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 $\land IsFiniteSet(E2TNodes)$
$\land \forall\, n \in E2TNodes : n \in \text{STRING}$

A set of $E2$ node identifiers
CONSTANT $E2Nodes$

ASSUME $\land IsFiniteSet(E2Nodes)$
$\land \forall\, n \in E2Nodes : n \in \text{STRING}$

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 $subs$

$vars \triangleq \langle nodes, masterships, conns, streams, chans, subs \rangle$

LOCAL $API \triangleq$ INSTANCE $E2TService$

LOCAL $E2AP \triangleq$ INSTANCE $E2AP$

$StartNode(n) \triangleq$
$\land\ nodes[n] = Stopped$
$\land\ nodes' = [nodes \text{ EXCEPT } ![n] = Started]$
$\land$ UNCHANGED $\langle masterships,\ conns,\ streams,\ chans,\ subs\rangle$

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

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

$HandleUnsubscribeRequest(n,\ c,\ r) \triangleq$
$\land\ \lor\ \land\ r.sub.id \notin streams[n]$
$\land\ streams' = [streams \text{ EXCEPT } ![n] = [i \in \{subId \in \text{DOMAIN } streams[n] : subId \neq r.id\} \mapsto streams[n]$
$\lor\ \land\ r.sub.id \in streams[n]$
$\land$ UNCHANGED $\langle streams\rangle$
$\land\ API!Server!Send!SubscribeResponse(c,\ [id \mapsto r.id])$
$\land$ UNCHANGED $\langle chans,\ subs\rangle$

$HandleControlRequest(n,\ c,\ r) \triangleq$
$\land\ API!Server!Send!ControlResponse(c,\ [foo \mapsto$ "bar", $bar \mapsto$ "baz"$])$
$\land$ UNCHANGED $\langle chans,\ subs\rangle$

$HandleE2TRequest(n,\ c) \triangleq$
$\land\ \lor\ API!Server!Handle!SubscribeRequest(c,$ LAMBDA $m : HandleSubscribeRequest(n,\ c,\ m))$
$\lor\ API!Server!Handle!UnsubscribeRequest(c,$ LAMBDA $m : HandleUnsubscribeRequest(n,\ c,\ m))$
$\lor\ API!Server!Handle!ControlRequest(c,$ LAMBDA $m : HandleControlRequest(n,\ c,\ m))$
$\land$ UNCHANGED $\langle nodes\rangle$

$ReconcileMastership(n,\ e) \triangleq$
$\land\ masterships[e].master \notin \text{DOMAIN } conns[e]$
$\land\ \exists\, c \in \text{DOMAIN } conns[e] : c \neq masterships[e].master$
$\land\ masterships' = [masterships \text{ EXCEPT } ![e] = [$

$term \mapsto masterships[e].term + 1,$
$conn \mapsto$ CHOOSE $c \in$ DOMAIN $conns[e] : c \neq masterships[e].master]]$
$\land$ UNCHANGED $\langle nodes, subs \rangle$

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

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

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

$HandleE2SetupRequest(node, conn, res) \triangleq$
$\land$ UNCHANGED $\langle chans, subs \rangle$

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

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

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

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

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

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

$$
\begin{array}{l}
Next \triangleq \\
\quad \lor \exists\, n \in E2TNodes : StartNode(n) \\
\quad \lor \exists\, n \in E2TNodes : StopNode(n) \\
\quad \lor \exists\, n \in E2TNodes,\ c \in API!Connections : HandleE2TRequest(n,\ c) \\
\quad \lor \exists\, n \in E2TNodes,\ c \in E2AP!Connections : HandleE2APRequest(n,\ c) \\
\quad \lor \exists\, n \in E2TNodes,\ e \in E2Nodes : ReconcileMastership(n,\ e) \\
\quad \lor \exists\, n \in E2TNodes : \exists\, s \in streams[n] : ReconcileStream(n,\ s) \\
\quad \lor \exists\, n \in E2TNodes,\ c \in chans : ReconcileChannel(n,\ c) \\
\quad \lor \exists\, n \in E2TNodes,\ s \in subs : ReconcileSubscription(n,\ s)
\end{array}
$$

* Modification History
* Last modified *Mon Sep* 13 19:08:57 *PDT* 2021 by *jordanhalterman*
* *Created Mon Sep* 13 03:23:39 *PDT* 2021 by *jordanhalterman*