ip } \xi, \text True)). \text{AODV(}) \\
> & \text{\phantom{\{ \xi. \text{ dip } \xi \in v D (rt \xi)})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
& \text{broadcast}(\lambda \xi. \text{ rreq(hops } \xi + 1, \text dip } \xi, \text max (sqn (rt \xi) (dip \xi)) (dsn } \xi), \\
& \text{dsk } \xi, \text oip } \xi, \text osn } \xi, \text ip } \xi, \text True)). \text{AODV(}) \\
> & \text{\phantom{\{ \xi. \text{ dip } \xi \in v D (rt \xi)})} \\
\end{align*}
\]

\[
\begin{align*}
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
\{ \xi. \text{ dip } \xi \in v D (rt \xi) \} \\
& \text{broadcast}(\lambda \xi. \text{ rreq(hops } \xi + 1, \text dip } \xi, \text max (sqn (rt \xi) (dip \xi)) (dsn } \xi), \\
& \text{dsk } \xi, \text oip } \xi, \text osn } \xi, \text ip } \xi, \text False)). \text{AODV(}) \\
> & \text{\phantom{\{ \xi. \text{ dip } \xi \in v D (rt \xi)})} \\
\end{align*}
\]
\[\begin{align*}
\{ \xi. \text{oip }\xi &= \text{ip }\xi \}\quad &\xrightarrow{\text{AODV}} \\
\{ \xi. \text{oip }\xi \neq \text{ip }\xi \}\quad &\xrightarrow{\text{AODV}} \\
\{ \xi. \text{oip }\xi \in \text{vD} (\text{rt }\xi) \land \text{dip }\xi \in \text{vD} (\text{rt }\xi)\} \\
&\quad \text{unicast}(\lambda_x. \text{the } (\text{nhop }\text{rt }\xi) (\text{oip }\xi)), \lambda_x. \text{rrep}((\text{dhops }\text{rt }\xi) (\text{dip }\xi)), \text{dip }\xi, \text{sqn }\text{rt }\xi (\text{dip }\xi), \text{oip }\xi, \text{ip }\xi). \\
&\quad \text{AODV()} \\
\langle \xi. \text{oip }\xi \neq \text{ip }\xi \rangle \\
\quad \quad \langle \xi. \text{dests }\xi \neq \text{Map.empty} \rangle \\
&\quad \quad \text{broadcast}(\lambda_x. \text{rerr}(\text{dests }\xi, \text{ip }\xi)). \text{AODV()} \\
\end{align*}\]
\( p \in \text{ctermsl} (\Gamma_{AODV} \text{PPkt}) \lor \)
\( p \in \text{ctermsl} (\Gamma_{AODV} \text{PRreq}) \lor \)
\( p \in \text{ctermsl} (\Gamma_{AODV} \text{PRrep}) \lor \)
\( p \in \text{ctermsl} (\Gamma_{AODV} \text{PRerr}) \)

by (cases \( pn \)) \( \text{simp\_all} \)

definition \( \sigma_{AODV} \) :: \( \text{ip} \Rightarrow (\text{state} \times (\text{state, msg, pseqp, pseqp label}) \text{ seqp}) \text{ set} \)
where "\( \sigma_{AODV} \ i \equiv \{(\text{aodv\_init} \ i, \ \Gamma_{AODV} \text{PAodv})\} \)"

abbreviation \( \text{paodv} \) :: \( \text{ip} \Rightarrow (\text{state} \times (\text{state, msg, pseqp, pseqp label}) \text{ seqp, msg seq\_action}) \text{ automaton} \)
where "\( \text{paodv} \ i \equiv (| \text{init} = \sigma_{AODV} \ i, \ \text{trans} = \text{seqp\_sos} \ \Gamma_{AODV} |) \)"

lemma \( \text{aodv\_trans} \) : "\( \text{trans} (\text{paodv} \ i) = \text{seqp\_sos} \ \Gamma_{AODV} \)"
by \( \text{simp} \)

lemma \( \text{aodv\_control\_within} \ [\text{simp}] \) : "\( \text{control\_within} \ \Gamma_{AODV} (\text{init} (\text{paodv} \ i)) \)"
unfolding \( \sigma_{AODV} \_\text{def} \) by (rule \( \text{control\_withinI} \)) \( \text{auto simp del:} \ \Gamma_{AODV} \_\text{simps} \)

lemma \( \text{aodv\_wf} \ [\text{simp}] \) :
"\( \text{wellformed} \ \Gamma_{AODV} \)"
proof (rule, intro allI)
  fix \( pn \ \ pn' \)
  show "\( \text{call(pn')} \notin \text{stermsl} (\Gamma_{AODV} \ pn) \)"
  by (cases \( pn \)) \( \text{simp\_all} \)
qed

lemmas \( \text{aodv\_labels\_not\_empty} \ [\text{simp}] = \text{labels\_not\_empty} \ [\text{OF aodv\_wf}] \)

lemma \( \text{aodv\_ex\_label} \ [\text{intro}] \) :
"\( \exists l. l \in \text{labels} \ \Gamma_{AODV} \)"
by (metis \( \text{aodv\_labels\_not\_empty} \_\text{all\_not\_in\_conv} \))

lemma \( \text{aodv\_ex\_labelE} \ [\text{elim}] \) :
  assumes "\( \forall l \in \text{labels} \ \Gamma_{AODV} \ p. \ P \ l \ p \)"
  and "\( \exists p. l. \ P \ l \ p \Longrightarrow Q \)"
  shows "\( Q \)"
using \( \text{assms} \) by (metis \( \text{aodv\_ex\_label} \))

lemma \( \text{aodv\_simple\_labels} \ [\text{simp}] \) :
"\( \text{simple\_labels} \ \Gamma_{AODV} \)"
proof
  fix \( pn \)
  assume "\( p \in \text{subterms}(\Gamma_{AODV} \ pn) \)"
  thus "\( \exists ! l. \text{labels} \ \Gamma_{AODV} \ p = \{l\} \)"
  by (cases \( pn \)) \( \text{simp\_all cong: seqp\_congs | elim \text{disjE}} \) +
qed

lemma \( \sigma_{AODV} \_\text{labels} \ [\text{simp}] \) :
"\( (\xi, \ p) \in \sigma_{AODV} \ i \Longrightarrow \text{labels} \ \Gamma_{AODV} \ p = \{\text{PAodv}:-0\} \)"
unfolding \( \sigma_{AODV} \_\text{def} \) by \( \text{simp} \)

lemma \( \text{aodv\_init\_kd\_empty} \ [\text{simp}] \) :
"\( (\xi, \ p) \in \sigma_{AODV} \ i \Longrightarrow \text{kd} (\text{rt} \ \xi) = \{\} \)"
unfolding \( \sigma_{AODV} \_\text{def} \) \( \text{kd\_def} \) by \( \text{simp} \)

lemma \( \text{aodv\_init\_sip\_not\_ip} \ [\text{simp}] \) :
"\neg (\text{sip} (\text{aodv\_init} \ i) = \ i) \)" by \( \text{simp} \)

lemma \( \text{aodv\_init\_sip\_not\_ip'} \ [\text{simp}] \) :
  assumes "\( (\xi, \ p) \in \sigma_{AODV} \ i \)"
  shows "\( \text{sip} \ \xi \neq \ \text{ip} \ \xi \)"
using \( \text{assms} \) unfolding \( \sigma_{AODV} \_\text{def} \) by \( \text{simp} \)

lemma \( \text{aodv\_init\_sip\_not\_i} \ [\text{simp}] \) :
  assumes "\( (\xi, \ p) \in \sigma_{AODV} \ i \)"
  shows "\( \text{sip} \ \xi \neq \ i \)"
using assms unfolding \( \sigma_{AODV\_def} \) by simp

lemma clear_locals_sip_not_ip':
assumes "\( ip \ \xi = i \)"
shows "\( \neg (sip (clear_locals \ \xi) = i) \)"
using assms by auto

Stop the simplifier from descending into process terms.
declare seqp_congs [cong]

Configure the main invariant tactic for AODV.
declare
\( \Gamma_{AODV\_simps} \) [cterms_env]
aodv_proc_cases [cterms1_cases]
seq_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms_intros]
seq_step_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans, cterms_intros]
end

5.4 Invariant assumptions and properties

theory E_Aodv_Predicates
imports E_Aodv
begin

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

abbreviation not_Pkt :: "msg \Rightarrow \text{bool}"
where "not_Pkt m \equiv \text{case } m \text{ of } Pkt _ _ _ \Rightarrow \text{False} \mid _ \Rightarrow \text{True}"
definition msg_sender :: "msg \Rightarrow \text{ip}"
where "msg_sender m \equiv \text{case } m \text{ of } Rreq _ _ _ _ _ _ ipc _ \Rightarrow ipc
\mid Rrep _ _ _ ipc \Rightarrow ipc
\mid Rerr _ ipc \Rightarrow ipc
\mid Pkt _ _ ipc \Rightarrow ipc"
lemma msg_sender_simps [simp]:
\[ \forall \text{hops dip dsn dsk oip osn sip handled}. \]
\[ \text{msg_sender (Rreq hops dip dsn dsk oip osn sip handled)} = sip \]
\[ \forall \text{hops dip dsn oip sip}. \text{msg_sender (Rrep hops dip dsn oip sip)} = sip \]
\[ \forall \text{dests sip}. \text{msg_sender (Rerr dests sip)} = sip \]
\[ \forall d dip sip. \text{msg_sender (Pkt d dip sip)} = sip \]
unfolding msg_sender_def by simp_all
definition msg_zhops :: "msg \Rightarrow \text{bool}"
where "msg_zhops m \equiv \text{case } m \text{ of } \]
\[ \text{Rreq hopsc dipc _ _ _ _ oipc _ _ osnc _ _ } \Rightarrow \text{hopsc = 0 } \rightarrow \text{oipc = sipc} \]
\[ \text{Rrep hopsc dipc _ _ _ sipc _ _ } \Rightarrow \text{hopsc = 0 } \rightarrow \text{dipc = sipc} \]
\[ _ \Rightarrow \text{True} \]
lemma msg_zhops_simps [simp]:
\[ \forall \text{hops dip dsn dsk oip osn sip handled}. \]
\[ \text{msg_zhops (Rreq hops dip dsn dsk oip osn sip handled)} = (\text{hops = 0 } \rightarrow \text{oip = sip}) \]
\[ \forall \text{hops dip dsn oip sip}. \text{msg_zhops (Rrep hops dip dsn oip sip)} = (\text{hops = 0 } \rightarrow \text{dip = sip}) \]
\[ \forall \text{dests sip}. \text{msg_zhops (Rerr dests sip)} = \text{True} \]
\[ \forall d dip. \text{msg_zhops (Newpkt d dip)} = \text{True} \]
\[ \forall d dip sip. \text{msg_zhops (Pkt d dip sip)} = \text{True} \]
unfolding msg_zhops_def by simp_all
definition rreq_rrep_sn :: "msg \Rightarrow \text{bool}"
where "rreq_rrep_sn m \equiv \text{case } m \text{ of } \]
\[ \text{Rreq _ _ _ _ _ _ osnc _ _ } \Rightarrow \text{osnc } \geq 1 \]
\[ \text{Rrep _ _ dsnc _ _ } \Rightarrow \text{dsnc } \geq 1 \]
| \_ \Rightarrow \text{True} |

lemma \textit{rreq\_rrep\_sn\_sims} \([\text{simp}]\):

- \(\forall \text{hops dip dsn osn sip handled.} \quad \text{rreq\_rrep\_sn} (\text{Rreq hops dip dsn osn sip handled}) = (\text{osn} \geq 1)\)
- \(\forall \text{hops dip dsn osn sip.} \quad \text{rreq\_rrep\_sn} (\text{Rrep hops dip dsn osn sip}) = (\text{dsn} \geq 1)\)
- \(\forall \text{dests sip.} \quad \text{rreq\_rrep\_sn} (\text{Rerr dests sip}) = \text{True}\)
- \(\forall \text{d dip.} \quad \text{rreq\_rrep\_sn} (\text{Newpkt d dip}) = \text{True}\)
- \(\forall \text{d dip sip.} \quad \text{rreq\_rrep\_sn} (\text{Pkt d dip sip}) = \text{True}\)

unfolding \textit{rreq\_rrep\_sn\_def} by \textit{simp_all}

definition \textit{rreq\_rrep\_fresh} :: "rt \Rightarrow msg \Rightarrow bool"
where "\textit{rreq\_rrep\_fresh} \textit{crt} \textit{m} \equiv \textit{case m of} \begin{cases} \text{Rreq hopsc _ _ _ oipc osnc ipcc _} \Rightarrow (\text{ipcc} \neq \text{oipc} \rightarrow \\ \quad (\text{oipc} \in \text{kD(crt)} \land (\text{sqn} \text{crt oipc} > \text{osnc} \\
\quad \lor (\text{sqn} \text{crt oipc} = \text{osnc} \\
\quad \land (\text{the (dhops} \text{crt oipc} \leq \text{hopsc} \\
\quad \land (\text{the (flag} \text{crt oipc} = \text{val}))) \\
\quad \lor \text{Rrep hopsc dipc dsnc _ ipcc} \Rightarrow (\text{ipcc} \neq \text{dipc} \\
\quad \rightarrow (\text{dipc} \in \text{kD(crt)} \\
\quad \land (\text{sqn} \text{crt dipc} = \text{dsnc} \\
\quad \land (\text{the (dhops} \text{crt dipc} = \text{hopsc} \\
\quad \land (\text{the (flag} \text{crt dipc} = \text{val}))) \\
\quad \lor \_ \Rightarrow \text{True}\end{cases}"

lemma \textit{rreq\_rrep\_fresh} \([\text{simp}]\):

- \(\forall \text{hops dip dsn osn sip handled.} \quad \text{rreq\_rrep\_fresh} (\text{Rreq hops dip dsn osn sip handled}) = (\text{osn} \neq \text{ipcc} \\
\quad \lor (\text{sqn} \text{crt oipc} > \text{osnc} \\
\quad \lor (\text{sqn} \text{crt oipc} = \text{osnc} \\
\quad \land (\text{the (dhops} \text{crt oipc} \leq \text{hopsc} \\
\quad \land (\text{the (flag} \text{crt oipc} = \text{val}))) \\
\quad \lor \_ \Rightarrow \text{True}\end{cases}"

unfolding \textit{rreq\_rrep\_fresh\_def} by \textit{simp_all}

definition \textit{rerr\_invalid} :: "rt \Rightarrow msg \Rightarrow bool"
where "\textit{rerr\_invalid} \textit{crt} \textit{m} \equiv \textit{case m of} \begin{cases} \text{Rerr destsc _} \Rightarrow (\forall \text{ripc} \in \text{dom(destsc).} \\
\quad (\text{ripc} \in \text{iD(crt)} \land (\text{the (destsc ripc} = \text{sqn} \text{crt ripc})) \\
\quad \lor \_ \Rightarrow \text{True}\end{cases}"

lemma \textit{rerr\_invalid} \([\text{simp}]\):

- \(\forall \text{hops dip dsn osn sip handled.} \quad \text{rerr\_invalid} (\text{Rreq hops dip dsn osn sip handled}) = \text{True}"
- \(\forall \text{hops dip dsn osn sip.} \quad \text{rerr\_invalid} (\text{Rrep hops dip dsn osn sip}) = \text{True}"
- \(\forall \text{dests sip.} \quad \text{rerr\_invalid} (\text{Rerr dests sip}) = (\forall \text{ripc} \in \text{dom(dests).} \\
\quad (\text{ripc} \in \text{iD(crt)} \land (\text{the (dests ripc} = \text{sqn} \text{crt ripc)))} \\
\quad \lor \_ \Rightarrow \text{True}\end{cases}"

unfolding \textit{rerr\_invalid\_def} by \textit{simp_all}

definition \textit{initmissing} :: "(nat \Rightarrow \text{state option}) \times \text{a} \Rightarrow (nat \Rightarrow \text{state}) \times \text{a}"
where "\textit{initmissing} \sigma = (\lambda i. \text{case (fst} \sigma) \text{i of None} \Rightarrow \text{aodv\_init} i | \text{Some s} \Rightarrow s, \text{snd} \sigma)"

lemma \textit{not\_in\_net\_ips\_fst\_init\_missing} \([\text{simp}]\):

assumes "\_ \notin \text{net\_ips} \sigma"
shows "fst (initmissing (netgmap fst σ)) i = aodv_init i"
using assms unfolding initmissing_def by simp

lemma fst_initmissing_netgmap_pair_fst [simp]:
  "fst (initmissing (netgmap (\(p, q\). (fst (id p), snd (id p), q)) s))
   = fst (initmissing (netgmap fst s))"
unfolding initmissing_def by auto

We introduce a streamlined alternative to initmissing with netgmap to simplify invariant statements and thus facilitate their comprehension and presentation.

lemma fst_initmissing_netgmap_default_aodv_init_netlift:
  "fst (initmissing (netgmap fst s)) = default aodv_init (netlift fst s)"
unfolding initmissing_def default_def by (simp add: fst_netgmap_netlift del: One_nat_def)

definition netglobal :: "\((nat ⇒ state) ⇒ bool\) ⇒ ((state × 'b) × 'c) net_state ⇒ bool"
where "netglobal P ≡ (λs. P (default aodv_init (netlift fst s)))"
end

5.5 Quality relations between routes

theory E_Fresher
imports E_Aodv_Data
begin

5.5.1 Net sequence numbers

On individual routes

definition nsqn_r :: "r ⇒ sqn"
where "nsqn_r r ≡ if \(π_4(r) = \text{val} ∨ π_2(r) = 0\) then π_2(r) else (π_2(r) - 1)"

lemma nsqnr_def': "nsqn_r r = (if \(π_4(r) = \text{inv}\) then π_2(r) - 1 else π_2(r))"
unfolding nsqn_r_def by simp

lemma nsqn_r_zero [simp]: "∀dsn dsk flag hops nhip. nsqn_r (0, dsk, flag, hops, nhip) = 0"
unfolding nsqn_r_def by clarsimp

lemma nsqn_r_val [simp]: "∀dsn dsk hops nhip. nsqn_r (dsn, dsk, val, hops, nhip) = dsn"
unfolding nsqn_r_def by clarsimp

lemma nsqn_r_inv [simp]: "∀dsn dsk hops nhip. nsqn_r (dsn, dsk, inv, hops, nhip) = dsn - 1"
unfolding nsqn_r_def by clarsimp

lemma nsqn_r_lte_dsn [simp]: "∀dsn dsk flag hops nhip. nsqn_r (dsn, dsk, flag, hops, nhip) ≤ dsn"
unfolding nsqn_r_def by clarsimp

On routes in routing tables

definition nsqn :: "rt ⇒ ip ⇒ sqn"
where "nsqn ≡ λrt dip. case σ_route(rt, dip) of None ⇒ 0 | Some r ⇒ nsqn_r(r)"
lemma nsqn_sqn_def:
"rt dip. sqn rt dip = (if flag rt dip = Some val ∨ sqn rt dip = 0
then sqn rt dip else sqn rt dip - 1)"

unfolding nsqn_def sqn_def by (clarsimp split: option.split)

lemma not_in_kD_nsqn [simp]:
  assumes "dip ∉ kD(rt)"
  shows "nsqn rt dip = 0"
  using assms unfolding nsqn_def by simp

lemma kD_nsqn:
  assumes "dip ∈ kD(rt)"
  shows "nsqn rt dip = nsqn r (the (σroute(rt, dip)))"
  using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

lemma nsqnr_r_flag_pred [simp, intro]:
  fixes dsn dsk flag hops nhip pre
  assumes "P (nsqn r (dsn, dsk, val, hops, nhip))"
  and "P (nsqn r (dsn, dsk, inv, hops, nhip))"
  shows "P (nsqn r (dsn, dsk, flag, hops, nhip))"
  using assms by (cases flag) auto

lemma sqn_nsqn:
"rt dip. sqn rt dip - 1 ≤ nsqn rt dip"

unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma val_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
  shows "nsqn rt ip = sqn rt ip"
  using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ vD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
  proof -
  from ⟨ip ∈ vD(rt)⟩ have "ip ∈ kD(rt)"
    and "the (flag rt ip) = val" by auto
    thus ?thesis ..
  qed

lemma inv_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
  using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
  proof -
  from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)"
    and "the (flag rt ip) = inv" by auto
    thus ?thesis ..
  qed

lemma nsqn_update_changed_kno_val [simp]: "rt ip dsn dsk hops nhip.
  rt ≠ update rt ip (dsn, kno, val, hops, nhip)
  ⇒ nsqn (update rt ip (dsn, kno, val, hops, nhip)) ip = dsn"

unfolding nsqn_def update_def
by (clarsimp simp: kD_nsqn split: option.split_asm option.split_if_split_asm)
lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip pre rt ip
  assumes "dip ≠ ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip)) dip = nsqn rt dip"
  using assms unfolding nsqn_def
  by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
  assumes "dip ∈ kD(rt)"
  and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
  using assms
  proof -
    from assms obtain dsk hops nhip
      where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip)"
      unfolding invalidate_def
      by auto
    moreover from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)"
      by simp
    ultimately show ?thesis
      using ⟨dests dip = Some rsn⟩ by simp
  qed

lemma nsqn_invalidate_other [simp]:
  assumes "dip ∈ kD(rt)"
  and "dip /∈ dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
  using assms
  by (clarsimp simp add: kD_nsqn)

5.5.2 Comparing routes

definition fresher :: "r ⇒ r ⇒ bool" ("(_/ ⊑ _)" [51, 51] 50)
where
  "fresher r r' ≡ ((nsqn, r < nsqn, r') ∨ (nsqn, r = nsqn, r' ∧ π5(r) ≥ π5(r')))"

lemma fresherI1 [intro]:
  assumes "nsqn, r < nsqn, r'"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
  assumes "nsqn, r = nsqn, r'" and "π5(r) ≥ π5(r')"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI [intro]:
  assumes "(nsqn, r < nsqn, r') ∨ (nsqn, r = nsqn, r' ∧ π5(r) ≥ π5(r'))"
  shows "r ⊑ r'"
  unfolding fresher_def using assms .

lemma fresherE [elim]:
  assumes "r ⊑ r'" and "nsqn, r < nsqn, r'" ⟹ P r r'" and "nsqn, r = nsqn, r' ∧ π5(r) ≥ π5(r')" ⟹ P r r'"
  shows "P r r'"
  using assms unfolding fresher_def by auto

lemma fresher_refl [simp]: "r ⊑ r"
  unfolding fresher_def by simp

lemma fresher_trans [elim, trans]:

"[ x ⊑ y; y ⊑ z ] ⇒ x ⊑ z"
unfolding fresher_def by auto

lemma not_fresher_trans [elim, trans]:
"[ ¬(x ⊑ y); ¬(z ⊑ x) ] ⇒ ¬(z ⊑ y)"
unfolding fresher_def by auto

lemma fresher_dsn_flag_hops_const [simp]:
fixes dsn dsd dsk flag hops nhip nhip'
shows "(dsn, dsd, flag, hops, nhip) ⊑ (dsn, dsk', flag, hops, nhip')"
unfolding fresher_def by (cases flag) simp_all

5.5.3 Comparing routing tables
definition rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where "rt_fresher ≡ λdip rt rt'. (the (σroute(rt, dip))) ⊑ (the (σroute(rt', dip)))"
abbreviation rt_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊑_ _)" [51, 999, 51] 50)
where "rt1 ⊑_i rt2 ≡ rt_fresher i rt1 rt2"

lemma rt_fresher_def':
"(rt1 ⊑_i rt2) = (nsqn rt1 dip < nsqn rt2 dip
          ∨ (nsqn rt1 dip = nsqn rt2 dip ∧ π5 (the (dhops rt1 dip)) ≥ π5 (the (dhops rt2 dip))))"
unfolding rt_fresher_def fresher_def by (rule refl)

lemma single_rt_fresher [intro]:
assumes "the (rt1 ip) ⊑ the (rt2 ip)"
shows "rt1 ⊑_ip rt2"
using assms unfolding rt_fresher_def .

lemma rt_fresher_single [intro]:
assumes "rt1 ⊑_ip rt2"
shows "the (rt1 ip) ⊑ the (rt2 ip)"
using assms unfolding rt_fresher_def .

lemma rt_fresher_def2:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
shows "(rt1 ⊑_dip rt2) = (nsqn rt1 dip < nsqn rt2 dip
          ∨ (nsqn rt1 dip = nsqn rt2 dip ∧ the (dhops rt1 dip) ≥ the (dhops rt2 dip)))"
using assms unfolding rt_fresher_def fresher_def by (simp add: kD_nsqn proj5_eq_dhops)

lemma rt_fresherI1 [intro]:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "nsqn rt1 dip < nsqn rt2 dip"
shows "rt1 ⊑_dip rt2"
unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherI2 [intro]:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "nsqn rt1 dip = nsqn rt2 dip"
and "the (dhops rt1 dip) ≥ the (dhops rt2 dip)"
shows "rt1 ⊑_dip rt2"
unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt_fresherE [elim]:
assumes "rt1 ⊑_dip rt2"
and "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "[ nsqn rt1 dip < nsqn rt2 dip ] → P rt1 rt2 dip"
and "[ nsqn rt1 dip = nsqn rt2 dip; the (dhops rt1 dip) ≥ the (dhops rt2 dip) ] → P rt1 rt2 dip"
shows "P rt1 rt2 dip"
using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "rt ⊑ dip rt"
unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt3"
  shows "rt1 ⊑ dip rt3"
using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip) ⊑ r"
  shows "rt ⊑ dip (∀ip. if ip = dip then Some r else rt ip)"
using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip ⇒ rt ⇒ rt ⇒ bool"
where "rt_fresh_as ≡ λdip rt1 rt2. (rt1 ⊑ dip rt2) ∧ (rt2 ⊑ dip rt1)"

abbreviation rt_fresh_as_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/≈/ _)" [51, 999, 51] 50)
where "rt1 ≈_1 rt2 ≡ rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: "∀rt dip. rt ≈ dip rt"
unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  "∀rt1 rt2 rt3 dip. [ rt1 ⊑ dip rt2; rt2 ⊑ dip rt3 ] → rt1 ⊑ dip rt3"
unfolding rt_fresh_as_def rt_fresher_def
by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1 ⊑ dip rt2"
  and "rt2 ⊑ dip rt1"
  shows "rt1 ≈ dip rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "the (rt1 dip) ⊑ the (rt2 dip)"
  and "the (rt2 dip) ⊑ the (rt1 dip)"
  shows "rt1 ≈ dip rt2"
using assms unfolding rt_fresh_as_def
by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip ∈ kD(rt)"
  and "dip ∈ kD(rt')"
  and "nsqn rt dip = nsqn rt' dip"
  and "π5(the (rt dip)) = π5(the (rt' dip))"
  shows "rt ≈ dip rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip) ≤ the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
with assms(1-3) show "rt \sqsubseteq dip rt'"
  by (rule rt_fresherI2)
next
from assms(1-2,4) have dhops: "the (dhops rt dip) \leq the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
with assms(2,1) assms(3) [symmetric] show "rt' \sqsubseteq dip rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1 \approx dip rt2"
  and "\[ rt1 \sqsubseteq dip rt2; rt2 \sqsubseteq dip rt1 ] \implies P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
  assumes "rt1 \approx dip rt2"
  shows "rt1 \sqsubseteq dip rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1 \approx dip rt2"
  shows "rt2 \sqsubseteq dip rt1"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1 \approx dip rt2"
  shows "rt2 \approx dip rt1"
using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes "\neg (rt1 \sqsubseteq dip rt2)"
  shows "\neg (rt1 \approx dip rt2)"
proof
  assume "rt1 \approx dip rt2"
  hence "rt1 \sqsubseteq dip rt2" ..
  with \neg (rt1 \sqsubseteq dip rt2) show False ..
qed

lemma not_rt_fresh_asI2 [intro]:
  assumes "\neg (rt2 \sqsubseteq dip rt1)"
  shows "\neg (rt1 \approx dip rt2)"
proof
  assume "rt1 \approx dip rt2"
  hence "rt2 \sqsubseteq dip rt1" ..
  with \neg (rt2 \sqsubseteq dip rt1) show False ..
qed

lemma not_single_rt_fresher [elim]:
  assumes "\neg (the (rt1 ip) \sqsubseteq the (rt2 ip))"
  shows "\neg (rt1 \subseteq ip rt2)"
proof
  assume "rt1 \subseteq ip rt2"
  hence "the (rt1 ip) \subseteq the (rt2 ip)" ..
  with \neg (the (rt1 ip) \subseteq the (rt2 ip)) show False ..
qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_singleI [elim]:
  assumes "\neg (rt1 \subseteq ip rt2)"
  shows "\neg (the (rt1 ip) \subseteq the (rt2 ip))"
proof
assume "the (rt1 ip) ⊑ the (rt2 ip)"
hence "rt1 ⊑_ip rt2" ..
with \neg(rt1 ⊑_ip rt2) show False ..
qed

lemma rt_fresh_as_nsqnr:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "rt1 ≈ dip rt2"
  shows "nsqn r (the (rt2 dip)) = nsqn r (the (rt1 dip))"
using assms unfolding rt_fresh_as_def
by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨dip ∈ kD(rt2)⟩]
kD_nsqn [OF ⟨dip ∈ kD(rt1)⟩]
kD_nsqn [OF ⟨dip ∈ kD(rt2)⟩])

lemma rt_fresher_mapupd [intro!]:
  assumes "dip∈kD(rt)"
  and "the (rt dip) ⊑ r"
  shows "rt ⊑ dip rt(dip ↦→ r)"
using assms unfolding rt_fresher_def
by simp

lemma rt_fresher_map_upd_other [intro!]:
  assumes "dip∈kD(rt)"
  and "dip ≠ ip"
  shows "rt ⊑ dip rt(ip ↦→ r)"
using assms unfolding rt_fresher_def
by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip ∈ kD(rt)"
  and "dip ≠ ip"
  shows "rt ⊑ dip update rt ip r"
using assms unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip∈kD(rt)"
  and "the (dhops rt dip) ≥ 1"
  and "update_arg_wf r"
  shows "rt ⊑ dip update rt ip r"
proof (cases "dip = ip")
  assume "dip ≠ ip" with ⟨dip ∈ kD(rt)⟩ show ?thesis
  by (rule rt_fresher_update_other)
next
  assume "dip = ip"

  from ⟨dip∈kD(rt)⟩ obtain dsn_n dsk_n f_n hops_n nhip_n
    where rtn [simp]: "the (rt dip) = (dsn_n, dsk_n, f_n, hops_n, nhip_n)"
    by (metis prod_cases5)
  with ⟨the (dhops rt dip) ≥ 1⟩ and ⟨dip∈kD(rt)⟩ have "hops_n ≥ 1"
    by (metis proj5_eq_dhops proj5_eq_dhops [symmetric])
  from ⟨dip∈kD(rt)⟩ rtn have [simp]: "sqn rt dip = dsn_n"
    and [simp]: "the (dhops rt dip) = hops_n"
    and [simp]: "the (flag rt dip) = f_n"
    by (simp add: sqn_def proj5_eq_dhops [symmetric]
                proj4_eq_flag [symmetric])
  from ⟨update_arg_wf r⟩ have "((dsn_n, dsk_n, f_n, hops_n, nhip_n)"
    ⊑ the ((update rt dip r) dip)"
    unfolding fresher_def sqn_def by (cases f_n) auto
thus \((dsn_n, dsk_n, f_n, hops_n, nhip_n)\) 
\(\subseteq\) the (update rt dip \((0, unk, val, Suc 0, nhip)\) dip)"

using \(\langle dip \in kD(rt)\rangle\) by - (rule update_cases_kD, simp_all)

next
fix dsn :: sqn and hops nhip
assume "0 < dsn"
show 
\((dsn_n, dsk_n, f_n, hops_n, nhip_n)\) 
\(\subseteq\) the (update rt dip \((dsn, kno, val, hops, nhip)\) dip)
proof (rule update_cases_kD [OF \(\langle dip \in kD(rt)\rangle\)], simp_all add: \(\langle 0 < dsn\rangle\))

unfolding fresher_def by auto

hence "rt \(\subseteq\) dip update rt dip r" 
by - (rule single_rt_fresher, simp)

with \(\langle dip = ip\rangle\) show ?thesis by simp

qed

theorem rt_fresher_invalidate [simp]:
assumes "dip \(\in\) kD(rt)"
and indests: "\(\forall\) rip \(\in\) dom(dests). rip \(\in\) vD(rt) \(\land\) sqn rt rip \(<\) the (dests rip)"
shows "rt \(\subseteq\) dip invalidate rt dests"
proof (cases "dip \(\in\) dom(dests)"
  assume "dip \(\notin\) dom(dests)"
  with \(\langle dip \in kD(rt)\rangle\) have "dip \(\in\) kD(invalidate rt dests)"
  by simp
  ultimately show ?thesis
  unfolding invalidate_def sqn_def
  by - (rule single_rt_fresher, auto simp: fresher_def)

next
assumee "dip \(\in\) dom(dests)"
moreover with indests have "dip \(\in\) vD(rt)"
and "sqn rt dip \(<\) the (dests dip)"
by auto
ultimately show ?thesis
unfolding invalidate_def sqn_def
by - (rule single_rt_fresher, auto simp: fresher_def)

qed

lemma nsqn_r_invalidate [simp]:
assumes "dip \(\in\) kD(rt)"
and "dip \(\in\) dom(dests)"
shows "nsqn r (the (invalidate rt dests dip)) = the (dests dip) - 1"
using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
assumes "dip \(\in\) kD(rt)"
and "\(\forall\) rip \(\in\) dom(dests). rip \(\in\) vD(rt) \(\land\) the (dests rip) = inc (sqn rt rip)"
shows "rt \(\approx\) dip invalidate rt dests"
proof (cases "dip \(\in\) dom(dests)"
  assume "dip \(\notin\) dom(dests)"
  with \(\langle dip \in kD(rt)\rangle\) have "dip \(\in\) kD(invalidate rt dests)"
  by simp
with \( \text{dip} \in kD(rt) \) show \( ?\text{thesis} \)
by rule \( \text{simp\_all add: \( \text{dip} \in \text{dom(dests)} \)} \)

next
assume "\( \text{dip} \in \text{dom(dests)} \)"
with \text{assms(2)} have "\( \text{dip} \in vD(rt) \)"
  and "the (\text{dests dip}) = \text{inc (sqn rt dip)}" by auto
from \( \text{dip} \in vD(rt) \) have "\( \text{dip} \in kD(rt) \)" by simp
moreover then have "\( \text{dip} \in kD(\text{invalidate rt dests}) \)" by simp
ultimately show \( ?\text{thesis} \)
proof
  (rule nsqn\_rt\_fresh\_asI)
from \( \text{dip} \in vD(rt) \) have "nsqn rt dip = sqn rt dip" by simp
also have "\( \text{sqn rt dip} = \text{nsqn, (the (\text{invalidate rt dests dip})} \)"
proof
  from \( \text{dip} \in kD(rt) \) have "nsqn, (the (\text{invalidate rt dests dip})) = the (\text{dests dip}) - 1"
  using \( \text{dip} \in \text{dom(dests)} \) by (rule nsqn\_invalidate)
  with \( \text{the (dests dip)} = \text{inc (sqn rt dip)} \)
  show "\( \text{sqn rt dip} = \text{nsqn, (the (\text{invalidate rt dests dip})} \)" by simp
qed
also from \( \text{dip} \in kD(\text{invalidate rt dests}) \)
have "\( \text{nsqn, (the (\text{invalidate rt dests dip})) = nsqn (\text{invalidate rt dests dip})} \)
  by (simp add: kD\_nsqn)
finally show "\( \text{nsqn rt dip} = \text{nsqn (\text{invalidate rt dests dip})} \)".
qed simp

lemmas rt\_fresher\_inc\_invalidate \[simp\] = rt\_fresh\_as\_inc\_invalidate \[THEN rt\_fresh\_asD1\]

5.5.4 Strictly comparing routing tables

definition rt\_strictly\_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
  "rt\_strictly\_fresher ≡ \( \lambda \text{dip rt1 rt2. (rt1 ⊑ i rt2) ∧ ¬(rt1 ≈ i rt2)} \)"

abbreviation
  rt\_strictly\_fresher\_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊏ _)") \[51, 999, 51\] \[50\]
where
  "rt1 ⊏ \_ rt2 ≡ rt\_strictly\_fresher i \_ rt2"

lemma rt\_strictly\_fresher\_def’’:
  "rt1 ⊏ \_ rt2 = ((rt1 ⊑\_ rt2) ∧ ¬(rt1 ⊑\_ rt1))"
unfolding rt\_strictly\_fresher\_def rt\_fresh\_as\_def by auto

lemma rt\_strictly\_fresher\_I’’ [intro]:
  assumes "rt1 ⊑\_ rt2"
  and "¬(rt2 ⊑\_ rt1)"
  shows "rt1 ⊏\_ rt2"
using \text{assms unfolding rt\_strictly\_fresher\_def’’} by simp

lemma rt\_strictly\_fresher\_E’’ [elim]:
  assumes "rt1 ⊏\_ rt2"
  and "\( [ \text{rt1 ⊑\_ rt2;} ¬(\text{rt2 ⊑\_ rt1}) ] \;⇒ \text{P rt1 rt2 i} \)"
  shows "P rt1 rt2 i"
using \text{assms unfolding rt\_strictly\_fresher\_def’’} by simp

lemma rt\_strictly\_fresher\_I [intro]:
  assumes "rt1 ⊑\_ rt2"
  and "¬(rt1 ≈\_ rt2)"
  shows "rt1 ⊏\_ rt2"
unfolding rt\_strictly\_fresher\_def using \text{assms ..}

lemmas rt\_strictly\_fresher\_single\_I [elim] = rt\_strictly\_fresher\_I [OF single\_rt\_fresher]

lemma rt\_strictly\_fresher\_E [elim]:
  assumes "rt1 ⊏\_ rt2"
  \( \text{assms} \)

447
and \([ rt1 \sqsubseteq_i rt2; \neg (rt1 \approx_i rt2) ] \implies P \ rt1 \ rt2 \ i \]
shows \(" P \ rt1 \ rt2 \ i\"
using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
"rt1 \sqsubseteq_i rt2 =
(nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i))
\lor (nsqn_r (the (rt1 i)) = nsqn_r (the (rt2 i)) \& \pi_5 (the (rt1 i)) > \pi_5 (the (rt2 i))))"
unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
assumes "rt1 \sqsubseteq dip rt2"
shows "(rt1 dip) \sqsubseteq (rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
assumes "rt1 \sqsubseteq dip rt2"
shows "\neg (rt1 \approx dip rt2)"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1 \sqsubseteq dip rt2"
and "rt2 \sqsubseteq dip rt3"
shows "rt1 \sqsubseteq dip rt3"
using assms proof -
from rt1 \sqsubseteq dip rt2 obtain "(the (rt1 dip) \sqsubseteq (the (rt2 dip))" by auto
also from rt2 \sqsubseteq dip rt3 obtain "(the (rt2 dip) \sqsubseteq (the (rt3 dip))" by auto
finally have "(the (rt1 dip) \sqsubseteq (the (rt3 dip))".
moreover have "\neg (rt1 \approx dip rt3)"
proof -
from rt1 \sqsubseteq dip rt2 obtain "\neg (the (rt2 dip) \sqsubseteq (the (rt1 dip))" by auto
also from rt2 \sqsubseteq dip rt3 obtain "\neg (the (rt3 dip) \sqsubseteq (the (rt2 dip))" by auto
finally have "\neg (the (rt3 dip) \sqsubseteq (the (rt1 dip))".
thus ?thesis ..
qed
ultimately show "rt1 \sqsubseteq dip rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: "\neg (rt \sqsubseteq dip rt)"
unfolding rt_strictly_fresher_def by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1 \sqsubseteq dip rt2"
and "rt2 \sqsubseteq dip rt3"
shows "rt1 \sqsubseteq dip rt3"
proof -
from rt1 \sqsubseteq dip rt2 have "rt1 \sqsubseteq dip rt2"
and "\neg (rt2 \sqsubseteq dip rt1)"
unfolding rt_strictly_fresher_def'' by auto
from this(1) and rt2 \sqsubseteq dip rt3 have "rt1 \sqsubseteq dip rt3"..
moreover from "\neg (rt2 \sqsubseteq dip rt1)" have "\neg (rt3 \sqsubseteq dip rt1)"
proof (rule contrapos_nn)
assume "rt3 \sqsubseteq dip rt1"
with rt2 \sqsubseteq dip rt3 show "rt2 \sqsubseteq dip rt1" ..
qed
ultimately show "rt1 \sqsubseteq dip rt3"
unfolding rt_strictly_fresher_def'' by auto
qed
lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
assumes "rt1 ⊑ dip rt2"
and "rt2 ⊑ dip rt3"
shows "rt1 ⊑ dip rt3"
proof -
from rt2 ⊑ dip rt3: have "rt2 ⊑ dip rt3"
and "¬(rt3 ⊑ dip rt2)"
unfolding rt_strictly_fresher_def'' by auto
from rt1 ⊑ dip rt2: and this(1) have "rt1 ⊑ dip rt3" ..
moreover from (¬(rt3 ⊑ dip rt2)) have "¬(rt3 ⊑ dip rt1)"
proof (rule contrapos_nn)
  assume "rt3 ⊑ dip rt1"
  hence "rt2 ⊑ dip rt1" ..
  hence "nsqn rt2 dip ≤ nsqn rt1 dip"
  using ⟨dip ∈ kD(rt2)⟩ ⟨dip ∈ kD(rt1)⟩
  by (rule rt_fresher_imp_nsqn_le)
with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
  by simp
qed
ultimately show "rt1 ⊑ dip rt3"
unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_imp_nsqn_le:
assumes "rt1 ⊑ ip rt2"
and "ip ∈ kD rt1"
and "ip ∈ kD rt2"
shows "nsqn rt1 ip ≤ nsqn rt2 ip"
using assms(1)
by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

lemma rt_strictly_fresher_ltI [intro]:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
and "nsqn rt1 dip < nsqn rt2 dip"
shows "rt1 ⊏ dip rt2"
proof
  from assms show "rt1 ⊑ dip rt2" ..
  next
  show "¬(rt1 ≈ dip rt2)"
  proof
    assume "rt1 ≈ dip rt2"
    hence "rt2 ⊑ dip rt1" ..
    hence "nsqn rt2 dip ≤ nsqn rt1 dip"
      using ⟨dip ∈ kD(rt2)⟩ ⟨dip ∈ kD(rt1)⟩
      by (rule rt_fresher_imp_nsqn_le)
    with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
      by simp
  qed
  qed

lemma rt_strictly_fresher_eqI [intro]:
assumes "i∈kD(rt1)"
and "i∈kD(rt2)"
and "nsqn rt1 i = nsqn rt2 i"
and "π₅(the (rt2 i)) < π₅(the (rt1 i))"
shows "rt1 ⊏ i rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]: "dests dip rt rt'. dests dip = None ⟷ (invalidate rt dests ⊑ dip rt') = (rt ⊑ dip rt')"
unfolding invalidate_def rt_strictly_fresher_def' by (rule iffl) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
assumes "dip ∈ vD(invalidate rt1 dests)"
shows "(invalidate rt1 dests ⊑ dip rt2) = (rt1 ⊑ dip rt2)"
proof (cases "dip ∈ dom(dests)"
assumption "dip ∈ dom(dests)"
  hence "dip ∉ vD(invalidate rt1 dests)"
    unfolding invalidate_def vD_def
    by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
  with ⟨dip ∈ vD(invalidate rt1 dests)⟩ show ?thesis by simp
next
  assume "dip ∉ dom(dests)"
  hence "dests dip = None" by auto
  moreover with ⟨dip ∈ vD(invalidate rt1 dests)⟩ have "dip ∈ vD(rt1)"
    unfolding invalidate_def vD_def
    by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
  ultimately show ?thesis unfolding invalidate_def rt_strictly_fresher_def' by auto
qed

lemma rt_strictly_fresher_update_other [elim!]:
"dip ip rt rt'. [ dip ≠ ip; rt ⊏ dip rt' ] ⇒ update rt ip r ⊏ dip rt'"
unfolding rt_strictly_fresher_def' by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
assumes "dip ∈ vD (rt2 nhip)"
and *: "osn < sqn (rt2 nhip) dip"
and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip)"
shows "update rt dip (osn, kno, val, hops, nhip) ⊏ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
  from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip)) dip = osn"
    by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
    have "nsqn_r (the (update rt dip (osn, kno, val, hops, nhip) dip)) = osn"
      by (simp add: kD_nsqn)
  also have "osn < sqn (rt2 nhip) dip" by (rule *)
  also have "sqn (rt2 nhip) dip = nsqn_r (the (rt2 nhip dip))"
    unfolding nsqn_r_def using ⟨dip ∈ vD (rt2 nhip)⟩
    by (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqn_r (the (update rt dip (osn, kno, val, hops, nhip) dip)) < nsqn_r (the (rt2 nhip dip))".
qed

lemma dhops_le_hops_imp_update_strictly_fresher:
assumes "dip ∈ vD(rt2 nhip)"
and sqn: "sqn (rt2 nhip) dip = osn"
and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip)"
shows "update rt dip (osn, kno, val, Suc hops, nhip) ⊏ dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
  from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip)) dip = osn"
    by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
    have "nsqn_r (the (update rt dip (osn, kno, val, Suc hops, nhip) dip)) = osn"
      by (simp add: kD_nsqn)
  also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
  also have "sqn (rt2 nhip) dip = nsqn_r (the (rt2 nhip dip))"
    unfolding nsqn_r_def using ⟨dip ∈ vD(rt2 nhip)⟩
    by (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqn_r (the (update rt dip (osn, kno, val, Suc hops, nhip) dip)) = nsqn_r (the (rt2 nhip dip))".
next
  have "the (dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
  also have "hops < hops + 1" by simp
  also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip)) dip)"
    using ** by simp
  finally show "update rt dip (osn, kno, val, Suc hops, nhip) ⊏ dip rt2 nhip".
finally have \( \text{the (dhops (rt2 nhip) dip) < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip)) dip)} \).
thus \( \pi_5 (\text{the (rt2 nhip dip)}) < \pi_5 (\text{the (update rt dip (osn, kno, val, Suc hops, nhip) dip)}) \)
using \( \langle \text{dip} \in vD(rt2 nhip) \rangle \) by \( \text{(simp add: proj5_eq_dhops)} \)
qd

lemma nsqn_invalidate:
assumes \( \text{dip} \in kD(rt) \)
and \( \forall \text{ip} \in \text{dom(dests)}. \text{ip} \in vD(rt) \land \text{the (dests ip) = inc (sqn rt ip)} \)
shows \( \text{nsqn (invalidate rt dests) dip = nsqn rt dip} \)
proof -
from \( \langle \text{dip} \in kD(rt) \rangle \) have \( \text{dip} \in kD(invalidate rt dests) \)
by \( \text{(simp add: proj5_eq_dhops)} \)
from assms have \( \text{rt fresh as inc invalidate} \)
with \( \langle \text{dip} \in kD(rt) \rangle \langle \text{dip} \in kD(invalidate rt dests) \rangle \) show ?thesis
by \( \text{(simp add: kD_nsqn del: invalidate_kD_inv)} \)
(erule(2) \( \text{rt fresh as nsqnr} \))
qd

end

5.6 Invariant proofs on individual processes

theory E_Seq_Invariants
imports AWN.Invariants E_Aodv E_Aodv_Data E_Aodv_Predicates E_Fresher
begin

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

lemma sequence_number_increases:
\( \text{paodv i} |\text{-} A onll \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). sn \xi \leq sn \xi') \)
by \( \text{inv_cterms} \)

lemma sequence_number_one_or_bigger:
\( \text{paodv i} |\text{-} A \Gamma_{AODV} (\lambda(\xi, _.). 1 \leq sn \xi) \)
by \( \text{(rule onll_step_to_invariantI [OF sequence_number_increases])} \)
(\( \text{auto simp: } \sigma_{AODV\_def} \))

We can get rid of the onl/onll if desired...

lemma sequence_number_increases':
\( \text{paodv i} |\text{-} A (\lambda((\xi, _), _, (\xi', _)). sn \xi \leq sn \xi') \)
by \( \text{(rule step_invariant_weakenE [OF sequence_number_increases])} \)
(\( \text{auto dest!: onllD} \))

lemma sequence_number_one_or_bigger':
\( \text{paodv i} |\text{-} A (\lambda(\xi, _.). 1 \leq sn \xi) \)
by \( \text{(rule invariant_weakenE [OF sequence_number_one_or_bigger])} \)

lemma sip_in_kD:
\( \text{paodv i} |\text{-} A \Gamma_{AODV} (\lambda(\xi, 1). l \in (\{PAodv-:7\} \cup \{PAodv-:5\} \cup \{PRrep-:0..PRrep-:4\} \cup \{PRreq-:0..PRreq-:3\}) \rightarrow \text{sip } \xi \in kD (rt \xi) ) \)
by \( \text{inv_cterms} \)

Proposition 7.38

lemma includes_nhip:
\( \text{paodv i} |\text{-} A \Gamma_{AODV} (\lambda(\xi, 1). \forall \text{dip} \in kD(rt \xi). \text{the (nhop (rt \xi) dip)} \in kD(rt \xi)) \)
proof -
\{ \text{fix ip and } \xi ' :: state} \nassume "\( \forall \text{dip} \in kD (rt \xi). \text{the (nhop (rt \xi) dip)} \in kD (rt \xi) "
and "\( \xi ' = [rt := \text{update (rt \xi) ip (0, unk, val, Suc 0, ip)}] \)"
\( \text{hence } "\forall \text{dip} \in kD (rt \xi). \)" 
the (nhop (update (rt \xi) ip (0, unk, val, Suc 0, ip)) dip) = ip
\( \forall \text{ the \( (\text{nhop \ (update \ (rt \ (rt) \ ip \ (0, \ \text{unk}, \ val, \ Suc \ 0, \ ip)) \ dip) \in kD \ (rt \ (rt))) \) by \ clarsimp \ (metis \ \text{nhop\_update\_unk\_val \ update\_another}) \) \}

\} \ \text{note one\_hop = this}

\{ \ \text{fix \ ip \ sip \ sn \ hops and \( (\xi', \ : \ : \ : \ : \ state) \) assume "\( \forall \text{ dip} \in kD \ (rt \ (rt)). \ \text{the \( (\text{nhop \ (rt \ (rt) \ dip) \in kD \ (rt \ (rt))) \)" and "(the \( (\text{nhop \ (rt \ (rt) \ ip \ (sn, \ kno, \ val, \ Suc \ hops, \ sip)) \ip) = ip \) ∨ \text{ the \( (\text{nhop \ (rt \ (rt) \ ip \ (sn, \ kno, \ val, \ Suc \ hops, \ sip)) \ip) \in kD \ (rt \ (rt))) \)" hence "\( (\text{the \( (\text{nhop \ (update \ (rt \ (rt) \ ip \ (sn, \ kno, \ val, \ Suc \ hops, \ sip)) \ip) = ip \) ∨ \text{ the \( (\text{nhop \ (update \ (rt \ (rt) \ ip \ (sn, \ kno, \ val, \ Suc \ hops, \ sip)) \ip) \in kD \ (rt \ (rt))) \)" by \ (metis \ \text{kD\_update\_unchanged \ nhop\_update\_changed \ update\_another}) \)"} \}

\} \ \text{note nhip\_is\_sip = this}

show \ ?thesis by \ (inv\_cterms \ inv \ add: \ \text{onl\_invariant\_sterms} \ [OF \ \text{aodv\_wf \ sip\_in\_kD}] \ solve: \ \text{one\_hop \ nhip\_is\_sip}) \qed

Proposition 7.4

lemma known\_destinations\_increase:

"\( \text{paodv \ i \ \vdash \ A \ \Gamma_{AODV} (\lambda((\xi, \ _), \ _, \ (\xi', \ _)). \ \text{kD \ (rt \ (rt)) \ \subseteq \ kD \ (rt \ (rt'))) \)" by \ (inv\_cterms \ simp \ add: \ subset\_insertI)

Proposition 7.5

lemma rreqs\_increase:

"\( \text{paodv \ i \ \vdash \ A \ \Gamma_{AODV} (\lambda((\xi, \ _), \ _, \ (\xi', \ _)). \ \text{rreqs} \ \xi \ \subseteq \ \text{rreqs} \ \xi') \)" by \ (inv\_cterms \ simp \ add: \ subset\_insertI)

lemma dests\_bigger\_than\_sqn:

"\( \text{paodv \ i \ \vdash \ \text{onl} \ \Gamma_{AODV} (\lambda(\xi, \ l). \ l \in \{\text{PAodv-:15..PAodv-:17}\} \cup \{\text{PPkt-:7..PPkt-:9}\} \cup \{\text{PRreq-:11..PRreq-:13}\} \cup \{\text{PRreq-:20..PRreq-:22}\} \cup \{\text{PRrep-7..PRrep-9}\} \cup \{\text{PRerr-1..PRerr-4}\} \cup \{\text{PRerr-6}\} \rightarrow (\forall \text{ ip} \in \text{dom(dests} \ \xi)). \ \text{ip} \in \text{kD}\ (rt \ (rt) \ \wedge \ \text{sqn} \ (rt \ (rt) \ \text{ip} \leq \ \text{the} \ (\text{dests} \ \xi \ \text{ip}))) \)"

proof -

have sqninv:

"\( \forall \text{ \dests \ rt \ rsn \ ip}. \text{ [ \ \forall \text{ip} \in \text{dom(dests)\ . \ ip} \in \text{kD}(rt) \ \wedge \ \text{sqn} \ (rt \ \text{ip} \leq \ \text{the} \ (\text{dests} \ \text{ip}) \ ; \ \text{dests} \ \text{ip} = \text{Some} \ \text{rsn] \ \rightarrow \text{sqn} \ (\text{invalidate} \ \text{rt dests}) \ \text{ip} \ \leq \ \text{rsn} \) \)"

by \ (rule \ \text{sqn\_invalidate\_in\_dests} \ [THEN \ \text{eq\_imp\_le}], \ \text{assumption}) \ \text{auto}

have indests:

"\( \forall \text{ \dests \ rt \ rsn \ ip}. \text{ [ \ \forall \text{ip} \in \text{dom(dests)\ . \ ip} \in \text{kD}(rt) \ \wedge \ \text{sqn} \ (rt \ \text{ip} \leq \ \text{the} \ (\text{dests} \ \text{ip}) \ ; \ \text{dests} \ \text{ip} = \text{Some} \ \text{rsn] \ \rightarrow \text{ip} \in \text{kD}(rt) \ \wedge \ \text{sqn} \ (rt \ \text{ip} \leq \ \text{rsn} \) \)"

by \ (metis \ \text{domI} \ \text{option.sel})

show \ ?thesis

by \ \text{inv\_cterms}

\ (\text{clarsimp} \ \text{split: \ \text{if\_split\_asm} \ \text{option\_split\_asm} \ \text{elim!}: \ \text{sqninv} \ \text{indests}))+

\qed

Proposition 7.6

lemma sqns\_increase:

"\( \text{paodv \ i \ \vdash \ A \ \Gamma_{AODV} (\lambda((\xi, \ _), \ _, \ (\xi', \ _)). \ \forall \text{ \ip}. \ \text{sqn} \ (rt \ (rt) \ \text{ip} \leq \ \text{sqn} \ (rt \ (rt') \ \text{ip}) \)" proof -

\{ \ \text{fix \ \xi \ as \ state}

assume \*: "\( \forall \text{ \ip} \in \text{dom(dests} \ \xi). \ \text{ip} \in \text{kD} \ (rt \ (rt) \ \wedge \ \text{sqn} \ (rt \ (rt) \ \text{ip} \leq \ \text{the} \ (\text{dests} \ \text{ip}) \)"

have "\( \forall \text{ \ip}. \ \text{sqn} \ (rt \ (rt) \ \text{ip} \leq \ \text{sqn} \ (\text{invalidate} \ \text{rt} \ \text{dests}) \ \text{ip}) \)"

proof

fix \ \ip

from \* \ have "\( \text{ip} \notin \text{dom(dests} \ \xi) \ \vee \ \text{sqn} \ (rt \ (rt) \ \text{ip} \leq \ \text{the} \ (\text{dests} \ \xi \ \text{ip}) \)" by \ \text{simp

452
thus "sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip"
by (metis domI invalidate_sqn option.sel)
qed

} note solve_invalidate = this
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn]
simp add: solve_invalidate)
qed

Proposition 7.7

lemma ip_constant:
"paodv i |= onl Γ_AODV (λ(ξ, _.). ip ξ = i)"
by (inv_cterms simp add: σAODV_0_def)

Proposition 7.8

lemma sender_ip_valid':
"paodv i |=A onll Γ_AODV (λ((ξ, _), a, _). anycast (λm. not_Pkt m −→ msg_sender m = ip ξ)) a)"
by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
(auto dest!: onlD onllD)

lemma sender_ip_valid:
"paodv i |=A onll Γ_AODV (λ((ξ, _), a, _). anycast (λm. not_Pkt m −→ msg_sender m = i) a)"
by (inv_cterms)

Proposition 7.9

lemma received_msg_inv:
"paodv i |= (recvmsg P →) onl Γ_AODV (λ(ξ, l). l ∈ {PAodv-:1} → P (msg ξ))"
by (inv_cterms)

Proposition 7.10

lemma hop_count_positive:
"paodv i |= onl Γ_AODV (λ(ξ, _.). ∀ip∈kd (rt ξ). the (dhops (rt ξ) ip) ≥ 1)"
by (inv_cterms)

lemma rreq_dip_in_vD_dip_eq_ip:
"paodv i |= onl Γ_AODV (λ(ξ, 1). (l ∈ {PRreq-:16..PRreq-:17} → dip ξ ∈ vD(rt ξ))
∧ (l ∈ {PRreq-:6, PRreq-:7} → dip ξ = ip ξ)
∧ (l ∈ {PRreq-:15..PRreq-:17} → dip ξ = ip ξ))"
by (inv_cterms)

lemma rrep_dip_in_vD:
"paodv i |= onl Γ_AODV (λ(ξ, 1). (l ∈ {PRrep-:4} → dip ξ ∈ vD(rt ξ)))"
by (inv_cterms)

Proposition 7.11

lemma anycast_msg_zhops:
"⋀ rreqid dip dsn dsk oip osn sip.
paodv i |=A onll Γ_AODV (λ(ξ, a, _. anycast msg_zhops a)"
proof (inv_cterms inv add:
lemma hop_count_zero_oip_dip_sip:

"paodv i = (recvmsg msg_zhops \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, 1).
(1 \in \{Paodv=-4..Paodv=-5\} \cup \{PRreq=n\n. True\} \rightarrow
(hops \xi = 0 \rightarrow oip \xi = sip \xi))
∧
((1 \in \{Paodv=-6..Paodv=-7\} \cup \{PRrep=-n\n. True\} \rightarrow
(hops \xi = 0 \rightarrow dip \xi = ip \xi)))"

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:

"paodv i = (recvmsg rreq_rrep_sn \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, 1).
1 \in \{Paodv=-4, Paodv=-5\} \cup \{PRreq=-n\n. True\} \rightarrow 1 \leq osn \xi)"

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq':

"paodv i = (recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, 1).
1 \in \{Paodv=-4, Paodv=-5\} \cup \{PRreq=-n\n. True\} \rightarrow 1 \leq osn \xi)"

proof (rule invariant_weakenE [OF osn_rreq])

fix a

assume "recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) a"

thus "recvmsg rreq_rrep_sn a"
  by (cases a) simp_all

qed

lemma dsn_rrep:

"paodv i = (recvmsg rreq_rrep_sn \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, 1).
1 \in \{Paodv=-6, Paodv=-7\} \cup \{PRrep=-n\n. True\} \rightarrow 1 \leq dsn \xi)"

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp
lemma dsn_rrep':
"paodv i \models (recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, l).
\begin{align*}
&1 \in \{\text{PAodv-:6, PAodv-:7}\} \cup \{\text{PRrep-:n|n. True}\} \rightarrow 1 \leq dsn \xi
\end{align*}
"
proof (rule invariant_weakenE [OF dsn_rrep])
fix a
assume "recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) a"
thus "recvmsg rreq_rrep_sn a"
by (cases a) simp_all
qed

lemma hop_count_zero_oip_dip_sip':
"paodv i \models (recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, l).
\begin{align*}
&\forall dip \in kD(rt \xi). (sqn (rt \xi) dip = 0 \rightarrow sqnf (rt \xi) dip = unk) \\
&\land (sqnf (rt \xi) dip = unk \rightarrow the (dhops (rt \xi) dip) = 1) \\
&\land (the (dhops (rt \xi) dip) = 1 \rightarrow the (nhop (rt \xi) dip) = dip))
\end{align*}
"
proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
fix a
assume "recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) a"
thus "recvmsg msg_zhops a"
by (cases a) simp_all
qed

Proposition 7.12

lemma zero_seq_unk_hops_one':
"paodv i \models (recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) \rightarrow) \text{ onl } \Gamma_{AODV} (\lambda(\xi, _).
\begin{align*}
&\forall dip \in kD(rt \xi). (sqn (rt \xi) dip = 0 \rightarrow sqnf (rt \xi) dip = unk) \\
&\land (sqnf (rt \xi) dip = unk \rightarrow the (dhops (rt \xi) dip) = Suc 0) \\
&\land (the (dhops (rt \xi) dip) = Suc 0 \rightarrow the (nhop (rt \xi) dip) = dip))
\end{align*}
"
proof -
{ fix dip and \xi :: state and P
assume "sqn (invalidate (rt \xi) (dests \xi)) dip = 0"
and all: "\forall ip. sqn (rt \xi) ip \leq sqn (invalidate (rt \xi) (dests \xi)) ip"
and *: "sqn (rt \xi) dip = 0 \Rightarrow P \xi dip"
have "P \xi dip"
proof -
from all have "sqn (rt \xi) dip \leq sqn (invalidate (rt \xi) (dests \xi)) dip" ..
with \sqn (invalidate (rt \xi) (dests \xi)) dip = 0 have "sqn (rt \xi) dip = 0" by simp
thus "P \xi dip" by (rule *)
qed
}
note sqn_invalidate_zero [elim!] = this

{ fix dsn hops :: nat and sip oip rt and dip :: ip
assume \text{\forall dip} \in kD(rt).
\begin{align*}
&\text{(sqn rt dip = 0 } \rightarrow \pi_3(\text{the (rt dip)) = unk)} \land \\
&\text{(\pi_3(\text{the (rt dip)) = unk } \rightarrow \text{the (dhops rt dip) = Suc 0}}) \land \\
&\text{(the (dhops rt dip) = Suc 0 } \rightarrow \text{the (nhop rt dip) = dip})
\end{align*}
and "hops = 0 \rightarrow sip = dip"
and "Suc 0 \leq dsn"
and "ip \neq dip \rightarrow ip \in kD(rt)"
hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip)) ip) = Suc 0 \rightarrow 
the (nhop (update rt dip (dsn, kno, val, Suc hops, sip)) ip) = ip"
by - (rule update_cases, auto simp add: sqn_def dest!: bepec)
}
note prreq-ok1 [simp] = this

{ fix ip dsn hops sip oip rt dip
assume \text{\forall dip} \in kD(rt).
\begin{align*}
&\text{(sqn rt dip = 0 } \rightarrow \pi_3(\text{the (rt dip)) = unk)} \land \\
&\text{(\pi_3(\text{the (rt dip)) = unk } \rightarrow \text{the (dhops rt dip) = Suc 0}}) \land \\
&\text{(the (dhops rt dip) = Suc 0 } \rightarrow \text{the (nhop rt dip) = dip})
\end{align*}
and "Suc 0 \leq dsn"
and "ip \neq dip \rightarrow ip \in kD(rt)"
hence "\(\pi_3(\text{the (update rt dip (dsn, kno, val, Suc hops, sip) ip)}) = \text{unk} \rightarrow \text{the (dhops (update rt dip (dsn, kno, val, Suc hops, sip)) ip}) = \text{Suc 0}\)"
by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
}

note prreq_ok2 [simp] = this
{
fix ip dsn hops sip oip rt dip
assume "\("sqn rt dip = 0 \rightarrow \pi_3(\text{the (rt dip)}) = \text{unk}\) ∧
(\(\pi_3(\text{the (rt dip)}) = \text{unk} \rightarrow \text{the (dhops rt dip)} = \text{Suc 0}\) ∧
(\(\text{the (dhops rt dip)} = \text{Suc 0} \rightarrow \text{the (nhop rt dip)} = \text{dip}\)"
and "\(\text{Suc 0} \leq \text{dsn}\)"
and "\(\text{ip} \neq \text{dip} \rightarrow \text{ip} \in \text{kD(rt)}\)"
hence "\(\text{sqn (update rt dip (dsn, kno, val, Suc hops, sip) ip)} = 0 \rightarrow \pi_3(\text{the (update rt dip (dsn, kno, val, Suc hops, sip) ip)}) = \text{unk}\)"
by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
}

note prreq_ok3 [simp] = this
{
fix rt sip
assume "\("sqn (update rt sip (0, unk, val, Suc 0, sip) sip) = 0 \rightarrow \pi_3(\text{the (update rt sip (0, unk, val, Suc 0, sip) sip)}) = \text{unk}\) ∧
(\(\pi_3(\text{the (update rt sip (0, unk, val, Suc 0, sip) sip)}) = \text{unk} \rightarrow \text{the (dhops (update rt sip (0, unk, val, Suc 0, sip) sip)) = Suc 0}\) ∧
(\(\text{the (dhops (update rt sip (0, unk, val, Suc 0, sip) sip)) = Suc 0} \rightarrow \text{the (nhop (update rt sip (0, unk, val, Suc 0, sip) sip)) = dip}\)"
by - (rule update_cases, simp_all add: sqnf_def sqn_def)
}

note prreq_ok4 [simp] = this
have prreq_ok5 [simp]: "\(\forall \text{dip rt}. \pi_3(\text{the (update rt sip (0, unk, val, Suc 0, sip) sip)}) = \text{unk} \rightarrow \text{the (dhops (update rt sip (0, unk, val, Suc 0, sip) sip)) = Suc 0}\)"
by (rule update_cases) simp_all

have prreq_ok6 [simp]: "\(\forall \text{dip rt}. \text{sqn (update rt sip (0, unk, val, Suc 0, sip) sip) = 0} \rightarrow \pi_3(\text{the (update rt sip (0, unk, val, Suc 0, sip) sip)}) = \text{unk}\)"
by (rule update_cases) simp_all

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip'] seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans] onl_invariant_sterms [OF aodv_wf osn_rreq'] onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+

qed

lemma zero_seq_unk_hops_one:
"paodv i ||= (recvmsg (\(\lambda m. rreq_rrep_sn m \land msg_zhops m\)) \rightarrow onl \Gamma_{AODV} (\lambda(\xi, _). \forall \text{dip \in kD(rt \xi)}. \text{sqn (rt \xi) dip = 0} \rightarrow \text{sqnf (rt \xi) dip = unk} \land \text{the (dhops (rt \xi) dip) = Suc 0} \land \text{the (nhop (rt \xi) dip) = dip}))" 
by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
"paodv i ||= (recvmsg (\(\lambda rreq_rrep_sn \rightarrow\)) onl \Gamma_{AODV} (\lambda(\xi, 1). \forall \text{dip \in kD(rt \xi)}. \pi_3(\text{(rt \xi dip)}) = \text{unk} \lor 1 \leq \pi_2(\text{(rt \xi dip)}))"
proof -

{ fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
assume "\(\text{dsk1 = unk} \lor \text{Suc 0} \leq \text{dsn2}\)"
hence "\(\pi_3(\text{(update rt sip (dsn1, dsk1, flag1, hops1, nhip1) sip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2) sip)}\)"

456
unfolding update_def by (cases "dsk1 = unk") (clarsimp split: option.split)+
}
note fromsip [simp] = this

{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
  assume allkd: "\(\forall dip \in kD(rt). \pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn rt dip}\)"
  and **: "dsk1 = unk \lor \text{Suc 0} \leq \text{dsn2}"
  have "\(\forall dip \in kD(rt). \pi_3(\text{the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1) dip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2)) dip}\)"
  (is "\(\forall dip \in kD(rt). \text{?prop dip}\)"
  proof
    fix dip
    assume "dip \in kD(rt)"
    thus "\(\text{?prop dip}\)"
    proof (cases "dip = sip")
      assume "dip = sip"
      with ** show ?thesis
      by simp
    next
      assume "dip \neq sip"
      with \(dip \in kD(rt)\) allkd show ?thesis
      by simp
    qed
  qed
}
note solve_update [simp] = this

{ fix dip rt dests
  assume *: "\(\forall ip \in \text{dom(dests)}. ip \in kD(rt) \land \text{sqn rt ip} \leq \text{the (dests ip)}\)"
  and **: "\(\forall ip \in kD(rt). \pi_3(\text{the (rt ip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn rt ip}\)"
  have "\(\forall dip \in kD(rt). \pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn (invalidate rt dests) dip}\)"
  proof
    fix dip
    assume "dip \in kD(rt)"
    with ** have "\(\pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn rt dip}\)" ..
    thus "\(\pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn (invalidate rt dests) dip}\)"
    proof
      assume "\(\pi_3(\text{the (rt dip)}) = \text{unk}\) thus \(\text{?thesis}..\)
    next
      assume "\(\text{Suc 0} \leq \text{sqn rt dip}\)
      have "\(\text{Suc 0} \leq \text{sqn (invalidate rt dests) dip}\)"
      proof (cases "dip \in \text{dom(dests)}")
        assume "dip \in \text{dom(dests)}"
        with * have "\(\text{sqn rt dip} \leq \text{the (dests dip)}\)" by simp
        with \(\text{Suc 0} \leq \text{sqn rt dip}\) have "\(\text{Suc 0} \leq \text{the (dests dip)}\)" by simp
        with \(dip \in \text{dom(dests)}\) dip \in kD(rt) [THEN kD_Some] show ?thesis
        unfolding invalidate_def sqn_def by auto
      next
      with \(\text{Suc 0} \leq \text{sqn rt dip} \lor dip \in kD(rt)\) [THEN kD_Some] show ?thesis
      unfolding invalidate_def sqn_def by auto
    qed
    thus ?thesis by (rule disjI2)
  qed
}
note solve_invalidate [simp] = this

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn
  [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aodv_wf osn_rreq]
    onl_invariant_sterms [OF aodv_wf dsn_rrep]
    simp add: proj3_inv proj2_eq_sqn)
qed

Proposition 7.13
Lemma rreq_rrep_sn_any_step_invariant:
"paodv i |=_A (recvmsg rreq_rrep_sn →) onll Γ_AODV (λ(_, a, _). anycast rreq_rrep_sn a)"
Proof -

have sqnf_kno: "paodv i |= onll Γ_AODV (λ(ξ, 1).
(1 ∈ {PRreq:-16} → dip ξ ∈ kD (rt ξ) ∧ sqn (rt ξ) (dip ξ) = kno))" by (inv_cterms)

have rrep_sqn_greater_dsn: "paodv i |= (recvmsg rreq_rrep_sn →) onl Γ_AODV (λ(ξ, l).
(l ∈ {PRrep-:1 .. PRrep-:4} → 1 ≤ sqn (rt ξ) (dip ξ)))" by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
(clarsimp simp: update_kno_dsn_greater_zero [simplified])

show ?thesis by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf sequence_number_one_or_bigger [THEN invariant_restrict_inD]
onl_invariant_sterms [OF aodv_wf kD_unk_or_atleast_one]
onl_invariant_sterms_TT [OF aodv_wf sqnf_kno]
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
onl_invariant_sterms [OF aodv_wf rrep_sqn_greater_dsn])

(auto simp: proj2_eq_sqn)

qed

Proposition 7.14

Lemma rreq_rrep_fresh_any_step_invariant:
"paodv i |= onll Γ_AODV (λ(ξ, 1). anycast (rreq_rrep_fresh (rt ξ)) a)"
Proof -

have rrep_prrep: "paodv i |= onll Γ_AODV (λ(ξ, 1).
(1 ∈ {PRreq:-15, PRreq-:24, PRreq-:26} → oip ξ ∈ kD(rt ξ)
∧ (sqn (rt ξ) (oip ξ) > (osn ξ)
∨ (sqn (rt ξ) (oip ξ) = (osn ξ)
∧ the (dhops (rt ξ) (oip ξ)) ≤ Suc (hops ξ)
∧ the (flag (rt ξ) (oip ξ)) = val))))" unfolding update_def by (clarsimp split: option.split)

(auto simp: proj2_eq_sqn)

qed
∧ the (dhops (rt ξ) (oip ξ)) ≤ Suc (hops ξ)
∧ the (flag (rt ξ) (oip ξ) = val)))))))

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip])
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip]
onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip]
onl_invariant_sterms [OF aodv_wf rrep_prrep]
onl_invariant_sterms [OF aodv_wf rreq_dip_kD_oip_sqn]
onl_invariant_sterms [OF aodv_wf rreq_dip_kD_oip_sqn])

qed

Proposition 7.15

lemma rerr_invalid_any_step_invariant:
"paodv i ||= onll Γ AODV (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a)"
proof -
  have dests_inv: "paodv i ||= onll Γ AODV (λ((ξ, l). (l ∈ {PAodv-:15, PPkt-:7, PRreq-:11, PRreq-:20, PRrep-:7, PRerr-:1}
                  ∨ (l = PPkt-:12...PAodv-:17)
                  ∨ (l ∈ {PRreq-:12...PRreq-:13}
                  ∨ (l = PRrep-:8...PRrep-:9)
                  ∨ (l = PRerr-:2...PRerr-:4) → (∀ip ∈ dom(dests ξ). ip ∈ vD(rt ξ))
                  ∧ (l = PRreq-:12 → dip ξ ∈ iD(rt ξ))))" by
  inv_cterms (clarsimp split: if_split_asm option.split_asm simp: domIff)+
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn])

qed

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs (rip, rsn) is assigned to the variable dests of type ip → sqn, or to the first argument of the function rerr, this set is a partial function, i.e., there is at most one entry (rip, rsn) for each destination rip: guaranteed by type.

lemma dests_vD_inc_sqn:
"paodv i ||= onll Γ AODV (λ((ξ, l). (l ∈ {PAodv-:15, PPkt-:7, PRreq-:11, PRreq-:20, PRrep-:7, PRerr-:1}
                  → (∀ip ∈ dom(dests ξ). ip ∈ vD(rt ξ))
                  ∧ (l = PPkt-:12 → dip ξ ∈ iD(rt ξ))))" by
inv_cterms (clarsimp split: if_split_asm option.split_asm simp: domIff)+

Proposition 7.27

lemma route_tables_fresher:
"paodv i ||=A (recvmsg rreq_rrep_sn → onll Γ AODV (λ(((ξ, _), _), (ξ', _)). ∀ dip ∈ kD(rt ξ). rt ξ ⊆ dip rt ξ'))"
proof (inv_cterms inv add:
onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
onl_invariant_sterms [OF aodv_wf invariant_restrict_inD])
5.7 The quality increases predicate

theory E_Quality_Increases
imports E_Aodv_Predicates E_Fresher
begin

definition quality_increases :: "state ⇒ state ⇒ bool"
where "quality_increases ξ ξ' ≡ (∀ dip ∈ kD(rt ξ)). dip ∈ kD(rt ξ') ∧ rt ξ ⊑ dip rt ξ' ∧ (∀ dip sqn (rt ξ) dip ≤ sqn (rt ξ') dip)"

lemma quality_increasesI [intro!]:
assumes "∀ dip. dip ∈ kD(rt ξ) ⇒ dip ∈ kD(rt ξ')" and "∀ dip. dip ∈ kD(rt ξ') ⇒ rt ξ ⊑ dip rt ξ'" and "∀ dip sqn (rt ξ) dip ≤ sqn (rt ξ') dip"
shows "quality_increases ξ ξ'"

unfolding quality_increases_def using assms by clarsimp

lemma quality_increasesE [elim]:
fixes dip
assumes "quality_increases ξ ξ'" and "dip ∈ kD(rt ξ)" and "[ dip ∈ kD(rt ξ'). rt ξ ⊑ dip rt ξ'; sqn (rt ξ) dip ≤ sqn (rt ξ') dip ] ⇒ R dip ξ ξ'"
shows "R dip ξ ξ'"

using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rt_fresherD [dest]:
fixes ip
assumes "quality_increases ξ ξ'"
and "ip∈kD(rt ξ)"
shows "rt ξ ⊑ ip rt ξ'"
using assms by auto

lemma quality_increases_sqnE [elim]:
fixes dip
assumes "quality_increases ξ ξ'"
and "sqn (rt ξ) dip ≤ sqn (rt ξ') dip ⇒ R dip ξ ξ'"
shows "R dip ξ ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases ξ ξ"
by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
fixes σ σ' dip
assumes "rt (σ i) ⊏ dip rt (σ' nhip)"
and qinc: "quality_increases (σ nhip) (σ' nhip)"
and "dip ∈ kD(rt (σ nhip))"
shows "rt (σ i) ⊏ dip rt (σ' nhip)"
proof -
  from qinc have "rt (σ nhip) ⊑ dip rt (σ' nhip)" using ⟨dip ∈ kD(rt (σ nhip))⟩
  by auto
  with ⟨rt (σ i) ⊏ dip rt (σ nhip)⟩ show ?thesis ..
qed

lemma kD_quality_increases [elim]:
assumes "i∈kD(rt ξ)"
and "quality_increases ξ ξ'"
shows "i∈kD(rt ξ')"
using assms by auto

lemma kD_nsqn_quality_increases [elim]:
assumes "i∈kD(rt ξ)"
and "quality_increases ξ ξ'"
sqn (rt ξ) i ≤ sqn (rt ξ') i
proof -
  from assms have "i∈kD(rt ξ')" ..
  moreover with assms have "rt ξ ⊑ i rt ξ'" by auto
  ultimately have "sqn (rt ξ) i ≤ sqn (rt ξ') i"
  using ⟨i∈kD(rt ξ)⟩ by (erule(2) rt_fresher_imp_nsqn_le)
  with ⟨i∈kD(rt ξ')⟩ show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
assumes "i∈kD(rt ξ)"
and "quality_increases ξ ξ'"
sqn (rt ξ) i ≤ sqn (rt ξ') i
using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
assumes "i∈kD(rt ξ)"
and "s ≤ nsqn (rt ξ) i"
and "quality_increases ξ ξ'"
sqn (rt ξ') i ≤ nsqn (rt ξ') i
proof
  from ⟨i∈kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ show "i∈kD(rt ξ')" ..
next
  from ⟨i∈kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with ⟨s ≤ nsqn (rt ξ) i⟩ show "s ≤ nsqn (rt ξ') i" by (rule le_trans)
qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality_increases ξ ξ'"
and "s < nsqn (rt ξ) i"
shows "s < nsqn (rt ξ') i"
proof -
from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
with s < nsqn (rt ξ) i: show "s < nsqn (rt ξ') i" by simp
qed

lemma nsqn_quality_increases_dhops [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality_increases ξ ξ'"
and "nsqn (rt ξ) i = nsqn (rt ξ') i"
shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
using assms unfolding quality_increases_def
by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
assumes "i ∈ kD(rt ξ)"
and "quality_increases ξ ξ'"
and "s = nsqn (rt ξ) i"
shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

lemma quality_increases_rreq_rrep_props [elim]:
fixes sn ip hops sip
assumes qinc: "quality_increases (σ sip) (σ' sip)"
and "1 ≤ sn"
and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
∧ (nsqn (rt (σ sip)) ip = sn
→ (the (dhops (rt (σ sip)) ip) ≤ hops
∨ the (flag (rt (σ sip)) ip) = inv))"
shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
∧ (nsqn (rt (σ' sip)) ip = sn
→ (the (dhops (rt (σ' sip)) ip) ≤ hops
∨ the (flag (rt (σ' sip)) ip) = inv))"
(is "_ ∨ ?nsqnafter")
proof -
from * obtain "ip ∈ kD(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto
from quality_increases (σ sip) (σ' sip)
have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
from quality_increases (σ sip) (σ' sip) and ip ∈ kD (rt (σ sip))
have "ip ∈ kD (rt (σ' sip))" ..
from sn ≤ nsqn (rt (σ sip)) ip have ?nsqnafter
proof
assume "sn < nsqn (rt (σ sip)) ip"
also from ip ∈ kD(rt (σ sip)) and quality_increases (σ sip) (σ' sip)
have "... ≤ nsqn (rt (σ' sip)) ip" ..
finally have "sn < nsqn (rt (σ' sip)) ip" .
thus ?thesis by simp
next
assume "sn = nsqn (rt (σ sip)) ip"
with ip ∈ kD(rt (σ sip)) and quality_increases (σ sip) (σ' sip)
have "sn < nsqn (rt (σ' sip)) ip
∨ (sn = nsqn (rt (σ' sip)) ip
∧ the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip))" ..
hence "sn < nsqn (rt (σ' sip)) ip
∨ (nsqn (rt (σ' sip)) ip = sn ∧ (the (dhops (rt (σ' sip)) ip) ≤ hops
∨ the (flag (rt (σ' sip)) ip) = inv))"
proof
assume "sn < nsqn (rt (σ sip)) ip" thus ?thesis ..
next
assume "sn = nsqn (rt (σ' sip)) ip 
∧ the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)"

hence "sn = nsqn (rt (σ' sip)) ip"
and "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)" by auto

from * and ⟨sn = nsqn (rt (σ sip)) ip⟩ have "the (dhops (rt (σ sip)) ip) ≤ hops 
∨ the (flag (rt (σ sip)) ip) = inv"

by simp 
thus ?thesis

proof
assume "the (dhops (rt (σ sip)) ip) ≤ hops"
with ⟨the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)⟩ have "the (dhops (rt (σ' sip)) ip) ≤ hops" by simp

next
assume "the (flag (rt (σ sip)) ip) = inv"
with ⟨ip ∈ kD(rt (σ' sip))⟩ have "nsqn (rt (σ' sip)) ip = sqn (rt (σ' sip)) ip" ..

from ⟨ip ∈ kD (rt (σ' sip))⟩ show ?thesis 

proof (rule vD_or_iD)
assume "ip ∈ iD(rt (σ' sip))"
hence "the (flag (rt (σ' sip)) ip) = inv" ..

next

assume "ip ∈ vD(rt (σ' sip))"
hence "nsqn (rt (σ' sip)) ip = sqn (rt (σ' sip)) ip" ..

with ⟨sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip⟩ have "nsqn (rt (σ' sip)) ip ≥ sqn (rt (σ sip)) ip" by simp

with ⟨sqn (rt (σ sip)) ip > 1⟩
  have "nsqn (rt (σ' sip)) ip > sqn (rt (σ sip)) ip - 1" by simp

with ⟨nsqn (rt (σ' sip)) ip = sqn (rt (σ sip)) ip - 1⟩
  have "nsqn (rt (σ' sip)) ip > nsqn (rt (σ sip)) ip" by simp

with ⟨sn = nsqn (rt (σ sip)) ip⟩ have "nsqn (rt (σ' sip)) ip > sn" by simp
thus ?thesis ..

qed

lemma quality_increases_rreq_rrep_props':
  fixes sn ip hops sip
assumes "∀ j. quality_increases (σ j) (σ' j)"
  and "1 ≤ sn"
  and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip 
     ∧ (nsqn (rt (σ sip)) ip = sn 
         → (the (dhops (rt (σ sip)) ip) ≤ hops 
            ∨ the (flag (rt (σ sip)) ip) = inv))"

  shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip 
       ∧ (nsqn (rt (σ' sip)) ip = sn 
           → (the (dhops (rt (σ' sip)) ip) ≤ hops 
               ∨ the (flag (rt (σ' sip)) ip) = inv))"

proof -

from assms(1) have "quality_increases (σ sip) (σ' sip)" ..
thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props)
lemma rreq_quality_increases:
  assumes "∀ j. j ≠ i → quality_increases (σ j) (σ' j)"
  and "rt (σ' i) = rt (σ i)"
  shows "∀ j. quality_increases (σ j) (σ' j)"
  using assms by clarsimp (metis order_refl quality_increasesI rt_fresher_refl)

definition msg_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"
where
  "msg_fresh σ m ≡
  case m of Rreq hopsc _ _ oipc osnc sipc _ ⇒
           osnc ≥ 1 ∧ (sipc ≠ oipc →
                       oipc ∈ kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc
                        ∧ (nsqn (rt (σ sipc)) oipc = osnc
                           → (hopsc ≥ (dhops (rt (σ sipc)) oipc)
                              ∨ the (flag (rt (σ sipc)) oipc) = inv)))))
| Rrep hopsc dipc dsnc _ sipc ⇒
              dsnc ≥ 1 ∧ (sipc ≠ dipc →
                       dipc ∈ kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc
                        ∧ (nsqn (rt (σ sipc)) dipc = dsnc
                           → (hopsc ≥ (dhops (rt (σ sipc)) dipc)
                              ∨ the (flag (rt (σ sipc)) dipc) = inv))))
| Rerr destsc sipc ⇒ (∀ ripc ∈ dom(destsc). (ripc ∈ kD(rt (σ sipc)))
                        ∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))
| _ ⇒ True"

lemma msg_fresh [simp]:
  "∃ hops dip dsn oip osn sip handled. msg_fresh σ (Rreq hops dip dsn dsk oip osn sip handled) =
   (osn ≥ 1 ∧ (sip ≠ oip → oip ∈ kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oip ≥ osn
    ∧ (nsqn (rt (σ sipc)) oip = osn
       → (hops ≥ (dhops (rt (σ sipc)) oip)
          ∨ the (flag (rt (σ sipc)) oip) = inv)))))"

lemma msg_fresh [simp]:
  "∃ hops dip dsn oip sip. msg_fresh σ (Rrep hops dip dsn oip sip) =
   (dsn ≥ 1 ∧ (sip ≠ dip → dip ∈ kD(rt (σ sipc))
    ∧ nsqn (rt (σ sipc)) dip ≥ dsn
    ∧ (nsqn (rt (σ sipc)) dip = dsn
       → (hops ≥ (dhops (rt (σ sipc)) dip)
          ∨ the (flag (rt (σ sipc)) dip) = inv)))))"

lemma msg_fresh [simp]:
  "∃ dests sip. msg_fresh σ (Rerr dests sip) =
   (∀ ripc ∈ dom(dests). (ripc ∈ kD(rt (σ sipc)))
    ∧ the (dests ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))"

lemma msg_fresh [simp]:
  "∃ d dip. msg_fresh σ (Newpkt d dip) = True"

lemma msg_fresh [simp]:
  "∃ d dip sip. msg_fresh σ (Pkt d dip sip) = True"

unfolding msg_fresh_def by simp_all

lemma msg_fresh_inc_sn [simp, elim]:
  "msg_fresh σ mimat++ rreq_rrep_sn m" by (cases m) simp_all

lemma recv_msg_fresh_inc_sn [simp, elim]:
  "orecvmsg (msg_fresh) σ mimat++ rreq_rrep_sn m" by (cases m) simp_all

lemma rreq_msg_fresh_inc_sn [simp, elim]:
  "msg_fresh σ mimat++ rreq_rrep_sn m" by (cases m) simp_all

lemma rreq_msg_fresh_is_fresh [simp]:
  fixes σ msg hops dip dsn dsk oip osn sip handled
  assumes "rreq_rrep_fresh (rt (σ sipc)) (Rreq hops dip dsn dsk oip osn sip handled)"
  and "rreq_rrep_sn (Rreq hops dip dsn dsk oip osn sip handled)"
  shows "msg_fresh σ (Rreq hops dip dsn dsk oip osn sip handled)"
  (is "msg_fresh σ ?msg")

proof -
  let ?rt = "rt (σ sipc)"
  from assms(2) have "1 ≤ osn" by simp
  thus ?thesis
  unfolding msg_fresh_def

qed
proof (simp only: msg.case, intro conjI impI)
assume "sip ≠ oip"
with assms(1) show "oip ∈ kD(?rt)" by simp
next
assume "sip ≠ oip"
and "nsqn ?rt oip = osn"
show "the (dhops ?rt oip) ≤ hops ∨ the (flag ?rt oip) = inv"
proof (cases "oip∈vD(?rt)"
assume "oip∈vD(?rt)"
"nsqn ?rt oip = sqn ?rt oip" ..
with ⟨nsqn ?rt oip = osn⟩ have "sqn ?rt oip = osn" by simp
with assms(1) and ⟨sip ≠ oip⟩ have "the (dhops ?rt oip) ≤ hops"
by simp
thus ?thesis ..
next
assume "oip∈vD(?rt)"
moreover from assms(1)
and ⟨sip ≠ oip⟩ have "oip∈kD(?rt)" by simp
ultimately have "oip∈iD(?rt)" by auto
hence "the (flag ?rt oip) = inv" ..
thus ?thesis ..
qed
next
assume "sip ≠ oip"
with assms(1) have "osn ≤ sqn ?rt oip" by auto
thus "osn ≤ nsqn ?rt oip" (rule nat_le_eq_or_lt)
assume "osn ≤ sqn ?rt oip" and "the (flag ?rt oip) = val"
also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
finally show "osn ≤ nsqn ?rt oip" .
next
assume "osn = sqn ?rt oip"
with assms(1) and ⟨sip ≠ oip⟩ have "oip∈kD(?rt)"
and "the (flag ?rt oip) = val"
by auto
hence "nsqn ?rt oip = sqn ?rt oip" ..
with ⟨osn = sqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
thus "osn ≤ nsqn ?rt oip" by simp
qed
qed simp

lemma rrep_nsqn_is_fresh [simp]:
fixes σ msg hops dip dsn oip sip
assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
(is "msg_fresh σ ?msg")
proof -
let ?rt = "rt (σ sip)"
from assms have "sip ≠ dip → dip∈kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
by simp
hence "sip ≠ dip → dip∈kD(?rt) ∧ nsqn ?rt dip ≥ dsn"
by clarsimp
with assms show "msg_fresh σ ?msg"
by clarsimp
qed

lemma rerr_nsqn_is_fresh [simp]:
fixes σ msg dests sip
assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
shows "msg_fresh σ (Rerr dests sip)"
(is "msg_fresh σ ?msg")
proof -
let \( rt = \text{"rt (} \sigma \text{ sip)"} \)

from assms have \
\( \forall \text{dests. } (\text{rip} \in \text{dom(dests)} \rightarrow \text{rip} \in \text{ID}(rt (\sigma \text{ sip})) \land \text{the (dests rip)} = \text{sqn (} rt (\sigma \text{ sip)) rip))" \)

by clarsimp
have \
\( \forall \text{dests. } (\text{rip} \in \text{KD}(rt (\sigma \text{ sip})) \land \text{the (dests rip)} - 1 \leq \text{nsqn (} rt (\sigma \text{ sip) rip)))" \)

proof
fix rip
assume "\text{rip} \in \text{dom dests}" with * have "\text{rip} \in \text{ID}(rt (\sigma \text{ sip}))" and "\text{the (dests rip)} = \text{sqn (} rt (\sigma \text{ sip)) rip)"

by auto
from this(2) have "\text{the (dests rip)} - 1 = \text{sqn (} rt (\sigma \text{ sip)) rip) - 1"

by simp
also have "... \leq \text{nsqn (} rt (\sigma \text{ sip) rip)}" by (rule sqn_nsqn)
finally have "\text{the (dests rip)} - 1 \leq \text{nsqn (} rt (\sigma \text{ sip)) rip)"

with \text{⟨} \text{rip} \in \text{ID}(rt (\sigma \text{ sip))⟩}

show "\text{rip} \in \text{KD}(rt (\sigma \text{ sip})) \land \text{the (dests rip)} - 1 \leq \text{nsqn (} rt (\sigma \text{ sip)) rip)"

by clarsimp
qed
thus "\text{msg_fresh (} \sigma \text{ ?msg)"}

by simp
qed

lemma quality_increases_msg_fresh [elim]:
assumes qinc: "\forall j. \text{quality_increases (} \sigma j \text{ (} \sigma' j\text{)}"
and "\text{msg_fresh (} \sigma \text{ m)"
shows "\text{msg_fresh (} \sigma' \text{ m)"
using assms(2)
proof (cases m)
fix hops dip dsn dsk oip osn sip handled
assume [simp]: "m = Rreq hops dip dsn dsk oip osn sip handled"
and "\text{msg_fresh (} \sigma \text{ m)" then have "\text{osn } \geq 1\text{ and (rip} \in \text{KD}(rt (\sigma \text{ sip})) \land \text{osn } \leq \text{nsqn (} rt (\sigma \text{ sip)) oip}
\land (\text{nsqn (} rt (\sigma \text{ sip)) oip } = \text{osn}
\longrightarrow (\text{the (dhops (} rt (\sigma \text{ sip)) oip)} \leq \text{hops}
\lor \text{the (flag (} rt (\sigma \text{ sip)) oip) } = \text{inv}))""

by auto
from this(2) show ?thesis
proof
assume "\text{rip} = \text{oip}" with (\text{osn } \geq 1) show ?thesis by simp
next
assume "\text{oip} \in \text{KD}(rt (\sigma \text{ sip})) \land \text{osn } \leq \text{nsqn (} rt (\sigma \text{ sip)) oip}
\land (\text{nsqn (} rt (\sigma \text{ sip)) oip } = \text{osn}
\longrightarrow (\text{the (dhops (} rt (\sigma \text{ sip)) oip)} \leq \text{hops}
\lor \text{the (flag (} rt (\sigma \text{ sip)) oip) } = \text{inv}))""

moreover from qinc have "\text{quality_increases (} \text{\sigma (} \text{\sigma' (\text{\sigma})" ..
ultimately have "\text{oip} \in \text{KD}(rt (\sigma' \text{ sip})) \land \text{osn } \leq \text{nsqn (} rt (\sigma' \text{ sip)) oip}
\land (\text{nsqn (} rt (\sigma' \text{ sip)) oip } = \text{osn}
\longrightarrow (\text{the (dhops (} rt (\sigma' \text{ sip)) oip)} \leq \text{hops}
\lor \text{the (flag (} rt (\sigma' \text{ sip)) oip) } = \text{inv}))""

using (\text{osn } \geq 1) by (rule quality_increases_rreq_rrep_props [rotated 2])
with (\text{osn } \geq 1) show "\text{msg_fresh (} \sigma' \text{ m)" by (clarsimp)
qed
next
fix hops dip dsn oip sip
assume [simp]: "m = Rrep hops dip dsn oip sip"
and "\text{msg_fresh (} \sigma \text{ m)" then have "\text{dsn } \geq 1\text{ and (rip} \in \text{KD}(rt (\sigma \text{ sip})) \land \text{dsn } \leq \text{nsqn (} rt (\sigma \text{ sip)) dip}
\land (\text{nsqn (} rt (\sigma \text{ sip)) dip } = \text{dsn}
\longrightarrow (\text{the (dhops (} rt (\sigma \text{ sip)) dip)} \leq \text{hops}
\lor \text{the (flag (} rt (\sigma \text{ sip)) dip) } = \text{inv}))""

by auto

466
from this(2) show "?thesis"
proof
  assume "sip = dip" with \( dsn \geq 1 \) show ?thesis by simp
next
  assume "dip \in kD(rt (\sigma sip)) \land dsn \leq nsqn (rt (\sigma sip)) dip \\
  \land (nsqn (rt (\sigma sip)) dip = dsn \\
  \rightarrow (the (dhops (rt (\sigma sip)) dip) \leq hops \\
  \lor the (flag (rt (\sigma sip)) dip) = inv))"
  moreover from qinc have "quality_increases (\sigma sip) (\sigma' sip)" ..
  ultimately have "dip \in kD(rt (\sigma' sip)) \land dsn \leq nsqn (rt (\sigma' sip)) dip \\
  \land (nsqn (rt (\sigma' sip)) dip = dsn \\
  \rightarrow (the (dhops (rt (\sigma' sip)) dip) \leq hops \\
  \lor the (flag (rt (\sigma' sip)) dip) = inv))"
  using \( dsn \geq 1 \) by (rule quality_increases_rreq_rrep_props [rotated 2])
  with \( dsn \geq 1 \) show "msg_fresh \sigma' m"
  by clarsimp
qed

next
fix dests sip
assume [simp]: "m = Rerr dests sip"
and "msg_fresh \sigma m"
then have *: "\( \forall rip \in \text{dom(dests)}. rip \in kD(rt (\sigma sip)) \\
  \land the (dests rip) - 1 \leq nsqn (rt (\sigma sip)) rip)"
  by simp
have "\( \forall rip \in \text{dom(dests)}. rip \in kD(rt (\sigma' sip)) \\
  \land the (dests rip) - 1 \leq nsqn (rt (\sigma' sip)) rip)"
proof
  fix rip
  assume "rip \in \text{dom(dests)}"
  with * have "rip \in kD(rt (\sigma sip))" and "the (dests rip) - 1 \leq nsqn (rt (\sigma sip)) rip)"
  by - (drule(1) bspec, clarsimp)+
  moreover from qinc have "quality_increases (\sigma sip) (\sigma' sip)" by simp
  ultimately show "rip \in kD(rt (\sigma' sip)) \land the (dests rip) - 1 \leq nsqn (rt (\sigma' sip)) rip)" ..
  qed
  thus ?thesis by simp
  qed simp_all
end

5.8 The ‘open’ AODV model

theory E_OAodv
imports E_Aodv AWN.OAWN_SOS_Labels AWN.OAWN_Convert
begin
Definitions for stating and proving global network properties over individual processes.
definition \( \sigma_{AODV}' :: ((ip \Rightarrow state) \times ((state, msg, pseqp, pseqp label) seqp)) set" where "\( \sigma_{AODV}' \equiv \{ (\lambda i. aodv_init i, \Gamma_{AODV} P_{Aodv}) \} \)"
abbreviation opaodv :: "ip \Rightarrow ((ip \Rightarrow state) \times ((state, msg, pseqp, pseqp label) seqp, msg seq_action)) automaton" where "opaodv i \equiv \{ \text{init = } \sigma_{AODV}', \text{trans = oseqp_sos } \Gamma_{AODV} i \}"
lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
  unfolding \( \sigma_{AODV}' \_def \) \( \sigma_{AODV}' \_def \) by rule simp_all
lemma oaedv_control_within [simp]: "control_within \Gamma_{AODV} (init (opaodv i))"
  unfolding \( \sigma_{AODV}' \_def \) by (rule control_withinI) (auto simp del: \( \Gamma_{AODV} \_sims \)\)\)
lemma \( \sigma_{AODV}' \_labels [simp]: "(\sigma, p) \in \sigma_{AODV}' \implies labels \Gamma_{AODV} p = \{ P_{Aodv} = 0 \}"
  unfolding \( \sigma_{AODV}' \_def \) by simp
lemma oaedv_init_kD_empty [simp]:

467
"(σ, p) ∈ σ_AODV' =⇒ kD (rt (σ i)) = {}"

unfolding σ_AODV'_def kD_def by simp

lemma oaedv_init_vD_empty [simp]:
"(σ, p) ∈ σ_AODV' =⇒ vD (rt (σ i)) = {}"

unfolding σ_AODV'_def vD_def by simp

lemma oaedv_trans: "trans (oaedv i) = oseqp_sos Γ_AODV i"

by simp

declare oseq_invariant_ctermsI [OF aodv_wf oaedv_control_within aodv_simple_labels oaedv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf oaedv_control_within aodv_simple_labels oaedv_trans, cterms_intros]

end

5.9 Global invariant proofs over sequential processes

theory E_Global_Invariants
imports
  E_Seq_Invariants
  E_Aodv_Predicates
  E_Fresher
  E_Quality_Increases
  AKN.OAWN_Convert
  E_OAodv
begin

lemma other_quality_increases [elim]:
assumes "other quality_increases I σ σ'"
shows "∀ j. quality_increases (σ j) (σ' j)"
using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
fixes m
assumes *: "otherwith P I (orecvmsg Q) σ σ' a"
  and weakenP: "∀ σ m. P σ m =⇒ P' σ m"
  and weakenQ: "∀ σ m. Q σ m =⇒ Q' σ m"
shows "otherwith P' I (orecvmsg Q') σ σ' a"
proof
  fix j
  assume "j /∈ I"
  with * have "P (σ j) (σ' j)" by auto
  thus "P' (σ j) (σ' j)" by (rule weakenP)

next
  from * have "orecvmsg Q σ a" by auto
  thus "orecvmsg Q' σ a"
  by rule (erule weakenQ)
qed

lemma orecieved_msg_inv:
assumes other: "∀ σ σ' m. [ P σ m; other Q {i} σ σ' ] =⇒ P σ' m"
  and local: "∀ σ m. P σ m =⇒ P (σ (i := σ i (msg := m))) m"
shows "oaedv i = (otherwith Q {i} (orecvmsg P), other Q {i} →)
      onl Γ_AODV (λ(σ, 1). l ∈ {PAodv-:1} =⇒ P σ (msg (σ i))))"
proof (inv_cterms, intro impI)
  fix σ σ' 1
  assume "1 = PAodv-:1 =⇒ P σ (msg (σ i))"
  and "1 = PAodv-:1"
  and "other Q {i} σ σ'"
  from this(1-2) have "P σ (msg (σ i))" ..
  hence "P σ' (msg (σ i))" using other Q {i} σ σ'
  by (rule other)
  moreover from other Q {i} σ σ' have "σ' i = σ i" ..
  ultimately show "P σ' (msg (σ' i))" by simp
next
fix \( \sigma \sigma' \) msg
assume "otherwith Q \( \{i\} (\text{orecmsg} P) \sigma \sigma' \) (receive msg)"
and "\( \sigma' i = \sigma i[\text{msg} := \text{msg}] \)"
from this(1) have "\( P \sigma \text{msg} \)"
and "\( \forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j) \)" by auto
from this(1) have "\( P (\sigma(i := \sigma i[\text{msg} := \text{msg}])) \) msg" by (rule local)
thus "\( P \sigma' \) msg"
proof (rule other)
fix \( s a s' \)
assume sr: "\( s \in \text{reachable (opaodv i) (recvmsg \ rreq_rrep_sn)} \)"
and tr: "\((s, a, s') \in \text{trans (opaodv i)}\)"
and rm: "recvmsg \( rreq_rrep_sn \) a"
from sr have srTT: "\( s \in \text{reachable (opaodv i) TT} \)"
from route_tables_fresher sr tr rm
have "\( \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{dip} \in \text{kD (rt } \xi). \text{rt } \xi \sqsubseteq \text{dip} \text{ rt } \xi') (s, a, s') \)"
by (rule step_invariantD)
moreover from known_destinations_increase srTT tr TT_True
have "\( \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{kD (rt } \xi) \subseteq \text{kD (rt } \xi') (s, a, s') \)"
by (rule step_invariantD)
moreover from sqns_increase srTT tr TT_True
have "\( \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{ip. sqn (rt } \xi) \text{ip} \leq \text{sqn (rt } \xi') \text{ip}) (s, a, s') \)"
by (rule step_invariantD)
ultimately show "\( \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{quality_increases } \xi \xi') (s, a, s') \)"
unfolding onll_def by auto
qed

lemmas olocal_quality_increases =
open_seq_step_invariant [OF local_quality_increases initiali_aodv oaodv_trans aodv_trans,
simplified seqll_onll_swap]

lemma equality_increases:
"\( \text{opaodv } i \models A (\text{recvmsg } rreq_rrep_sn \rightarrow) \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{quality_increases } \xi \xi') \)"
proof (rule onll_onestep_invariantI, simp)
fix \( \sigma P p l \sigma' p' l' \)
assume or: "\( (\sigma, p) \in \text{oreachable (opaodv i) } ?S \) (other quality_increases \( \{i\} \))"
and 11: "\( l \in \text{labels } \Gamma_{AODV} p \)"
and 22: "\( ?S \sigma \sigma' a \)"
and tr: "\( ((\sigma, p), a, (\sigma', p')) \in \text{oseqp_sos } \Gamma_{AODV} i \)"
and 11': "\( l' \in \text{labels } \Gamma_{AODV} p' \)"
from this(1-3) have "\( \text{recvmsg } (\lambda_. \text{rreq_rrep_sn}) \sigma a \)"
by (auto dest!: orachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
and QU="other quality_increases \( \{i\} \)"]
otherwith_actionD
with or have orw: "\( (\sigma, p) \in \text{oreachable (opaodv i) } (\text{act (recvmsg rreq_rrep_sn)}) \) (other quality_increases \( \{i\} \))"
by - (erule orachable_weakenE, auto)
with tr l l' and \( \text{recvmsg}(\lambda_. \text{req}_{rrep_{-sn}}) \) a) have "quality_increases (\( \sigma \) i) (\( \sigma' \) i)"
by (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def)
with \( \forall \sigma \sigma' a: \) show "\( \forall j. \) quality_increases (\( \sigma \) j) (\( \sigma' \) j)"
by (auto dest!: otherwith_syncD)
qed

lemma req_{rrep_{-nsq_{-}fres_{-}any_{-}step_{-}invariant}:
"opaodv i \models_{A} (\text{act (recvmsg req}_{rrep_{-}sn}), \text{other A \{i\} \rightarrow})
onll \Gamma_{AODV} (\lambda(\sigma, \_, a, \_). \text{anycast (msg}_{fres_{-}} \sigma a))"
proof (rule ostep_invariantI, simp del: act_simp)
fix \( \sigma a \sigma' p' \)
assume or: "\((\sigma, p) \in \text{oreachable (opaodv i) (act (recvmsg req}_{rrep_{-}sn})) (\text{other A \{i\}})\)"
and recv: "\(\text{act (recvmsg req}_{rrep_{-}sn}) \sigma \sigma' a\)"
obtain l l' where "\(l \in \text{labels} \Gamma_{AODV} p\)" and "\(l' \in \text{labels} \Gamma_{AODV} p'\)"
by (metis aodv_ex_label)
from \(\langle((\sigma, p), a, (\sigma', p'))\rangle \in \text{oseqp}\_\text{sos} \Gamma_{AODV} i\)
have tr: "\(\text{trans (opaodv i)}\)
\(\langle((\sigma, p), a, (\sigma', p'))\rangle \in \text{trans (opaodv i)}\)"
by simp
have "\(\text{anycast (rreq}_{rrep_{-}fresh (rt (\sigma i))) a}\)"
proof
\begin{itemize}
\item have "\(\text{opaodv i \models}_{A} (\text{act (recvmsg req}_{rrep_{-}sn}), \text{other A \{i\} \rightarrow})\)
onll \Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{anycast (rreq}_{rrep_{-}fresh (rt \xi)) a}))"
\end{itemize}
by (rule ostep_invariantE [OF open_seq_step_invariant [OF rreq_{rrep_{-}fresh_{-}any_{-}step_{-}invariant initiali_{aodv},
simplified seqll_onll_swap]]) auto
hence "\(\text{onll} \Gamma_{AODV} (\text{seqll } \lambda((\xi, \_), a, \_). \text{anycast (rreq}_{rrep_{-}fresh (rt \xi)) a})) ((\sigma, p), a, (\sigma', p'))\)"
using or tr recv by - (erule(4) ostep_invariantE)
thus ?thesis
using \(\langle l \in \text{labels} \Gamma_{AODV} p\rangle\) and \(\langle l' \in \text{labels} \Gamma_{AODV} p'\rangle\) by auto
qed
moreover have "\(\text{anycast (rerr}_{invalid (rt (\sigma i))) a}\)"
proof
\begin{itemize}
\item have "\(\text{opaodv i \models}_{A} (\text{act (recvmsg req}_{rrep_{-}sn}), \text{other A \{i\} \rightarrow})\)
onll \Gamma_{AODV} (\lambda((\xi', \_), a, \_). \text{anycast (rerr}_{invalid (rt \xi) a}))"
\end{itemize}
by (rule ostep_invariantE [OF open_seq_step_invariant [OF rerr_{invalid}_{any_{-}step_{-}invariant initiali_{aodv},
simplified seqll_onll_swap]]) auto
hence "\(\text{onll} \Gamma_{AODV} (\text{seqll } \lambda((\xi', \_), a, \_). \text{anycast (rerr}_{invalid (rt \xi) a})) ((\sigma, p), a, (\sigma', p'))\)"
using or tr recv by - (erule(4) ostep_invariantE)
thus ?thesis
using \(\langle l \in \text{labels} \Gamma_{AODV} p\rangle\) and \(\langle l' \in \text{labels} \Gamma_{AODV} p'\rangle\) by auto
qed
moreover have "\(\text{anycast req}_{rrep_{-}sn a}\)"
proof
from or tr recv
have "\(\text{onll} \Gamma_{AODV} (\text{seqll } \lambda((\xi, \_), a, \_). \text{anycast req}_{rrep_{-}sn a}) ((\sigma, p), a, (\sigma', p'))\)"
by (rule ostep_invariantE [OF open_seq_step_invariant [OF req_{rrep_{-}sn}_{-}any_{-}step_{-}invariant initiali_{aodv},
aodv_{-}trans aodv_{-}trans,
simplified seqll_onll_swap]])
thus ?thesis
using \(\langle l \in \text{labels} \Gamma_{AODV} p\rangle\) and \(\langle l' \in \text{labels} \Gamma_{AODV} p'\rangle\) by auto
qed
moreover have "\(\text{anycast (\lambda m. \text{not}_{Pkt} m \rightarrow \text{msg}_{sender} m = i) a}\)"
proof
have "\(\text{opaodv i \models}_{A} (\text{act (recvmsg req}_{rrep_{-}sn}), \text{other A \{i\} \rightarrow})\)
onll \Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{anycast (\lambda m. \text{not}_{Pkt} m \rightarrow \text{msg}_{sender} m = i) a}))"
by (rule ostep_invariantE [OF open_seq_step_invariant [OF req_{rrep_{-}sn}_{-}any_{-}step_{-}invariant initiali_{aodv},
aodv_{-}trans aodv_{-}trans,
simplified seqll_onll_swap]])
thus ?thesis
using \(\langle l \in \text{labels} \Gamma_{AODV} p\rangle\) and \(\langle l' \in \text{labels} \Gamma_{AODV} p'\rangle\) by auto
qed

open_seq_step_invariant [OF sender_ip_valid initiali_aodv, simplified seqll_onll_swap]) auto
thus ?thesis using or tr recv (l ∈ labels Γ_aodv p) and (l’ ∈ labels Γ_aodv p’)
by (drule(3) onll_ostep_invariantD, auto)
qed

ultimately have "anycast (msg_fresh σ) a"
by (simp_all add: anycast_def
del: msg_fresh
split: seq.action.split_asm msg.split_asm) simp_all
by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} |→
onl Γ_aodv (λ(σ, l). l ∈ {PAodv-:1} −→ msg_fresh σ (msg (σ i)))"
proof (rule oreceived_msg_inv)
fix σ σ' m
assume *: "msg_fresh σ m"
and "other quality_increases {i} σ σ'"
from this(2) have "∀ j. quality_increases (σ j) (σ’ j) " ..
thus "msg_fresh σ’ m" using * ..

next
fix σ m
assume "msg_fresh σ m"
thus "msg_fresh (σ(i := σ i(msg := m))) m"
proof (cases m)
fix dests sip
assume "m = Rerr dests sip"
with ⟨msg_fresh σ m⟩ show ?thesis by auto
qed auto
qed

lemma oquality_increases_nsqn_fresh:
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} |→
onl Γ_aodv (λ((σ, _), _, (σ', _)). ∀ j. quality_increases (σ j) (σ’ j))"
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} |→
onl Γ_aodv (seq l (λ(ξ, l). l ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-:n |n. True} −→ 1 ≤ osn ξ)))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])
(auto simp: seql_onl_swap)

lemma rreq_sip:
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} |→
onl Γ_aodv (λ(σ, l). (l ∈ {PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2} ∧ sip (σ i) ≠ oip (σ i))
→ oip (σ i) ∈ kD(rt (σ (sip (σ i))))
∧ nsqn (rt (σ (sip (σ i)))) (oip (σ i)) ≥ osn (σ i))
∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i))
→ (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (oip (σ i)))
∨ the (flag (rt (σ (sip (σ i)))) (oip (σ i))) = inv))")
(is "_ |= (?S, ?U → _)")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf oadv_trans]
onl_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq]
simp add: seqlonsnip

471
fix $\sigma, \tau \vdash l$
assumption "(?l, p) in reachable (opaodv i) ?S ?U"
and "1 in labels $\Gamma_{AODV}$ p"
and pre:

\[
(1 = PAodv-:4 \land 1 = PAodv-:5 \land 1 = PRreq-:0 \land 1 = PRreq-:2) \land \sigma \neq oip (s i) \\
\rightarrow \sigma (s i) \in kD (rt (\sigma (s i)))) \\
\land \sigma (s i) \leq nsqn (rt (\sigma (s i)))) (oip (s i)) \\
\land (nsqn (rt (\sigma (s i)))) (oip (s i)) = \sigma (s i) \\
\rightarrow \sigma (s i) \leq hops (s i) \\
\lor \sigma (s i) = inv" 
\]

and "other quality_increases {i} σ σ'"
and hyp: "(l = PAodv-:4 ∨ l = PAodv-:5 ∨ l = PRreq-:0 ∨ l = PRreq-:2) \land \sigma (s i) \neq oip (s i)"
(is "labels \land \sigma (s i) \neq oip (s i)"
from this(4) have "\sigma (s i) = s i" ...
with hyp have hyp': "labels \land \sigma (s i) \neq oip (s i)" by simp
show "oip (s i) \in kD (rt (\sigma (s i)))) \\
\land \sigma (s i) \leq nsqn (rt (\sigma (s i)))) (oip (s i)) \\
\land (nsqn (rt (\sigma (s i)))) (oip (s i)) = \sigma (s i) \\
\rightarrow \sigma (s i) \leq hops (s i) \\
\lor \sigma (s i) = inv"
proof (cases "\sigma (s i) = s i")
assume "\sigma (s i) \neq s i"
from "other quality_increases {i} σ σ'
have "quality_increases (\sigma (s i)) (\sigma (s i))"
by (rule otherE) (clarsimp simp:
moreover from "other quality_increases {i} (other (l \in \{PAodv-:6, PAodv-:7\} \{PRrep-:n\n.pre: "oip (s i) = s i) hyp and pre by auto
qed

next
assume "\sigma (s i) = s i" thus ?thesis
using (s i = s i) hyp and pre by auto

lemma odsn_rrep:
"opaodv i \vdash (otherwith quality_increases {i} (orecvmsg msg_fresh), 
other quality_increases {i}) \rightarrow 
\land \Gamma_{AODV} (seql (\lambda (l, 1), l \in \{PAodv-:6, PAodv-:7\} \{PRrep-:n\n.by (rule oinvariant_weakenE [OF open_seq_invariant [OF dsn_rrep initiali_aodv]])
(auto simp: seql_onl_swap)

lemma rrep_sip:
"opaodv i \vdash (otherwith quality_increases {i} (orecvmsg msg_fresh), 
other quality_increases {i}) \rightarrow 
\land \Gamma_{AODV} (\lambda (s, l). 
(l \in \{PAodv-:6, PAodv-:7, PRrep-:0, PRrep-:1\} \land \sigma (s i) \neq dip (s i)) \\
\rightarrow \sigma (s i) \in kD (rt (\sigma (s i)))) \\
\land \sigma (s i) \leq nsqn (rt (\sigma (s i)))) (dip (s i)) \geq dsn (s i) \\
\land (nsqn (rt (\sigma (s i)))) (dip (s i)) = dsn (s i) \\
\rightarrow (dip (s i) \geq \sigma (s i) \leq \sigma (dip (s i))) \\
\lor \sigma (dip (s i)) = inv))"
(is "_ \vdash (?S, ?U \rightarrow _)"
proof (inv_c terms inv add: oseq_step_invariant_sterms [OF equality_increases_nsqn_fresh aodv_wf

472
lemma rerr_sip:
"oapov i |= (otherwith quality_increases {i} (orecmsg msg_fresh), other_quality_increases {i} ->)
    onl Γ_AODV (λ(σ, l).
    l ∈ {PAodv-:8, PAodv-:9, PRerr-:0, PRerr-:1} ->
      (∀ripc∈dom(dests (σ i)). ripc∈kD(rt (σ (sip (σ i)))) ∧
      the (dests (σ i) ripc) - 1 ≤ nsqn (rt (σ (sip (σ i)))) ripc))"
(is "_ |= (♂s, ✅ ℰ −→ _) _")
proof -
{ fix dests rip sip rsn and σ σ' :: "ip ⇒ state"
  assume qinc: "∀j. quality_increases (σ j) (σ' j)"
  and *: "∀ripc∈dom dests. rip ∈ kD (rt (σ sip)) ∧
      the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
  and "dests rip = Some rsn"
  from this(3) have "rip∈dom dests" by auto
  with * and (dests rip = Some rsn) have "rip∈kD(rt (σ sip))"
}

fix σ σ' p l
assume "("σ, p) ∈ reachable (oapov i) ✅ S ✅ U"
and "l ∈ labels Γ_AODV p"
and pre:
"(l = PAodv-:6 ∨ l = PAodv-:7 ∨ l = PRrep-:0 ∨ l = PRrep-:1) ∧ sip (σ i) ≠ dip (σ i)
  → dip (σ i) ∈ kD (rt (σ (sip (σ i))))
  ∧ dsn (σ i) ≤ nsqn (rt (σ (sip (σ i)))) (dip (σ i))
  ∧ (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dsn (σ i))
  → the (dhops (rt (σ (sip (σ i)))) (dip (σ i)) ≤ hops (σ i)
      ∨ the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv)"
and "other_quality_increases {i} σ σ'"
and hyp: "(l=PAdv-:6 ∨ l=PAdv-:7 ∨ l=PRrep-:0 ∨ l=PRrep-:1) ∧ sip (σ', i) ≠ dip (σ' i)"
(is "?labels ∧ sip (σ', i) ≠ dip (σ' i)"
from this(4) have "σ' i = σ i" ..
with hyp have hyp': "?labels ∧ dip (σ i) ≠ dip (σ i)" by simp
show "dip (σ i) ∈ kD (rt (σ' (sip (σ' i))))
      ∧ dsn (σ' i) ≤ nsqn (rt (σ' (sip (σ' i)))) (dip (σ' i))
      ∧ (nsqn (rt (σ' (sip (σ' i)))) (dip (σ' i)) = dsn (σ' i))
      → the (dhops (rt (σ' (sip (σ' i)))) (dip (σ' i)) ≤ hops (σ' i)
          ∨ the (flag (rt (σ' (sip (σ' i)))) (dip (σ' i))) = inv)"
proof (cases "sip (σ i) = i")
assume "sip (σ i) ≠ i"
from "other_quality_increases {i} σ σ'"
have "quality_increases (σ (sip (σ i))) (σ' (sip (σ' i)))"
  by (rule otherE) (clarsimp simp: by auto)
moreover from "σ, p) ∈ reachable (oapov i) ✅ S ✅ U" l ∈ labels Γ_AODV p and hyp
have "l ≤ dsn (σ' i)"
  by (auto dest!: onl_o invariant weakenD [OF odsn_rrep]
simp add: seqlsimp σ' i = σ i)
moreover from "sip (σ i) ≠ i" hyp' and pre
have "dip (σ' i) ∈ kD (rt (σ' (sip (σ' i))))
      ∧ dsn (σ' i) ≤ nsqn (rt (σ' (sip (σ' i)))) (dip (σ' i))
      ∧ (nsqn (rt (σ' (sip (σ' i)))) (dip (σ' i)) = dsn (σ' i))
      → the (dhops (rt (σ' (sip (σ' i)))) (dip (σ' i)) ≤ hops (σ' i)
          ∨ the (flag (rt (σ' (sip (σ' i)))) (dip (σ' i))) = inv)"
  by (auto simp: σ' i = σ i)
ultimately show ?thesis
  by (rule quality_increases_rreq_rrep_props)
next
assume "sip (σ i) = i" thus ?thesis
  using σ' i = σ i hyp and pre by auto
qed

oaov_trans]
onl_o invariant stems [OF aodv_wf oreceived_rreq_rrep nsqn_fresh_inv]
onl_o invariant stems [OF aodv_wf odsn_rrep]
simp del: One_nat_def, rule implI)
by (auto dest!: bspec)
from qinc have "quality_increases (σ sip) (σ' sip)"
have "rip ∈ kD(rt (σ sip)) ∧ rsn - 1 ≤ nsqn (rt (σ sip)) rip"
proof
from (rip∈kD(rt (σ sip))) and (quality_increases (σ sip) (σ' sip))
show "rip ∈ kD(rt (σ sip))"
next
from (rip∈kD(rt (σ sip))) and (quality_increases (σ sip) (σ' sip))
have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ sip)) rip"
with (rsn - 1 ≤ nsqn (rt (σ sip)) rip) show "rsn - 1 ≤ nsqn (rt (σ sip)) rip"
by (rule le_trans)
qed

} note partial = this

show ?thesis
by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf
oaodv_trans]
onl_onvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
other_quality_increases other_localD
simp del: One_nat_def, intro conjI)
(clarsimp simp del: One_nat_def split: if_split_asm option.split_asm, erule(2) partial)+
qed

lemma prerr_guard: "paodv i ||= onl Γ AODV (λ(ξ, l). (l = PRerr−:1
→ (∀ip∈dom(dests ξ). ip∈vD(rt ξ)
∧ the (nhop (rt ξ) ip) = sip ξ
∧ sqn (rt ξ) ip < the (dests ξ ip)))"
by (inv_cterms) (clarsimp split: option.split_asm if_split_asm)

lemmas odests_vD_inc_sqn =
open_seq_invariant [OF dests_vD_inc_sqn initiali_aodv aoaodv_trans aodv_trans,
simplified seql_onl_swap,
THEN oinvariant_anyact]

lemmas oprerr_guard =
open_seq_invariant [OF prerr_guard initiali_aodv aoaodv_trans aodv_trans,
simplified seql_onl_swap,
THEN oinvariant_anyact]

Proposition 7.28
lemma seq_compare_next_hop':
"opaodv i ||= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} → onl Γ AODV (λ(σ, _).
∀dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →
dip ∈ kD(rt (σ nhip)) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"
(is "_= (?[S, ?U →) _")
proof -

fix nhop and σ :: "ip ⇒ state"
assume pre: "∀dip∈kD(rt (σ i)). nhop dip ≠ dip →
dip∈kD(rt (σ (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
and qinc: "∀j. quality_increases (σ j) (σ' j)"
have "∀dip∈kD(rt (σ i)). nhop dip ≠ dip →
dip∈kD(rt (σ' (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
proof (intro ballI impI)
fix dip
assume "dip∈kD(rt (σ i))"
and "nhop dip ≠ dip"
with pre have "dip∈kD(rt (σ (nhop dip)))"
and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
by auto

474
from qinc have qinc\_nhop: "quality\_increases (σ (nhop dip)) (σ' (nhop dip))" ..
with \( \text{dip} \in \text{kD}(rt (σ (nhop dip))) \) have "dip\(\in\)kD (rt (σ' (nhop dip)))" ..

moreover have "nsqn (rt (σ i)) dip \leq nsqn (rt (σ' (nhop dip))) dip" proof -
from \( \text{dip} \in \text{kD}(rt (σ (nhop dip))) \) qinc\_nhop
have "nsqn (rt (σ (nhop dip))) dip \leq nsqn (rt (σ' (nhop dip))) dip" ..
with nsqn (rt (σ i)) dip \leq nsqn (rt (σ (nhop dip))) dip show ?thesis
by simp
qed
ultimately show "dip\(\in\)kD(rt (σ' (nhop dip))) ∧ nsqn (rt (σ i)) dip \leq nsqn (rt (σ' (nhop dip))) dip" ..
qed
}

\{ fix nhop and σ σ' :: "ip ⇒ state"
assumepre: "∀ dip\(\in\)kD(rt (σ i)). nhop dip ≠ dip \(\Rightarrow\) dip\(\in\)kD(rt (σ (nhop dip))) ∧ nsqn (rt (σ i)) dip \leq nsqn (rt (σ (nhop dip))) dip"
and ndest: "∀ ripc\(\in\)dom (dests (σ i)). ripc \(\in\) kD (rt (σ (sip (σ i)))) ∧ the (dests (σ i)) ripc) - 1 \leq nsqn (rt (σ (sip (σ i)))) ripc"
and issip: "∀ ip\(\in\)dom (dests (σ i)). nhop ip = sip (σ i)"
and qinc: "∀ j. quality\_increases (σ j) (σ' j)"
have "∀ dip\(\in\)kD(rt (σ i)). nhop dip ≠ dip \(\Rightarrow\) dip \(\in\) kD (rt (σ' (nhop dip))) ∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip \leq nsqn (rt (σ' (nhop dip))) dip" proof (intro ballI impI)
fix dip
assume "dip\(\in\)kD(rt (σ i))"
and "nhop dip ≠ dip"
with pre and qinc have "dip\(\in\)kD(rt (σ' (nhop dip)))"
and "nsqn (rt (σ i)) dip \leq nsqn (rt (σ' (nhop dip))) dip"
by (auto dest!: basic)
have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip \leq nsqn (rt (σ' (nhop dip))) dip" proof (cases "dip\(\in\)dom (dests (σ i))")
assume "dip\(\in\)dom (dests (σ i))"
with \( \text{dip} \in \text{kD}(rt (σ i)) \) obtain dsn where "dests (σ i) dip = Some dsn" by auto
with \( \text{dip} \in \text{kD}(rt (σ i)) \) have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = dsn - 1" by (rule nsqn\_invalidate\_eq)
moreover have "dsn - 1 \leq nsqn (rt (σ' (nhop dip))) dip" proof -
from \( \text{dip} \in \text{kD}(rt (σ i)) \) have "the (dests (σ i) dip) = dsn" by simp
with ndest and \( \text{dip} \in \text{kD}(rt (σ i)) \) have "dip \(\in\) kD (rt (σ (sip (σ i)))))"
"dsn - 1 \leq nsqn (rt (σ (sip (σ i)))) dip"
by auto
moreover from issip and \( \text{dip} \in \text{kD}(rt (σ i)) \) have "nhop dip = sip (σ i)" ..
ultimately have "dip \(\in\) kD (rt (σ (nhop dip)))"
and "dsn - 1 \leq nsqn (rt (σ (nhop dip))) dip" by auto
with qinc show "dsn - 1 \leq nsqn (rt (σ' (nhop dip))) dip"
by simp (metis kD\_nsqn\_quality\_increases\_trans)
qed
ultimately show ?thesis by simp
next
assume "dip \notin dom (dests (σ i))"
with \( \text{dip} \in \text{kD}(rt (σ i)) \) have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = nsqn (rt (σ i)) dip"
by (rule nsqn\_invalidate\_other)
with nsqn (rt (σ i)) dip \leq nsqn (rt (σ' (nhop dip))) dip show ?thesis by simp
qed
with \( \text{dip} \in \text{kD}(rt (σ' (nhop dip))) \) show "dip \(\in\) kD (rt (σ' (nhop dip))) ∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip \leq nsqn (rt (σ' (nhop dip))) dip" ..
qed
note basic_prerr = this

{ fix σ σ' :: "ip ⇒ state"
  assume a1: "∀ dip∈kD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
    ∧ nsqn (σ i) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
  and a2: "∀ j. quality_increases (σ j) (σ' j)"
  have "∀ dip∈kD(rt (σ i)).
    the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
    ∧ nsqn (σ i) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip" by (drule(1) basic, auto)
  thus "∃ dip. ?P dip" by (cases "dip = sip (σ i)") auto
  qed
} note nhop_update_sip = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀ dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
    ∧ nsqn (σ i) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀ j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip∈kD(rt (σ sip))
    ∧ osn ≤ nsqn (σ sip) oip
    ∧ (nsqn (σ sip) oip = osn
    → the (dhops (rt (σ sip)) oip) ≤ hops
    ∨ the (flag (rt (σ sip)) oip) = inv)"
  from pre and qinc
  have pre': "∀ dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
    ∧ nsqn (σ i) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
  by (rule basic)
  have "(the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) oip) ≠ oip
    → oip∈kD(rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip)) oip))))
    ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) oip
    ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip)) oip)))) oip)"
  (is "?nhop_not_oip → ?oip_in_kD ∧ ?nsqn_le_nsqn")
  proof (rule, split update_rt_split_asm)
  assume rt: "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)"
  and the (nhop (rt (σ i)) oip) ≠ oip" with pre' show "?oip_in_kD ∧ ?nsqn_le_nsqn" by auto
  next
    assume rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)"
    and notoip: ?nhop_not_oip
    with * qinc have ?oip_in_kD
      by (clarsimp elim!: kD_quality_increases)
    moreover with * pre qinc rtnot notoip have ?nsqn_le_nsqn
      by simp (metis kD_nsqn_quality_increases_trans)
    ultimately show "?oip_in_kD ∧ ?nsqn_le_nsqn" ..
  qed
} note update1 = this

476
{ fix σ σ' oip sip osn hops
assume pre: "∀ dip∈KD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
→ dip∈KD(rt (σ (the (nhop (rt (σ i)) dip))))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
and qinc: "∀ j. quality_increases (σ j) (σ' j)"
and *: "sip ≠ oip → oip∈KD(rt (σ sip))
∧ osn ≤ nsqn (rt (σ sip)) oip
∧ (nsqn (rt (σ sip)) oip = osn
→ the (dhops (rt (σ sip)) oip) ≤ hops
∨ the ((flag (rt (σ sip)) oip) = inv)"

from pre and qinc
have pre': "∀ dip∈KD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
→ dip∈KD(rt (σ' (the (nhop (rt (σ i)) dip))))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
by (rule basic)

have "∀ dip∈KD(rt (σ i)).
the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip) ≠ dip
→ dip∈KD(rt (σ' (the (nhop (rt (σ i)) dip))))
∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip
≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
(is "∀ dip∈KD(rt (σ i)). _ → ?dip_in_kD dip ∧ ?nsqn_le_nsqn dip")
proof (intro ballI impI, split update_rt_split_asm)
fix dip
assume "dip∈KD(rt (σ i))"
and "the (nhop (rt (σ i)) dip) ≠ dip"
and "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)"
with pre' show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip" by simp
next
fix dip
assume "dip∈KD(rt (σ i))"
and notdip: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip) ≠ dip"
and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)"
show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip"
proof (cases "dip = oip")
assume "dip ≠ oip"
with pre' (dip∈KD(rt (σ i))) notdip
show ?thesis by clarsimp
next
assume "dip = oip"
with rtnot qinc (dip∈KD(rt (σ i))) notdip *
have "?dip_in_kD dip" by simp (metis kD_quality_increases)
moreover from (dip = oip) rtnot qinc (dip∈KD(rt (σ i))) notdip *
have "?nsqn_le_nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)
ultimately show ?thesis ..
qed
}
Note update2 = this

have "opaodv i |= (?S, ?U →) onl Γ AODV (λ(σ, _).
∀ dip ∈ kD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
→ dip ∈ kD(rt (σ (the (nhop (rt (σ i)) dip))))
∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip)"
by (inv_cterms inv add: oseq_step_invariant_sterms [OF equality_increases_nsqn_fresh aodv_wf oadodv_trans]
onl_ooinvariant_sterms [OF aodv_wf odests_vD_inc_sqn]
onl_ooinvariant_sterms [OF aodv_wf oprerr_guard]
onl_ooinvariant_sterms [OF aodv_wf rreq_sip]
onl_ooinvariant_sterms [OF aodv_wf rrep_sip]
onl_ooinvariant_sterms [OF aodv_wf rerr_sip]
other_quality_increases
other_localD
solve: basic basic_prerr
simp add: seqlsimp nsqn_invalidate nhop_update_sip
simp del: One_nat_def)

(rule conjI, erule(2) update1, erule(2) update2)+

thus ?thesis unfolding Let_def by auto

qed

Proposition 7.30
lemmas okD_unk_or_atleast_one =
open_seq_invariant [OF kD_unk_or_atleast_one initiali_aodv,
simplified seql_onl_swap]

lemmas ozero_seq_unk_hops_one =
open_seq_invariant [OF zero_seq_unk_hops_one initiali_aodv,
simplified seql_onl_swap]

lemma oreachable_fresh_okD_unk_or_atleast_one:
  fixes dip
  assumes "((σ, p) ∈ oreachable (opaodv i))
    (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
    ∧ msg_zhops m)))

    (other_quality_increases {i})"
    and "dip∈kD(rt (σ i))"
  shows "π3 (the (rt (σ i) dip)) = unk ∨ 1 ≤ π2 (the (rt (σ i) dip))"
(is "?P dip")

proof -
  have "∃ l. l∈labels ΓAODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip∈kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaodv_trans aodv_trans]],
    auto dest!: otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip∈kD(rt (σ i))⟩ show ?thesis by simp
qed

lemma oreachable_fresh_ozero_seq_unk_hops_one:
  fixes dip
  assumes "((σ, p) ∈ oreachable (opaodv i))
    (otherwith ((=)) {i} (orecvmsg (λσ m. msg_fresh σ m
    ∧ msg_zhops m)))

    (other_quality_increases {i})"
    and "dip∈kD(rt (σ i))"
  shows "sqn (rt (σ i)) dip = 0 → sqnf (rt (σ i)) dip = unk
    ∧ the (dhops (rt (σ i)) dip) = 1
    ∧ the (nhop (rt (σ i)) dip) = dip"
(is "?P dip")

proof -
  have "∃ l. l∈labels ΓAODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip∈kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]],
    auto dest!: onlD otherwith_actionD simp: seqlsimp)
  with ⟨dip∈kD(rt (σ i))⟩ show ?thesis by simp
qed

lemma seq_nhop_quality_increases':
  shows "opaodv i |=( otherwith ((=)) {i})
    (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →)
    onl ΓAODV (λ(σ, _). ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip))
∧ nhip ≠ dip
→ (rt (σ i)) □ dip (rt (σ nhip))"
(is "_ |= (?S i, _ ->) _")

proof
  have weaken:
    "\& I Q R P. p |= (otherwith quality_increases I (orecvmsg Q), other quality_increases I ->) P
      \implies p |= (otherwith ((m)) I (orecvmsg (\lambda \sigma m. Q \sigma m \land R \sigma m)), other quality_increases I ->) P"
    by auto

  fix i a and \sigma \sigma' :: "ip => state"
  assume a1: "\forall dip. dip \in V (rt (\sigma i))
    \land dip \in V (rt (\sigma' (the (nhop (rt (\sigma i)) dip))))
    \land (the (nhop (rt (\sigma i)) dip)) \neq dip
    \implies rt (\sigma i) \sqsubseteq dip rt (\sigma (the (nhop (rt (\sigma i)) dip)))"
  and ov: "?S i \sigma \sigma' a"
  have "\forall dip. dip \in V (rt (\sigma i))
    \land dip \in V (rt (\sigma' (the (nhop (rt (\sigma i)) dip))))
    \land (the (nhop (rt (\sigma i)) dip)) \neq dip
    \implies rt (\sigma i) \sqsubseteq dip rt (\sigma' (the (nhop (rt (\sigma i)) dip)))"

  proof clarify
    fix dip
    assume a2: "dip \in V (rt (\sigma i))"
    and a3: "dip \in V (rt (\sigma' (the (nhop (rt (\sigma i)) dip))))"
    and a4: "(the (nhop (rt (\sigma i)) dip)) \neq dip"
    from ov have "\forall j. j \neq i \implies \sigma j = \sigma' j" by auto
    show "rt (\sigma i) \sqsubseteq dip rt (the (nhop (rt (\sigma i)) dip)))"
    proof (cases "(the (nhop (rt (\sigma i)) dip)) = i")
      assume "(the (nhop (rt (\sigma i)) dip)) = i"
      with dip \in V (rt (\sigma i)) have "dip \in V (rt (\sigma (the (nhop (rt (\sigma i)) dip))))" by simp
      with a1 a2 a4 have "rt (\sigma i) \sqsubseteq dip rt (\sigma (the (nhop (rt (\sigma i)) dip))))" by simp
      with (the (nhop (rt (\sigma i)) dip)) = i have "rt (\sigma i) \sqsubseteq dip rt (\sigma i)" by simp
      hence False by simp
      thus ?thesis ..
    next
      assume "(the (nhop (rt (\sigma i)) dip)) \neq i"
      with \forall j. j \neq i \implies \sigma j = \sigma' j
      have *: "\sigma (the (nhop (rt (\sigma i)) dip)) = \sigma' (the (nhop (rt (\sigma i)) dip)))" by simp
      with dip \in V (rt (\sigma i)) have "dip \in V (rt (\sigma (the (nhop (rt (\sigma i)) dip))))" by simp
      with a1 a2 a4 have "rt (\sigma i) \sqsubseteq dip rt (\sigma (the (nhop (rt (\sigma i)) dip))))" by simp
      with * show ?thesis by simp
    qed
  qed

  } note basic = this

  { fix \sigma \sigma' a dip sip i
    assume a1: "\forall dip. dip \in V (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip))
      \land dip \in V (update (\sigma' (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip)) dip)))))
      \land (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip)) dip)) \neq dip
      \implies update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip)
      \sqsubseteq dip rt (\sigma' (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip)) dip))"
    proof clarify
      fix dip
      assume a2: "dip \in V (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip))"
      and a3: "dip \in V (update (\sigma' (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip)) dip))))"
      and a4: "the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip)) dip) \neq dip"
      show "update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip)
        \sqsubseteq dip rt (\sigma' (the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip)) dip)))"
      proof (cases "dip = sip")
        assume "dip = sip"
    qed

    479
with (\(\text{the (nhop (update (rt (\(\sigma\) i)) sip (0, unk, val, Suc 0, sip)) dip) \neq dip}\))

have False by simp

thus \(?\text{thesis} \ldots\) .

next

assume \([\text{simp}]: \text{"dip \neq sip"}\)

from a2 have "dip\(\in\)\(\ \text{vD}\)(rt (\(\sigma\) i)) \lor dip = sip"

by \([\text{rule vD_update_val}]\)

with \(\text{dip \neq sip} \) have "dip\(\in\)\(\ \text{vD}\)(rt (\(\sigma\) i))" by simp

moreover from a3 have "dip\(\in\)\(\ \text{vD}\)(rt (\(\sigma\)' (the (nhop (rt (\(\sigma\) i)) dip))))" by simp

moreover from a4 have "the (nhop (rt (\(\sigma\) i)) dip) \neq dip" by simp

ultimately have "rt (\(\sigma\) i) \supset dip rt (\(\sigma\)' (the (nhop (rt (\(\sigma\) i)) dip)))"

using a1 ow by \(-\) (drule(1) basic, simp)

with \(\text{dip \neq sip} \) show \(?\text{thesis} \ldots\) .

by \(-\) (erule rt_strictly_fresher_update_other, simp)

qed

qed

} note update_0_unk = this

\{ fix \(\sigma\) a \(\sigma\)' nhop

assume pre: "\(\forall\) dip. dip\(\in\)\(\ \text{vD}\) (rt (\(\sigma\) i)) \land dip\(\in\)\(\ \text{vD}\) (rt (\(\sigma\)' (nhop dip))) \land nhop dip \neq dip

\quad \rightarrow rt (\(\sigma\) i) \supset dip rt (\(\sigma\)' (nhop dip))"

and ow: "\(?S i \ \sigma\ \ \sigma\)' a" 

have "\(\forall\) dip. dip \in \text{vD} (\text{invalidate (rt (\(\sigma\) i)) (dests (\(\sigma\) i)))} \\

\land dip \in \text{vD} (rt (\(\sigma\)' (nhop dip))) \land nhop dip \neq dip

\quad \rightarrow rt (\(\sigma\) i) \supset dip rt (\(\sigma\)' (nhop dip))"

proof clarify

fix dip

assume "dip\(\in\)\(\ \text{vD}\) (\text{invalidate (rt (\(\sigma\) i)) (dests (\(\sigma\) i)))}"

and "dip\(\in\)\(\ \text{vD}\) (rt (\(\sigma\)' (nhop dip)))"

and "nhop dip \neq dip"

from this(1) have "dip\(\in\)\(\ \text{vD}\) (rt (\(\sigma\) i))" by (clarsimp dest!: vD_invalidate_vD_not_dests)

moreover from ow have "\(\forall\) j. j \neq i \rightarrow j = \sigma\)' j" by auto

ultimately have "rt (\(\sigma\) i) \supset dip rt (\(\sigma\)' (nhop dip))"

using pre \(\langle\text{dip} \in \text{vD} (rt (\(\sigma\)' (nhop dip)))\rangle\) \(\langle\text{nhop dip} \neq dip\rangle\) by metis

with \(\langle\forall\) j. j \neq i \rightarrow j = \sigma\)' j show "rt (\(\sigma\) i) \supset dip rt (\(\sigma\)' (nhop dip))"

by (metis rt_strictly_fresher_irefl)

qed

} note invalidate = this

\{ fix \(\sigma\) a \(\sigma\)' dip oip osn sip hops i

assume pre: "\(\forall\) dip. dip \in \text{vD} (rt (\(\sigma\) i)) \\

\land dip \in \text{vD} (rt (\(\sigma\) (the (nhop (rt (\(\sigma\) i)) dip)))) \\

\land (the (nhop (rt (\(\sigma\) i)) dip) \neq dip

\quad \rightarrow rt (\(\sigma\) i) \supset dip rt (\(\sigma\) (the (nhop (rt (\(\sigma\) i)) dip))))"

and ow: "\(?S i \ \sigma\ \ \sigma\)' a"

and "Suc 0 \leq osn"

and a6: "sip \neq oip \rightarrow oip \in kD (rt (\(\sigma\) sip)) \\

\land osn \leq nsqn (rt (\(\sigma\) sip)) oip \\

\land (nsqn (rt (\(\sigma\) sip)) oip = osn \\

\quad \rightarrow (dhop (rt (\(\sigma\) sip)) oip) \leq hops \\

\quad \lor (the (flag (rt (\(\sigma\) sip)) oip) = inv)"

and after: "\(\sigma\)' i = i (rt := update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc hops, sip))"

have "\(\forall\) dip. dip \in \text{vD} (\text{update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc hops, sip)}) \\

\land dip \in \text{vD} (rt (\(\sigma\)' (the (nhop (update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc hops, sip))) dip))) \\

\land (the (nhop (update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc hops, sip)) dip) \neq dip

\rightarrow update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc hops, sip) \\

\supset dip rt (\(\sigma\)' (the (nhop (update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc hops, sip)) dip)))"

proof clarify

fix dip

assume a2: "dip\(\in\)\(\ \text{vD}\) (\text{update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc (hops), sip)})"
and a3: "\( \text{dip} \in vD(\text{rt} (\sigma' \ (\text{the} \ (\text{nhop} \ (\text{update} \ (\text{rt} (\sigma \ i)) \ \text{oip} \\
\quad (\text{osn}, \ \kno, \ \val, \ \ Suc \ \hops, \ \sip))) \ \text{dip}))) \)"

and a4: "the (\text{nhop} \ (\text{update} \ (\text{rt} (\sigma \ i)) \ \text{oip} \ (\text{osn}, \ \kno, \ \val, \ \ Suc \ \hops, \ \sip))) \ \text{dip} \neq \ \text{dip}"

from \( \sigma_0 \) have a5: "\( \forall j. \ j \neq i \rightarrow \sigma_j = \sigma'_j \)" by auto

show "\( \text{update} \ (\text{rt} (\sigma \ i)) \ \text{oip} \ (\text{osn}, \ \kno, \ \val, \ \ Suc \ \hops, \ \sip) \ \text{dip} \)
\[
\subseteq \text{dip}
\]
\( \quad \text{rt} (\sigma' \ (\text{the} \ (\text{nhop} \ (\text{update} \ (\text{rt} (\sigma \ i)) \ \text{oip} \\
\quad (\text{osn}, \ \kno, \ \val, \ \ Suc \ \hops, \ \sip)) \ \text{dip}))) \)"

(is "\( ?rt1 \subseteq \text{dip} \ ?rt2 \ \text{dip} \)"

proof (cases "\( ?rt1 = \text{rt} (\sigma \ i) \)"
assume nochange [simp]:
"\( \text{update} \ (\text{rt} (\sigma \ i)) \ \text{oip} \ (\text{osn}, \ \kno, \ \val, \ \ Suc \ \hops, \ \sip) = \text{rt} (\sigma \ i) \)"

from after have "\( \forall j. \ j \neq i \rightarrow \sigma_j = \sigma'_j \)" by simp

with a5 have "\( \forall j. \ j = \sigma'_j \)" by metis

from a2 have "\( \text{dip} \in vD (\text{rt} (\sigma \ i)) \)" by simp

moreover from a3 have "\( \text{dip} \in vD(\text{rt} (\sigma' (\text{the} (\text{nhop} \ (\text{rt} (\sigma \ i)) \ \text{dip})))) \)"

using nochange and \( \forall j. \ \sigma_j = \sigma'_j \) by clarsimp

moreover from a4 have "the (\text{nhop} \ (\text{rt} (\sigma \ i)) \ \text{dip}) \neq \ \text{dip}" by simp

ultimately have "\( \text{rt} (\sigma \ i) \ \subseteq \text{dip} \ \text{rt} (\sigma' (\text{the} \ (\text{nhop} \ (\text{rt} (\sigma \ i)) \ \text{dip})))) \)"

using pre by simp

hence "\( \text{rt} (\sigma \ i) \ \subseteq \text{dip} \ \text{rt} (\sigma' (\text{the} \ (\text{nhop} \ (\sigma \ i)) \ \text{dip}))) \)"

using \( \forall j. \ \sigma_j = \sigma'_j \) by simp

thus "?thesis" by simp

next
assume change: "\( ?rt1 \neq \text{rt} (\sigma \ i) \)"

from after a2 have "\( \text{dip} \in kD(\text{rt} (\sigma' \ i)) \)" by auto

show ?thesis
proof (cases "\( \text{dip} = \text{oip} \)"
assume "\( \text{dip} \neq \text{oip} \)"

with a2 have "\( \text{dip} \in vD (\text{rt} (\sigma \ i)) \)" by auto

moreover with a3 a5 after and \( \text{dip} \neq \text{oip} \)

have "\( \text{dip} \in vD(\text{rt} (\sigma' (\text{the} (\text{nhop} \ (\text{rt} (\sigma \ i)) \ \text{dip}))))) \)"

by simp metis

moreover from a4 and \( \text{dip} \neq \text{oip} \) have "the (\text{nhop} \ (\text{rt} (\sigma \ i)) \ \text{dip}) \neq \ \text{dip}" by simp

ultimately have "\( \text{rt} (\sigma \ i) \ \subseteq \text{dip} \ \text{rt} (\sigma' (\text{the} \ (\text{nhop} \ (\sigma \ i)) \ \text{dip})) \)"

using pre by simp

with after and a5 and \( \text{dip} \neq \text{oip} \) show ?thesis

by simp (metis rt_strictly_fresher_update_other
\quad rt_strictly_fresher_irefl)

next
assume "\( \text{dip} = \text{oip} \)"

with a4 and change have "\( \text{sip} \neq \text{oip} \)" by simp

with a6 have "\( \text{oip} \in kD(\text{rt} (\sigma \ \text{sip})) \)"

and "\( \text{osn} \leq \text{nsqn} (\text{rt} (\sigma \ \text{sip})) \ \text{oip} \)" by auto

from a3 change \( \text{dip} = \text{oip} \) have "\( \text{oip} \in vD(\text{rt} (\sigma' \ \text{sip})) \)" by simp

hence "\( \text{the} (\text{flag} (\text{rt} (\sigma' \ \text{sip})) \ \text{oip}) = \text{val} \)" by simp

from \( \text{oip} \in kD(\text{rt} (\sigma \ \text{sip})) \)

have "\( \text{osn} < \text{nsqn} (\text{rt} (\sigma' \ \text{sip})) \ \text{oip} \lor (\text{osn} = \text{nsqn} (\text{rt} (\sigma' \ \text{sip})) \ \text{oip} \land \text{the} (\text{dhops} (\text{rt} (\sigma' \ \text{sip})) \ \text{oip}) \leq \text{hops}) \)"

proof
assume "\( \text{oip} \in vD(\text{rt} (\sigma \ \text{sip})) \)"

hence "\( \text{the} (\text{flag} (\text{rt} (\sigma \ \text{sip})) \ \text{oip}) = \text{val} \)" by simp

with a6 \( \text{sip} \neq \text{oip} \) have "\( \text{nsqn} (\text{rt} (\sigma \ \text{sip})) \ \text{oip} = \text{osn} \rightarrow \text{the} (\text{dhops} (\text{rt} (\sigma \ \text{sip})) \ \text{oip}) \leq \text{hops} \)"

by simp

show ?thesis
proof (cases "\( \text{sip} = \text{i} \)"

assume "sip \neq i"
with a5 have "\sigma \ sip = \sigma' \ sip" by simp
with \osn \leq \ nsqn (rt (\sigma \ sip)) \ oip\nand \nsqn (rt (\sigma \ sip)) \ oip = \osn \rightarrow \the (dhops (rt (\sigma \ sip)) \ oip) \leq \hops"
show \?thesis by auto

next
— alternative to using sip_not_ip
assume [simp]: "sip = i"

have "?rt1 = rt (\sigma \ i)"
proof (rule update_cases_kD, simp_all)
  from \Suc 0 \leq \osn: show "0 < \osn" by simp
next
  from \oip\in kD(rt (\sigma \ sip)) and \sip = i: show "oip\in kD(rt (\sigma \ i))"
  by simp
next
assume "sqn (rt (\sigma \ i)) \ oip < \osn"
also from \osn \leq \ nsqn (rt (\sigma \ sip)) \ oip:
  have "\ldots \leq \ nsqn (rt (\sigma \ i)) \ oip" by simp
also have "\ldots \leq \ sqn (rt (\sigma \ i)) \ oip"
  by (rule nsqn_sqn)
finally have "sqn (rt (\sigma \ i)) \ oip < sqn (rt (\sigma \ i)) \ oip".
  hence False by simp
  thus "(\lambda a. if a = oip
    then Some (\osn, kno, val, Suc \ hops, i)
    else rt (\sigma \ i) a) = rt (\sigma \ i)"..
next
assume "sqn (rt (\sigma \ i)) \ oip = \osn"
  and "Suc \ hops < the (dhops (rt (\sigma \ i)) \ oip)"
from this(1) and \oip \in vD (rt (\sigma \ sip)): have "nsqn (rt (\sigma \ i)) \ oip = \osn"
  by simp
with \nsqn (rt (\sigma \ sip)) \ oip = \osn \rightarrow \the (dhops (rt (\sigma \ sip)) \ oip) \leq \hops:
  have "the (dhops (rt (\sigma \ i)) \ oip) \leq \hops" by simp
with \Suc \ hops < the (dhops (rt (\sigma \ i)) \ oip): have False by simp
  thus "(\lambda a. if a = oip
    then Some (\osn, kno, val, Suc \ hops, i)
    else rt (\sigma \ i) a) = rt (\sigma \ i)"..
next
assume "the (flag (rt (\sigma \ i)) \ oip) = inv"
  with the (flag (rt (\sigma \ sip))) \ oip: have False by simp
  thus "(\lambda a. if a = oip
    then Some (\osn, kno, val, Suc \ hops, i)
    else rt (\sigma \ i) a) = rt (\sigma \ i)"..
next
from \oip\in kD(rt (\sigma \ sip))
  show "(\lambda a. if a = oip then Some (the (rt (\sigma \ i)) \ oip)) else rt (\sigma \ i) a) = rt (\sigma \ i)"
    by (auto dest!: kD_Some)
qed
with change have False ..
  thus ?thesis ..
qed

next
assume "oip\in iD(rt (\sigma \ sip))"
with \the (flag (rt (\sigma' \ sip)) \ oip) = val\ and a5 have "sip = i"
  by (metis f.distinct(1) iD_flag_is_inv)
from \oip\in iD(rt (\sigma \ sip)): have "the (flag (rt (\sigma \ sip)) \ oip) = inv" by auto
with \sip = i: \Suc 0 \leq \osn: change after \oip\in kD(rt (\sigma \ sip))
  have "nsqn (rt (\sigma \ sip)) \ oip < nsqn (rt (\sigma' \ sip)) \ oip"
    unfolding update_def
    by (clarsimp split: option.split_asm if_split_asm)
      (auto simp: sqn_def)
with \osn \leq \ nsqn (rt (\sigma \ sip)) \ oip: have "\osn < \nsqn (rt (\sigma' \ sip)) \ oip"
  by simp
  thus ?thesis ..
qed
thus \(\text{thesis}\)

proof

assume osnlt: "osn < nsqn (rt (\(\sigma\)' sip)) oip"

from \(\langle\text{dip} \in kD(rt (\(\sigma\)' i))\rangle\) and \(\langle\text{dip} = oip\rangle\) have "\(\text{dip} \in kD(?rt1)\)" by simp

moreover from a3 have "\(\text{dip} \in kD(?rt2 dip)\)" by simp

moreover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"

proof -

have "nsqn ?rt1 oip = osn"

by (simp add: \(\langle\text{dip} = oip\rangle\) nsqn_update_changed_kno_val [OF change [THEN not_sym]])

also have "... < nsqn (rt (\(\sigma\)' sip)) oip" using osnlt .

also have "... = nsqn (?rt2 oip) oip" by (simp add: change)

finally show \(\text{thesis}\)

using \(\langle\text{dip} = oip\rangle\) by simp

qed

ultimately show \(\text{thesis}\)

by (rule rt_strictly_fresher_ltI)

next

assume osneq: "osn = nsqn (rt (\(\sigma\)' sip)) oip ∧ the (dhops (rt (\(\sigma\)' sip)) oip) \(\leq\) hops"

have "oip \(\in\) kD(?rt1)" by simp

moreover from a3 \(\langle\text{dip} = oip\rangle\) have "oip \(\in\) kD(?rt2 oip)" by simp

moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"

proof -

from osneq have "osn = nsqn (rt (\(\sigma\)' sip)) oip" ..

also have "osn = nsqn ?rt1 oip"

by (simp add: \(\langle\text{dip} = oip\rangle\) nsqn_update_changed_kno_val [OF change [THEN not_sym]])

also have "nsqn (rt (\(\sigma\)' sip)) oip = nsqn (?rt2 oip) oip"

by (simp add: change)

finally show \(\text{thesis}\)

qed

ultimately have "\(\pi_5(\text{the (?rt2 oip oip)}) < \pi_5(\text{the (?rt1 oip)})\)"

proof -

from osneq have "the (dhops (rt (\(\sigma\)' sip)) oip) \(\leq\) hops" ..

moreover from \(\langle\text{oip} \in vD (rt (\(\sigma\)' sip))\rangle\) have "oip \(\in\) D(rt (\(\sigma\)' sip))" by auto

ultimately have "\(\pi_5(\text{the (rt (\(\sigma\)' sip) oip)}) \(\leq\) hops\)"

by (auto simp add: proj5_eq_dhops)

also from change after have "hops < \(\pi_5(\text{the (rt (\(\sigma\)' i) oip)})\)"

by (simp add: proj5_eq_dhops) (metis dhops_update_changed lessI)

finally have "\(\pi_5(\text{the (rt (\(\sigma\)' sip) oip)}) < \pi_5(\text{the (rt (\(\sigma\)' i) oip)})\)" .

with change after show \(\text{thesis}\) by simp

qed

ultimately have "\(?rt1 \subseteq oip \ ?rt2 oip\)"

by (rule rt_strictly_fresher_eqI)

with \(\langle\text{dip} = oip\rangle\) show \(\text{thesis}\) by simp

qed

qed

note rreq_rrep_update = this

have "opaodv i \implies\ (otherwith ((\oplus)) \{i\} (orecmsg (\(\lambda\sigma m. msg\_fresh \sigma m \
∧ msg\_zhops m)))
∧ other\_quality\_increases \{i\} \rightarrow)

\(\text{onl } \Gamma_{AODV}\)

(\(\lambda(\sigma, _). \forall\text{dip}. \text{dip} \in vD (rt (\(\sigma\) i)) \cap vD (rt (\sigma (the (nhop (rt (\(\sigma\) i)) dip)))) \\wedge \text{the (nhop (rt (\(\sigma\) i)) dip)} \neq \text{dip} \rightarrow \text{rt (\(\sigma\) i) \subseteq dip rt (\sigma (the (nhop (rt (\(\sigma\) i)) dip))))}\)

proof (inv_cterms inv add: onl_oinvariant_sterms [OF aodv_wf rreq_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rerr_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]])
hence

or: "\((\sigma, p) \in \text{reachable} \ (\text{opaodv} \ i) \ (\exists S \ i) \ \text{(other quality Increases} \ \{i\})\)"

and "other quality Increases \ \{i\} \ \sigma \ \sigma'"

and ll: "\(l \in \text{labels} \ \Gamma_{\text{AODV}} \ p\)"

and pre: "\(\forall \ dip. \ dip \in vD \ (rt \ (\sigma \ i))\)
\(\wedge \ dip \in vD \ (rt \ (\sigma \ i)) \wedge \ text{the (nhop (rt (\sigma \ i)) dip)) \wedge \ the \ (nhop (rt (\sigma \ i)) dip) \neq dip \rightarrow rt (\sigma \ i) \sqsubseteq dip \ rt (\sigma \ i) \sqsubseteq dip (\sigma \ (\text{nhop (rt (\sigma \ i)) dip}))\)"

from this(1-2)

have or': "\((\sigma', p) \in \text{reachable} \ (\text{opaodv} \ i) \ (\exists S \ i) \ \text{(other quality Increases} \ \{i\})\)"

by - (rule reachable_other')

from or and ll have next_hop: "\(\forall \ dip. \ let \ nhip = \text{the (nhop (rt (\sigma \ i)) dip)} \in dip \in kD (rt (\sigma \ i)) \wedge nhip \neq dip \rightarrow dip \in kD (rt (\sigma \ nhip)) \wedge nsqn (rt (\sigma \ i)) dip \leq nsqn (rt (\sigma \ nhip)) dip\)"

by (auto dest!: onl_invariant_weakenD [OF seq_compare_next_hop'])

from or and ll have unk_hops_one: "\(\forall \ dip \in kD \ (rt (\sigma \ i)) \cdot sqn (rt (\sigma \ i)) dip = 0 \rightarrow sqnf (rt (\sigma \ i)) dip = unk \wedge \ the (dhops (rt (\sigma \ i)) dip)) = 1 \wedge \ the (nhop (rt (\sigma \ i)) dip) = dip\)"

by (auto dest!: onl_invariant_weakenD [OF ozero_seq_unk_hops_one [OF oadv_trans aodv_trans]]

otherwith_actionD

simp: seqlsimp)

from \text{other quality Increases} \ \{i\} \ \sigma \ \sigma' have "\(\sigma' \ i = \sigma \ i\)" by auto

hence "\text{quality Increases} \ (\sigma \ i) \ (\sigma' \ i)" by auto

with \text{other quality Increases} \ \{i\} \ \sigma \ \sigma' have "\(\forall j. \ \text{quality Increases} \ (\sigma \ j) \ (\sigma' \ j)\)"

by - (rule otherE, metis singleton_iff)

show "\(\forall \ dip. \ dip \in vD \ (rt (\sigma' \ i)) \wedge dip \in vD \ (rt (\sigma' \ i)) \wedge dip \in kD (rt (\sigma \ i))) \wedge dip \in kD (rt (\sigma \ i)) \wedge dip \in kD (rt (\sigma \ i)) \wedge dip = unk \wedge the (nhop (rt (\sigma \ i))) dip \not= dip \rightarrow rt (\sigma' \ i) \sqsubseteq dip \ rt (\sigma' \ i) \sqsubseteq dip (\sigma \ (the (nhop (rt (\sigma \ i)) dip)))\)"

proof clarify

fix dip

assume "dip \in vD (rt (\sigma' \ i))"

and "dip \in vD (rt (\sigma' \ (the (nhop (rt (\sigma' \ i)) dip)))"

and "the (nhop (rt (\sigma' \ i)) dip) \not= dip"

from this(i) and \(\sigma' \ i = \sigma \ i\) have "dip \in vD (rt (\sigma \ i))"

and "dip \in kD (rt (\sigma \ i))"

by auto

from \(the \ (nhop (rt (\sigma' \ i)) dip) \not= dip\) and \(\sigma' \ i = \sigma \ i\)

have "the (nhop (rt (\sigma \ i)) dip) \not= dip" (is "\(?nhip \not= \)"") by simp

with \(dip \in kD (rt (\sigma \ i))\) and next hop

have "dip \in kD (rt (\sigma \ ?nhip))"

and nsqns: "nsqn (rt (\sigma \ i)) dip \leq nsqn (rt (\sigma \ ?nhip)) dip"

by (auto simp: Let_def)

have "0 < sqn (rt (\sigma \ i)) dip"

proof (rule neqD_conv [THEN iffD1, OF notI])

assume "sqn (rt (\sigma \ i)) dip = 0"

with \(dip \in kD (rt (\sigma \ i))\) and unk_hops_one

have "?nhip = dip" by simp

with \(?nhip \not= dip\) show False ..

qed

also have "... = nsqn (rt (\sigma \ i)) dip"

by (rule vD_slnsqn_sqn [OF \(dip \in vD (rt (\sigma \ i))\), THEN sym])
also have \( \ldots \leq \text{nsqn} (\text{rt} (\sigma \ ?nhip)) \text{ dip} \)
by (rule nsqns)
also have \( \ldots \leq \text{sqn} (\text{rt} (\sigma \ ?nhip)) \text{ dip} \)
by (rule nsqn_sqn)
finally have \( 0 < \text{sqn} (\text{rt} (\sigma \ ?nhip)) \text{ dip} \).

have \( \text{rt} (\sigma \ i) \sqsubseteq \text{dip} (\sigma' \ ?nhip) \)
proof (cases \( \text{dip} \in \text{vD}(\text{rt} (\sigma \ ?nhip)) \))
assume \( \text{dip} \in \text{vD}(\text{rt} (\sigma \ ?nhip)) \)
with \( \text{pre} \ (\text{dip} \in \text{vD}(\text{rt} (\sigma \ i))) \) and \( ?nhip \neq \text{dip} \)
have \( \text{rt} (\sigma \ i) \sqsubseteq \text{dip} (\sigma \ ?nhip) \) by auto
moreover from \( \forall j. \text{quality\_increases} (\sigma \ j) (\sigma' \ j) \)
have \( \text{quality\_increases} (\sigma \ ?nhip) (\sigma' \ ?nhip) \)
ultimately show \( \text{thesis} \)
using \( \text{dip} \in \text{kD}(\text{rt} (\sigma \ ?nhip)) \)
by (rule strictly\_fresher\_quality\_increases\_right)
next
assume \( \text{dip} \notin \text{vD}(\text{rt} (\sigma \ ?nhip)) \)
with \( \text{dip} \in \text{kD}(\text{rt} (\sigma' \ ?nhip)) \)
have \( \ldots = \text{sqn} (\text{rt} (\sigma \ ?nhip)) \text{ dip} - 1 \)
also have \( \ldots < \text{sqn} (\text{rt} (\sigma' \ ?nhip)) \text{ dip} \)
proof -
from \( \forall j. \text{quality\_increases} (\sigma \ j) (\sigma' \ j) \)
have \( \text{quality\_increases} (\sigma \ ?nhip) (\sigma' \ ?nhip) \)
hence \( \forall \text{ip}. \text{sqn} (\text{rt} (\sigma \ ?nhip)) \text{ ip} \leq \text{sqn} (\text{rt} (\sigma \ ?nhip)) \text{ dip} \) by auto
with \( 0 < \text{sqn} (\text{rt} (\sigma \ ?nhip)) \text{ dip} \) show \( \text{thesis} \) by auto
qed
also have \( \ldots = \text{sqn} (\text{rt} (\sigma \ ?nhip)) \text{ dip} \)
proof (rule vD_nsqn_sqn [THEN sym])
from \( \text{dip} \in \text{vD}(\text{rt} (\sigma' \ (the \ (\text{nhop} (\text{rt} (\sigma' \ i)) \text{ dip})))) \) and \( \sigma' \ i = \sigma \ i \)
show \( \text{dip} \in \text{vD}(\text{rt} (\sigma' \ ?nhip)) \) by simp
qed
finally have \( \text{sqn} (\text{rt} (\sigma \ i)) \text{ dip} < \text{sqn} (\text{rt} (\sigma \ ?nhip)) \text{ dip} \).

moreover from \( \text{dip} \in \text{vD}(\text{rt} (\sigma' \ (the \ (\text{nhop} (\text{rt} (\sigma' \ i)) \text{ dip})))) \) and \( \sigma' \ i = \sigma \ i \)
have \( \text{dip} \in \text{kD}(\text{rt} (\sigma' \ ?nhip)) \) by auto
ultimately show \( \text{rt} (\sigma \ i) \sqsubseteq \text{dip} (\sigma' \ ?nhip) \)
using \( \text{dip} \in \text{kD}(\text{rt} (\sigma \ i)) \) by - (rule rt\_strictly\_fresher\_ltI)
qed
with \( \sigma' \ i = \sigma \ i \) show \( \text{rt} (\sigma \ i) \sqsubseteq \text{dip} (\sigma' \ (the \ (\text{nhop} (\text{rt} (\sigma' \ i)) \text{ dip}))) \)
by simp
qed

thus \( \text{thesis} \) unfolding Let\_def.
qed

lemma seq\_compare\_next\_hop:
fixes \( w \)
shows \( \text{opaodv} \ i \models (\text{otherwith} (\text{(=)}) \ {i}) \ (\text{orecvmsg} \ \text{msg\_fresh}, \text{other quality\_increases} \ {i} \rightarrow) \)
\begin{align*}
global (\lambda \sigma. \forall \text{dip}. \text{let nhip} = \text{the} (\text{nhop} (\text{rt} (\sigma \ i)) \text{ dip})
\text{ in dip} \in \text{kD}(\text{rt} (\sigma \ i)) \land \text{nhip} \neq \text{dip} \rightarrow \\
\text{ dip} \in \text{kD}(\text{rt} (\sigma \ \text{nhip}) ) \land \text{nsqn} (\text{rt} (\sigma \ i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma \ \text{nhip})) \text{ dip} )
\end{align*}
by (rule oinvariant\_weakenE [OF seq\_compare\_next\_hop']) (auto dest!: onlD)

lemma seq\_nhop\_quality\_increases:
shows \( \text{opaodv} \ i \models (\text{otherwith} (\text{(=)}) \ {i}) \)
5.10 Routing graphs and loop freedom

theory E_Loop_Freedom
imports E_Aodv_Predicates E_Fresher
begin

Define the central theorem that relates an invariant over network states to the absence of loops in the associate routing graph.

definition
rt_graph :: "(ip ⇒ state) ⇒ ip ⇒ ip rel"
where
"rt_graph σ = (λdip. {(ip, ip') | ip ip' dsn dsk hops.
 ip ≠ dip ∧ rt (σ ip) dip = Some (dsn, dsk, val, hops, ip')})"

Given the state of a network σ, a routing graph for a given destination ip address dip abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

lemma rt_graphE [elim]:
fixes n dip ip ip'
assumes "(ip, ip') ∈ rt_graph σ dip"
shows "ip ≠ dip ∧ (∃r. rt (σ ip) = r ∧ (∃dsn dsk hops. r dip = Some (dsn, dsk, val, hops, ip')))"
using assms unfolding rt_graph_def by auto

lemma rt_graph_vD [dest]:
"\(\forall ip ip' σ. (ip, ip') ∈ rt_graph σ dip \implies dip ∈ vD(σ ip)\)"
unfolding rt_graph_def vD_def by auto

lemma rt_graph_vD_trans [dest]:
"\(\forall ip ip' σ. (ip, ip') ∈ (rt_graph σ dip) + \implies dip ∈ vD(σ ip)\)"
by (erule converse_tranclE) auto

lemma rt_graph_not_dip [dest]:
"\(\forall ip ip' σ. (ip, ip') ∈ rt_graph σ dip \implies ip ≠ dip\)"
unfolding rt_graph_def by auto

lemma rt_graph_not_dip_trans [dest]:
"\(\forall ip ip' σ. (ip, ip') ∈ (rt_graph σ dip) + \implies ip ≠ dip\)"
by (erule converse_tranclE) auto

NB: the property below cannot be lifted to the transitive closure

lemma rt_graph_nhip_is_nhop [dest]:
"\(\forall ip ip' σ. (ip, ip') ∈ rt_graph σ dip \implies ip' = the (nhop (σ ip)) dip\)"
unfolding rt_graph_def by auto

theorem inv_to_loop_freedom:
assumes "\(∀ i dip. let nhip = the (nhop (σ i)) dip\)
in dip ∈ vD (σ i) ∩ vD (σ nhip) ∧ nhip ≠ dip
\implies (rt (σ i)) ⊏ dip (rt (σ nhip))"
shows "\(∀ dip. irrefl ((rt_graph σ dip)+)\)"
using assms proof (intro allI)
fix σ :: "ip ⇒ state" and dip
assume inv: "\(∀ ip dip. let nhip = the (nhop (σ ip)) dip\)

in dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip)) ∧
    nhip ≠ dip → rt (σ ip) ⊑ dip rt (σ nhip)

{ fix ip' }
assume "(ip', ip') ∈ (rt_graph σ dip)⁺"  
  and "dip ∈ vD(rt (σ ip'))"  
  and "ip' ≠ dip"
hence "rt (σ ip) ⊑ dip rt (σ ip')"
proof induction

fix nhip
assume "(ip, nhip) ∈ rt_graph σ dip"  
  and "dip ∈ vD(rt (σ ip))"  
  and "nhip ≠ dip"
from ⟨(ip, nhip) ∈ rt_graph σ dip⟩ have "dip ∈ vD(rt (σ ip))"  
  and "nhip = the (nhop (rt (σ ip)) dip)"
  by (clarsimp simp: Let_def)
from ⟨dip ∈ vD(rt (σ ip))⟩ and ⟨dip ∈ vD(rt (σ nhip))⟩
  have "dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip))" ..
with "nhip = the (nhop (rt (σ ip)) dip)"
  and "nhip ≠ dip"
  and inv
show "rt (σ ip) ⊑ dip rt (σ nhip)"
  by (clarsimp simp: Let_def)
next

fix nhip nhip'
assume "(ip, nhip) ∈ (rt_graph σ dip)⁺" 
  and "(nhip, nhip') ∈ rt_graph σ dip" 
  and IH: "[[ dip ∈ vD(rt (σ nhip)); nhip ≠ dip ]] ⇒ rt (σ ip) ⊑ dip rt (σ nhip)"
  and "dip ∈ vD(rt (σ nhip'))"  
  and "nhip' ≠ dip"
from ⟨(nhip, nhip') ∈ rt_graph σ dip⟩ have 1: "dip ∈ vD(rt (σ nhip))"  
  and 2: "nhip ≠ dip"
  and "nhip' = the (nhop (rt (σ nhip)) dip)"
  by auto
from 1 2 have "rt (σ ip) ⊑ dip rt (σ nhip)" by (rule IH)
also have "rt (σ nhip) ⊑ dip rt (σ nhip')"  
  proof -
  from ⟨dip ∈ vD(rt (σ nhip))⟩ and ⟨dip ∈ vD(rt (σ nhip'))⟩ 
  have "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
  with "nhip' ≠ dip" 
    and "nhip' = the (nhop (rt (σ nhip)) dip)" 
    and inv
  show "rt (σ nhip) ⊑ dip rt (σ nhip')" 
    by (clarsimp simp: Let_def)
  qed
finally show "rt (σ ip) ⊑ dip rt (σ nhip')" .
note fresher = this }
show "irrefl ((rt_graph σ dip)⁺)"
unfolding irrefl_def proof (intro allI notI)

fix ip
assume "(ip, ip) ∈ (rt_graph σ dip)⁺" 
  moreover then have "dip ∈ vD(rt (σ ip))"  
    and "ip ≠ dip"
  by auto
ultimately have "rt (σ ip) ⊑ dip rt (σ ip)" by (rule fresher)
thus False by simp
qed

end

5.11 Lift and transfer invariants to show loop freedom

theory E_Aodv_Loop_Freedom
importsAWN.OClosed_TransferAWN.Qmsg_LiftingEGlobal_InvariantsE_Loop_Freedom

begin

5.11.1 Lift to parallel processes with queues

lemma par_step_no_change_on_send_or_receive:
  fixes σ s a σ' s'
  assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos ΓADV i) (seqp_sos ΓQMSG)"
  and "a ≠ τ"
  shows "σ' i = σ i"
using assms by (rule qmsg_no_change_on_send_or_receive)

lemma par_nhop_quality_increases:
  shows "opaodv i ⟨⟨qmsg ⟩⟩ (otherwith ((=)) {i}) (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m))
  other_quality_increases {i} →
  global (λσ. ∀ dip. let nhip = the (nhop (rt (σ i)) dip)
  in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
  → (rt (σ i)) ⊏ dip (rt (σ nhip)))"
proof (rule lift_into_qmsg [OF seq_nhop_quality_increases])
show "opaodv i =A (otherwith ((=)) {i}) (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other_quality_increases {i} →
  globala (λσ, _. σ'). quality_increases (σ i) (σ' i))"
proof (rule ostep_invariant_weakenE [OF olocal_quality_increases], simp_all)
fix t :: "(nat ⇒ state) × (state, msg, pseq, pseq label) seqp", msg seq_action) transition"
assume "call ΓADV (λ((σ, _, (σ', _)). ∀ j. quality_increases (σ j) (σ' j)) t"
thus "quality_increases (fst (fst t)) i (fst (snd (snd t))) i"
by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)
qed

lemma par_rreq_rrep_sn_quality_increases:
"opaodv i ⟨⟨qmsg ⟩⟩ (otherwith ((=)) {i}) (orecvmsg (λσ . rreq_rrep_sn) σ, other (λ_. True) {i} →
  globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
proof -
have "opaodv i =A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →
  globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i)))"
proof (rule ostep_invariant_weakenE [OF olocal_quality_increases])
(auto dest!: onllD seq1D elim!: aodv_ex_labelE)
by (rule lift_step_into_qmsg_statelessassm) simp_all
hence "opaodv i ⟨⟨qmsg ⟩⟩ =A (λσ __. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →
  globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thus thesis by rule auto
qed

lemma par_rreq_rrep_nsqn_fresh_any_step:
"opaodv i ⟨⟨qmsg ⟩⟩ =A (λσ __. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →
  globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
proof -
have "opaodv i =A (λσ __. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_. True) {i} →
  globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
proof (rule ostep_invariant_weakenE [OF rreq_rrep_nsqn_fresh_any_step_invariant])
fix t
assume "call ΓADV (λ((σ, a, _). anycast (msg_fresh σ) a) t"
thus "globala (λ(σ, a, σ'). anycast (msg_fresh σ) a) t" by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)

488
qed auto
hence "opaodv i ((i qmsg = A (λσ _. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)
globala (λ(σ, a, _). anycast (msg_fresh σ) a)"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

lemma par_anycast_msg_zhops:
  shows "opaodv i ((i qmsg = A (λσ _. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)
globala (λ(_, a, _). anycast msg_zhops a)"
proof -
  from anycast_msg_zhops initiali_aodv oaodv_trans aodv_trans
  have "opaodv i ((act TT, other (λ_ _. True) {i} →)
  seqll i (onll Γ (λ(_, a, _). anycast msg_zhops a))) t"
    by (rule open_seq_step_invariant)
  hence "opaodv i ((λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
globala (λ(_, a, _). anycast msg_zhops a)"
    by (rule ostep_invariant_weakenE)
  fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
  assume "seqll i (onll Γ (λ(_, a, _). anycast msg_zhops a)) t"
  thus "globala (λ(_, a, _). anycast msg_zhops a) t"
    by (cases t) (clarsimp dest!: seqllD onllD, metis aodv_ex_label)
qed simp_all
hence "opaodv i ((i qmsg = A (λσ _. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)
globala (λ(_, a, _). anycast msg_zhops a)"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

5.11.2 Lift to nodes

lemma node_step_no_change_on_send_or_receive:
  assumes "((σ, NodeS i P R), a, (σ', NodeS i' P' R')) ∈ onode_sos
(oparp_sos i (oseqp_sos Γ AODV i) (seqp_sos Γ QMSG))"
  and "a = τ"
  shows "σ' i = σ i"
using assms
  by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

lemma node_nhop_quality_increases:
  shows "((i : opaodv i ((i qmsg = A (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
globala (λ(_, a, _). castmsg (msg_fresh σ) a))) t"
by (rule node_lift [OF par_nhop_quality_increases]) auto

lemma node_quality_increases:
  shows "((i : opaodv i ((i qmsg = A (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
globala (λ(_, a, _). quality_increases σ i)) t"
by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

lemma node_rreq_rrep_nsqn_fresh_any_step:
  shows "((i : opaodv i ((i qmsg = A (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
globala (λ(σ, a, _). castmsg (msg_fresh σ) a)) t"
by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

lemma node_anycast_msg_zhops:
  shows "((i : opaodv i ((i qmsg = A (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
globala (λ(_, a, _). anycast msg_zhops a)) t"
by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp
globala (λ(_, a, _). castmsg msg_zhops a)"
by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

lemma node_silent_change_only:
shows "\( i : \text{opaadv} i \langle\i qmsg : R_i \rangle_o \models (\lambda \sigma _. \text{arrivmsg} (\lambda _. \text{True}) \sigma, \text{other} (\lambda _. \text{True}) \{i\} \to)\)
globala (λ(σ, a, σ'). a ≠ τ → σ' = i = σ i)"
proof (rule ostep_invariantI, simp (no_asm), rule impl)
fix σ ζ a σ' ζ'
assume or: "(σ, ζ) ∈ oreachable \((i : \text{opaadv} i \langle\i qmsg : R_i \rangle_o)\)
(λσ _. \text{arrivmsg} (λ_. \text{True}) σ)
(\text{other} (λ_. \text{True}) \{i\})"
and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans \((i : \text{opaadv} i \langle\i qmsg : R_i \rangle_o)\)
"a ≠ τ_n" from or obtain p R where "ζ = NodeS i p R"
by (drule node_net_state, metis)
with tr have "((σ, NodeS i p R), a, (σ', ζ')) ∈ onode_sos (oparp_sos i (trans (opaadv i)) (trans qmsg))"
by simp
thus "σ' = i = σ i" using (a ≠ τ_n)
by (cases rule: onode_sos_cases)
(auto elim: qmsg_no_change_on_send_or_receive)
qed

5.11.3 Lift to partial networks

lemma arrive_rreq_rrep_nsqn_fresh_inc_sn [simp]:
assumes "oarrivmsg (λσ m. msg_fresh σ m ∧ P σ m) σ m"
shows "oarrivmsg (λ_. rreq_rrep_sn) σ m"
using assms by (cases m) auto

lemma opnet_nhup_quality_increases:
shows "\( opnet (λi. \text{opaadv} i \langle\i qmsg : R_i \rangle p \models\)
(without ((=)) (net_tree_ips p))
(oarrivmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)), \text{other quality increases} (\text{net_tree_ips} p) \to)\)
global (λσ. \forall i∈\text{net_tree_ips} p. \forall dip.
let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) \cap vD (rt (σ nhip)) ∧ nhip ≠ dip
→ (rt (σ i)) ⊏ dip (rt (σ nhip)))"
proof (rule pnet_lift [OF node_nhup_quality_increases])
fix i R
have "\( i : \text{opaadv} i \langle\i qmsg : R_i \rangle_o \models (\lambda \sigma _. \text{arrivmsg} (\lambda_. \text{rreq_rrep_sn}) σ, \text{other} (\lambda_. \text{True}) \{i\} \to)\) globala (λ(σ, a, σ')).
castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
proof (rule ostep_invariantI, simp (no_asm))
fix σ s a σ' s'
assume or: "(σ, s) ∈ oreachable \((i : \text{opaadv} i \langle\i qmsg : R_i \rangle_o)\)
(λσ _. \text{arrivmsg} (λ_. \text{rreq_rrep_sn}) σ)
(\text{other} (λ_. \text{True}) \{i\})"
and tr: "((σ, s), a, (σ', s')) ∈ trans \((i : \text{opaadv} i \langle\i qmsg : R_i \rangle_o)\)
"from or tr am have "castmsg (msg_fresh σ) a"
by (auto dest!: ostep_invariantD [OF node_rreq_rrep_nsqn_fresh_any_step])
moreover from or tr am have "castmsg (msg_zhops) a"
by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
ultimately show "castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a"
by (case_tac a) auto
qed
thus "\( i : \text{opaadv} i \langle\i qmsg : R_i \rangle_o \models (\lambda \sigma _. \text{arrivmsg} (\lambdaσ m. msg_fresh σ m ∧ msg_zhops m) σ, \text{other quality increases} \{i\} \to)\) globala (λ(σ, a, _).
castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
by rule auto
fix i R
show "(i : opaodv i (⟨i qmsg : R⟩)) |−
(λσ _. oarrivemsg (λm. msg_fresh σ m ∧ msg_zhops m) σ,
other quality_increases {i} →) globala (λ(σ, a, σ').
a = τ ∧ (∀d. a ≠ i:deliver(d)) → σ i = σ' i)"
by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto

5.11.4 Lift to closed networks

lemma onet_nhop_quality_increases:
shows "oclosed (opnet (λi. opaodv i ⟨⟨i qmsg) p)
|− (λ_:_. True, other quality_increases (net_tree_ips p) →)
global (λσ. ∀i∈net_tree_ips p. ∀dip.
let nhip = the (nhop (rt (σ i)) dip)
in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
→ (rt (σ i)) ⊏ dip (rt (σ nhip)))"
(is "_ |− (_, ?U →) ?inv")
proof (rule inclosed_closed)
from opnet_nhop_quality_increases
show "opnet (λi. opaodv i ⟨⟨i qmsg) p
|− (otherwith ((=)) (net_tree_ips p) inoclosed, ?U →) ?inv"
proof (rule oinvariant_weakenE)
fix σ σ' :: "ip ⇒ state" and a :: "msg node_action"
assume "otherwith ((=)) (net_tree_ips p) inoclosed σ σ' a"
thus "otherwith ((=)) (net_tree_ips p)
(oarrivemsg (λm. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"
proof (rule otherwithEI)
fix σ :: "ip ⇒ state" and a :: "msg node_action"
assume "inoclosed σ a"
thus "oarrivemsg (λm. msg_fresh σ m ∧ msg_zhops m) σ a"
proof (cases a)
fix ii ni ms
assume "a = ii−ni:arrive(ms)"
moreover with :inoclosed σ a: obtain d di where "ms = newpkt(d, di)"
by (cases ms) auto
ultimately show ?thesis by simp
qed simp_all
qed
qed

5.11.5 Transfer into the standard model

interpretation aodv_openproc: openproc paodv opaodv id
rewrites "aodv_openproc.initmissing = initmissing"
proof -
show "openproc paodv opaodv id"
proof unfold_locales
fix i :: ip
have "{(σ, ζ). (σ i, ζ) ∈ σ AODV i ∧ (∀j. j ≠ i → σ j ∈ fst (σ AODV j))} ⊆ σ AODV"" unfolding σ AODV_def σ AODV'_def
proof (rule equalityD1)
show "∀f p. {(σ, ζ). (σ i, ζ) ∈ {(f i, p)} ∧ (∀j. j ≠ i → σ j ∈ fst (f j, p))} = {(f, p)}" by (rule set_eqI) auto

491
thus "\{(\sigma, \zeta) \mid \sigma \zeta s. s \in \text{init (paodv i)}
\wedge (\sigma i, \zeta) = \text{id s}
\wedge (\forall j. j \neq i \rightarrow \sigma j \in (\text{fst o id} \ ' \ \text{init (paodv j)})) \subseteq \text{init (opaodv i)}\"
by simp
next
show "\forall j. \text{init (paodv j)} \neq \{\}"
unfolding \sigma_{AODV-def} by simp
next
fix i s a s' \sigma \sigma'
assume "\sigma i = \text{fst (id s)}"
and "\sigma' i = \text{fst (id s')}"
and "(s, a, s') \in \text{trans (paodv i)}"
then obtain q q'
where "s = (\sigma i, q)"
and "s' = (\sigma' i, q')"
and "((\sigma i, q), a, (\sigma' i, q')) \in \text{trans (paodv i)}"
by (cases s, cases s') auto
from this(3)
have "((\sigma, q), a, (\sigma', q')) \in \text{trans (opaodv i)}"
by simp (rule open_seqp_action [OF aodv_wf])
with \langle s = (\sigma i, q) \rangle
\langle s' = (\sigma' i, q') \rangle
show "((\sigma, \text{snd (id s)}), a, (\sigma', \text{snd (id s')}) \in \text{trans (opaodv i)}"
by simp
qed
then interpret opn: openproc paodv opaodv id.
have \{simp\}: "\forall i. (\text{SOME x. x } \in (\text{fst o id} \ ' \ \text{init (paodv i)}) = \text{aodv_init i}"
unfolding \sigma_{AODV-def} \sigma_{QMSG-def} by simp
hence "\forall i. \text{openproc.initmissing paodv id i } = \text{initmissing i}"
unfolding opn.initmissing_def opn.someinitial_def initmissing_def
by (auto split: option.split)
thus "\text{openproc.initmissing paodv id } = \text{initmissing}" ..
qed
interpretation aodv_openproc_par_qmsg: openproc_parq paodv opaodv id qmsg
rewrites "aodv_openproc_par_qmsg.netglobal = netglobal"
and "aodv_openproc_par_qmsg.initmissing = initmissing"
proof -
show "\forall i. \text{openproc.netglobal (\lambda i. paodv i \langle\langle qmsg \rangle \rangle (\lambda(p, q). (\text{fst (id p)}, \text{snd (id p), q)) \sigma = netglobal \sigma) = netglobal\sigma}"
unfolding opq.netglobal_def
unfolding \sigma_{AODV-def} \sigma_{QMSG-def} by (clarsimp cong: option.case_cong)
thus "\text{openproc.netglobal (\lambda i. paodv i \langle\langle qmsg \rangle \rangle (\lambda(p, q). (\text{fst (id p)}, \text{snd (id p), q)) = netglobal\sigma) = netglobal\sigma}"
unfolding opq.netglobal_def
unfolding \sigma_{AODV-def} \sigma_{QMSG-def} by (clarsimp cong: option.case_cong)
simp add: \text{fstd_initmissing_netmap_default_aodv_init_netlift}
[symmetric, unfolded initmissing_def]
thus "\text{openproc.netglobal (\lambda i. paodv i \langle\langle qmsg \rangle \rangle (\lambda(p, q). (\text{fst (id p)}, \text{snd (id p), q})) = netglobal}"
by auto
qed
lemma net_nhop_quality_increases:
assumes "\text{wf_net_tree n}"
shows "\text{closed (pnet (\lambda i. paodv i \langle\langle qmsg \rangle \rangle n) } \models \text{netglobal}
(\lambda\sigma. \forall i \text{ dip. let nhip = the (nhop (rt (\sigma i)) dip)}
in dip \in vD (rt (\sigma i)) \cap vD (rt (\sigma nhip)) \wedge nhip \neq dip"
\[ \rightarrow (rt (\sigma i)) \sqsubseteq dip (rt (\sigma nhip)) \]

(is "\_ \models netglobal (\lambda \sigma. \forall i. ?inv \sigma i")

\[ \begin{array}{l}
\text{proof -} \\
\quad \text{from } \langle \text{wf_net_tree } n \rangle \\
\quad \text{have proto: } "\text{closed (pnet (\lambda i. paodv i (\{ qmsg \}) n) } \models \text{netglobal (\lambda \sigma. } \forall i \in \text{net_tree_ips n. } \forall \text{dip.} \\
\quad \quad \text{let nhip = the (nhop (rt (\sigma i)) dip) } \\
\quad \quad \text{in dip } \in \text{Vd (rt (\sigma i)) } \cap \text{Vd (rt (\sigma nhip)) } \land \text{nhip } \neq \text{dip} \\
\quad \quad \rightarrow (rt (\sigma i)) \sqsubseteq dip (rt (\sigma nhip))" \\
\quad \text{by } (\text{rule aodv_openproc_par_qmsg.close_opnet [OF _ onet_nhop_quality_increases]})) \\
\quad \text{show } ?\text{thesis} \\
\quad \text{unfolding invariant_def opnet_sos.opnet_tau1} \\
\quad \text{proof } (\text{rule, simp only: aodv_openproc_par_qmsg.netglobalsimp} \\
\quad \quad \text{fst_initmissing_netgmap_pair_fst, rule allI}) \\
\quad \quad \text{fix } \sigma i \\
\quad \quad \text{assume sr: } "\sigma \in \text{reachable (closed (pnet (\lambda i. paodv i (\{ qmsg \}) n)) } \text{TT"} \\
\quad \quad \text{hence } "\forall i \in \text{net_tree_ips n. } \exists \text{?inv (fst (initmissing (netgmap fst \sigma))) } i" \\
\quad \quad \text{by } - (\text{drule invariantD [OF proto],} \\
\quad \quad \quad \text{simp only: aodv_openproc_par_qmsg.netglobalsimp} \\
\quad \quad \quad \quad \text{fst_initmissing_netgmap_pair_fst}) \\
\quad \quad \text{thus } "?\text{inv (fst (initmissing (netgmap fst \sigma))) } i" \\
\quad \text{proof } (\text{cases } "i \in \text{net_tree_ips n")} \\
\quad \quad \text{assume } "i \notin \text{net_tree_ips n"} \\
\quad \quad \text{from sr have } "\sigma \in \text{reachable (pnet (\lambda i. paodv i (\{ qmsg \}) n)) } \text{TT" } .. \\
\quad \quad \text{hence } "\text{net_ips } \sigma = \text{net_tree_ips n" } .. \\
\quad \quad \text{with } (i \notin \text{net_tree_ips n) have } "i \notin \text{net_ips } \sigma" \text{ by simp} \\
\quad \quad \text{hence } "(\text{fst (initmissing (netgmap fst \sigma))) } i = \text{aodv_init } i" \\
\quad \quad \quad \text{by simp} \\
\quad \quad \quad \text{thus } ?\text{thesis by simp} \\
\quad \text{qed} \\
\text{qed} \\
\end{array} \]

5.11.6 Loop freedom of AODV

\[ \text{theorem aodv_loop_freedom:} \]
\[ \begin{array}{l}
\text{ assumes } "\text{wf_net_tree } n" \\
\text{ shows } "\text{closed (pnet (\lambda i. paodv i (\{ qmsg \}) n) } \models \text{netglobal (\lambda \sigma. } \forall \text{dip. irrefl ((rt_graph } \sigma \text{ dip)}^+)\)" } \\
\text{ using } \text{assms } \text{by } (\text{rule aodv_openproc_par_qmsg.netglobal_weakenE} \\
\quad \text{ [OF net_nhop_quality_increases inv_to_loop_freedom")} \\
\end{array} \]

end


