Building Correct SDN-Based Components from a Global Formal Model
Event-B Models, generated by tghe Rodin toolkit

christian.attiogbe@univ-nantes.fr

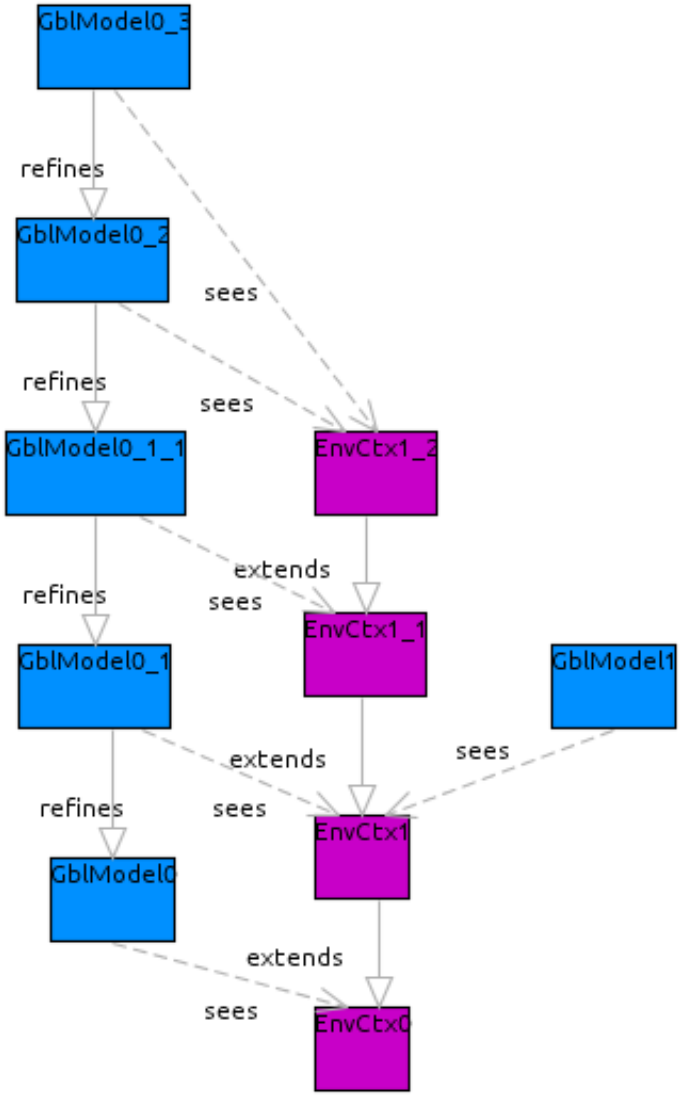
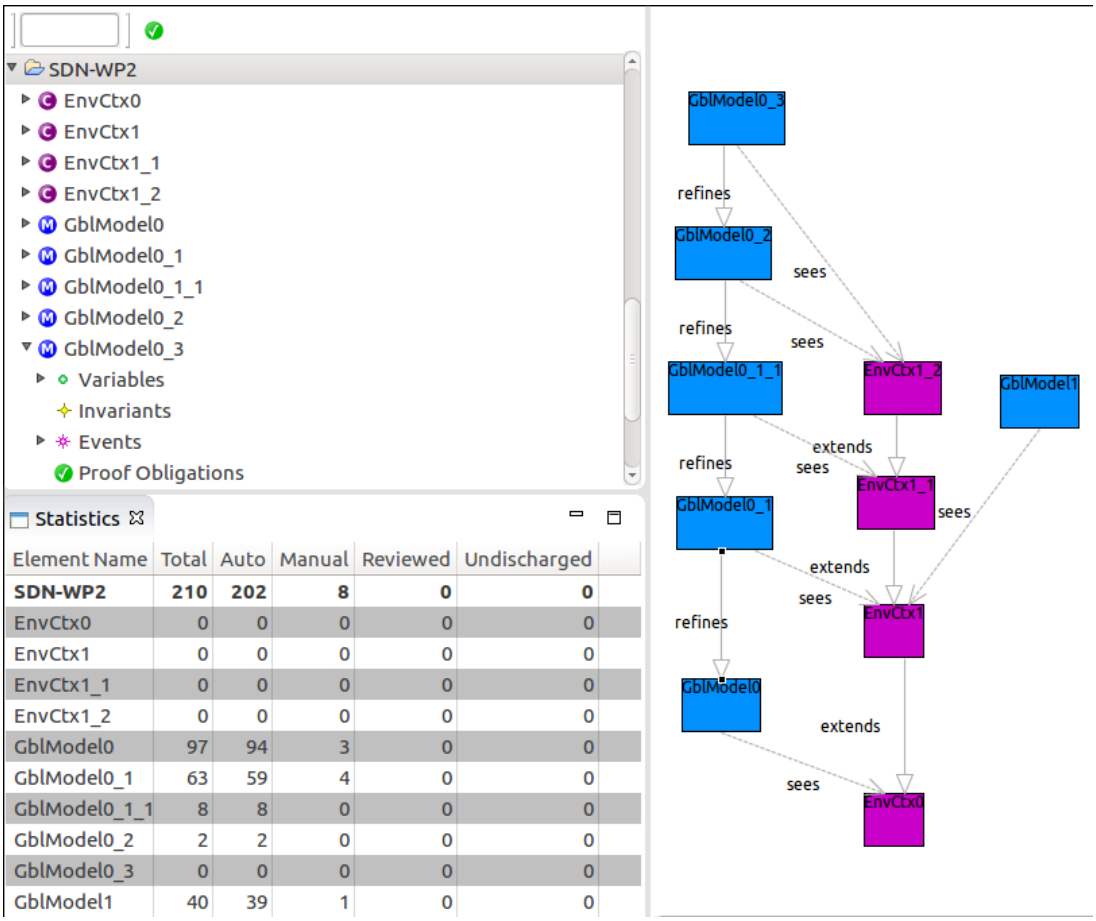


Figure 1: The architecture of the development (from Rodin)



Element Name	Total	Auto	Manual	Reviewed	Undischarged
SDN-WP2	**210**	**202**	**8**	**0**	**0**
EnvCtx0	0	0	0	0	0
EnvCtx1	0	0	0	0	0
EnvCtx1_1	0	0	0	0	0
EnvCtx1_2	0	0	0	0	0
GblModel0	97	94	3	0	0
GblModel0_1	63	59	4	0	0
GblModel0_1_1	8	8	0	0	0
GblModel0_2	2	2	0	0	0
GblModel0_3	0	0	0	0	0
GblModel1	40	39	1	0	0

Figure 2: The proof statistics of the development (from Rodin)

Contents

CONTEXT EnvCtx0

SETS

PACKET set of packets (exchanged between Switches,Controllers,Hosts)

MESG set of messages (exchanged between Swithes and Controler on the secure control channel)

MESGTYPE message types

ENTRY set of entries of the flow table

HEADER

header

ACTION // set of actions applied by switch to maching packets

SW_ID Open flow swith ID

SW_STATE Open FLow Switch State

CONSTANTS

PKIn

PKOut

BarrierQ

BarrierR

FlowMd

askStatus

Status

AddE

DelE

ModE ALL CONTROLLER LOCAL TABLE INPORT

AXIOMS

ax2: $MESGTYPE = \{PKIn, PKOut, BarrierQ, BarrierR, FlowMd, askStatus, Status, AddE, DelE, ModE\}$

@ax4 ACTION = {ALL, CONTROLLER, LOCAL, TABLE, INPORT} // Virtual Actions: Forward: OpenFlow switches support not only forwarding packets to physical ports, but also the virtual ones

END

CONTEXT EnvCtx1

EXTENDS EnvCtx0

SETS

PORTID_ACTION

set of ports ID which equip switches

set of actions applied by switch to maching packets

CONSTANTS

ALL

CONTROLLER

LOCAL

TABLE

INPORT

AXIOMS

ax4: $PORTID\_ACTION = \{ALL, CONTROLLER, LOCAL, TABLE, INPORT\}$

Virtual Actions: Forward: OpenFlow switches support not only forwarding packets to physical ports, but also the virtual ones

END

CONTEXT EnvCtx1_1
EXTENDS EnvCtx1
SETS

IPADR IP Adresses

MACADR MAC Adresses

MACTYPE MAC Type : 800

IPPROTO IPProto :

CONSTANTS

DummyIP

DummyMac

DummyProto

DummyMacType

AXIOMS

ax1: $DummyIP \in IPADR$

ax2: $DummyMac \in MACADR$

ax3: $DummyProto \in IPPROTO$

ax4: $DummyMacType \in MACTYPE$

END

CONTEXT EnvCtx1_2
EXTENDS EnvCtx1_1
SETS

MSG_PRIORITY

END

MACHINE GblModel0

SEES EnvCtx0

VARIABLES

switches set of switches ID

swIPk each switch has a set of inconming packets (its incomping buffer)

swIncomingPk the incomingPK of all the switches

swOPk each switch has a set of outgoing packets (outgoing packet buffer)

swOutgoingPk outgoingPk of all switches

swIncomingMsg each switch has a set of inconming

swOutgoingMsg each switch has a set of outgoing packets for controller ; they don't match any entry, and should be sent to controller

swStatus status of each switch

ctlOutgoingPk controller outgoing packets

ctlSentPkts all packets sent by the controller (this is for Ppty checking)

swSentPkts all packets sent by (all) the switches

ctlIncomingPk controller incoming packets

flowTable each switch has a flow table, it is a set of entries

secureChan

a secure channer between switch and controller

broadCChan // a secure broadCast channel

dataChan

data channel between the switches

actionsQueues // the queues of packet for each action

eHeader1 entry header : actions // actios of an entry

pHeader1 packet header

mesgType the mesg type

mesgPk the mesg packet

INVARIANTS

i10: $switches \subseteq SW\_ID$

i20: $flowTable \in ENTRY \nrightarrow switches$

it is a set of entries

i30: $swIPk \in switches \leftrightarrow PACKET$

each switch has a set of packets (its buffer)

i40: $swIncomingPk \subseteq PACKET$

i50: $finite(swIncomingPk)$

i60: $swIncomingPk \subseteq ran(swIPk)$

i70: $swOPk \in switches \leftrightarrow PACKET$

i80: $swOutgoingPk \subseteq PACKET$

i90: $finite(swOutgoingPk)$

i100: $swOutgoingPk \subseteq ran(swOPk)$

i110: $swIncomingMsg \subseteq MESG \times switches$

i120: $swOutgoingMsg \subseteq MESG$

i130: $ctlIncomingPk \subseteq PACKET$

set of packets coming in from ctlr

i140: $ctlOutgoingPk \subseteq PACKET$

set of packets going out from ctlr

i150: $ctlSentPkts \subseteq PACKET$

pkts in transit via Ctl

i160: $swSentPkts \subseteq PACKET$

pkts in transit via switches

i170: $secureChan \subseteq MESG \times switches$

i172: $dataChan \subseteq PACKET \times switches$

for the refinement it will be (PACKET x OFSW_ID x switches

i180: $dom(dataChan) \subseteq swSentPkts \cup ctlSentPkts$

PROPERTY Pa: any thing on the data chan was sent by the switches

i182: $swIncomingPk \subseteq ctlSentPkts \cup swSentPkts$

PROPERTY Pb: any thing in the sw IncomingPks was sent by the ctl or the switches

i190: $eHeader1 \in ENTRY \nrightarrow HEADER$

i200: $pHeader1 \in PACKET \nrightarrow HEADER$

i210: $dom(eHeader1) = dom(flowTable)$

i220: $swStatus \in SW\_ID \nrightarrow SW\_STATE$

each open flow switch has a state

i230: $mesgType \in MESG \nrightarrow MESGTYPE$

each mesg has a type

i240: $mesgPk \in MESG \nrightarrow PACKET$

each mesg has a packet field

i250: $ran(mesgPk) \subseteq ctlSentPkts$

PROPERTY Pc: the packets sent via messages are included in the sentPkts

i260: $swOutgoingPk \subseteq swSentPkts \cup ctlSentPkts$

PROPERTY Pd: outgoing packets are sent by one of the switches or the controller

EVENTS

Initialisation

begin

a1: $switches := \varnothing$

a2: $swIncomingMsg := \varnothing$

a3: $swOutgoingMsg := \varnothing$

a4: $ctlIncomingPk := \varnothing$

a5: $ctlOutgoingPk := \varnothing$

a6: $ctlSentPkts := \varnothing$

a8: $swSentPkts := \varnothing$

a9: $flowTable := \varnothing$

a10: $swIPk := \varnothing$

a11: $swIncomingPk := \varnothing$

a12: $swOPk := \varnothing$

a13: $swOutgoingPk := \varnothing$

a14: $secureChan := \varnothing$

a15: $dataChan := \varnothing$

a16: $eHeader1 := \varnothing$

a17: $pHeader1 := \varnothing$

a18: $swStatus := \varnothing$

a19: $mesgType := \varnothing$

a20: $mesgPk := \varnothing$

end

Event new_switch ⟨ordinary⟩ $\widehat{=}$

a new switch

any

sw

st

where

g0: $sw \in SW\_ID$

g1: $sw \notin switches$

g2: $st \in SW\_STATE$

then

a1: $switches := switches \cup \{sw\}$

a2: $swStatus(sw) := st$

end

Event sw_rcv_machingPkt ⟨ordinary⟩ $\widehat{=}$

a switch receives a packet (from another switch) which header matches with a flow table entry

any

sw

pkt

swpk

ahd msg theActions theQueues theQueuesDash

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $ahd \in HEADER$

g4: $pkt \in dom(pHeader1)$

g5: $ahd = pHeader1(pkt)$

g6: $\exists ee \cdot (((ee \in ENTRY) \land (ee \in dom(flowTable))) \land (eHeader1(ee) = ahd))$

g7: $swpk \subseteq PACKET$

g8: $sw \in dom(swIPk)$

g9: $sw \mapsto pkt \notin swIPk$

then

a3: $swIncomingPk := swIncomingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pkt\}$

end

Event sw_rcv_unmachingPkt ⟨ordinary⟩ $\widehat{=}$

receives a packet (from another switch) which header does not mach any entry of the flow table

any

sw

pkt

ahd

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $pkt \in dom(pHeader1)$

g4: $ahd \in HEADER$

g5: $ahd = pHeader1(pkt)$

g6: $\forall ee \cdot (((ee \in ENTRY) \land (ee \in dom(flowTable))) \Rightarrow (eHeader1(ee) \neq ahd))$

does not match any entry

then

a3: $swOutgoingPk := swOutgoingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swOPk := swOPk \cup \{sw \mapsto pkt\}$

end

Event sw_rcv_Msg ⟨ordinary⟩ $\widehat{=}$

any

sw

msg

where

g0: $sw \in switches$

g3: $msg \in MESG$

g61: $msg \mapsto sw \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto sw\}$

a2: $swIncomingMsg := swIncomingMsg \cup \{msg \mapsto sw\}$

end

Event sw_sndPk2ctrl ⟨ordinary⟩ $\widehat{=}$

a swith emits a msg with an unmached packets to the controller /!\ TO BE REFINED with ORDERING

any

sw

pkt

msg

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g5: $pkt \in swOutgoingPk$

g6: $msg \in MESG$

g7: $(msg \mapsto PKIn) \in mesgType$

g8: $(msg \mapsto pkt) \in mesgPk$

then

a3: $swSentPkts := swSentPkts \cup \{pkt\}$

a1: $swOutgoingPk := swOutgoingPk \setminus \{pkt$

$\mapsto$ pt$\}$

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event sw_sendPcktALL ⟨ordinary⟩ $\widehat{=}$

send each pkt in the ALL queue to all interface, not including the incoming interface

any

pkt

sws

sw

ahd

where

g0: $sw \in switches$

g1: $sws \subseteq switches$

g2: $sws = switches \setminus \{sw\}$

g3: $pkt \in PACKET$

g4: $ahd \in HEADER$

to make a header for the packet

g41: $pkt \mapsto ahd \in pHeader1$

g5: $sw \in dom(swIPk)$

g6: $pkt \in swIncomingPk$

g7: $sw \mapsto pkt \in swIPk$

swIncomingPk

then

a4: $dataChan := dataChan \cup (\{pkt\} \times sws)$

a5: $swSentPkts := swSentPkts \cup \{pkt\}$

a2: $swIPk := swIPk \setminus \{sw \mapsto pkt\}$

but not necessarily swIncomingPk := swIncomingPk \ {pkt}

a3: $swIncomingPk := swIncomingPk \setminus \{pkt\}$

end

Event sw_sendPckt2sw ⟨ordinary⟩ $\widehat{=}$

a switch sends a packet to another switch via the data channel

any

sw

pk

dsw ANY destination switche

where

g0: $sw \in switches$

g1: $dsw \in switches$

g2: $dsw \neq sw$

g3: $pk \in PACKET$

g4: $pk \in swIncomingPk$

g5: $sw \in dom(swIPk)$

g6: $sw \mapsto pk \in swIPk$

swIncomingPk

then

a4: $swSentPkts := swSentPkts \cup \{pk\}$

a3: $dataChan := dataChan \cup \{pk \mapsto sw\}$

dsw

a2: $swIPk := swIPk \setminus \{sw \mapsto pk\}$

but not necesarily swIncomingPk := swIncomingPk \ {pk} // ↦ pt}

a1: $swIncomingPk := swIncomingPk \setminus \{pk\}$

end

Event sw_newFTentry ⟨ordinary⟩ $\widehat{=}$

a new entry is added to the flow table

any

sw

ne

nh

msg aas

where

g0: $sw \in switches$

g1: $ne \in ENTRY$

g2: $ne \notin dom(flowTable)$

g3: $nh \in HEADER$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto AddE) \in mesgType$

then

a3: $eHeader1(ne) := nh$

@a4 actions(ne) := aas

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $flowTable(ne) := sw$

$\cup$ {ne}

end

Event sw_modFTentry ⟨ordinary⟩ $\widehat{=}$

modifies an entry of the FT according to the actions of received packet

any

sw

ne

oe

nh

msg aas

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $(msg \mapsto sw) \in swIncomingMsg$

g3: $(msg \mapsto ModE) \in mesgType$

g4: $ne \in ENTRY$

g5: $ne \notin dom(flowTable)$

g6: $ne \notin dom(eHeader1)$

g8: $oe \in ENTRY$

g9: $oe \in dom(flowTable)$

g10: $oe \neq ne$

g11: $nh \in HEADER$

nheader

@g12 aas $\subseteq$ ACTION // actions actions(ne) = aas

then

a4: $flowTable := \{oe\} \lhd (flowTable \cup \{ne \mapsto sw\})$

:= (flowTable \ {oe}) ∪ {ne}

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{oe\} \lhd (eHeader1 \cup \{ne \mapsto nh\})$

@a3 actions := {oe} ⩤ (actions ∪ {ne ↦ aas})

end

Event sw_delFTentry ⟨ordinary⟩ $\widehat{=}$

delete an entry ed from the flow table

any

sw

ed

msg

where

g0: $sw \in switches$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto DelE) \in mesgType$

g5: $ed \in ENTRY \land ed \in dom(flowTable)$

then

a4: $flowTable := \{ed\} \lhd flowTable$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{ed\} \lhd eHeader1$

end

Event sw_rcvDataPk ⟨ordinary⟩ $\widehat{=}$

a switch receives a data pk from controller

any

sw

msg

pk

where

g0: $sw \in switches$

g2: $msg \in MESG$

g3: $msg \mapsto sw \in swIncomingMsg$

g4: $(msg \mapsto PKOut) \in mesgType$

g1: $pk \in PACKET$

g6: $(msg \mapsto pk) \in mesgPk$

g7: $pk \notin swIncomingPk$

g10: $sw \mapsto pk \notin swIPk$

then

a3: $swIncomingPk := swIncomingPk \cup \{pk\}$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pk\}$

end

Event sw_getStatus ⟨ordinary⟩ $\widehat{=}$

the controler asks for a swith status

any

swid

pkt

msg

msgt

nmsg

nmsgt

where

g0: $swid \in SW\_ID$

g1: $swid \in dom(swStatus)$

g61: $(msg \mapsto swid) \in secureChan$

g2: $pkt \in PACKET$

the switch builds a packet with its status and sends it to the controler via the secure Chaannel

g3: $msg \in MESG$

g4: $msgt \in MESGTYPE$

g5: $msgt = askStatus$

g6: $msg \mapsto msgt \in mesgType$

g7: $nmsg \in MESG$

g8: $nmsgt \in MESGTYPE$

g9: $nmsgt = Status$

g10: $(nmsg \mapsto nmsgt) \in mesgType$

swStatus(swid)

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $swOutgoingMsg := swOutgoingMsg \cup \{nmsg\}$

with the state os the swithc {swStatus(swid)}

end

Event sw_emitMsg ⟨ordinary⟩ $\widehat{=}$

a switch sends one of its message to the controller

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $msg \in swOutgoingMsg$

then

a1: $swOutgoingMsg := swOutgoingMsg \setminus \{msg\}$

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_havePacket ⟨ordinary⟩ $\widehat{=}$

simulate the disposal of packets

any

pkt

where

g1: $pkt \in PACKET$

g2: $pkt \notin ctlOutgoingPk$

then

a1: $ctlOutgoingPk := ctlOutgoingPk \cup \{pkt\}$

end

Event ctl_emitPkt ⟨ordinary⟩ $\widehat{=}$

the controler emits one of its pending packets, on the port portid of the switch swid, through a msg

any

swid

pkt

msg

ahd

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g2: $ctlOutgoingPk \neq \varnothing$

g3: $pkt \in ctlOutgoingPk$

one of the pakt to be sent on the sw

g4: $msg \in MESG$

to build a mesg with the pkt

g5: $ahd \in HEADER$

a header for the packet

then

a5: $ctlOutgoingPk := ctlOutgoingPk \setminus \{pkt\}$

a0: $mesgType(msg) := PKOut$

the controler emit a msg with PKOut

a3: $secureChan := secureChan \cup \{msg \mapsto swid\}$

emission on the appropriate channel

a4: $ctlSentPkts := ctlSentPkts \cup \{pkt\}$

a1: $mesgPk(msg) := pkt$

a2: $pHeader1(pkt) := ahd$

end

Event ctl_rcvPacketIn ⟨ordinary⟩ $\widehat{=}$

the controler receives a packet from a switch swid which previously received it on its port portid, and was unmatched

any

swid

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto PKIn) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $ctlIncomingPk := ctlIncomingPk \cup \{pkt\}$

(pkt):= swid

end

Event ctl_rcvBarrierRp ⟨ordinary⟩ $\widehat{=}$

any

swid

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g7: $(msg \mapsto BarrierR) \in mesgType$

Barrier Response

g8: $(msg \mapsto pkt) \in mesgPk$

g5: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_rcvStatus ⟨ordinary⟩ $\widehat{=}$

any

swid

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto Status) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $msg \mapsto swid \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_askStatusMsg ⟨ordinary⟩ $\widehat{=}$

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

then

a1: $mesgType(msg) := askStatus$

status request

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_askBarrier ⟨ordinary⟩ $\widehat{=}$

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

then

a1: $mesgType(msg) := BarrierQ$

Barrier reQuest

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

END

MACHINE GblModel0_1
REFINES GblModel0
SEES EnvCtx1
VARIABLES

switches set of switches ID

swIPk

swOPk

swIncomingPk each switch has a set of inconming packets

swOutgoingPk each switch has a set of outgoing packets

swIncomingMsg each switch has a set of inconming

swOutgoingMsg each switch has a set of outgoing packets for controller ; they don't match any entry, and should be sent to controller

swStatus

ctlOutgoingPk controller outgoing packets

ctlSentPkts

swSentPkts all packets sent by the switches

ctlIncomingPk controller incoming packets

flowTable a switch has a flow table, it is a se of entries

secureChan a secure channer between switch and controller

dataChan data channel between the switches

eHeader1 entry header

pHeader1 packet header

mesgType the mesg type

mesgPk

the mesg packet

———— new variables

actionsQueues the queues of packet for each action

actions actios of an entry

swPorts switches now have ports

mesgPt the mesg port

rswIncomingPk now we associate a port with each (pack) for delivery

rswOutgoingPk now we associate a port with each (pack) fo delivery

rDataChan refined data channel

rCtlOutgoingPk refined ctlOutgoingPk

rCtlIncomingPk refined incoming Pk

INVARIANTS

ii10: $swPorts \in switches \rightarrow \mathbb{P}(PORTID\_ACTION)$

ii20: $mesgPt \in MESG \pfun PORTID\_ACTION$

each mesg has now a port

ii30: $rswIncomingPk \in (PACKET \times PORTID\_ACTION) \rel switches$

ii40: $dom(rswIncomingPk^{-1}[switches]) \subseteq swIncomingPk$

inclusion should be ike that since, we may hae the packets on several ports !

ii50: $rswOutgoingPk \in (PACKET \times PORTID\_ACTION) \rel switches$

hard to establish invariant binding with rswOutgoingPk ∈ switches ↛ ℙ (PACKET × PORTID_ACTION)

ii60: $dom(rswOutgoingPk^{-1}[switches]) \subseteq swOutgoingPk$

swOutgoingPk = dom(rswOutgoingPk[switches]) // inclusion should be ike that since, we may hae the packets on several ports !

ii70: $rDataChan \in (PACKET \times SW\_ID) \pfun PORTID\_ACTION$

ii80: $dataChan \subseteq dom(rDataChan)$

inclusion should be ike that since, we may hae the packets on several ports !

ii90: $rCtlOutgoingPk \in (PACKET \times PORTID\_ACTION) \pfun SW\_ID$

ii100: $ctlOutgoingPk \subseteq dom(dom(rCtlOutgoingPk))$

ii110: $rCtlIncomingPk \in (PACKET \times PORTID\_ACTION) \pfun SW\_ID$

ii120: $ctlIncomingPk \subseteq dom(dom(rCtlIncomingPk))$

ii130: $actionsQueues \in PORTID\_ACTION \pfun \mathbb{P}(PACKET)$

set of packets for each action ; the action will be applied to the packets

ii140: $actions \in ENTRY \pfun \mathbb{P}(PORTID\_ACTION)$

ii150: $dom(actions) = dom(flowTable)$

EVENTS

Initialisation ⟨extended⟩

begin

a1: $switches := \varnothing$

a2: $swIncomingMsg := \varnothing$

a3: $swOutgoingMsg := \varnothing$

a4: $ctlIncomingPk := \varnothing$

a5: $ctlOutgoingPk := \varnothing$

a6: $ctlSentPkts := \varnothing$

a8: $swSentPkts := \varnothing$

a9: $flowTable := \varnothing$

a10: $swIPk := \varnothing$

a11: $swIncomingPk := \varnothing$

a12: $swOPk := \varnothing$

a13: $swOutgoingPk := \varnothing$

a14: $secureChan := \varnothing$

a15: $dataChan := \varnothing$

a16: $eHeader1 := \varnothing$

a17: $pHeader1 := \varnothing$

a18: $swStatus := \varnothing$

a19: $mesgType := \varnothing$

a20: $mesgPk := \varnothing$

aa1: $swPorts := \varnothing$

aa2: $mesgPt := \varnothing$

aa3: $rswIncomingPk := \varnothing$

aa4: $rswOutgoingPk := \varnothing$

aa5: $rDataChan := \varnothing$

aa6: $rCtlOutgoingPk := \varnothing$

aa7: $rCtlIncomingPk := \varnothing$

aa8: $actionsQueues := PORTID\_ACTION \times \{\varnothing\}$

each queue is empty

aa9: $actions := \varnothing$

end

Event new_switch ⟨ordinary⟩ $\widehat{=}$

a new switch

extends new_switch

any

sw

st

swps each sw has ports (linked to other switches)

where

g0: $sw \in SW\_ID$

g1: $sw \notin switches$

g2: $st \in SW\_STATE$

gg0: $swps \subseteq PORTID\_ACTION$

gg1: $swps \neq \varnothing$

then

a1: $switches := switches \cup \{sw\}$

a2: $swStatus(sw) := st$

@a3 swIPk(sw) := $\varnothing$

@a4 swOPk(sw) := $\varnothing$

aa0: $swPorts(sw) := swps$

end

Event sw_rcv_machingPkt ⟨ordinary⟩ $\widehat{=}$

a switch receives a packet (from another switch) which header matches with a flow table entry

extends sw_rcv_machingPkt

any

sw

pkt

swpk

ahd msg theActions theQueues theQueuesDash

pt

theActions

theQueues

pkpts

theQueuesDash now a port is specified

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $ahd \in HEADER$

g4: $pkt \in dom(pHeader1)$

g5: $ahd = pHeader1(pkt)$

g6: $\exists ee \cdot (((ee \in ENTRY) \land (ee \in dom(flowTable))) \land (eHeader1(ee) = ahd))$

g7: $swpk \subseteq PACKET$

g8: $sw \in dom(swIPk)$

g9: $sw \mapsto pkt \notin swIPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

gg4: $((pkt \mapsto sw) \mapsto pt) \in rDataChan$

gg5: $sw \in ran(rswIncomingPk)$

sw ∈ dom(rswIncomingPk)

gg6: $(pkt \mapsto pt) \notin dom(rswIncomingPk)$

(pkt ↦ pt) ∉ rswIncomingPk(sw)

gg7: $theActions \subseteq PORTID\_ACTION$

gg8: $\exists ee \cdot ((ee \in ENTRY) \land (ee \in dom(flowTable)) \land (eHeader1(ee) = ahd) \land theActions = actions(ee))$

the actions of the matching entry, to be applied

gg9: $theQueues \in PORTID\_ACTION \nrightarrow \mathbb{P}(PACKET)$

gg10: $theQueues = theActions \lhd actionsQueues$

the queues of the current actions

gg11: $theQueuesDash \in PORTID\_ACTION \rightarrow \mathbb{P}(PACKET)$

gg12: $theQueuesDash = theActions \lhd\!\!\!\!- actionsQueues$

gg13: $pkpts \subseteq PACKET \times PORTID\_ACTION$

gg14: $pkpts = rswIncomingPk^{-1}[\{sw\}] \cup \{pkt \mapsto pt\}$

then

a3: $swIncomingPk := swIncomingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pkt\}$

aa1: $rDataChan := rDataChan \setminus \{(pkt \mapsto sw) \mapsto pt\}$

aa2: $rswIncomingPk := rswIncomingPk \cup \{(pkt \mapsto pt) \mapsto sw\}$

(sw) := pkpts

aa3: $actionsQueues := theQueuesDash \cup (\bigcup aa \cdot (aa \in PORTID\_ACTION \wedge aa \in theActions \wedge aa \in dom(theQueues)) | \{aa \mapsto (theQueues(aa) \cup \{pkt\})\})$

add pkt to all actionqueue

end

Event sw_rcv_unmachingPkt ⟨ordinary⟩ $\widehat{=}$

receives a packet (from another switch) which header does not mach any entry of the flow table

extends sw_rcv_unmachingPkt

any

sw

pkt

ahd

pt now a port is specified

pkpts

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $pkt \in dom(pHeader1)$

g4: $ahd \in HEADER$

g5: $ahd = pHeader1(pkt)$

g6: $\forall ee \cdot (((ee \in ENTRY) \wedge (ee \in dom(flowTable))) \Rightarrow (eHeader1(ee) \neq ahd))$

does not match any entry

gg1: $pt \in PORTID\_ACTION$

gg2: $pt \in swPorts(sw)$

gg3: $((pkt \mapsto sw) \mapsto pt) \in rDataChan$

gg4: $pkpts \subseteq PACKET \times PORTID\_ACTION$

gg5: $sw \in ran(rswOutgoingPk)$

sw ∈ dom(rswOutgoingPk)

@gg6 pkpts = rswOutgoingPk(sw) ∪ {pkt ↦ pt}

then

a3: $swOutgoingPk := swOutgoingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swOPk := swOPk \cup \{sw \mapsto pkt\}$

aa1: $rDataChan := rDataChan \setminus \{(pkt \mapsto sw) \mapsto pt\}$

aa2: $rswOutgoingPk := rswOutgoingPk \cup \{(pkt \mapsto pt) \mapsto sw\}$

rswOutgoingPk(sw) := pkpts// in this case the Pk should be sent to the controller

end

Event sw_rcv_Msg ⟨ordinary⟩ $\widehat{=}$

swIncomingMsg := swIncomingMsg ∪ {msg ↦ sw}

extends sw_rcv_Msg

any

sw

msg

where

g0: $sw \in switches$

g3: $msg \in MESG$

g61: $msg \mapsto sw \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto sw\}$

a2: $swIncomingMsg := swIncomingMsg \cup \{msg \mapsto sw\}$

end

Event sw_sndPk2ctrl ⟨ordinary⟩ $\widehat{=}$

a swith emits a msg with an unmached packets to the controller /!\ TO BE REFINED with ORDERING

extends sw_sndPk2ctrl

any

sw

pkt

msg

pt now a port is specified

pkpts

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g5: $pkt \in swOutgoingPk$

g6: $msg \in MESG$

g7: $(msg \mapsto PKIn) \in mesgType$

g8: $(msg \mapsto pkt) \in mesgPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

gg4: $(msg \mapsto pt) \in mesgPt$

now mesg has port

gg5: $sw \in ran(rswOutgoingPk)$

sw ∈ dom(rswOutgoingPk)

gg6: $((pkt \mapsto pt) \mapsto sw) \in rswOutgoingPk$

(pkt ↦ pt) ∈ rswOutgoingPk(sw)

gg7: $pkpts \subseteq PACKET \times PORTID\_ACTION$

@gg7 pkpts = rswOutgoingPk(sw) \ {pkt ↦ pt}

then

a3: $swSentPkts := swSentPkts \cup \{pkt\}$

a1: $swOutgoingPk := swOutgoingPk \setminus \{pkt$

↦ pt}

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

aa1: $rswOutgoingPk := \{(pkt \mapsto pt) \mapsto sw\} \setminus rswOutgoingPk$

rswOutgoingPk(sw) := pkpts

end

Event sw_sendPcktALL ⟨ordinary⟩ $\widehat{=}$

send each pkt in the ALL queue to all interface, not including the incoming interface

extends sw_sendPcktALL

any

pkt

sws

sw

ahd

where

g0: $sw \in switches$

g1 pkt ∈ swOutgoingPk

g1: $sws \subseteq switches$

g2: $sws = switches \setminus \{sw\}$

g3: $pkt \in PACKET$

g4: $ahd \in HEADER$

to make a header for the packet

g41: $pkt \mapsto ahd \in pHeader1$

g5: $sw \in dom(swIPk)$

g6: $pkt \in swIncomingPk$

g7: $sw \mapsto pkt \in swIPk$

swIncomingPk

gg1: $sw \in switches$

gg2: $(pkt \mapsto sw) \in dataChan$

gg3: $(pkt \mapsto sw) \notin dom(rDataChan)$

gg4: $ALL \in dom(actionsQueues)$

tge port for Broadcasting

gg5: $actionsQueues(ALL) \neq \varnothing$

gg6: $pkt \in PACKET \land pkt \in actionsQueues(ALL)$

thtat is pk should be brodcast

then

a4: $dataChan := dataChan \cup (\{pkt\} \times sws)$

a5: $swSentPkts := swSentPkts \cup \{pkt\}$

a2: $swIPk := swIPk \setminus \{sw \mapsto pkt\}$

but not necessarily swIncomingPk := swIncomingPk \ {pkt}

a3: $swIncomingPk := swIncomingPk \setminus \{pkt\}$

aa1: $rDataChan(pkt \mapsto sw) := ALL$

Brdo := Broad ∪ ({pkt}×PORTID_ACTION)

remain update of actionsQueues(ALL) TODO here

end

Event sw_sendPckt2sw ⟨ordinary⟩ $\widehat{=}$

a switch sends a paxket to another swithc via the data channel

extends sw_sendPckt2sw

any

sw

pk

dsw ANY destination switche

pt

pkpts

where

g0: $sw \in switches$

g1: $dsw \in switches$

g2: $dsw \neq sw$

g3: $pk \in PACKET$

g4: $pk \in swIncomingPk$

g5: $sw \in dom(swIPk)$

g6: $sw \mapsto pk \in swIPk$

swIncomingPk

gg1: $pt \in PORTID\_ACTION$

gg2: $pt \in swPorts(sw)$

gg3: $sw \in ran(rswIncomingPk)$

sw ∈ dom(rswIncomingPk)

gg4: $((pk \mapsto pt) \mapsto sw) \in rswIncomingPk$

∈ rswIncomingPk(sw)

gg5: $(pk \mapsto sw) \notin dom(rDataChan)$

gg6: $pkpts \subseteq PACKET \times PORTID\_ACTION$

@gg8 pkpts = {pk ↦ pt} ⩤ rswIncomingPk // rswIncomingPk(sw) \ {pk ↦ pt}

then

a4: $swSentPkts := swSentPkts \cup \{pk\}$

a3: $dataChan := dataChan \cup \{pk \mapsto sw\}$

dsw

a2: $swIPk := swIPk \setminus \{sw \mapsto pk\}$

but not necesarily swIncomingPk := swIncomingPk \ {pk} // ↦ pt}

a1: $swIncomingPk := swIncomingPk \setminus \{pk\}$

aa1: $rswIncomingPk := \{(pk \mapsto pt) \mapsto sw\} \setminus rswIncomingPk$

// rswIncomingPk(sw) := pkpts

aa2: $rDataChan(pk \mapsto sw) := pt$

end

Event sw_newFTentry ⟨ordinary⟩ $\widehat{=}$

a new entry is added to the flow table

extends sw_newFTentry

any

sw

ne

nh

msg aas

aas

where

g0: $sw \in switches$

g1: $ne \in ENTRY$

g2: $ne \notin dom(flowTable)$

g3: $nh \in HEADER$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto AddE) \in mesgType$

gg1: $aas \subseteq PORTID\_ACTION$

actions

then

a3: $eHeader1(ne) := nh$

@a4 actions(ne) := aas

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $flowTable(ne) := sw$

∪ {ne}

aa2: $actions(ne) := aas$

end

Event sw_modFTentry ⟨ordinary⟩ $\widehat{=}$

modifies an entry of the FT according to the actions of received packet

extends sw_modFTentry

any

sw

ne

oe

nh

msg aas

aas

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $(msg \mapsto sw) \in swIncomingMsg$

g3: $(msg \mapsto ModE) \in mesgType$

g4: $ne \in ENTRY$

g5: $ne \notin dom(flowTable)$

g6: $ne \notin dom(eHeader1)$

@g7 ne $\notin$ dom(actions)

g8: $oe \in ENTRY$

g9: $oe \in dom(flowTable)$

g10: $oe \neq ne$

g11: $nh \in HEADER$

nheader

@g12 aas $\subseteq$ ACTION // actions actions(ne) = aas

gg7: $ne \notin dom(actions)$

g12: $aas \subseteq PORTID\_ACTION$

actions actions(ne) = aas

then

a4: $flowTable := \{oe\} \lhd (flowTable \cup \{ne \mapsto sw\})$

:= (flowTable \ {oe}) $\cup$ {ne}

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{oe\} \lhd (eHeader1 \cup \{ne \mapsto nh\})$

@a3 actions := {oe} $\lhd$ (actions $\cup$ {ne $\mapsto$ aas})

aa3: $actions := \{oe\} \lhd (actions \cup \{ne \mapsto aas\})$

end

Event sw_delFTentry ⟨ordinary⟩ $\widehat{=}$

delete an entry ed from the flow table

extends sw_delFTentry

any

sw

ed

msg

where

g0: $sw \in switches$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto DelE) \in mesgType$

g5: $ed \in ENTRY \land ed \in dom(flowTable)$

then

a4: $flowTable := \{ed\} \lhd flowTable$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{ed\} \lhd eHeader1$

@a1 flowTable := flowTable − {ed}

@a3 actions := {ed} ⩤ actions

aa3: $actions := \{ed\} \lhd actions$

end

Event sw_rcvDataPk ⟨ordinary⟩ $\widehat{=}$

a switch receives a data pk

extends sw_rcvDataPk

any

sw

msg

pk

pt

where

g0: $sw \in switches$

g2: $msg \in MESG$

g3: $msg \mapsto sw \in swIncomingMsg$

g4: $(msg \mapsto PKOut) \in mesgType$

g1: $pk \in PACKET$

g6: $(msg \mapsto pk) \in mesgPk$

g7: $pk \notin swIncomingPk$

g10: $sw \mapsto pk \notin swIPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

@gg6 sw ∈ ran(rswIncomingPk) // ∈ dom(rswIncomingPk)

@gg8 (pk ↦ pt) ∉ dom(rswIncomingPk) // (pk ↦ pt) ∉ rswIncomingPk(sw)

@gg9 pkpts ⊆ PACKET × PORTID_ACTION

@gg8 pkpts = // pkpts = rswIncomingPk(sw) ∪ {pk ↦ pt}

then

a3: $swIncomingPk := swIncomingPk \cup \{pk\}$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pk\}$

aa1: $rswIncomingPk := rswIncomingPk \cup \{(pk \mapsto pt) \mapsto sw\}$

rswIncomingPk(sw) := pkpts

end

Event sw_getStatus ⟨ordinary⟩ $\widehat{=}$

the controler asks for a swith status

extends sw_getStatus

any

swid

pkt

msg

msgt

nmsg

nmsgt

where

g0: $swid \in SW\_ID$

g1: $swid \in dom(swStatus)$

g61: $(msg \mapsto swid) \in secureChan$

g2: $pkt \in PACKET$

the switch builds a packet with its status and sends it to the controler via the secure Chaannel

g3: $msg \in MESG$

g4: $msgt \in MESGTYPE$

g5: $msgt = askStatus$

g6: $msg \mapsto msgt \in mesgType$

g7: $nmsg \in MESG$

g8: $nmsgt \in MESGTYPE$

g9: $nmsgt = Status$

g10: $(nmsg \mapsto nmsgt) \in mesgType$

swStatus(swid)

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $swOutgoingMsg := swOutgoingMsg \cup \{nmsg\}$

with the state os the swithc {swStatus(swid)}

end

Event sw_emitMsg ⟨ordinary⟩ $\widehat{=}$

a switch sends one of its message to the controller

extends sw_emitMsg

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $msg \in swOutgoingMsg$

then

a1: $swOutgoingMsg := swOutgoingMsg \setminus \{msg\}$

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_havePacket ⟨ordinary⟩ $\widehat{=}$

simulate the disposal of packets

extends ctl_havePacket

any

pkt

sw

pt

where

g1: $pkt \in PACKET$

g2: $pkt \notin ctlOutgoingPk$

gg1: $sw \in switches$

gg2: $pt \in PORTID\_ACTION$

gg3: $sw \in dom(swPorts)$

gg4: $pt \in swPorts(sw)$

then

a1: $ctlOutgoingPk := ctlOutgoingPk \cup \{pkt\}$

aa1: $rCtlOutgoingPk(pkt \mapsto pt) := sw$

end

Event ctl_emitPkt ⟨ordinary⟩ $\widehat{=}$

the controler emits one of its pending packets, on the port portid of the switch swid, through a msg

extends ctl_emitPkt

any

swid

pkt

msg

ahd

pt

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g2: $ctlOutgoingPk \neq \varnothing$

g3: $pkt \in ctlOutgoingPk$

one of the pakt to be sent on the sw

g4: $msg \in MESG$

to build a mesg with the pkt

g5: $ahd \in HEADER$

a header for the packet

gg2: $pt \in PORTID\_ACTION$

gg3: $pt \in swPorts(swid)$

gg4: $((pkt \mapsto pt) \mapsto swid) \in rCtlOutgoingPk$

then

a5: $ctlOutgoingPk := ctlOutgoingPk \setminus \{pkt\}$

a0: $mesgType(msg) := PKOut$

the controler emit a msg with PKOut

a3: $secureChan := secureChan \cup \{msg \mapsto swid\}$

emission on the appropriate channel

a4: $ctlSentPkts := ctlSentPkts \cup \{pkt\}$

a1: $mesgPk(msg) := pkt$

a2: $pHeader1(pkt) := ahd$

aa4: $rCtlOutgoingPk := rCtlOutgoingPk \setminus \{(pkt \mapsto pt) \mapsto swid\}$

end

Event ctl_rcvPacketIn ⟨ordinary⟩ $\widehat{=}$

the controler receives a packet from a switch swid which previously received it on its port portid, and was unmatched

extends ctl_rcvPacketIn

any

$swid$

pkt

msg

pt

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto PKIn) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

gg2: $pt \in PORTID\_ACTION$

gg7: $(msg \mapsto pt) \in mesgPt$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $ctlIncomingPk := ctlIncomingPk \cup \{pkt\}$

(pkt):= swid

aa1: $rCtlIncomingPk(pkt \mapsto pt) := swid$

end

Event ctl_rcvBarrierRp ⟨ordinary⟩ $\widehat{=}$

extends ctl_rcvBarrierRp

any

swid

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g7: $(msg \mapsto BarrierR) \in mesgType$

Barrier Response

g8: $(msg \mapsto pkt) \in mesgPk$

g5: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_rcvStatus ⟨ordinary⟩ $\widehat{=}$

extends ctl_rcvStatus

any

swid

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto Status) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $msg \mapsto swid \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_askStatusMsg ⟨ordinary⟩ $\widehat{=}$
extends ctl_askStatusMsg
any
sw
msg
where
g0: $sw \in switches$
g1: $msg \in MESG$
then
a1: $mesgType(msg) := askStatus$
status request
a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$
end
Event ctl_askBarrier ⟨ordinary⟩ $\widehat{=}$
extends ctl_askBarrier
any
sw
msg
where
g0: $sw \in switches$
g1: $msg \in MESG$
then
a1: $mesgType(msg) := BarrierQ$
Barrier reQuest
a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$
end
END

MACHINE GblModel0_1_1
REFINES GblModel0_1
SEES EnvCtx1_1
VARIABLES

switches set of switches ID

swIncomingPk each switch has a set of inconming packets

swOutgoingPk each switch has a set of outgoing packets

swIPk

swOPk

swIncomingMsg each switch has a set of inconming

swOutgoingMsg each switch has a set of outgoing packets for controller ; they don't match any entry, and should be sent to controller

swStatus

ctlOutgoingPk controller outgoing packets

ctlSentPkts

swSentPkts all packets sent by the switches

ctlIncomingPk controller incoming packets

flowTable a switch has a flow table, it is a se of entries

secureChan

a secure channer between switch and controller

broadCChan // a secure broadCast channel

dataChan

data channel between the switches

actionsQueues // the queues of packet for each action

eHeader1

entry header

actions // actios of an entry

pHeader1 packet header

mesgType the mesg type

mesgPk

the mesg packet

new variables

actionsQueues the queues of packet for each action

actions actios of an entry

swPorts switches now have ports

mesgPt the mesg port

rswIncomingPk now we associate a port with each (pack) for delivery

rswOutgoingPk

now we associate a port with each (pack) fo delivery

rBroadcChan

rDataChan

rCtlOutgoingPk

rCtlIncomingPk

new variables

—- new variables ——

macSrc MAC source address,

macDst MAC destination address,

macType MAC type,

IpSrc IP source address,

IpDst IP destination address,

IpProto IP protocol,

TpSrcPt transport source port,

TpDstPt transport destination port

switchesIP the set of IP of connected switches (bijection with switches)

ctlIP controller'IO

ctlMac contraller's MAC

switchesMac the MacAdr of switches : each switch has a mac address

INVARIANTS

i1: $macSrc \in PACKET \pfun MACADR$

i2: $macDst \in PACKET \pfun MACADR$

i3: $macType \in PACKET \pfun MACTYPE$

i4: $IpSrc \in PACKET \pfun IPADR$

i5: $IpDst \in PACKET \pfun IPADR$

i6: $IpProto \in PACKET \pfun IPPROTO$

i7: $TpSrcPt \in PACKET \pfun PORTID\_ACTION$

i8: $TpDstPt \in PACKET \pfun PORTID\_ACTION$

i0: $switchesIP \in switches \rightarrowtail IPADR$

injection

i13: $switchesMac \in switches \rightarrowtail MACADR$

i11: $ctlIP \in IPADR$

i12: $ctlMac \in MACADR$

EVENTS

Initialisation ⟨extended⟩

begin

a1: $switches := \varnothing$

a2: $swIncomingMsg := \varnothing$

a3: $swOutgoingMsg := \varnothing$

a4: $ctlIncomingPk := \varnothing$

a5: $ctlOutgoingPk := \varnothing$

a6: $ctlSentPkts := \varnothing$

a8: $swSentPkts := \varnothing$

a9: $flowTable := \varnothing$

a10: $swIPk := \varnothing$

a11: $swIncomingPk := \varnothing$

a12: $swOPk := \varnothing$

a13: $swOutgoingPk := \varnothing$

a14: $secureChan := \varnothing$

a15: $dataChan := \varnothing$

a16: $eHeader1 := \varnothing$

a17: $pHeader1 := \varnothing$

a18: $swStatus := \varnothing$

a19: $mesgType := \varnothing$

a20: $mesgPk := \varnothing$

aa1: $swPorts := \varnothing$

aa2: $mesgPt := \varnothing$

aa3: $rswIncomingPk := \varnothing$

aa4: $rswOutgoingPk := \varnothing$

aa5: $rDataChan := \varnothing$

aa6: $rCtlOutgoingPk := \varnothing$

aa7: $rCtlIncomingPk := \varnothing$

aa8: $actionsQueues := PORTID\_ACTION \times \{\varnothing\}$

each queue is empty

aa9: $actions := \varnothing$

aaa1: $macSrc := PACKET \times \{DummyMac\}$

aaa2: $macDst := PACKET \times \{DummyMac\}$

aaa3: $macType := PACKET \times \{DummyMacType\}$

aaa4: $IpSrc := PACKET \times \{DummyIP\}$

aaa5: $IpDst := PACKET \times \{DummyIP\}$

aaa6: $IpProto := PACKET \times \{DummyProto\}$

aaa7: $TpSrcPt := PACKET \times \{LOCAL\}$

aaa8: $TpDstPt := PACKET \times \{LOCAL\}$

aaa9: $switchesIP := \varnothing$

aaa10: $ctlIP := DummyIP$

aaa11: $ctlMac := DummyMac$

aaa12: $switchesMac := \varnothing$

end

Event new_switch ⟨ordinary⟩ $\widehat{=}$

a new switch has its swId, IP and mac

extends new_switch

any

sw

st

$swps$ each sw has ports (linked to other switches)

nIp new fresh Ip

nMac

where

g0: $sw \in SW\_ID$

g1: $sw \notin switches$

g2: $st \in SW\_STATE$

gg0: $swps \subseteq PORTID\_ACTION$

gg1: $swps \neq \varnothing$

gggg1: $nIp \in IPADR$

gggg2: $nIp \notin ran(switchesIP)$

fresh IP

gggg3: $sw \mapsto nIp \notin switchesIP$

gggg4: $nMac \in MACADR$

gggg5: $nMac \notin ran(switchesMac)$

gggg6: $sw \mapsto nMac \notin switchesMac$

then

a1: $switches := switches \cup \{sw\}$

a2: $swStatus(sw) := st$

@a3 swIPk(sw) := $\varnothing$

@a4 swOPk(sw) := $\varnothing$

aa0: $swPorts(sw) := swps$

aaaa1: $switchesIP(sw) := nIp$

aaaa2: $switchesMac(sw) := nMac$

end

Event sw_rcv_machingPkt ⟨ordinary⟩ $\widehat{=}$

a switch receives a packet (from another switch) which header matches with a flow table entry

extends sw_rcv_machingPkt

any

sw

pkt

$swpk$

ahd msg theActions theQueues theQueuesDash

pt

theActions

theQueues

pkpts

theQueuesDash now a port is specified

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $ahd \in HEADER$

g4: $pkt \in dom(pHeader1)$

g5: $ahd = pHeader1(pkt)$

g6: $\exists ee \cdot (((ee \in ENTRY) \land (ee \in dom(flowTable))) \land (eHeader1(ee) = ahd))$

g7: $swpk \subseteq PACKET$

g8: $sw \in dom(swIPk)$

g9: $sw \mapsto pkt \notin swIPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

gg4: $((pkt \mapsto sw) \mapsto pt) \in rDataChan$

gg5: $sw \in ran(rswIncomingPk)$

sw ∈ dom(rswIncomingPk)

gg6: $(pkt \mapsto pt) \notin dom(rswIncomingPk)$

(pkt ↦ pt) ∉ rswIncomingPk(sw)

gg7: $theActions \subseteq PORTID\_ACTION$

gg8: $\exists ee \cdot ((ee \in ENTRY) \land (ee \in dom(flowTable)) \land (eHeader1(ee) = ahd) \land theActions = actions(ee))$

the actions of the matching entry, to be applied

gg9: $theQueues \in PORTID\_ACTION \nrightarrow \mathbb{P}(PACKET)$

gg10: $theQueues = theActions \lhd actionsQueues$

the queues of the current actions

gg11: $theQueuesDash \in PORTID\_ACTION \rightarrow \mathbb{P}(PACKET)$

gg12: $theQueuesDash = theActions \lhd\!\!\!- actionsQueues$

gg13: $pkpts \subseteq PACKET \times PORTID\_ACTION$

gg14: $pkpts = rswIncomingPk^{-1}[\{sw\}] \cup \{pkt \mapsto pt\}$

then

a3: $swIncomingPk := swIncomingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pkt\}$

aa1: $rDataChan := rDataChan \setminus \{(pkt \mapsto sw) \mapsto pt\}$

aa2: $rswIncomingPk := rswIncomingPk \cup \{(pkt \mapsto pt) \mapsto sw\}$

(sw) := pkpts

aa3: $actionsQueues := theQueuesDash \cup (\bigcup aa \cdot (aa \in PORTID\_ACTION \wedge aa \in theActions \wedge aa \in dom(theQueues)) | \{aa \mapsto (theQueues(aa) \cup \{pkt\})\})$

add pkt to all actionqueue

end

Event sw_rcv_unmachingPkt ⟨ordinary⟩ $\widehat{=}$

receives a packet (from another switch) which header does not mach any entry of the flow table

extends sw_rcv_unmachingPkt

any

sw

pkt

ahd

pt now a port is specified

$pkpts$

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $pkt \in dom(pHeader1)$

g4: $ahd \in HEADER$

g5: $ahd = pHeader1(pkt)$

g6: $\forall ee \cdot (((ee \in ENTRY) \wedge (ee \in dom(flowTable))) \Rightarrow (eHeader1(ee) \neq ahd))$

does not match any entry

gg1: $pt \in PORTID\_ACTION$

gg2: $pt \in swPorts(sw)$

gg3: $((pkt \mapsto sw) \mapsto pt) \in rDataChan$

gg4: $pkpts \subseteq PACKET \times PORTID\_ACTION$

gg5: $sw \in ran(rswOutgoingPk)$

sw ∈ dom(rswOutgoingPk)

@gg6 pkpts = rswOutgoingPk(sw) ∪ {pkt ↦ pt}

then

a3: $swOutgoingPk := swOutgoingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swOPk := swOPk \cup \{sw \mapsto pkt\}$

aa1: $rDataChan := rDataChan \setminus \{(pkt \mapsto sw) \mapsto pt\}$

aa2: $rswOutgoingPk := rswOutgoingPk \cup \{(pkt \mapsto pt) \mapsto sw\}$

rswOutgoingPk(sw) := pkpts// in this case the Pk should be sent to the controller

end

Event sw_rcv_Msg ⟨ordinary⟩ $\widehat{=}$

extends sw_rcv_Msg

any

sw

msg

where

g0: $sw \in switches$

g3: $msg \in MESG$

g61: $msg \mapsto sw \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto sw\}$

a2: $swIncomingMsg := swIncomingMsg \cup \{msg \mapsto sw\}$

end

Event sw_sndPk2ctrl ⟨ordinary⟩ $\widehat{=}$

a swith emits a msg with an unmached packets to the controller /!\ TO BE REFINED with ORDERING

extends sw_sndPk2ctrl

any

sw

pkt

msg

pt now a port is specified

$pkpts$

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g5: $pkt \in swOutgoingPk$

g6: $msg \in MESG$

g7: $(msg \mapsto PKIn) \in mesgType$

g8: $(msg \mapsto pkt) \in mesgPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

gg4: $(msg \mapsto pt) \in mesgPt$

now mesg has port

gg5: $sw \in ran(rswOutgoingPk)$

sw ∈ dom(rswOutgoingPk)

gg6: $((pkt \mapsto pt) \mapsto sw) \in rswOutgoingPk$

(pkt ↦ pt) ∈ rswOutgoingPk(sw)

gg7: $pkpts \subseteq PACKET \times PORTID\_ACTION$

@gg7 pkpts = rswOutgoingPk(sw) \ {pkt ↦ pt}

then

a3: $swSentPkts := swSentPkts \cup \{pkt\}$

a1: $swOutgoingPk := swOutgoingPk \setminus \{pkt\}$

$\mapsto$ pt}

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

aa1: $rswOutgoingPk := \{(pkt \mapsto pt) \mapsto sw\} \setminus rswOutgoingPk$

rswOutgoingPk(sw) := pkpts

end

Event sw_sendPcktALL ⟨ordinary⟩ $\widehat{=}$

send each pkt in the ALL queue to all interface, not including the incoming interface

extends sw_sendPcktALL

any

pkt

sws

sw

ahd

where

g0: $sw \in switches$

g1 pkt ∈ swOutgoingPk

g1: $sws \subseteq switches$

g2: $sws = switches \setminus \{sw\}$

g3: $pkt \in PACKET$

g4: $ahd \in HEADER$

to make a header for the packet

g41: $pkt \mapsto ahd \in pHeader1$

g5: $sw \in dom(swIPk)$

g6: $pkt \in swIncomingPk$

g7: $sw \mapsto pkt \in swIPk$

swIncomingPk

gg1: $sw \in switches$

gg2: $(pkt \mapsto sw) \in dataChan$

gg3: $(pkt \mapsto sw) \notin dom(rDataChan)$

gg4: $ALL \in dom(actionsQueues)$

tge port for Broadcasting

gg5: $actionsQueues(ALL) \neq \varnothing$

gg6: $pkt \in PACKET \wedge pkt \in actionsQueues(ALL)$

thtat is pk should be brodcast

then

a4: $dataChan := dataChan \cup (\{pkt\} \times sws)$

a5: $swSentPkts := swSentPkts \cup \{pkt\}$

a2: $swIPk := swIPk \setminus \{sw \mapsto pkt\}$

but not necessarily swIncomingPk := swIncomingPk \ {pkt}

a3: $swIncomingPk := swIncomingPk \setminus \{pkt\}$

aa1: $rDataChan(pkt \mapsto sw) := ALL$

Brdo := Broad ∪ ({pkt}×PORTID_ACTION)

remain update of actionsQueues(ALL) TODO here

end

Event sw_sendPckt2sw ⟨ordinary⟩ $\widehat{=}$

a switch sends a paxket to another swithc via the data channel

extends sw_sendPckt2sw

any

sw

pk

dsw ANY destination switche

pt

$pkpts$

where

g0: $sw \in switches$

g1: $dsw \in switches$

g2: $dsw \neq sw$

g3: $pk \in PACKET$

g4: $pk \in swIncomingPk$

g5: $sw \in dom(swIPk)$

g6: $sw \mapsto pk \in swIPk$

swIncomingPk

gg1: $pt \in PORTID\_ACTION$

gg2: $pt \in swPorts(sw)$

gg3: $sw \in ran(rswIncomingPk)$

sw ∈ dom(rswIncomingPk)

gg4: $((pk \mapsto pt) \mapsto sw) \in rswIncomingPk$

∈ rswIncomingPk(sw)

gg5: $(pk \mapsto sw) \notin dom(rDataChan)$

gg6: $pkpts \subseteq PACKET \times PORTID\_ACTION$

@gg8 pkpts = {pk ↦ pt} ⩤ rswIncomingPk // rswIncomingPk(sw) \ {pk ↦ pt}

then

a4: $swSentPkts := swSentPkts \cup \{pk\}$

a3: $dataChan := dataChan \cup \{pk \mapsto sw\}$

dsw

a2: $swIPk := swIPk \setminus \{sw \mapsto pk\}$

but not necesarily swIncomingPk := swIncomingPk \ {pk} // ↦ pt}

a1: $swIncomingPk := swIncomingPk \setminus \{pk\}$

aa1: $rswIncomingPk := \{(pk \mapsto pt) \mapsto sw\} \setminus rswIncomingPk$

// rswIncomingPk(sw) := pkpts

aa2: $rDataChan(pk \mapsto sw) := pt$

end

Event sw_newFTentry ⟨ordinary⟩ $\widehat{=}$

a new entry is added to the flow table

extends sw_newFTentry

any

sw

ne

nh

msg aas

aas

where

g0: $sw \in switches$

g1: $ne \in ENTRY$

g2: $ne \notin dom(flowTable)$

g3: $nh \in HEADER$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto AddE) \in mesgType$

gg1: $aas \subseteq PORTID\_ACTION$

actions

then

a3: $eHeader1(ne) := nh$

@a4 actions(ne) := aas

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $flowTable(ne) := sw$

$\cup$ {ne}

aa2: $actions(ne) := aas$

end

Event sw_modFTentry ⟨ordinary⟩ $\widehat{=}$

modifies an entry of the FT according to the actions of received packet

extends sw_modFTentry

any

sw

ne

oe

nh

msg aas

aas

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $(msg \mapsto sw) \in swIncomingMsg$

g3: $(msg \mapsto ModE) \in mesgType$

g4: $ne \in ENTRY$

g5: $ne \notin dom(flowTable)$

g6: $ne \notin dom(eHeader1)$

@g7 ne $\notin$ dom(actions)

g8: $oe \in ENTRY$

g9: $oe \in dom(flowTable)$

g10: $oe \neq ne$

g11: $nh \in HEADER$

nheader

@g12 aas $\subseteq$ ACTION // actions actions(ne) = aas

gg7: $ne \notin dom(actions)$

g12: $aas \subseteq PORTID\_ACTION$

actions actions(ne) = aas

then

a4: $flowTable := \{oe\} \lhd (flowTable \cup \{ne \mapsto sw\})$

:= (flowTable \ {oe}) $\cup$ {ne}

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{oe\} \lhd (eHeader1 \cup \{ne \mapsto nh\})$

@a3 actions := {oe} $\lhd$ (actions $\cup$ {ne $\mapsto$ aas})

aa3: $actions := \{oe\} \lhd (actions \cup \{ne \mapsto aas\})$

end

Event sw_delFTentry ⟨ordinary⟩ $\widehat{=}$

delete an entry ed from the flow table

extends sw_delFTentry

any

sw

ed

msg

where

g0: $sw \in switches$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto DelE) \in mesgType$

g5: $ed \in ENTRY \land ed \in dom(flowTable)$

then

a4: $flowTable := \{ed\} \lhd flowTable$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{ed\} \lhd eHeader1$

@a1 flowTable := flowTable − {ed}

@a3 actions := {ed} ⩤ actions

aa3: $actions := \{ed\} \lhd actions$

end

Event sw_rcvDataPk ⟨ordinary⟩ $\widehat{=}$

a switch receives a data pk

extends sw_rcvDataPk

any

sw

msg

pk

pt

where

g0: $sw \in switches$

g2: $msg \in MESG$

g3: $msg \mapsto sw \in swIncomingMsg$

g4: $(msg \mapsto PKOut) \in mesgType$

g1: $pk \in PACKET$

g6: $(msg \mapsto pk) \in mesgPk$

g7: $pk \notin swIncomingPk$

g10: $sw \mapsto pk \notin swIPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

@gg6 sw ∈ ran(rswIncomingPk) // ∈ dom(rswIncomingPk)

@gg8 (pk ↦ pt) ∉ dom(rswIncomingPk) // (pk ↦ pt) ∉ rswIncomingPk(sw)

@gg9 pkpts ⊆ PACKET × PORTID_ACTION

@gg8 pkpts = // pkpts = rswIncomingPk(sw) ∪ {pk ↦ pt}

then

a3: $swIncomingPk := swIncomingPk \cup \{pk\}$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pk\}$

aa1: $rswIncomingPk := rswIncomingPk \cup \{(pk \mapsto pt) \mapsto sw\}$

rswIncomingPk(sw) := pkpts

end

Event sw_getStatus ⟨ordinary⟩ $\widehat{=}$

the controler asks for a swith status

extends sw_getStatus

any

$swid$

pkt

msg

$msgt$

$nmsg$

$nmsgt$

where

g0: $swid \in SW\_ID$

g1: $swid \in dom(swStatus)$

g61: $(msg \mapsto swid) \in secureChan$

g2: $pkt \in PACKET$

the switch builds a packet with its status and sends it to the controler via the secure Chaannel

g3: $msg \in MESG$

g4: $msgt \in MESGTYPE$

g5: $msgt = askStatus$

g6: $msg \mapsto msgt \in mesgType$

g7: $nmsg \in MESG$

g8: $nmsgt \in MESGTYPE$

g9: $nmsgt = Status$

g10: $(nmsg \mapsto nmsgt) \in mesgType$

swStatus(swid)

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $swOutgoingMsg := swOutgoingMsg \cup \{nmsg\}$

with the state os the swithc {swStatus(swid)}

end

Event sw_emitMsg ⟨ordinary⟩ $\widehat{=}$

a switch sends one of its message to the controller

extends sw_emitMsg

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $msg \in swOutgoingMsg$

then

a1: $swOutgoingMsg := swOutgoingMsg \setminus \{msg\}$

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_havePacket ⟨ordinary⟩ $\widehat{=}$

simulate the disposal of packets

extends ctl_havePacket

any

pkt

sw

pt

where

g1: $pkt \in PACKET$

g2: $pkt \notin ctlOutgoingPk$

gg1: $sw \in switches$

gg2: $pt \in PORTID\_ACTION$

gg3: $sw \in dom(swPorts)$

gg4: $pt \in swPorts(sw)$

then

a1: $ctlOutgoingPk := ctlOutgoingPk \cup \{pkt\}$

aa1: $rCtlOutgoingPk(pkt \mapsto pt) := sw$

end

Event ctl_emitPkt ⟨ordinary⟩ $\widehat{=}$

the controler emits one of its pending packets, on the port portid of the switch swid, through a msg

extends ctl_emitPkt

any

$swid$

pkt

msg

ahd

pt

macsrc

macdst

ips

ipd

iproto

srcPt

dstPt

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g2: $ctlOutgoingPk \neq \varnothing$

g3: $pkt \in ctlOutgoingPk$

one of the pakt to be sent on the sw

g4: $msg \in MESG$

to build a mesg with the pkt

g5: $ahd \in HEADER$

a header for the packet

gg2: $pt \in PORTID\_ACTION$

gg3: $pt \in swPorts(swid)$

gg4: $((pkt \mapsto pt) \mapsto swid) \in rCtlOutgoingPk$

ggg1: $macsrc \in MACADR$

ggg2: $macdst \in MACADR$

ggg3: $ips \in IPADR$

ggg4: $ipd \in IPADR$

ggg5: $iproto \in IPPROTO$

ggg6: $srcPt \in PORTID\_ACTION$

ggg7: $dstPt \in PORTID\_ACTION$

ggg8: $ips = ctlIP$

ggg9: $swid \in dom(switchesIP)$

ggg9,1: $ipd = switchesIP(swid)$

ggg10: $swid \in dom(switchesMac)$

ggg10,1: $macdst = switchesMac(swid)$

ggg11: $macsrc = ctlMac$

then

a5: $ctlOutgoingPk := ctlOutgoingPk \setminus \{pkt\}$

a0: $mesgType(msg) := PKOut$

the controler emit a msg with PKOut

a3: $secureChan := secureChan \cup \{msg \mapsto swid\}$

emission on the appropriate channel

a4: $ctlSentPkts := ctlSentPkts \cup \{pkt\}$

a1: $mesgPk(msg) := pkt$

a2: $pHeader1(pkt) := ahd$

aa4: $rCtlOutgoingPk := rCtlOutgoingPk \setminus \{(pkt \mapsto pt) \mapsto swid\}$

aaa1: $macSrc(pkt) := macsrc$

mac src of the contriller

aaa2: $macDst(pkt) := macdst$

aaa3: $IpSrc(pkt) := ips$

aaa4: $IpDst(pkt) := ipd$

ip src of the ctl

aaa5: $IpProto(pkt) := iproto$

aaa6: $TpSrcPt(pkt) := srcPt$

aaa7: $TpDstPt(pkt) := dstPt$

end

Event ctl_rcvPacketIn ⟨ordinary⟩ $\widehat{=}$

the controler receives a packet from a switch swid which previously received it on its port portid, and was unmatched

extends ctl_rcvPacketIn

any

$swid$

pkt

msg

pt

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto PKIn) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

gg2: $pt \in PORTID\_ACTION$

gg7: $(msg \mapsto pt) \in mesgPt$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $ctlIncomingPk := ctlIncomingPk \cup \{pkt\}$

(pkt):= swid

aa1: $rCtlIncomingPk(pkt \mapsto pt) := swid$

end

Event ctl_rcvBarrierRp ⟨ordinary⟩ $\widehat{=}$

extends ctl_rcvBarrierRp

any

$swid$

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g7: $(msg \mapsto BarrierR) \in mesgType$

Barrier Response

g8: $(msg \mapsto pkt) \in mesgPk$

g5: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_rcvStatus ⟨ordinary⟩ $\widehat{=}$

extends ctl_rcvStatus

any

$swid$

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto Status) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $msg \mapsto swid \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_askStatusMsg ⟨ordinary⟩ $\widehat{=}$

extends ctl_askStatusMsg

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

then

a1: $mesgType(msg) := askStatus$

status request

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_askBarrier ⟨ordinary⟩ $\widehat{=}$

extends ctl_askBarrier

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

then

a1: $mesgType(msg) := BarrierQ$

Barrier reQuest

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

END

MACHINE GblModel0_2
REFINES GblModel0_1_1
SEES EnvCtx1_2
VARIABLES

switches set of switches ID

swIPk

swOPk

swIncomingPk each switch has a set of inconming packets

swOutgoingPk each switch has a set of outgoing packets

swIncomingMsg each switch has a set of inconming

swOutgoingMsg each switch has a set of outgoing packets for controller ; they don't match any entry, and should be sent to controller

swStatus

ctlOutgoingPk controller outgoing packets

ctlSentPkts

swSentPkts all packets sent by the switches

ctlIncomingPk controller incoming packets

flowTable a switch has a flow table, it is a se of entries

secureChan

a secure channer between switch and controller

broadCChan // a secure broadCast channel

dataChan data channel between the switches

actionsQueues the queues of packet for each action

eHeader1 entry header

pHeader1 packet header

actions actios of an entry

mesgType the mesg type

mesgPk

macSrc MAC source address,

macDst MAC destination address,

macType MAC type,

IpSrc IP source address,

IpDst IP destination address,

IpProto IP protocol,

TpSrcPt transport source port,

TpDstPt transport destination port

switchesIP

switchesMac

ctlIP controller'IO

ctlMac

contraller's MAC

switches IP Id

the mesg packet

new variables

swPorts switches now have ports

mesgPt the mesg port

rswIncomingPk now we associate a port with each (pack) for delivery

rswOutgoingPk now we associate a port with each (pack) fo delivery

rDataChan

rCtlOutgoingPk

rCtlIncomingPk additional variables

msgPriority SecureChan now msgs may have a priority

INVARIANTS

iii10: $msgPriority \in MESG \pfun MSG\_PRIORITY$

EVENTS

Initialisation ⟨extended⟩

begin

a1: $switches := \varnothing$

a2: $swIncomingMsg := \varnothing$

a3: $swOutgoingMsg := \varnothing$

a4: $ctlIncomingPk := \varnothing$

a5: $ctlOutgoingPk := \varnothing$

a6: $ctlSentPkts := \varnothing$

a8: $swSentPkts := \varnothing$

a9: $flowTable := \varnothing$

a10: $swIPk := \varnothing$

a11: $swIncomingPk := \varnothing$

a12: $swOPk := \varnothing$

a13: $swOutgoingPk := \varnothing$

a14: $secureChan := \varnothing$

a15: $dataChan := \varnothing$

a16: $eHeader1 := \varnothing$

a17: $pHeader1 := \varnothing$

a18: $swStatus := \varnothing$

a19: $mesgType := \varnothing$

a20: $mesgPk := \varnothing$

aa1: $swPorts := \varnothing$

aa2: $mesgPt := \varnothing$

aa3: $rswIncomingPk := \varnothing$

aa4: $rswOutgoingPk := \varnothing$

aa5: $rDataChan := \varnothing$

aa6: $rCtlOutgoingPk := \varnothing$

aa7: $rCtlIncomingPk := \varnothing$

aa8: $actionsQueues := PORTID\_ACTION \times \{\varnothing\}$

each queue is empty

aa9: $actions := \varnothing$

aaa1: $macSrc := PACKET \times \{DummyMac\}$

aaa2: $macDst := PACKET \times \{DummyMac\}$

aaa3: $macType := PACKET \times \{DummyMacType\}$

aaa4: $IpSrc := PACKET \times \{DummyIP\}$

aaa5: $IpDst := PACKET \times \{DummyIP\}$

aaa6: $IpProto := PACKET \times \{DummyProto\}$

aaa7: $TpSrcPt := PACKET \times \{LOCAL\}$

aaa8: $TpDstPt := PACKET \times \{LOCAL\}$

aaa9: $switchesIP := \varnothing$

aaa10: $ctlIP := DummyIP$

aaa11: $ctlMac := DummyMac$

aaa12: $switchesMac := \varnothing$

aaa: $msgPriority := \varnothing$

end

Event new_switch ⟨ordinary⟩ $\widehat{=}$

a new switch

extends new_switch

any

sw

st

$swps$ each sw has ports (linked to other switches)

nIp new fresh Ip

$nMac$

where

g0: $sw \in SW\_ID$

g1: $sw \notin switches$

g2: $st \in SW\_STATE$

gg0: $swps \subseteq PORTID\_ACTION$

gg1: $swps \neq \varnothing$

gggg1: $nIp \in IPADR$

gggg2: $nIp \notin ran(switchesIP)$

fresh IP

gggg3: $sw \mapsto nIp \notin switchesIP$

gggg4: $nMac \in MACADR$

gggg5: $nMac \notin ran(switchesMac)$

gggg6: $sw \mapsto nMac \notin switchesMac$

then

a1: $switches := switches \cup \{sw\}$

a2: $swStatus(sw) := st$

@a3 swIPk(sw) := $\varnothing$

@a4 swOPk(sw) := $\varnothing$

aa0: $swPorts(sw) := swps$

aaaa1: $switchesIP(sw) := nIp$

aaaa2: $switchesMac(sw) := nMac$

end

Event sw_rcv_machingPkt ⟨ordinary⟩ $\widehat{=}$

a switch receives a packet (from another switch) which header matches with a flow table entry

extends sw_rcv_machingPkt

any

sw

pkt

$swpk$

ahd msg theActions theQueues theQueuesDash

pt

$theActions$

$theQueues$

$pkpts$

$theQueuesDash$ now a port is specified

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $ahd \in HEADER$

g4: $pkt \in dom(pHeader1)$

g5: $ahd = pHeader1(pkt)$

g6: $\exists ee \cdot (((ee \in ENTRY) \wedge (ee \in dom(flowTable))) \wedge (eHeader1(ee) = ahd))$

g7: $swpk \subseteq PACKET$

g8: $sw \in dom(swIPk)$

g9: $sw \mapsto pkt \notin swIPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

gg4: $((pkt \mapsto sw) \mapsto pt) \in rDataChan$

gg5: $sw \in ran(rswIncomingPk)$

sw ∈ dom(rswIncomingPk)

gg6: $(pkt \mapsto pt) \notin dom(rswIncomingPk)$

(pkt ↦ pt) ∉ rswIncomingPk(sw)

gg7: $theActions \subseteq PORTID\_ACTION$

gg8: $\exists ee \cdot ((ee \in ENTRY) \land (ee \in dom(flowTable)) \land (eHeader1(ee) = ahd) \land theActions = actions(ee))$

the actions of the matching entry, to be applied

gg9: $theQueues \in PORTID\_ACTION \nrightarrow \mathbb{P}(PACKET)$

gg10: $theQueues = theActions \lhd actionsQueues$

the queues of the current actions

gg11: $theQueuesDash \in PORTID\_ACTION \rightarrow \mathbb{P}(PACKET)$

gg12: $theQueuesDash = theActions \lhd\!\!\!- actionsQueues$

gg13: $pkpts \subseteq PACKET \times PORTID\_ACTION$

gg14: $pkpts = rswIncomingPk^{-1}[\{sw\}] \cup \{pkt \mapsto pt\}$

then

a3: $swIncomingPk := swIncomingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pkt\}$

aa1: $rDataChan := rDataChan \setminus \{(pkt \mapsto sw) \mapsto pt\}$

aa2: $rswIncomingPk := rswIncomingPk \cup \{(pkt \mapsto pt) \mapsto sw\}$

(sw) := pkpts

aa3: $actionsQueues := theQueuesDash \cup (\bigcup aa \cdot (aa \in PORTID\_ACTION \land aa \in theActions \land aa \in dom(theQueues)) | \{aa \mapsto (theQueues(aa) \cup \{pkt\})\})$

add pkt to all actionqueue

end

Event sw_rcv_unmachingPkt ⟨ordinary⟩ $\widehat{=}$

receives a packet (from another switch) which header does not mach any entry of the flow table

extends sw_rcv_unmachingPkt

any

sw

pkt

ahd

pt now a port is specified

pkpts

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $pkt \in dom(pHeader1)$

g4: $ahd \in HEADER$

g5: $ahd = pHeader1(pkt)$

g6: $\forall ee \cdot (((ee \in ENTRY) \land (ee \in dom(flowTable))) \Rightarrow (eHeader1(ee) \neq ahd))$

does not match any entry

gg1: $pt \in PORTID\_ACTION$

gg2: $pt \in swPorts(sw)$

gg3: $((pkt \mapsto sw) \mapsto pt) \in rDataChan$

gg4: $pkpts \subseteq PACKET \times PORTID\_ACTION$

gg5: $sw \in ran(rswOutgoingPk)$

sw ∈ dom(rswOutgoingPk)

@gg6 pkpts = rswOutgoingPk(sw) ∪ {pkt ↦ pt}

then

a3: $swOutgoingPk := swOutgoingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swOPk := swOPk \cup \{sw \mapsto pkt\}$

aa1: $rDataChan := rDataChan \setminus \{(pkt \mapsto sw) \mapsto pt\}$

aa2: $rswOutgoingPk := rswOutgoingPk \cup \{(pkt \mapsto pt) \mapsto sw\}$

rswOutgoingPk(sw) := pkpts// in this case the Pk should be sent to the controller

end

Event sw_rcv_Msg ⟨ordinary⟩ $\widehat{=}$

extends sw_rcv_Msg

any

sw

msg

where

g0: $sw \in switches$

g3: $msg \in MESG$

g61: $msg \mapsto sw \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto sw\}$

a2: $swIncomingMsg := swIncomingMsg \cup \{msg \mapsto sw\}$

end

Event sw_sndPk2ctrl ⟨ordinary⟩ $\widehat{=}$

a swith emits a msg with an unmached packets to the controller /!\ TO BE REFINED with ORDERING

extends sw_sndPk2ctrl

any

sw

pkt

msg

pt now a port is specified

pkpts

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g5: $pkt \in swOutgoingPk$

g6: $msg \in MESG$

g7: $(msg \mapsto PKIn) \in mesgType$

g8: $(msg \mapsto pkt) \in mesgPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

gg4: $(msg \mapsto pt) \in mesgPt$

now mesg has port

gg5: $sw \in ran(rswOutgoingPk)$

sw ∈ dom(rswOutgoingPk)

gg6: $((pkt \mapsto pt) \mapsto sw) \in rswOutgoingPk$

(pkt ↦ pt) ∈ rswOutgoingPk(sw)

gg7: $pkpts \subseteq PACKET \times PORTID\_ACTION$

@gg7 pkpts = rswOutgoingPk(sw) \ {pkt ↦ pt}

then

a3: $swSentPkts := swSentPkts \cup \{pkt\}$

a1: $swOutgoingPk := swOutgoingPk \setminus \{pkt$

↦ pt}

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

aa1: $rswOutgoingPk := \{(pkt \mapsto pt) \mapsto sw\} \setminus rswOutgoingPk$

rswOutgoingPk(sw) := pkpts

end

Event sw_sendPcktALL ⟨ordinary⟩ $\widehat{=}$

send each pkt in the ALL queue to all interface, not including the incoming interface

extends sw_sendPcktALL

any

pkt

sws

sw

ahd

where

g0: $sw \in switches$

g1 pkt ∈ swOutgoingPk

g1: $sws \subseteq switches$

g2: $sws = switches \setminus \{sw\}$

g3: $pkt \in PACKET$

g4: $ahd \in HEADER$

to make a header for the packet

g41: $pkt \mapsto ahd \in pHeader1$

g5: $sw \in dom(swIPk)$

g6: $pkt \in swIncomingPk$

g7: $sw \mapsto pkt \in swIPk$

swIncomingPk

gg1: $sw \in switches$

gg2: $(pkt \mapsto sw) \in dataChan$

gg3: $(pkt \mapsto sw) \notin dom(rDataChan)$

gg4: $ALL \in dom(actionsQueues)$

tge port for Broadcasting

gg5: $actionsQueues(ALL) \neq \varnothing$

gg6: $pkt \in PACKET \wedge pkt \in actionsQueues(ALL)$

thtat is pk should be brodcast

then

a4: $dataChan := dataChan \cup (\{pkt\} \times sws)$

a5: $swSentPkts := swSentPkts \cup \{pkt\}$

a2: $swIPk := swIPk \setminus \{sw \mapsto pkt\}$

but not necessarily swIncomingPk := swIncomingPk \ {pkt}

a3: $swIncomingPk := swIncomingPk \setminus \{pkt\}$

aa1: $rDataChan(pkt \mapsto sw) := ALL$

Brdo := Broad ∪ ({pkt}×PORTID_ACTION)

remain update of actionsQueues(ALL) TODO here

end

Event sw_sendPckt2sw ⟨ordinary⟩ $\widehat{=}$

a switch sends a paxket to another swithc via the data channel

extends sw_sendPckt2sw

any

sw

pk

dsw ANY destination switche

pt

$pkpts$

where

g0: $sw \in switches$

g1: $dsw \in switches$

g2: $dsw \neq sw$

g3: $pk \in PACKET$

g4: $pk \in swIncomingPk$

g5: $sw \in dom(swIPk)$

g6: $sw \mapsto pk \in swIPk$

swIncomingPk

gg1: $pt \in PORTID\_ACTION$

gg2: $pt \in swPorts(sw)$

gg3: $sw \in ran(rswIncomingPk)$

sw ∈ dom(rswIncomingPk)

gg4: $((pk \mapsto pt) \mapsto sw) \in rswIncomingPk$

∈ rswIncomingPk(sw)

gg5: $(pk \mapsto sw) \notin dom(rDataChan)$

gg6: $pkpts \subseteq PACKET \times PORTID\_ACTION$

@gg8 pkpts = {pk ↦ pt} ⩤ rswIncomingPk // rswIncomingPk(sw) \ {pk ↦ pt}

then

a4: $swSentPkts := swSentPkts \cup \{pk\}$

a3: $dataChan := dataChan \cup \{pk \mapsto sw\}$

dsw

a2: $swIPk := swIPk \setminus \{sw \mapsto pk\}$

but not necesarily swIncomingPk := swIncomingPk \ {pk} // ↦ pt}

a1: $swIncomingPk := swIncomingPk \setminus \{pk\}$

aa1: $rswIncomingPk := \{(pk \mapsto pt) \mapsto sw\} \setminus rswIncomingPk$

// rswIncomingPk(sw) := pkpts

aa2: $rDataChan(pk \mapsto sw) := pt$

end

Event sw_newFTentry ⟨ordinary⟩ $\widehat{=}$

a new entry is added to the flow table

extends sw_newFTentry

any

sw

ne

nh

msg aas

aas

where

g0: $sw \in switches$

g1: $ne \in ENTRY$

g2: $ne \notin dom(flowTable)$

g3: $nh \in HEADER$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto AddE) \in mesgType$

gg1: $aas \subseteq PORTID\_ACTION$

actions

then

a3: $eHeader1(ne) := nh$

@a4 actions(ne) := aas

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $flowTable(ne) := sw$

$\cup$ {ne}

aa2: $actions(ne) := aas$

end

Event sw_modFTentry ⟨ordinary⟩ $\widehat{=}$

modifies an entry of the FT according to the actions of received packet

extends sw_modFTentry

any

sw

ne

oe

nh

msg aas

aas

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $(msg \mapsto sw) \in swIncomingMsg$

g3: $(msg \mapsto ModE) \in mesgType$

g4: $ne \in ENTRY$

g5: $ne \notin dom(flowTable)$

g6: $ne \notin dom(eHeader1)$

@g7 ne $\notin$ dom(actions)

g8: $oe \in ENTRY$

g9: $oe \in dom(flowTable)$

g10: $oe \neq ne$

g11: $nh \in HEADER$

nheader

@g12 aas $\subseteq$ ACTION // actions actions(ne) = aas

gg7: $ne \notin dom(actions)$

g12: $aas \subseteq PORTID\_ACTION$

actions actions(ne) = aas

then

a4: $flowTable := \{oe\} \lhd (flowTable \cup \{ne \mapsto sw\})$

:= (flowTable \ {oe}) ∪ {ne}

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{oe\} \lhd (eHeader1 \cup \{ne \mapsto nh\})$

@a3 actions := {oe} ⩤ (actions ∪ {ne ↦ aas})

aa3: $actions := \{oe\} \lhd (actions \cup \{ne \mapsto aas\})$

end

Event sw_delFTentry ⟨ordinary⟩ $\widehat{=}$

delete an entry ed from the flow table

extends sw_delFTentry

any

sw

ed

msg

where

g0: $sw \in switches$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto DelE) \in mesgType$

g5: $ed \in ENTRY \land ed \in dom(flowTable)$

then

a4: $flowTable := \{ed\} \lhd flowTable$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{ed\} \lhd eHeader1$

@a1 flowTable := flowTable − {ed}

@a3 actions := {ed} ⩤ actions

aa3: $actions := \{ed\} \lhd actions$

end

Event sw_rcvDataPk ⟨ordinary⟩ $\widehat{=}$

a switch receives a data pk

extends sw_rcvDataPk

any

sw

msg

pk

pt

where

g0: $sw \in switches$

g2: $msg \in MESG$

g3: $msg \mapsto sw \in swIncomingMsg$

g4: $(msg \mapsto PKOut) \in mesgType$

g1: $pk \in PACKET$

g6: $(msg \mapsto pk) \in mesgPk$

g7: $pk \notin swIncomingPk$

g10: $sw \mapsto pk \notin swIPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

@gg6 sw ∈ ran(rswIncomingPk) // ∈ dom(rswIncomingPk)

@gg8 (pk ↦ pt) ∉ dom(rswIncomingPk) // (pk ↦ pt) ∉ rswIncomingPk(sw)

@gg9 pkpts ⊆ PACKET × PORTID_ACTION

@gg8 pkpts = // pkpts = rswIncomingPk(sw) ∪ {pk ↦ pt}

then

a3: $swIncomingPk := swIncomingPk \cup \{pk\}$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pk\}$

aa1: $rswIncomingPk := rswIncomingPk \cup \{(pk \mapsto pt) \mapsto sw\}$

rswIncomingPk(sw) := pkpts

end

Event sw_getStatus ⟨ordinary⟩ $\widehat{=}$

the controler asks for a swith status

extends sw_getStatus

any

swid

pkt

msg

msgt

nmsg

nmsgt

where

g0: $swid \in SW\_ID$

g1: $swid \in dom(swStatus)$

g61: $(msg \mapsto swid) \in secureChan$

g2: $pkt \in PACKET$

the switch builds a packet with its status and sends it to the controler via the secure Chaannel

g3: $msg \in MESG$

g4: $msgt \in MESGTYPE$

g5: $msgt = askStatus$

g6: $msg \mapsto msgt \in mesgType$

g7: $nmsg \in MESG$

g8: $nmsgt \in MESGTYPE$

g9: $nmsgt = Status$

g10: $(nmsg \mapsto nmsgt) \in mesgType$

swStatus(swid)

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $swOutgoingMsg := swOutgoingMsg \cup \{nmsg\}$

with the state os the swithc {swStatus(swid)}

end

Event sw_emitMsg ⟨ordinary⟩ $\widehat{=}$

a switch sends one of its message to the controller

extends sw_emitMsg

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $msg \in swOutgoingMsg$

then

a1: $swOutgoingMsg := swOutgoingMsg \setminus \{msg\}$

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_havePacket ⟨ordinary⟩ $\widehat{=}$

simulate the disposal of packets

extends ctl_havePacket

any

pkt

sw

pt

where

g1: $pkt \in PACKET$

g2: $pkt \notin ctlOutgoingPk$

gg1: $sw \in switches$

gg2: $pt \in PORTID\_ACTION$

gg3: $sw \in dom(swPorts)$

gg4: $pt \in swPorts(sw)$

then

a1: $ctlOutgoingPk := ctlOutgoingPk \cup \{pkt\}$

aa1: $rCtlOutgoingPk(pkt \mapsto pt) := sw$

end

Event ctl_emitPkt ⟨ordinary⟩ $\widehat{=}$

the controler emits one of its pending packets, on the port portid of the switch swid, through a msg

extends ctl_emitPkt

any

$swid$

pkt

msg

ahd

pt

$macsrc$

$macdst$

ips

ipd

$iproto$

$srcPt$

$dstPt$

pty

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g2: $ctlOutgoingPk \neq \varnothing$

g3: $pkt \in ctlOutgoingPk$

one of the pakt to be sent on the sw

g4: $msg \in MESG$

to build a mesg with the pkt

g5: $ahd \in HEADER$

a header for the packet

gg2: $pt \in PORTID\_ACTION$

gg3: $pt \in swPorts(swid)$

gg4: $((pkt \mapsto pt) \mapsto swid) \in rCtlOutgoingPk$

ggg1: $macsrc \in MACADR$

ggg2: $macdst \in MACADR$

ggg3: $ips \in IPADR$

ggg4: $ipd \in IPADR$

ggg5: $iproto \in IPPROTO$

ggg6: $srcPt \in PORTID\_ACTION$

ggg7: $dstPt \in PORTID\_ACTION$

ggg8: $ips = ctlIP$

ggg9: $swid \in dom(switchesIP)$

ggg9,1: $ipd = switchesIP(swid)$

ggg10: $swid \in dom(switchesMac)$

ggg10,1: $macdst = switchesMac(swid)$

ggg11: $macsrc = ctlMac$

gg0: $pty \in MSG\_PRIORITY$

then

a5: $ctlOutgoingPk := ctlOutgoingPk \setminus \{pkt\}$

a0: $mesgType(msg) := PKOut$

the controler emit a msg with PKOut

a3: $secureChan := secureChan \cup \{msg \mapsto swid\}$

emission on the appropriate channel

a4: $ctlSentPkts := ctlSentPkts \cup \{pkt\}$

a1: $mesgPk(msg) := pkt$

a2: $pHeader1(pkt) := ahd$

aa4: $rCtlOutgoingPk := rCtlOutgoingPk \setminus \{(pkt \mapsto pt) \mapsto swid\}$

aaa1: $macSrc(pkt) := macsrc$

mac src of the contriller

aaa2: $macDst(pkt) := macdst$

aaa3: $IpSrc(pkt) := ips$

aaa4: $IpDst(pkt) := ipd$

ip src of the ctl

aaa5: $IpProto(pkt) := iproto$

aaa6: $TpSrcPt(pkt) := srcPt$

aaa7: $TpDstPt(pkt) := dstPt$

aaaa1: $msgPriority(msg) := pty$

now the messahe has prority

end

Event ctl_rcvPacketIn ⟨ordinary⟩ $\widehat{=}$

the controler receives a packet from a switch swid which previously received it on its port portid, and was unmatched

extends ctl_rcvPacketIn

any

swid

pkt

msg

pt

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto PKIn) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

gg2: $pt \in PORTID\_ACTION$

gg7: $(msg \mapsto pt) \in mesgPt$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $ctlIncomingPk := ctlIncomingPk \cup \{pkt\}$

(pkt):= swid

aa1: $rCtlIncomingPk(pkt \mapsto pt) := swid$

end

Event ctl_rcvBarrierRp ⟨ordinary⟩ $\widehat{=}$

extends ctl_rcvBarrierRp

any

$swid$

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g7: $(msg \mapsto BarrierR) \in mesgType$

Barrier Response

g8: $(msg \mapsto pkt) \in mesgPk$

g5: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_rcvStatus ⟨ordinary⟩ $\widehat{=}$

extends ctl_rcvStatus

any

$swid$

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto Status) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $msg \mapsto swid \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_askStatusMsg ⟨ordinary⟩ $\widehat{=}$

extends ctl_askStatusMsg

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

then

a1: $mesgType(msg) := askStatus$

status request

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_askBarrier ⟨ordinary⟩ $\widehat{=}$

extends ctl_askBarrier

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

then

a1: $mesgType(msg) := BarrierQ$

Barrier reQuest

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

END

MACHINE GblModel0_3
REFINES GblModel0_2
SEES EnvCtx1_2
VARIABLES

switches set of switches ID

swIPk

swOPk

swIncomingPk each switch has a set of inconming packets

swOutgoingPk each switch has a set of outgoing packets

swIncomingMsg each switch has a set of inconming

swOutgoingMsg each switch has a set of outgoing packets for controller ; they don't match any entry, and should be sent to controller

swStatus

ctlOutgoingPk controller outgoing packets

ctlSentPkts

swSentPkts all packets sent by the switches

ctlIncomingPk controller incoming packets

flowTable a switch has a flow table, it is a se of entries

secureChan

a secure channer between switch and controller

broadCChan // a secure broadCast channel

dataChan data channel between the switches

actionsQueues the queues of packet for each action

eHeader1 entry header

pHeader1 packet header

actions actios of an entry

mesgType the mesg type

mesgPk

macSrc MAC source address,

macDst MAC destination address,

macType MAC type,

IpSrc IP source address,

IpDst IP destination address,

IpProto IP protocol,

TpSrcPt transport source port,

TpDstPt transport destination port

switchesIP

switchesMac

ctlIP controller'IO

ctlMac

contraller's MAC

the mesg packet

new variables

swPorts switches now have ports

mesgPt the mesg port

rswIncomingPk now we associate a port with each (pack) for delivery

rswOutgoingPk now we associate a port with each (pack) fo delivery

rDataChan

rCtlOutgoingPk

rCtlIncomingPk additional variables

msgPriority SecureChan now msgs may have a priority

EVENTS

Initialisation ⟨extended⟩

begin

a1: $switches := \varnothing$

a2: $swIncomingMsg := \varnothing$

a3: $swOutgoingMsg := \varnothing$

a4: $ctlIncomingPk := \varnothing$

a5: $ctlOutgoingPk := \varnothing$

a6: $ctlSentPkts := \varnothing$

a8: $swSentPkts := \varnothing$

a9: $flowTable := \varnothing$

a10: $swIPk := \varnothing$

a11: $swIncomingPk := \varnothing$

a12: $swOPk := \varnothing$

a13: $swOutgoingPk := \varnothing$

a14: $secureChan := \varnothing$

a15: $dataChan := \varnothing$

a16: $eHeader1 := \varnothing$

a17: $pHeader1 := \varnothing$

a18: $swStatus := \varnothing$

a19: $mesgType := \varnothing$

a20: $mesgPk := \varnothing$

aa1: $swPorts := \varnothing$

aa2: $mesgPt := \varnothing$

aa3: $rswIncomingPk := \varnothing$

aa4: $rswOutgoingPk := \varnothing$

aa5: $rDataChan := \varnothing$

aa6: $rCtlOutgoingPk := \varnothing$

aa7: $rCtlIncomingPk := \varnothing$

aa8: $actionsQueues := PORTID\_ACTION \times \{\varnothing\}$

each queue is empty

aa9: $actions := \varnothing$

aaa1: $macSrc := PACKET \times \{DummyMac\}$

aaa2: $macDst := PACKET \times \{DummyMac\}$

aaa3: $macType := PACKET \times \{DummyMacType\}$

aaa4: $IpSrc := PACKET \times \{DummyIP\}$

aaa5: $IpDst := PACKET \times \{DummyIP\}$

aaa6: $IpProto := PACKET \times \{DummyProto\}$

aaa7: $TpSrcPt := PACKET \times \{LOCAL\}$

aaa8: $TpDstPt := PACKET \times \{LOCAL\}$

aaa9: $switchesIP := \varnothing$

aaa10: $ctlIP := DummyIP$

aaa11: $ctlMac := DummyMac$

aaa12: $switchesMac := \varnothing$

aaa: $msgPriority := \varnothing$

end

Event new_switch ⟨ordinary⟩ $\widehat{=}$

a new switch

extends new_switch

any

sw

st

$swps$ each sw has ports (linked to other switches)

nIp new fresh Ip

$nMac$

where

g0: $sw \in SW\_ID$

g1: $sw \notin switches$

g2: $st \in SW\_STATE$

gg0: $swps \subseteq PORTID\_ACTION$

gg1: $swps \neq \varnothing$

gggg1: $nIp \in IPADR$

gggg2: $nIp \notin ran(switchesIP)$

fresh IP

gggg3: $sw \mapsto nIp \notin switchesIP$

gggg4: $nMac \in MACADR$

gggg5: $nMac \notin ran(switchesMac)$

gggg6: $sw \mapsto nMac \notin switchesMac$

then

a1: $switches := switches \cup \{sw\}$

a2: $swStatus(sw) := st$

@a3 swIPk(sw) := $\varnothing$

@a4 swOPk(sw) := $\varnothing$

aa0: $swPorts(sw) := swps$

aaaa1: $switchesIP(sw) := nIp$

aaaa2: $switchesMac(sw) := nMac$

end

Event sw_rcv_machingPkt ⟨ordinary⟩ $\widehat{=}$

a switch receives a packet (from another switch) which header matches with a flow table entry

extends sw_rcv_machingPkt

any

sw

pkt

swpk

ahd msg theActions theQueues theQueuesDash

pt

theActions

theQueues

pkpts

theQueuesDash now a port is specified

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $ahd \in HEADER$

g4: $pkt \in dom(pHeader1)$

g5: $ahd = pHeader1(pkt)$

g6: $\exists ee \cdot (((ee \in ENTRY) \land (ee \in dom(flowTable))) \land (eHeader1(ee) = ahd))$

g7: $swpk \subseteq PACKET$

g8: $sw \in dom(swIPk)$

g9: $sw \mapsto pkt \notin swIPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

gg4: $((pkt \mapsto sw) \mapsto pt) \in rDataChan$

gg5: $sw \in ran(rswIncomingPk)$

sw ∈ dom(rswIncomingPk)

gg6: $(pkt \mapsto pt) \notin dom(rswIncomingPk)$

(pkt ↦ pt) ∉ rswIncomingPk(sw)

gg7: $theActions \subseteq PORTID\_ACTION$

gg8: $\exists ee \cdot ((ee \in ENTRY) \land (ee \in dom(flowTable)) \land (eHeader1(ee) = ahd) \land theActions = actions(ee))$

the actions of the matching entry, to be applied

gg9: $theQueues \in PORTID\_ACTION \nrightarrow \mathbb{P}(PACKET)$

gg10: $theQueues = theActions \lhd actionsQueues$

the queues of the current actions

gg11: $theQueuesDash \in PORTID\_ACTION \rightarrow \mathbb{P}(PACKET)$

gg12: $theQueuesDash = theActions \lhd\!\!\!- actionsQueues$

gg13: $pkpts \subseteq PACKET \times PORTID\_ACTION$

gg14: $pkpts = rswIncomingPk^{-1}[\{sw\}] \cup \{pkt \mapsto pt\}$

then

a3: $swIncomingPk := swIncomingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pkt\}$

aa1: $rDataChan := rDataChan \setminus \{(pkt \mapsto sw) \mapsto pt\}$

aa2: $rswIncomingPk := rswIncomingPk \cup \{(pkt \mapsto pt) \mapsto sw\}$

(sw) := pkpts

aa3: $actionsQueues := theQueuesDash \cup (\bigcup aa \cdot (aa \in PORTID\_ACTION \land aa \in theActions \land aa \in dom(theQueues)) | \{aa \mapsto (theQueues(aa) \cup \{pkt\})\})$

add pkt to all actionqueue

end

Event sw_rcv_unmachingPkt ⟨ordinary⟩ $\widehat{=}$

receives a packet (from another switch) which header does not mach any entry of the flow table

extends sw_rcv_unmachingPkt

any

sw

pkt

ahd

pt now a port is specified

pkpts

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $(pkt \mapsto sw) \in dataChan$

g3: $pkt \in dom(pHeader1)$

g4: $ahd \in HEADER$

g5: $ahd = pHeader1(pkt)$

g6: $\forall ee \cdot (((ee \in ENTRY) \land (ee \in dom(flowTable))) \Rightarrow (eHeader1(ee) \neq ahd))$

does not match any entry

gg1: $pt \in PORTID\_ACTION$

gg2: $pt \in swPorts(sw)$

gg3: $((pkt \mapsto sw) \mapsto pt) \in rDataChan$

gg4: $pkpts \subseteq PACKET \times PORTID\_ACTION$

gg5: $sw \in ran(rswOutgoingPk)$

sw ∈ dom(rswOutgoingPk)

@gg6 pkpts = rswOutgoingPk(sw) ∪ {pkt ↦ pt}

then

a3: $swOutgoingPk := swOutgoingPk \cup \{pkt\}$

a1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

a2: $swOPk := swOPk \cup \{sw \mapsto pkt\}$

aa1: $rDataChan := rDataChan \setminus \{(pkt \mapsto sw) \mapsto pt\}$

aa2: $rswOutgoingPk := rswOutgoingPk \cup \{(pkt \mapsto pt) \mapsto sw\}$

rswOutgoingPk(sw) := pkpts// in this case the Pk should be sent to the controller

end

Event sw_rcv_Msg ⟨ordinary⟩ $\widehat{=}$

extends sw_rcv_Msg

any

sw

msg

where

g0: $sw \in switches$

g3: $msg \in MESG$

g61: $msg \mapsto sw \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto sw\}$

a2: $swIncomingMsg := swIncomingMsg \cup \{msg \mapsto sw\}$

end

Event sw_sndPk2ctrl ⟨ordinary⟩ $\widehat{=}$

a swith emits a msg with an unmached packets to the controller /!\ TO BE REFINED with ORDERING

extends sw_sndPk2ctrl

any

sw

pkt

msg

pt now a port is specified

pkpts

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g5: $pkt \in swOutgoingPk$

g6: $msg \in MESG$

g7: $(msg \mapsto PKIn) \in mesgType$

g8: $(msg \mapsto pkt) \in mesgPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

gg4: $(msg \mapsto pt) \in mesgPt$

now mesg has port

gg5: $sw \in ran(rswOutgoingPk)$

sw ∈ dom(rswOutgoingPk)

gg6: $((pkt \mapsto pt) \mapsto sw) \in rswOutgoingPk$

(pkt ↦ pt) ∈ rswOutgoingPk(sw)

gg7: $pkpts \subseteq PACKET \times PORTID\_ACTION$

@gg7 pkpts = rswOutgoingPk(sw) \ {pkt ↦ pt}

ggg1: $rswOutgoingPk \neq \varnothing$

rswOutgoingPk is prioritary

then

a3: $swSentPkts := swSentPkts \cup \{pkt\}$

a1: $swOutgoingPk := swOutgoingPk \setminus \{pkt\}$

↦ pt}

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

aa1: $rswOutgoingPk := \{(pkt \mapsto pt) \mapsto sw\} \setminus rswOutgoingPk$

rswOutgoingPk(sw) := pkpts

end

Event sw_sendPcktALL ⟨ordinary⟩ $\widehat{=}$

send each pkt in the ALL queue to all interface, not including the incoming interface

extends sw_sendPcktALL

any

pkt

sws

sw

ahd

where

g0: $sw \in switches$

g1 pkt ∈ swOutgoingPk

g1: $sws \subseteq switches$

g2: $sws = switches \setminus \{sw\}$

g3: $pkt \in PACKET$

g4: $ahd \in HEADER$

to make a header for the packet

g41: $pkt \mapsto ahd \in pHeader1$

g5: $sw \in dom(swIPk)$

g6: $pkt \in swIncomingPk$

g7: $sw \mapsto pkt \in swIPk$

swIncomingPk

gg1: $sw \in switches$

gg2: $(pkt \mapsto sw) \in dataChan$

gg3: $(pkt \mapsto sw) \notin dom(rDataChan)$

gg4: $ALL \in dom(actionsQueues)$

tge port for Broadcasting

gg5: $actionsQueues(ALL) \neq \varnothing$

gg6: $pkt \in PACKET \wedge pkt \in actionsQueues(ALL)$

thtat is pk should be brodcast

then

a4: $dataChan := dataChan \cup (\{pkt\} \times sws)$

a5: $swSentPkts := swSentPkts \cup \{pkt\}$

a2: $swIPk := swIPk \setminus \{sw \mapsto pkt\}$

but not necessarily swIncomingPk := swIncomingPk \ {pkt}

a3: $swIncomingPk := swIncomingPk \setminus \{pkt\}$

aa1: $rDataChan(pkt \mapsto sw) := ALL$

Brdo := Broad ∪ ({pkt}×PORTID_ACTION)

remain update of actionsQueues(ALL) TODO here

end

Event sw_sendPckt2sw ⟨ordinary⟩ $\widehat{=}$

a switch sends a packet to another switch via the data channel

extends sw_sendPckt2sw

any

sw

pk

dsw ANY destination switche

pt

$pkpts$

where

g0: $sw \in switches$

g1: $dsw \in switches$

g2: $dsw \neq sw$

g3: $pk \in PACKET$

g4: $pk \in swIncomingPk$

g5: $sw \in dom(swIPk)$

g6: $sw \mapsto pk \in swIPk$

swIncomingPk

gg1: $pt \in PORTID\_ACTION$

gg2: $pt \in swPorts(sw)$

gg3: $sw \in ran(rswIncomingPk)$

sw ∈ dom(rswIncomingPk)

gg4: $((pk \mapsto pt) \mapsto sw) \in rswIncomingPk$

∈ rswIncomingPk(sw)

gg5: $(pk \mapsto sw) \notin dom(rDataChan)$

gg6: $pkpts \subseteq PACKET \times PORTID\_ACTION$

@gg8 pkpts = {pk ↦ pt} ⩤ rswIncomingPk // rswIncomingPk(sw) \ {pk ↦ pt}

then

a4: $swSentPkts := swSentPkts \cup \{pk\}$

a3: $dataChan := dataChan \cup \{pk \mapsto sw\}$

dsw

a2: $swIPk := swIPk \setminus \{sw \mapsto pk\}$

but not necesarily swIncomingPk := swIncomingPk \ {pk} // ↦ pt}

a1: $swIncomingPk := swIncomingPk \setminus \{pk\}$

aa1: $rswIncomingPk := \{(pk \mapsto pt) \mapsto sw\} \setminus rswIncomingPk$

// rswIncomingPk(sw) := pkpts

aa2: $rDataChan(pk \mapsto sw) := pt$

end

Event sw_newFTentry ⟨ordinary⟩ $\widehat{=}$

a new entry is added to the flow table

extends sw_newFTentry

any

sw

ne

nh

msg aas

aas

where

g0: $sw \in switches$

g1: $ne \in ENTRY$

g2: $ne \notin dom(flowTable)$

g3: $nh \in HEADER$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto AddE) \in mesgType$

gg1: $aas \subseteq PORTID\_ACTION$

actions

then

a3: $eHeader1(ne) := nh$

@a4 actions(ne) := aas

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $flowTable(ne) := sw$

$\cup$ {ne}

aa2: $actions(ne) := aas$

end

Event sw_modFTentry ⟨ordinary⟩ $\widehat{=}$

modifies an entry of the FT according to the actions of received packet

extends sw_modFTentry

any

sw

ne

oe

nh

msg aas

aas

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $(msg \mapsto sw) \in swIncomingMsg$

g3: $(msg \mapsto ModE) \in mesgType$

g4: $ne \in ENTRY$

g5: $ne \notin dom(flowTable)$

g6: $ne \notin dom(eHeader1)$

@g7 ne $\notin$ dom(actions)

g8: $oe \in ENTRY$

g9: $oe \in dom(flowTable)$

g10: $oe \neq ne$

g11: $nh \in HEADER$

nheader

@g12 aas $\subseteq$ ACTION // actions actions(ne) = aas

gg7: $ne \notin dom(actions)$

g12: $aas \subseteq PORTID\_ACTION$

actions actions(ne) = aas

then

a4: $flowTable := \{oe\} \lhd (flowTable \cup \{ne \mapsto sw\})$

:= (flowTable \ {oe}) ∪ {ne}

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{oe\} \lhd (eHeader1 \cup \{ne \mapsto nh\})$

@a3 actions := {oe} ⩤ (actions ∪ {ne ↦ aas})

aa3: $actions := \{oe\} \lhd (actions \cup \{ne \mapsto aas\})$

end

Event sw_delFTentry ⟨ordinary⟩ $\widehat{=}$

delete an entry ed from the flow table

extends sw_delFTentry

any

sw

ed

msg

where

g0: $sw \in switches$

g6: $msg \in MESG$

g7: $msg \mapsto sw \in swIncomingMsg$

g8: $(msg \mapsto DelE) \in mesgType$

g5: $ed \in ENTRY \wedge ed \in dom(flowTable)$

then

a4: $flowTable := \{ed\} \lhd flowTable$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $eHeader1 := \{ed\} \lhd eHeader1$

@a1 flowTable := flowTable − {ed}

@a3 actions := {ed} ⩤ actions

aa3: $actions := \{ed\} \lhd actions$

end

Event sw_rcvDataPk ⟨ordinary⟩ $\widehat{=}$

a switch receives a data pk

extends sw_rcvDataPk

any

sw

msg

pk

pt

where

g0: $sw \in switches$

g2: $msg \in MESG$

g3: $msg \mapsto sw \in swIncomingMsg$

g4: $(msg \mapsto PKOut) \in mesgType$

g1: $pk \in PACKET$

g6: $(msg \mapsto pk) \in mesgPk$

g7: $pk \notin swIncomingPk$

g10: $sw \mapsto pk \notin swIPk$

gg1: $pt \in PORTID\_ACTION$

gg2: $sw \in dom(swPorts)$

gg3: $pt \in swPorts(sw)$

@gg6 sw $\in$ ran(rswIncomingPk) // $\in$ dom(rswIncomingPk)

@gg8 (pk $\mapsto$ pt) $\notin$ dom(rswIncomingPk) // (pk $\mapsto$ pt) $\notin$ rswIncomingPk(sw)

@gg9 pkpts $\subseteq$ PACKET $\times$ PORTID_ACTION

@gg8 pkpts = // pkpts = rswIncomingPk(sw) $\cup$ {pk $\mapsto$ pt}

then

a3: $swIncomingPk := swIncomingPk \cup \{pk\}$

a1: $swIncomingMsg := swIncomingMsg \setminus \{msg \mapsto sw\}$

a2: $swIPk := swIPk \cup \{sw \mapsto pk\}$

aa1: $rswIncomingPk := rswIncomingPk \cup \{(pk \mapsto pt) \mapsto sw\}$

rswIncomingPk(sw) := pkpts

end

Event sw_getStatus ⟨ordinary⟩ $\widehat{=}$

the controler asks for a swith status

extends sw_getStatus

any

swid

pkt

msg

msgt

nmsg

nmsgt

where

g0: $swid \in SW\_ID$

g1: $swid \in dom(swStatus)$

g61: $(msg \mapsto swid) \in secureChan$

g2: $pkt \in PACKET$

the switch builds a packet with its status and sends it to the controler via the secure Chaannel

g3: $msg \in MESG$

g4: $msgt \in MESGTYPE$

g5: $msgt = askStatus$

g6: $msg \mapsto msgt \in mesgType$

g7: $nmsg \in MESG$

g8: $nmsgt \in MESGTYPE$

g9: $nmsgt = Status$

g10: $(nmsg \mapsto nmsgt) \in mesgType$

swStatus(swid)

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $swOutgoingMsg := swOutgoingMsg \cup \{nmsg\}$

with the state os the swithc {swStatus(swid)}

end

Event sw_emitMsg ⟨ordinary⟩ $\widehat{=}$

a switch sends one of its message to the controller

extends sw_emitMsg

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

g2: $msg \in swOutgoingMsg$

then

a1: $swOutgoingMsg := swOutgoingMsg \setminus \{msg\}$

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_havePacket ⟨ordinary⟩ $\widehat{=}$

simulate the disposal of packets

extends ctl_havePacket

any

pkt

sw

pt

where

g1: $pkt \in PACKET$

g2: $pkt \notin ctlOutgoingPk$

gg1: $sw \in switches$

gg2: $pt \in PORTID\_ACTION$

gg3: $sw \in dom(swPorts)$

gg4: $pt \in swPorts(sw)$

then

a1: $ctlOutgoingPk := ctlOutgoingPk \cup \{pkt\}$

aa1: $rCtlOutgoingPk(pkt \mapsto pt) := sw$

end

Event ctl_emitPkt ⟨ordinary⟩ $\widehat{=}$

the controler emits one of its pending packets, on the port portid of the switch swid, through a msg

extends ctl_emitPkt

any

$swid$

pkt

msg

ahd

pt

$macsrc$

$macdst$

ips

ipd

$iproto$

$srcPt$

$dstPt$

pty

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g2: $ctlOutgoingPk \neq \varnothing$

g3: $pkt \in ctlOutgoingPk$

one of the pakt to be sent on the sw

g4: $msg \in MESG$

to build a mesg with the pkt

g5: $ahd \in HEADER$

a header for the packet

gg2: $pt \in PORTID\_ACTION$

gg3: $pt \in swPorts(swid)$

gg4: $((pkt \mapsto pt) \mapsto swid) \in rCtlOutgoingPk$

ggg1: $macsrc \in MACADR$

ggg2: $macdst \in MACADR$

ggg3: $ips \in IPADR$

ggg4: $ipd \in IPADR$

ggg5: $iproto \in IPPROTO$

ggg6: $srcPt \in PORTID\_ACTION$

ggg7: $dstPt \in PORTID\_ACTION$

ggg8: $ips = ctlIP$

ggg9: $swid \in dom(switchesIP)$

ggg9,1: $ipd = switchesIP(swid)$

ggg10: $swid \in dom(switchesMac)$

ggg10,1: $macdst = switchesMac(swid)$

ggg11: $macsrc = ctlMac$

gg0: $pty \in MSG\_PRIORITY$

then

a5: $ctlOutgoingPk := ctlOutgoingPk \setminus \{pkt\}$

a0: $mesgType(msg) := PKOut$

the controler emit a msg with PKOut

a3: $secureChan := secureChan \cup \{msg \mapsto swid\}$

emission on the appropriate channel

a4: $ctlSentPkts := ctlSentPkts \cup \{pkt\}$

a1: $mesgPk(msg) := pkt$

a2: $pHeader1(pkt) := ahd$

aa4: $rCtlOutgoingPk := rCtlOutgoingPk \setminus \{(pkt \mapsto pt) \mapsto swid\}$

aaa1: $macSrc(pkt) := macsrc$

mac src of the contriller

aaa2: $macDst(pkt) := macdst$

aaa3: $IpSrc(pkt) := ips$

aaa4: $IpDst(pkt) := ipd$

ip src of the ctl

aaa5: $IpProto(pkt) := iproto$

aaa6: $TpSrcPt(pkt) := srcPt$

aaa7: $TpDstPt(pkt) := dstPt$

aaaa1: $msgPriority(msg) := pty$

now the messahe has prority

end

Event ctl_rcvPacketIn ⟨ordinary⟩ $\widehat{=}$

the controler receives a packet from a switch swid which previously received it on its port portid, and was unmatched

extends ctl_rcvPacketIn

any

swid

pkt

msg

pt

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto PKIn) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

gg2: $pt \in PORTID\_ACTION$

gg7: $(msg \mapsto pt) \in mesgPt$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

a2: $ctlIncomingPk := ctlIncomingPk \cup \{pkt\}$

(pkt):= swid

aa1: $rCtlIncomingPk(pkt \mapsto pt) := swid$

end

Event ctl_rcvBarrierRp ⟨ordinary⟩ $\widehat{=}$

extends ctl_rcvBarrierRp

any

swid

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g7: $(msg \mapsto BarrierR) \in mesgType$

Barrier Response

g8: $(msg \mapsto pkt) \in mesgPk$

g5: $(msg \mapsto swid) \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_rcvStatus ⟨ordinary⟩ $\widehat{=}$

extends ctl_rcvStatus

any

swid

pkt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g4: $msg \in MESG$

g5: $(msg \mapsto Status) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g8: $msg \mapsto swid \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg \mapsto swid\}$

end

Event ctl_askStatusMsg ⟨ordinary⟩ $\widehat{=}$

extends ctl_askStatusMsg

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

then

a1: $mesgType(msg) := askStatus$

status request

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

Event ctl_askBarrier ⟨ordinary⟩ $\widehat{=}$

extends ctl_askBarrier

any

sw

msg

where

g0: $sw \in switches$

g1: $msg \in MESG$

then

a1: $mesgType(msg) := BarrierQ$

Barrier reQuest

a2: $secureChan := secureChan \cup \{msg \mapsto sw\}$

end

END

MACHINE GblModel1

SEES EnvCtx1

VARIABLES

switches set of switches ID

swPorts each switch has a set of ports

swIncomingPk each switch has a set of inconming packets

swOutgoingPk each switch has a set of outgoing packets

swIncomingMsg each switch has a set of inconming

swOutgoingMsg each switch has a set of outgoing packets for controller ; they don't match any entry, and should be sent to controller

swStatus

ctlOutgoingPk controller outgoing packets

ctlIncomingPk controller incoming packets

flowTable a switch has a flow table, it is a se of entries

secureChan a secure channer between switch and controller

broadCChan a secure broadCast channel

dataChan data channel between the switches

actionsQueues the queues of packet for each action

eHeader1 entry header

pHeader1 packet header

actions actios of an entry

mesgType the mesg type

mesgPk the mesg packet

mesgPt the mesg port

INVARIANTS

i10: $switches \subseteq SW\_ID$

i11: $flowTable \in ENTRY \nrightarrow switches$

it is a se of entries

i20: $swIncomingPk \subseteq PACKET \times PORTID\_ACTION$

i30: $swOutgoingPk \subseteq PACKET \times PORTID\_ACTION$

i32: $swIncomingMsg \subseteq MESG$

i34: $swOutgoingMsg \subseteq MESG$

i36: $swPorts \in switches \rightarrow PORTID\_ACTION$

i38: $ctlIncomingPk \in (PACKET \times PORTID\_ACTION) \nrightarrow SW\_ID$

i39: $ctlOutgoingPk \in (PACKET \times PORTID\_ACTION) \nrightarrow SW\_ID$

i40: $secureChan \subseteq MESG$

i50: $broadCChan \subseteq PACKET \times PORTID\_ACTION$

i52: $dataChan \subseteq PACKET \times SW\_ID$

for the refinement (PACKET x OFSW_ID x PORTID_ACTION)

i60: $actionsQueues \in PORTID\_ACTION \pfun \mathbb{P}(PACKET)$

set of packets for each action ; the action will be applied to the packets

i62: $eHeader1 \in ENTRY \pfun HEADER$

i66: $pHeader1 \in PACKET \pfun HEADER$

i63: $dom(eHeader1) = dom(flowTable)$

i64: $actions \in ENTRY \pfun \mathbb{P}(PORTID\_ACTION)$

i65: $dom(actions) = dom(flowTable)$

i100: $swStatus \in SW\_ID \pfun SW\_STATE$

each open flow switch has a state

i110: $mesgType \in MESG \pfun MESGTYPE$

i120: $mesgPk \in MESG \pfun PACKET$

i130: $mesgPt \in MESG \pfun PORTID\_ACTION$

EVENTS

Initialisation

begin

a1: $switches := \varnothing$

a2: $swPorts := \varnothing$

a3: $swIncomingMsg := \varnothing$

a4: $swOutgoingMsg := \varnothing$

a5: $ctlIncomingPk := \varnothing$

a6: $ctlOutgoingPk := \varnothing$

a7: $flowTable := \varnothing$

a8: $swIncomingPk := \varnothing$

a9: $swOutgoingPk := \varnothing$

a10: $secureChan := \varnothing$

a11: $broadCChan := \varnothing$

a20: $dataChan := \varnothing$

a12: $actionsQueues := PORTID\_ACTION \times \{\varnothing\}$

each queue is empty

a13: $eHeader1 := \varnothing$

a14: $actions := \varnothing$

a15: $swStatus := \varnothing$

a16: $mesgType := \varnothing$

a17: $mesgPk := \varnothing$

a18: $mesgPt := \varnothing$

20: $pHeader1 := \varnothing$

end

Event sw_rcv_machingPkt ⟨ordinary⟩ $\widehat{=}$

a switch receives a packet (from another switch) which header matches with a flow table entry

any

sw

pkt

pt

msg

ahd

theActions

theQueues

theQueuesDash

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $pt \in PORTID\_ACTION$

g3: $sw \mapsto pt \in swPorts$

g4: $(pkt \mapsto sw) \in dataChan$

g5: $msg \in MESG$

g6: $(msg \mapsto PKIn) \in mesgType$

g7: $(msg \mapsto pkt) \in mesgPk$

g8: $(msg \mapsto pt) \in mesgPt$

g9: $ahd \in HEADER$

g91: $pkt \in dom(pHeader1)$

g10: $ahd = pHeader1(pkt)$

g11: $theActions \subseteq PORTID\_ACTION$

g12: $\exists ee \cdot ((ee \in ENTRY) \land (ee \in dom(flowTable)) \land (eHeader1(ee) = ahd) \land theActions = actions(ee))$

the actions of the matching entry, to be applied

g13: $theQueues \in PORTID\_ACTION \nrightarrow \mathbb{P}(PACKET)$

g14: $theQueues = theActions \lhd actionsQueues$

the queues of the current actions

g15: $theQueuesDash \in PORTID\_ACTION \rightarrow \mathbb{P}(PACKET)$

g16: $theQueuesDash = theActions \lhd\!\!\!- actionsQueues$

then

aa1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

aa2: $swIncomingPk := swIncomingPk \cup \{pkt \mapsto pt\}$

the list of actions should be applied to the packet body

aa3: $actionsQueues := theQueuesDash \cup (\bigcup aa \cdot (aa \in PORTID\_ACTION \land aa \in theActions \land aa \in dom(theQueues)) | \{aa \mapsto (theQueues(aa) \cup \{pkt\})\})$

add pkt to all actionqueue

end

Event sw_rcv_unmachingPkt ⟨ordinary⟩ $\widehat{=}$

receives a packet (from another switch) which header does not mach any entry of the flow table

any

sw

pkt

pt

ahd

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $pt \in PORTID\_ACTION$

g4: $(pkt \mapsto sw) \in dataChan$

g41: $pkt \in dom(pHeader1)$

g5: $ahd \in HEADER \land ahd = pHeader1(pkt)$

g6: $\forall ee \cdot (((ee \in ENTRY) \land (ee \in dom(flowTable))) \Rightarrow (eHeader1(ee) \neq ahd))$

then

aa1: $dataChan := dataChan \setminus \{pkt \mapsto sw\}$

aa2: $swOutgoingPk := swOutgoingPk \cup \{pkt \mapsto pt\}$

in this case the Pk should be sent to the controller

end

Event sw_sndPk2ctrl ⟨ordinary⟩ $\widehat{=}$

a swith emits a msg with an unmached packets to the controller /!\ TO BE REFINED with ORDERING

any

sw

pkt

pt

msg

where

g0: $sw \in switches$

g1: $pkt \in PACKET$

g2: $pt = swPorts(sw)$

g5: $(pkt \mapsto pt) \in swOutgoingPk$

g6: $msg \in MESG$

g7: $(msg \mapsto PKIn) \in mesgType$

g8: $(msg \mapsto pkt) \in mesgPk$

g9: $(msg \mapsto pt) \in mesgPt$

then

a1: $swOutgoingPk := swOutgoingPk \setminus \{pkt \mapsto pt\}$

a2: $secureChan := secureChan \cup \{msg\}$

end

Event sw_sendPcktALL ⟨ordinary⟩ $\widehat{=}$

send each pkt in the ALL queue to all interface, not including the incoming interface

any

pkt

pt

where

g1: $ALL \in dom(actionsQueues)$

g2: $actionsQueues(ALL) \neq \varnothing$

g3: $pkt \in PACKET \wedge pkt \in actionsQueues(ALL)$

g4: $pt \in PORTID\_ACTION$

then

a1: $broadCChan := broadCChan \cup \{pkt \mapsto pt\}$

$$$$$$$$$$$$$$$$$$$$$$$$$$

@a2 actionsQueues(ALL) := actionsQueues(ALL) − {pkt}

event sendPckINPORT

end

Event sw_sendPckt2sw ⟨ordinary⟩ $\widehat{=}$

a switch sends a paxket to another swithc via the data channel

any

sw

pk

pt

where

g0: $sw \in switches$

g1: $pk \in PACKET$

g2: $pt \in PORTID\_ACTION$

g3: $(pk \mapsto pt) \in swOutgoingPk$

then

a1: $swOutgoingPk := swOutgoingPk \setminus \{pk \mapsto pt\}$

a2: $dataChan := dataChan \cup \{pk \mapsto sw\}$

end

Event sw_newFTentry ⟨ordinary⟩ $\widehat{=}$

a new entry is added to the flow table

any

sw

ne

nh

aas

where

g0: $sw \in switches$

g1: $ne \in ENTRY$

g2: $ne \notin dom(flowTable)$

g3: $nh \in HEADER$

nheader

g4: $aas \subseteq PORTID\_ACTION$

actions

then

a1: $flowTable(ne) := sw$

$\cup$ {ne}

a2: $eHeader1(ne) := nh$

a3: $actions(ne) := aas$

end

Event sw_modFTentry ⟨ordinary⟩ $\widehat{=}$

modifies an entry of the FT according to the actions of received packet

any

sw

ne

oe

nh

aas

where

g0: $sw \in switches$

g1: $ne \in ENTRY$

g2: $ne \notin dom(flowTable)$

g3: $ne \notin dom(eHeader1)$

g4: $ne \notin dom(actions)$

g5: $oe \in ENTRY$

g6: $oe \in dom(flowTable)$

g61: $oe \neq ne$

g7: $nh \in HEADER$

nheader

g8: $aas \subseteq PORTID\_ACTION$

actions actions(ne) = aas

then

a1: $eHeader1 := \{oe\} \lhd (eHeader1 \cup \{ne \mapsto nh\})$

a2: $actions := \{oe\} \lhd (actions \cup \{ne \mapsto aas\})$

a3: $flowTable := \{oe\} \lhd (flowTable \cup \{ne \mapsto sw\})$

:= (flowTable \ {oe}) ∪ {ne}

end

Event sw_delFTentry ⟨ordinary⟩ $\widehat{=}$

delete an entry ed from the flow table

any

ed

where

g0: $ed \in ENTRY \land ed \in dom(flowTable)$

then

a2: $eHeader1 := \{ed\} \lhd eHeader1$

@a1 flowTable := flowTable − {ed}

a3: $actions := \{ed\} \lhd actions$

a4: $flowTable := \{ed\} \lhd flowTable$

end

Event sw_getStatus ⟨ordinary⟩ $\widehat{=}$

the controler asks for a swith status

any

swid

pkt

msg

msgt

nmsg

nmsgt

where

g0: $swid \in SW\_ID$

g1: $swid \in dom(swStatus)$

g2: $pkt \in PACKET$

the switch builds a packet with its status and sends it to the controler via the secure Chaannel

g3: $msg \in MESG$

g4: $msgt \in MESGTYPE$

g5: $msgt = askStatus$

g6: $msg \mapsto msgt \in mesgType$

g7: $nmsg \in MESG$

g8: $nmsgt \in MESGTYPE$

g9: $nmsgt = Status$

g10: $nmsg \mapsto nmsgt \in mesgType$

swStatus(swid)

then

a1: $secureChan := secureChan \setminus \{msg\}$

a2: $swOutgoingMsg := swOutgoingMsg \cup \{nmsg\}$

with the state os the swithc {swStatus(swid)}

end

Event sw_emitMsg ⟨ordinary⟩ $\widehat{=}$

a switch sends one of its message to the controller

any

msg

where

g1: $msg \in MESG$

g2: $msg \in swOutgoingMsg$

then

a1: $swOutgoingMsg := swOutgoingMsg \setminus \{msg\}$

a2: $secureChan := secureChan \cup \{msg\}$

end

Event ctl_emitPkt ⟨ordinary⟩ $\widehat{=}$

the controler emits one of its pending packets, on the port portid of the switch swid, through a msg

any

swid

pkt

pt

msg

where

g0: $swid \in SW\_ID$

g1: $pkt \in PACKET$

g2: $pt \in PORTID\_ACTION$

g3: $((pkt \mapsto pt) \mapsto swid) \in ctlOutgoingPk$

one of the pakt to be sent on the sw

g4: $msg \in MESG$

g5: $(msg \mapsto PKIn) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g7: $(msg \mapsto pt) \in mesgPt$

then

a0: $secureChan := secureChan \cup \{msg\}$

emission on the appropriate channel

a1: $ctlOutgoingPk := ctlOutgoingPk \setminus \{(pkt \mapsto pt) \mapsto swid\}$

end

Event ctl_rcvPacketIn ⟨ordinary⟩ $\widehat{=}$

the controler receives a packet from a switch swid which previously received it on its port portid, and was unmatched

any

swid

pkt

pt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g2: $pt \in PORTID\_ACTION$

g3: $swid \mapsto pt \in swPorts$

g4: $msg \in MESG$

g5: $(msg \mapsto PKIn) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g7: $(msg \mapsto pt) \in mesgPt$

g8: $msg \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

then

a1: $secureChan := secureChan \setminus \{msg\}$

a2: $ctlIncomingPk(pkt \mapsto pt) := swid$

end

Event ctl_rcvBarrierRp ⟨ordinary⟩ $\widehat{=}$

any

swid

pkt

pt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g2: $pt \in PORTID\_ACTION$

g3: $swid \mapsto pt \in swPorts$

g4: $msg \in MESG$

g7: $(msg \mapsto BarrierR) \in mesgType$

Barrier Response

g8: $(msg \mapsto pkt) \in mesgPk$

g9: $(msg \mapsto pt) \in mesgPt$

g5: $msg \in secureChan$

to be refined in unmacth...{(pkt ↦ pt) ↦ sw} +++ PkIn

then

a1: $secureChan := secureChan \setminus \{msg\}$

end

Event ctl_rcvStatus ⟨ordinary⟩ $\widehat{=}$

any

swid

pkt

pt

msg

where

g0: $swid \in switches$

g1: $pkt \in PACKET$

g2: $pt \in PORTID\_ACTION$

g3: $swid \mapsto pt \in swPorts$

g4: $msg \in MESG$

g5: $(msg \mapsto Status) \in mesgType$

g6: $(msg \mapsto pkt) \in mesgPk$

g7: $(msg \mapsto pt) \in mesgPt$

g8: $msg \in secureChan$

then

a1: $secureChan := secureChan \setminus \{msg\}$

end

Event ctl_askStatusMsg ⟨ordinary⟩ $\widehat{=}$

any

msg

where

g1: $msg \in MESG$

g2: $msg \mapsto askStatus \in mesgType$

then

a1: $secureChan := secureChan \cup \{msg\}$

end

Event ctl_askBarrier ⟨ordinary⟩ $\widehat{=}$

any

msg

where

g1: $msg \in MESG$

g2: $msg \mapsto BarrierQ \in mesgType$

Barrier reQuest

then

a1: $secureChan := secureChan \cup \{msg\}$

end

END