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*

The set of $E2$ node identifiers
CONSTANT *E2Node*

ASSUME $\land\ IsFiniteSet(E2Node)$
$\quad\quad\quad\ \land\ \forall\, n \in E2Node : n \in \text{STRING}$

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

ASSUME $\land\ IsFiniteSet(RICNode)$
$\quad\quad\quad\ \land\ \forall\, n \in RICNode : n \in \text{STRING}$

The state of the $E2$ node
VARIABLE *state*

The state of the network
VARIABLE *network*

The primary management connection
VARIABLE *mgmtConn*

The state of *E2AP* connections
VARIABLE *dataConn*

The set of outstanding transactions
VARIABLE *transactions*

Subscriptions
VARIABLE *subs*

```
vars ≜ ⟨state, network, mgmtConn, dataConn, subs⟩

LOCAL E2AP ≜ INSTANCE E2AP WITH conns ← network
```

StartNode: Starting an *E*2 node

```
StartNode(e2Node) ≜
    ∧ state[e2Node] = Stopped
    ∧ state′ = [state EXCEPT ![e2Node] = Started]
    ∧ UNCHANGED ⟨network, mgmtConn, dataConn, subs, transactions⟩
```

StopeNode: *Stoping* an *E*2 node

```
StopNode(e2Node) ≜
    ∧ state[e2Node] = Started
    ∧ state′ = [state EXCEPT ![e2Node] = Stopped]
    ∧ UNCHANGED ⟨network, mgmtConn, dataConn, subs, transactions⟩
```

Reconciling an *E*2 node connection

```
ReconcileConnection(e2NodeId, ricNodeId) ≜
    ∧ ricNodeId ∈ dataConn[e2NodeId]
    ∧ ∨ ∧ dataConn[e2NodeId].state = Connecting
        ∧ E2AP!Client(e2NodeId)!Connect(ricNodeId)
        ∧ LET newConnId ≜ CHOOSE i ∈ {conn.id : conn ∈ network[e2NodeId]} :
                                 i ∉ {conn.id : conn ∈ network′[e2NodeId]}
          IN
            ∧ dataConn′ = [dataConn EXCEPT ![e2NodeId] =
                             dataConn[e2NodeId] @@ (ricNodeId :>
                             [state ↦ Connected, conn ↦ newConnId])]
            ∧ UNCHANGED ⟨transactions⟩
      ∨ ∧ dataConn[e2NodeId].state ≠ Connecting
        ∧ ∨ ∧ ∃ conn ∈ E2AP!Client(e2NodeId)!Connections :
                ∧ conn.id = dataConn[e2NodeId].conn
                ∧ ∨ ∧ dataConn[e2NodeId].state = Connecting
                    ∧ dataConn′ = [dataConn EXCEPT ![e2NodeId] = [
                                     dataConn[e2NodeId] EXCEPT ![ricNodeId].state = Connected]]
                    ∧ UNCHANGED ⟨transactions⟩
                  ∨ ∧ dataConn[e2NodeId].state = Connected
                    ∧ Len(transactions[e2NodeId]) < 256
                    ∧ LET txId ≜ CHOOSE i ∈ 0 .. 255 : i ∉ DOMAIN transactions[e2NodeId]
                          req ≜ [txId ↦ txId, e2NodeId ↦ e2NodeId]
                      IN
                        ∧ E2AP!Client(e2NodeId)!Send!E2NodeConfigurationUpdate(conn, req)
                        ∧ transactions′ = [transactions EXCEPT ![e2NodeId] =
                                             transactions[e2NodeId] @@ (txId :> req)]
                    $\land\ dataConn' = [dataConn\ \text{EXCEPT}\ ![e2NodeId] = [$
                              $dataConn[e2NodeId]\ \text{EXCEPT}\ ![ricNodeId].state = Configuring]]$
              $\lor\ \land\ dataConn[e2NodeId].state = Configuring$
                 $\land\ E2AP!Client(e2NodeId)!Ready(conn)$
                 $\land\ \text{LET}\ res \triangleq E2AP!Client(e2NodeId)!Read(conn)$
                    IN
                    $\land\ E2AP!Client(e2NodeId)!Receive$
                          $!E2NodeConfigurationUpdateAcknowledge(conn, res)$
                    $\land\ dataConn' = [dataConn\ \text{EXCEPT}\ ![e2NodeId] = [$
                              $dataConn[e2NodeId]\ \text{EXCEPT}\ ![ricNodeId].state = Configured]]$
                 $\land\ \text{UNCHANGED}\ \langle transactions \rangle$
              $\lor\ \land\ dataConn[e2NodeId].state = Configured$
                 $\land\ \text{UNCHANGED}\ \langle dataConn \rangle$
        $\lor\ \land\ \neg\exists\ conn \in E2AP!Client(e2NodeId)!Connections : conn.id = dataConn[e2NodeId].conn$
           $\land\ dataConn' = [dataConn\ \text{EXCEPT}\ ![e2NodeId] = [$
                       $dataConn[e2NodeId]\ \text{EXCEPT}\ ![ricNodeId] =$
                       $[state \mapsto Connecting, conn \mapsto Nil]]]$
  $\land\ \text{UNCHANGED}\ \langle subs \rangle$
```

An $E2$ node connects to a RIC instance

```
$Connect(e2NodeId, ricNodeId) \triangleq$
  $\land\ E2AP!Client(e2NodeId)!Connect(ricNodeId)$
  $\land\ \text{UNCHANGED}\ \langle state, dataConn, transactions \rangle$
```

An $E2$ node disconnects from a RIC instance

```
$Disconnect(e2NodeId, conn) \triangleq$
  $\land\ E2AP!Client(e2NodeId)!Disconnect(conn)$
  $\land\ \text{UNCHANGED}\ \langle state, dataConn, transactions \rangle$
```

An $E2$ node Sends an $E2$ setup request

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

Handles an $E2$ Setup Response

```
$HandleE2SetupResponse(e2NodeId, conn, res) \triangleq$
  $\land\ E2AP!Client(E2Node)!Receive!E2SetupResponse(conn, res)$
  $\land\ \lor\ \land\ res.txId \in \text{DOMAIN}\ transactions[e2NodeId]$
```

```
            $\land\ mgmtConn' = [mgmtConn\ \textsc{except}\ ![e2NodeId] = [connId \mapsto conn.id]]$
            $\land\ transactions' = [transactions\ \textsc{except}\ ![e2NodeId] = [$
                        $t \in \textsc{domain}\ transactions[e2NodeId] \setminus \{res.txId\} \mapsto transactions[e2NodeId][t]]]$
        $\lor\ \land\ res.txId \notin transactions[e2NodeId]$
            $\land\ \textsc{unchanged}\ \langle mgmtConn, transactions \rangle$
    $\land\ \textsc{unchanged}\ \langle dataConn, subs \rangle$
```

Handles a *RIC* Subscription Request

```
$HandleRICSubscriptionRequest(e2NodeId, conn, req) \triangleq$
    $\land\ E2AP!Client(E2Node)!Receive!RICSubscriptionRequest(conn, req)$
    $\land\ \textsc{unchanged}\ \langle dataConn, subs \rangle$
```

Handles a *RIC* Subscription Delete Request

```
$HandleRICSubscriptionDeleteRequest(e2NodeId, conn, req) \triangleq$
    $\land\ E2AP!Client(E2Node)!Receive!RICSubscriptionDeleteRequest(conn, req)$
    $\land\ \textsc{unchanged}\ \langle dataConn, subs \rangle$
```

Handles a *RIC* Control Request

```
$HandleRICControlRequest(e2NodeId, conn, req) \triangleq$
    $\land\ E2AP!Client(E2Node)!Receive!RICControlRequest(conn, req)$
    $\land\ E2AP!Client(E2Node)!Reply!RICControlAcknowledge(conn, [foo \mapsto \text{"bar"}, bar \mapsto \text{"baz"}])$
    $\land\ \textsc{unchanged}\ \langle dataConn, subs \rangle$
```

Handles an *E*2 Connection Update Request

```
$HandleE2ConnectionUpdate(e2NodeId, conn, req) \triangleq$
    $\land\ E2AP!Client(E2Node)!Receive!E2ConnectionUpdate(conn, req)$
    $\land\ \textsc{let}\ add \triangleq \textsc{if}\ \text{"add"} \in \textsc{domain}\ req\ \textsc{then}\ req[\text{"add"}]\ \textsc{else}\ \{\}$
            $update \triangleq \textsc{if}\ \text{"update"} \in \textsc{domain}\ req\ \textsc{then}\ req[\text{"update"}]\ \textsc{else}\ \{\}$
            $remove \triangleq \textsc{if}\ \text{"remove"} \in \textsc{domain}\ req\ \textsc{then}\ req[\text{"remove"}]\ \textsc{else}\ \{\}$
      $\textsc{in}$
        $\land\ dataConn' = [dataConn\ \textsc{except}\ ![e2NodeId] = [$
                        $n \in (\textsc{domain}\ dataConn[e2NodeId] \cup add) \setminus remove \mapsto$
                          $\textsc{if}\ n \notin update \land n \in \textsc{domain}\ dataConn\ \textsc{then}$
                            $dataConn[n]$
                           $\textsc{else}$
                            $[state \mapsto Connecting, conn \mapsto Nil]]]$
    $\land\ \textsc{unchanged}\ \langle subs \rangle$
```

Handles an Incoming *E*2 Node Configuration Update *Ack*

```
$HandleE2NodeConfigurationUpdateAcknowledge(e2NodeId, conn, res) \triangleq$
    $\land\ E2AP!Client(E2Node)!Receive!E2NodeConfigurationUpdateAcknowledge(conn, res)$
    $\land\ res.txId \in transactions$
    $\land\ dataConn[conn.dst].state = Configuring$
    $\land\ transactions' = [t \in \textsc{domain}\ transactions \setminus \{res.txId\} \mapsto transactions[t]]$
    $\land\ dataConn' = [dataConn\ \textsc{except}\ ![conn.dst].state = Configured]$
    $\land\ \textsc{unchanged}\ \langle subs \rangle$
```

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)
    ∨ ∃ e2NodeId ∈ E2Node :
        StopNode(e2NodeId)
    ∨ ∃ e2NodeId ∈ E2Node, ricNodeId ∈ RICNode :
        Connect(e2NodeId, ricNodeId)
    ∨ ∃ e2NodeId ∈ E2Node, ricNodeId ∈ RICNode :
        ∃ conn ∈ E2AP!Client(e2NodeId)!Connections :
          Disconnect(e2NodeId, conn)
    ∨ ∃ e2NodeId ∈ E2Node :
        ∃ conn ∈ E2AP!Client(e2NodeId)!Connections :
          E2Setup(e2NodeId, conn)
    ∨ ∃ e2NodeId ∈ E2Node :
        ∃ conn ∈ E2AP!Client(e2NodeId)!Connections :
          HandleRequest(e2NodeId, conn)
```

\* Modification History
\* Last modified *Wed Sep* 22 15:36:29 *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*