MODULE $E2T$

LOCAL INSTANCE *Naturals*

LOCAL INSTANCE *Sequences*

LOCAL INSTANCE *FiniteSets*

LOCAL INSTANCE *TLC*

An empty value

CONSTANT *Nil*

Node states

CONSTANT *Stopped*, *Started*

A set of $E2T$ node identifiers

CONSTANT *E2Term*

ASSUME $\land$ *IsFiniteSet*(*E2Term*)
$\land \forall n \in E2Term : n \in$ STRING

A mapping of node states

VARIABLE *state*

gRPC connection states

VARIABLE *grpc*

SCTP connection states

VARIABLE *sctp*

A global store of mastership for each $E2$ node

VARIABLE *masterships*

A global store of configuration for each $E2$ node

VARIABLE *nodes*

A global store of connections for each $E2$ node

VARIABLE *conns*

A store of streams for each node

VARIABLE *streams*

A global store of channel states

VARIABLE *chans*

A global store of subscription states

VARIABLE *subs*

```
$vars \triangleq \langle state, masterships, grpc, sctp, streams, chans, subs \rangle$

$\textsc{local}\ API \triangleq \textsc{instance}\ E2TService\ \textsc{with}\ conns \leftarrow grpc$

$\textsc{local}\ E2AP \triangleq \textsc{instance}\ E2AP\ \textsc{with}\ conns \leftarrow sctp$
```

```
$StartNode(e2TermID) \triangleq$
    $\land state[e2TermID] = Stopped$
    $\land state' = [state\ \textsc{except}\ ![e2TermID] = Started]$
    $\land E2AP!Server(e2TermID)!Start$
    $\land \textsc{unchanged}\ \langle masterships, conns, streams, chans, subs \rangle$

$StopNode(e2TermID) \triangleq$
    $\land state[e2TermID] = Started$
    $\land state' = [state\ \textsc{except}\ ![e2TermID] = Stopped]$
    $\land E2AP!Server(e2TermID)!Start$
    $\land streams' = [streams\ \textsc{except}\ ![e2TermID] = [id \in \{\} \mapsto [id \mapsto Nil]]]$
    $\land \textsc{unchanged}\ \langle masterships, conns, chans, subs \rangle$
```

```
$HandleSubscribeRequest(e2TermID, apiConn, apiReq) \triangleq$
    $\land \lor \land apiReq.sub.id \notin streams[e2TermID]$
            $\land streams' = [streams\ \textsc{except}\ ![e2TermID] = streams[e2TermID] \mathbin{@@} (apiReq.sub.id :> [id \mapsto apiReq$
       $\lor \land apiReq.sub.id \in streams[e2TermID]$
            $\land \textsc{unchanged}\ \langle streams \rangle$
    $\land \textsc{unchanged}\ \langle chans, subs \rangle$

$SendSubscribeResponse(e2TermID, apiConn, s) \triangleq$
    $\land Len(streams[e2TermID][s]) > 0$
    $\land API!Server!Send!SubscribeResponse(apiConn, [indication \mapsto streams[e2TermID][s][1]])$
    $\land streams' = [streams\ \textsc{except}\ ![e2TermID] = [streams[e2TermID]\ \textsc{except}\ ![s] = SubSeq(streams[e2Tern$
    $\land \textsc{unchanged}\ \langle chans, subs \rangle$

$HandleUnsubscribeRequest(e2TermID, apiConn, apiReq) \triangleq$
    $\land \lor \land apiReq.sub.id \notin streams[e2TermID]$
            $\land streams' = [streams\ \textsc{except}\ ![e2TermID] = [i \in \{subId \in \textsc{domain}\ streams[e2TermID] : subId \neq a$
       $\lor \land apiReq.sub.id \in streams[e2TermID]$
            $\land \textsc{unchanged}\ \langle streams \rangle$
    $\land API!Server!Reply!UnsubscribeResponse(apiConn, [id \mapsto apiReq.id])$
    $\land \textsc{unchanged}\ \langle chans, subs \rangle$

$HandleControlRequest(e2TermID, apiConn, apiReq) \triangleq$
    $\land API!Server!Reply!ControlResponse(apiConn, [foo \mapsto \text{“bar”}, bar \mapsto \text{“baz”}])$
    $\land \textsc{unchanged}\ \langle chans, subs \rangle$
```

$$\begin{array}{l}
\mathit{HandleE2TRequest}(\mathit{e2TermID},\ \mathit{apiConn}) \triangleq \\
\quad \land\ \lor\ \mathit{API}!\mathit{Server}!\mathit{Handle}!\mathit{SubscribeRequest}(\mathit{apiConn},\ \textsc{lambda}\ m : \mathit{HandleSubscribeRequest}(\mathit{e2TermID},\ \mathit{ap} \\
\quad \ \lor\ \mathit{API}!\mathit{Server}!\mathit{Handle}!\mathit{UnsubscribeRequest}(\mathit{apiConn},\ \textsc{lambda}\ m : \mathit{HandleUnsubscribeRequest}(\mathit{e2TermI} \\
\quad \ \lor\ \mathit{API}!\mathit{Server}!\mathit{Handle}!\mathit{ControlRequest}(\mathit{apiConn},\ \textsc{lambda}\ m : \mathit{HandleControlRequest}(\mathit{e2TermID},\ \mathit{apiCo} \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{state} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{ReconcileMastership}(\mathit{e2TermID},\ \mathit{e2NodeID}) \triangleq \\
\quad \land\ \mathit{masterships}[\mathit{e2NodeID}].\mathit{master} \notin \textsc{domain}\ \mathit{conns}[\mathit{e2NodeID}] \\
\quad \land\ \exists\, c \in \textsc{domain}\ \mathit{conns}[\mathit{e2NodeID}] : c \neq \mathit{masterships}[\mathit{e2NodeID}].\mathit{master} \\
\quad \land\ \mathit{masterships}' = [\mathit{masterships}\ \textsc{except}\ ![\mathit{e2NodeID}] = [\\
\qquad\qquad\qquad\qquad \mathit{term} \mapsto \mathit{masterships}[\mathit{e2NodeID}].\mathit{term} + 1, \\
\qquad\qquad\qquad\qquad \mathit{conn} \mapsto \textsc{choose}\ c \in \textsc{domain}\ \mathit{conns}[\mathit{e2NodeID}] : c \neq \mathit{masterships}[\mathit{e2NodeID}].\mathit{master}] \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{state},\ \mathit{subs} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{ReconcileStream}(n,\ s) \triangleq \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{state},\ \mathit{subs} \rangle
\end{array}$$

ReconcileChannel reconciles a channel's state

$$\begin{array}{l}
\mathit{ReconcileChannel}(n,\ c) \triangleq \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{state},\ \mathit{streams} \rangle
\end{array}$$

ReconcileSubscription reconciles a subscription's state

$$\begin{array}{l}
\mathit{ReconcileSubscription}(n,\ s) \triangleq \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{state},\ \mathit{streams},\ \mathit{chans} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{HandleE2SetupRequest}(\mathit{node},\ \mathit{conn},\ \mathit{res}) \triangleq \\
\quad \land\ \mathit{E2AP}!\mathit{Server}(\mathit{node})!\mathit{Reply}!\mathit{E2SetupResponse}(\mathit{conn},\ [\mathit{foo} \mapsto \text{“bar”},\ \mathit{bar} \mapsto \text{“baz”}]) \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{chans},\ \mathit{subs} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{HandleRICControlResponse}(\mathit{node},\ \mathit{conn},\ \mathit{res}) \triangleq \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{chans},\ \mathit{subs} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{HandleRICSubscriptionResponse}(\mathit{node},\ \mathit{conn},\ \mathit{res}) \triangleq \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{chans},\ \mathit{subs} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{HandleRICSubscriptionDeleteResponse}(\mathit{node},\ \mathit{conn},\ \mathit{res}) \triangleq \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{chans},\ \mathit{subs} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{HandleRICIndication}(\mathit{node},\ \mathit{conn},\ \mathit{res}) \triangleq \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{chans},\ \mathit{subs} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{HandleE2APRequest}(\mathit{node},\ \mathit{conn}) \triangleq \\
\quad \land\ \lor\ \mathit{E2AP}!\mathit{Server}(\mathit{node})!\mathit{Handle}!\mathit{E2SetupRequest}(\mathit{conn},\ \textsc{lambda}\ c,\ m : \mathit{HandleE2SetupRequest}(\mathit{node},\ \mathit{co} \\
\quad \ \lor\ \mathit{E2AP}!\mathit{Server}(\mathit{node})!\mathit{Handle}!\mathit{RICControlResponse}(\mathit{conn},\ \textsc{lambda}\ c,\ m : \mathit{HandleRICControlResponse} \\
\quad \ \lor\ \mathit{E2AP}!\mathit{Server}(\mathit{node})!\mathit{Handle}!\mathit{RICSubscriptionResponse}(\mathit{conn},\ \textsc{lambda}\ c,\ m : \mathit{HandleRICSubscription}
\end{array}$$

```
        $\lor E2AP!Server(node)!Handle!RICSubscriptionDeleteResponse(conn,$ LAMBDA $c, m : HandleRICSubsc$
        $\lor E2AP!Server(node)!Handle!RICIndication(conn,$ LAMBDA $c, m : HandleRICIndication(node, conn,$
    $\land$ UNCHANGED $\langle state \rangle$
```

```
Init $\triangleq$
    $\land state = [e2TermID \in E2Term \mapsto Stopped]$
    $\land masterships = [e2TermID \in E2Term \mapsto [e \in \{\} \mapsto [master \mapsto Nil, term \mapsto 0]]]$
    $\land nodes = [e \in \{\} \mapsto [version \mapsto 0, conns \mapsto \{\}]]$
    $\land conns = [e \in \{\} \mapsto [id \mapsto Nil]]$
    $\land streams = [n \in E2Term \mapsto [x \in \{\} \mapsto [id \mapsto x]]]$
    $\land chans = [x \in \{\} \mapsto [id \mapsto x]]$
    $\land subs = [x \in \{\} \mapsto [id \mapsto x]]$

Next $\triangleq$
    $\lor \exists n \in E2Term :$
        $StartNode(n)$
    $\lor \exists n \in E2Term :$
        $StopNode(n)$
    $\lor \exists n \in E2Term, c \in API!Connections :$
        $HandleE2TRequest(n, c)$
    $\lor \exists n \in E2Term, c \in API!Connections :$
        $\exists s \in$ DOMAIN $streams[n] :$
          $SendSubscribeResponse(n, c, s)$
    $\lor \exists n \in E2Term :$
        $\exists c \in E2AP!Server(n)!Connections :$
          $HandleE2APRequest(n, c)$
    $\lor \exists n \in E2Term :$
        $\exists e \in$ DOMAIN $nodes[n] :$
          $ReconcileMastership(n, e)$
    $\lor \exists n \in E2Term :$
        $\exists s \in$ DOMAIN $streams[n] :$
          $ReconcileStream(n, s)$
    $\lor \exists n \in E2Term, c \in chans :$
        $ReconcileChannel(n, c)$
    $\lor \exists n \in E2Term, s \in subs :$
        $ReconcileSubscription(n, s)$
```

---

* Modification History
* Last modified *Tue Sep* 21 18:16:38 *PDT* 2021 by *jordanhalterman*
* Created *Mon Sep* 13 03:23:39 *PDT* 2021 by *jordanhalterman*