MODULE *RANSim*

LOCAL INSTANCE *Naturals*

LOCAL INSTANCE *Sequences*

LOCAL INSTANCE *FiniteSets*

LOCAL INSTANCE *TLC*

An empty value
CONSTANT *Nil*

Node states
CONSTANT *Stopped*, *Started*

Connection states
CONSTANT *Connecting*, *Connected*, *Configuring*, *Configured*, *Disconnecting*, *Disconnected*

Connection Phase
CONSTANT *Open*, *Close*

The set of *E*2 node identifiers
CONSTANT *E2Node*

ASSUME $\land$ *IsFiniteSet*(*E2Node*)
$\land \forall\, n \in E2Node : n \in$ STRING

A set of *RIC* node identifiers
CONSTANT *RICNode*

ASSUME $\land$ *IsFiniteSet*(*RICNode*)
$\land \forall\, n \in RICNode : n \in$ STRING

The state of the *E*2 node
VARIABLE *state*

The state of the network
VARIABLE *network*

The primary management connection
VARIABLE *mgmtConn*

The state of *E*2*AP* connections
VARIABLE *dataConn*

The set of outstanding transactions

VARIABLE *transactions*

Subscriptions
VARIABLE *subs*

$vars \triangleq \langle state,\ network,\ mgmtConn,\ dataConn,\ subs \rangle$

LOCAL $E2AP \triangleq$ INSTANCE $E2AP$ WITH $conns \leftarrow network$

StartNode: Starting an $E2$ node
$StartNode(e2Node) \triangleq$
$\land\ state[e2Node] = Stopped$
$\land\ state' = [state \text{ EXCEPT } ![e2Node] = Started]$
$\land$ UNCHANGED $\langle network,\ mgmtConn,\ dataConn,\ subs,\ transactions \rangle$

StopeNode: *Stoping* an $E2$ node
$StopNode(e2Node) \triangleq$
$\land\ state[e2Node] = Started$
$\land\ state' = [state \text{ EXCEPT } ![e2Node] = Stopped]$
$\land$ UNCHANGED $\langle network,\ mgmtConn,\ dataConn,\ subs,\ transactions \rangle$

Reconcile opening an $E2$ connection
$ReconcileOpenConnection(e2NodeId,\ ricNodeId) \triangleq$
$\land\ \lor\ \land\ dataConn[e2NodeId].state = Connecting$
$\land\ E2AP!Client(e2NodeId)!Connect(ricNodeId)$
$\land$ LET $newConnId \triangleq$ CHOOSE $i \in \{conn.id : conn \in network[e2NodeId]\} :$
$i \notin \{conn.id : conn \in network'[e2NodeId]\}$
IN
$\land\ dataConn' = [dataConn \text{ EXCEPT } ![e2NodeId] =$
$dataConn[e2NodeId] \mathbin{@@} (ricNodeId :>$
$[state \mapsto Connected,\ conn \mapsto newConnId])]$
$\land$ UNCHANGED $\langle transactions \rangle$
$\lor\ \land\ dataConn[e2NodeId].state \neq Connecting$
$\land\ \lor\ \land \exists\ conn \in E2AP!Client(e2NodeId)!Connections :$
$\land\ conn.id = dataConn[e2NodeId].conn$
$\land\ \lor\ \land\ dataConn[e2NodeId].state = Connecting$
$\land\ dataConn' = [dataConn \text{ EXCEPT } ![e2NodeId] = [$
$dataConn[e2NodeId] \text{ EXCEPT } ![ricNodeId].state = Connected]]$
$\land$ UNCHANGED $\langle transactions \rangle$
$\lor\ \land\ dataConn[e2NodeId].state = Connected$
$\land\ Len(transactions[e2NodeId]) < 256$
$\land$ LET $txId \triangleq$ CHOOSE $i \in 0\,..\,255 : i \notin$ DOMAIN $transactions[e2NodeId]$
$req \triangleq [txId \mapsto txId,\ e2NodeId \mapsto e2NodeId]$

```
                    IN
                        ∧ E2AP!Client(e2NodeId)!Send!E2NodeConfigurationUpdate(conn, req)
                        ∧ transactions' = [transactions EXCEPT ![e2NodeId] =
                                              transactions[e2NodeId] @@ (txId :> req)]
                        ∧ dataConn' = [dataConn EXCEPT ![e2NodeId] = [
                                          dataConn[e2NodeId] EXCEPT ![ricNodeId].state = Configuring]]
                ∨ ∧ dataConn[e2NodeId].state = Configuring
                  ∧ E2AP!Client(e2NodeId)!Ready(conn)
                  ∧ LET res ≜ E2AP!Client(e2NodeId)!Read(conn)
                    IN
                        ∧ E2AP!Client(e2NodeId)!Receive
                                !E2NodeConfigurationUpdateAcknowledge(conn, res)
                        ∧ dataConn' = [dataConn EXCEPT ![e2NodeId] = [
                                          dataConn[e2NodeId] EXCEPT ![ricNodeId].state = Configured]]
                  ∧ UNCHANGED ⟨transactions⟩
                ∨ ∧ dataConn[e2NodeId].state = Configured
                  ∧ UNCHANGED ⟨dataConn⟩
            ∨ ∧ ¬∃ conn ∈ E2AP!Client(e2NodeId)!Connections : conn.id = dataConn[e2NodeId].conn
              ∧ dataConn' = [dataConn EXCEPT ![e2NodeId] = [
                                dataConn[e2NodeId] EXCEPT ![ricNodeId] =
                                [state ↦ Connecting, conn ↦ Nil]]]
    ∧ UNCHANGED ⟨subs⟩

 Reconcile closing an E2 connection
ReconcileCloseConnection(e2NodeId, ricNodeId) ≜
    ∧ ∨ ∧ dataConn[e2NodeId].state = Disconnecting
        ∧ E2AP!Client(e2NodeId)!Disconnect(ricNodeId)

 Reconcile an E2 connection
ReconcileConnection(e2NodeId, ricNodeId) ≜
    ∧ ricNodeId ∈ dataConn[e2NodeId]
    ∨ ∧ dataConn[e2NodeId].phase = Open
      ∧ ReconcileOpenConnection(e2NodeId, ricNodeId)
    ∨ ∧ dataConn[e2NodeId].phase = Close
      ∧ ReconcileCloseConnection(e2NodeId, ricNodeId)

──────────────────────────────────────────────────────────────────────────

 An E2 node connects to a RIC instance
Connect(e2NodeId, ricNodeId) ≜
    ∧ E2AP!Client(e2NodeId)!Connect(ricNodeId)
    ∧ UNCHANGED ⟨state, dataConn, transactions⟩

 An E2 node disconnects from a RIC instance
```

$Disconnect(e2NodeId,\ conn) \triangleq$
$\quad \land\ E2AP!Client(e2NodeId)!Disconnect(conn)$
$\quad \land$ UNCHANGED $\langle state,\ dataConn,\ transactions\rangle$

An $E2$ node Sends an $E2$ setup request
$E2Setup(e2NodeId,\ conn) \triangleq$
$\quad \land \neg\exists\, c \in E2AP!Client(e2NodeId)!Connections : c.id = mgmtConn[e2NodeId].connId$
$\quad \land Len(transactions[e2NodeId]) < 256$
$\quad \land$ LET $txId \triangleq$ CHOOSE $i \in 0\,..\,255 : i \notin$ DOMAIN $transactions$
$\qquad\quad req \triangleq [txId \mapsto txId,\ e2NodeId \mapsto E2Node]$
$\qquad$ IN
$\qquad\quad \land transactions' = transactions \mathbin{@@} (txId :> req)$
$\qquad\quad \land E2AP!Client(E2Node)!Send!E2SetupRequest(conn,\ req)$
$\quad \land$ UNCHANGED $\langle mgmtConn,\ dataConn,\ subs\rangle$

Handles an $E2$ Setup Response
$HandleE2SetupResponse(e2NodeId,\ conn,\ res) \triangleq$
$\quad \land E2AP!Client(E2Node)!Receive!E2SetupResponse(conn,\ res)$
$\quad \land\ \lor \land res.txId \in$ DOMAIN $transactions[e2NodeId]$
$\qquad\quad \land mgmtConn' = [mgmtConn$ EXCEPT $![e2NodeId] = [connId \mapsto conn.id]]$
$\qquad\quad \land transactions' = [transactions$ EXCEPT $![e2NodeId] = [$
$\qquad\qquad\qquad\qquad t \in$ DOMAIN $transactions[e2NodeId] \setminus \{res.txId\} \mapsto transactions[e2NodeId][t]]]$
$\qquad \lor \land res.txId \notin transactions[e2NodeId]$
$\qquad\quad \land$ UNCHANGED $\langle mgmtConn,\ transactions\rangle$
$\quad \land$ UNCHANGED $\langle dataConn,\ subs\rangle$

Handles a *RIC* Subscription Request
$HandleRICSubscriptionRequest(e2NodeId,\ conn,\ req) \triangleq$
$\quad \land E2AP!Client(E2Node)!Receive!RICSubscriptionRequest(conn,\ req)$
$\quad \land$ UNCHANGED $\langle dataConn,\ subs\rangle$

Handles a *RIC* Subscription Delete Request
$HandleRICSubscriptionDeleteRequest(e2NodeId,\ conn,\ req) \triangleq$
$\quad \land E2AP!Client(E2Node)!Receive!RICSubscriptionDeleteRequest(conn,\ req)$
$\quad \land$ UNCHANGED $\langle dataConn,\ subs\rangle$

Handles a *RIC* Control Request
$HandleRICControlRequest(e2NodeId,\ conn,\ req) \triangleq$
$\quad \land E2AP!Client(E2Node)!Receive!RICControlRequest(conn,\ req)$
$\quad \land E2AP!Client(E2Node)!Reply!RICControlAcknowledge(conn,\ [foo \mapsto$ "bar", $bar \mapsto$ "baz"$])$
$\quad \land$ UNCHANGED $\langle dataConn,\ subs\rangle$

Handles an $E2$ Connection Update Request
$HandleE2ConnectionUpdate(e2NodeId,\ conn,\ req) \triangleq$
$\quad \land E2AP!Client(E2Node)!Receive!E2ConnectionUpdate(conn,\ req)$
$\quad \land$ LET $add \triangleq$ IF "add" $\in$ DOMAIN req THEN req["add"] ELSE $\{\}$
$\qquad\quad update \triangleq$ IF "update" $\in$ DOMAIN req THEN req["update"] ELSE $\{\}$

```
        remove ≜ IF "remove" ∈ DOMAIN req THEN req["remove"] ELSE {}
    IN
        ∧ dataConn' = [dataConn EXCEPT ![e2NodeId] = [
                        n ∈ (DOMAIN dataConn[e2NodeId] ∪ add) \ remove ↦
                          IF n ∉ update ∧ n ∈ DOMAIN dataConn THEN
                            dataConn[n]
                          ELSE
                            [state ↦ Connecting, conn ↦ Nil]]]
    ∧ UNCHANGED ⟨subs⟩

 Handles an Incoming E2 Node Configuration Update Ack
HandleE2NodeConfigurationUpdateAcknowledge(e2NodeId, conn, res) ≜
    ∧ E2AP!Client(E2Node)!Receive!E2NodeConfigurationUpdateAcknowledge(conn, res)
    ∧ res.txId ∈ transactions
    ∧ dataConn[conn.dst].state = Configuring
    ∧ transactions' = [t ∈ DOMAIN transactions \ {res.txId} ↦ transactions[t]]
    ∧ dataConn' = [dataConn EXCEPT ![conn.dst].state = Configured]
    ∧ UNCHANGED ⟨subs⟩

 Handle E2AP procedure requests and responses
HandleRequest(e2NodeId, conn) ≜
    ∧ ∨ E2AP!Client(E2Node)!Handle!RICSubscriptionRequest(conn, LAMBDA c, m :
                                HandleRICSubscriptionRequest(e2NodeId, c, m))
      ∨ E2AP!Client(E2Node)!Handle!RICSubscriptionDeleteRequest(conn, LAMBDA c, m :
                                HandleRICSubscriptionDeleteRequest(e2NodeId, c, m))
      ∨ E2AP!Client(E2Node)!Handle!RICControlRequest(conn, LAMBDA c, m :
                                HandleRICControlRequest(e2NodeId, c, m))
      ∨ E2AP!Client(E2Node)!Handle!E2ConnectionUpdate(conn, LAMBDA c, m :
                                HandleE2ConnectionUpdate(e2NodeId, c, m))
      ∨ E2AP!Client(E2Node)!Handle!E2NodeConfigurationUpdateAcknowledge(conn, LAMBDA c, m :
                                HandleE2NodeConfigurationUpdateAcknowledge(e2NodeId, c, m))
    ∧ UNCHANGED ⟨state⟩

├──────────────────────────────────────────────────────────────────────────┤

Init ≜
    ∧ E2AP!Init
    ∧ state = [n ∈ E2Node ↦ Stopped]
    ∧ mgmtConn = [n ∈ E2Node ↦ [connId ↦ Nil]]
    ∧ dataConn   = [n ∈ E2Node ↦ [c ∈ {} ↦ [connId ↦ Nil]]]
    ∧ transactions = [n ∈ E2Node ↦ [t ∈ {} ↦ [id      ↦ Nil]]]
    ∧ subs = [n ∈ E2Node ↦ [i ∈ {} ↦ [id   ↦ Nil]]]

Next ≜
    ∨ ∃ e2NodeId ∈ E2Node :
        StartNode(e2NodeId)
```

$\lor \exists\, e2NodeId \in E2Node :$
$\quad StopNode(e2NodeId)$
$\lor \exists\, e2NodeId \in E2Node,\ ricNodeId \in RICNode :$
$\quad Connect(e2NodeId,\ ricNodeId)$
$\lor \exists\, e2NodeId \in E2Node,\ ricNodeId \in RICNode :$
$\quad \exists\, conn \in E2AP!Client(e2NodeId)!Connections :$
$\quad\ Disconnect(e2NodeId,\ conn)$
$\lor \exists\, e2NodeId \in E2Node :$
$\quad \exists\, conn \in E2AP!Client(e2NodeId)!Connections :$
$\quad\ E2Setup(e2NodeId,\ conn)$
$\lor \exists\, e2NodeId \in E2Node :$
$\quad \exists\, conn \in E2AP!Client(e2NodeId)!Connections :$
$\quad\ HandleRequest(e2NodeId,\ conn)$

* Modification History
* Last modified *Thu Sep* 23 10:21:49 *PDT* 2021 by *adibrastegarnia*
* Last modified *Tue Sep* 21 15:04:44 *PDT* 2021 by *jordanhalterman*
* Created *Tue Sep* 21 13:27:29 *PDT* 2021 by *jordanhalterman*