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 E2TNodes

ASSUME ∧ IsFiniteSet(E2TNodes)
       ∧ ∀ n ∈ E2TNodes : n ∈ STRING

 A set of E2 node identifiers
CONSTANT E2Nodes

ASSUME ∧ IsFiniteSet(E2Nodes)
       ∧ ∀ n ∈ E2Nodes : n ∈ STRING

 A mapping of node states
VARIABLE nodes

 A global store of mastership for each E2 node
VARIABLE masterships

 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 ≜ ⟨nodes, masterships, conns, streams, chans, subs⟩

LOCAL API ≜ INSTANCE E2TService
```

```
$\textsc{local}\ E2AP \triangleq \textsc{instance}\ E2AP$
```

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

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

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

$SendSubscribeResponse(n,\ c,\ s) \triangleq$
    $\land\ Len(streams[n][s]) > 0$
    $\land\ API!Server!Send!SubscribeResponse(c,\ [indication \mapsto streams[n][s][1]])$
    $\land\ streams' = [streams\ \textsc{except}\ ![n] = [streams[n]\ \textsc{except}\ ![s] = SubSeq(streams[n][s],\ 2,\ Len(streams[n][s$
    $\land\ \textsc{unchanged}\ \langle chans,\ subs \rangle$

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

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

$HandleE2TRequest(n,\ c) \triangleq$
    $\land\ \lor\ API!Server!Handle!SubscribeRequest(c,\ \textsc{lambda}\ m : HandleSubscribeRequest(n,\ c,\ m))$
        $\lor\ API!Server!Handle!UnsubscribeRequest(c,\ \textsc{lambda}\ m : HandleUnsubscribeRequest(n,\ c,\ m))$
        $\lor\ API!Server!Handle!ControlRequest(c,\ \textsc{lambda}\ m : HandleControlRequest(n,\ c,\ m))$
    $\land\ \textsc{unchanged}\ \langle nodes \rangle$
```

```
$ReconcileMastership(n, e) \triangleq$
    $\land masterships[e].master \notin \text{DOMAIN } conns[e]$
    $\land \exists c \in \text{DOMAIN } conns[e] : c \neq masterships[e].master$
    $\land masterships' = [masterships \text{ EXCEPT } ![e] = [$
                              $term \mapsto masterships[e].term + 1,$
                              $conn \mapsto \text{CHOOSE } c \in \text{DOMAIN } conns[e] : c \neq masterships[e].master]]$
    $\land \text{UNCHANGED } \langle nodes, subs \rangle$

$ReconcileStream(n, s) \triangleq$
    $\land \text{UNCHANGED } \langle nodes, subs \rangle$

 ReconcileChannel reconciles a channel's state
$ReconcileChannel(n, c) \triangleq$
    $\land \text{UNCHANGED } \langle nodes, streams \rangle$

 ReconcileSubscription reconciles a subscription's state
$ReconcileSubscription(n, s) \triangleq$
    $\land \text{UNCHANGED } \langle nodes, streams, chans \rangle$
```

```
$HandleE2SetupRequest(node, conn, res) \triangleq$
    $\land E2AP!RIC!Reply!E2SetupResponse(conn, [foo \mapsto \text{"bar"}, bar \mapsto \text{"baz"}])$
    $\land \text{UNCHANGED } \langle chans, subs \rangle$

$HandleRICControlResponse(node, conn, res) \triangleq$
    $\land \text{UNCHANGED } \langle chans, subs \rangle$

$HandleRICSubscriptionResponse(node, conn, res) \triangleq$
    $\land \text{UNCHANGED } \langle chans, subs \rangle$

$HandleRICSubscriptionDeleteResponse(node, conn, res) \triangleq$
    $\land \text{UNCHANGED } \langle chans, subs \rangle$

$HandleRICIndication(node, conn, res) \triangleq$
    $\land \text{UNCHANGED } \langle chans, subs \rangle$

$HandleE2APRequest(node, conn) \triangleq$
    $\land \lor E2AP!RIC!Handle!E2SetupRequest(conn, \text{LAMBDA } m : HandleE2SetupRequest(node, conn, m))$
       $\lor E2AP!RIC!Handle!RICControlResponse(conn, \text{LAMBDA } m : HandleRICControlResponse(node, conn$
       $\lor E2AP!RIC!Handle!RICSubscriptionResponse(conn, \text{LAMBDA } m : HandleRICSubscriptionResponse($
       $\lor E2AP!RIC!Handle!RICSubscriptionDeleteResponse(conn, \text{LAMBDA } m : HandleRICSubscriptionDele$
       $\lor E2AP!RIC!Handle!RICIndication(conn, \text{LAMBDA } m : HandleRICIndication(node, conn, m))$
    $\land \text{UNCHANGED } \langle nodes \rangle$
```

$Init \triangleq$

$\quad \land\ nodes = [n \in E2TNodes \mapsto Stopped]$
$\quad \land\ masterships = [e \in E2Nodes \mapsto [master \mapsto Nil,\ term \mapsto 0]]$
$\quad \land\ conns = [e \in E2Nodes \mapsto [c \in \{\} \mapsto [id \mapsto c,\ e2node \mapsto Nil,\ e2t \mapsto Nil]]]$
$\quad \land\ streams = [n \in E2TNodes \mapsto [x \in \{\} \mapsto [id \mapsto x]]]$
$\quad \land\ chans = [x \in \{\} \mapsto [id \quad\quad \mapsto x]]$
$\quad \land\ subs = [x \in \{\} \mapsto [id \mapsto x]]$

$Next \triangleq$

$\quad \lor\ \exists\, n \in E2TNodes : StartNode(n)$
$\quad \lor\ \exists\, n \in E2TNodes : StopNode(n)$
$\quad \lor\ \exists\, n \in E2TNodes,\ c \in API!Connections : HandleE2TRequest(n,\ c)$
$\quad \lor\ \exists\, n \in E2TNodes,\ c \in API!Connections : \exists\, s \in \textsc{domain}\ streams[n] : SendSubscribeResponse(n,\ c,\ s)$
$\quad \lor\ \exists\, n \in E2TNodes,\ c \in E2AP!Connections : HandleE2APRequest(n,\ c)$
$\quad \lor\ \exists\, n \in E2TNodes,\ e \in E2Nodes : ReconcileMastership(n,\ e)$
$\quad \lor\ \exists\, n \in E2TNodes : \exists\, s \in \textsc{domain}\ streams[n] : ReconcileStream(n,\ s)$
$\quad \lor\ \exists\, n \in E2TNodes,\ c \in chans : ReconcileChannel(n,\ c)$
$\quad \lor\ \exists\, n \in E2TNodes,\ s \in subs : ReconcileSubscription(n,\ s)$

\* Modification History
\* Last modified *Mon Sep* 13 19:25:13 *PDT* 2021 by *jordanhalterman*
\* *Created Mon Sep* 13 03:23:39 *PDT* 2021 by *jordanhalterman*