──────────────────────── MODULE *E2TService* ────────────────────────

The *E2TService* module provides a formal specification of the *E2T* service. The spec defines the client and server interfaces for *E2T* and provides helpers for managing and operating on connections.

CONSTANT *Nil*

VARIABLE *conns*

$gRPC \triangleq$ INSTANCE *gRPC* WITH
    $OK \leftarrow$ "OK",
    $Error \leftarrow$ "Error"

LOCAL INSTANCE *TLC*

$vars \triangleq \langle conns \rangle$

──────────────────────── MODULE *Messages* ────────────────────────

The *Messages* module defines predicates for receiving, sending, and verifying all the messages supported by *E2T*.

   Message type constants
CONSTANT
    SubscribeRequest,
    SubscribeResponse
CONSTANTS
    UnsubscribeRequest,
    UnsubscribeResponse
CONSTANTS
    ControlRequest,
    ControlResponse

LOCAL *messageTypes* $\triangleq$
    {*SubscribeRequest*,
     SubscribeResponse,
     UnsubscribeRequest,
     UnsubscribeResponse,
     ControlRequest,
     ControlResponse}

   Message types should be defined as strings to simplify debugging
ASSUME $\forall\, m \in messageTypes : m \in$ STRING

├──┤

This section defines predicates for identifying *E2T* message types on the network.

$IsSubscribeRequest(m) \triangleq m.type = SubscribeRequest$

$IsSubscribeResponse(m) \triangleq m.type = SubscribeResponse$

1

$IsUnsubscribeRequest(m) \triangleq m.type = UnsubscribeRequest$

$IsUnsubscribeResponse(m) \triangleq m.type = UnsubscribeResponse$

$IsControlRequest(m) \triangleq m.type = ControlRequest$

$IsControlResponse(m) \triangleq m.type = ControlResponse$

This section defines predicates for validating $E2T$ message contents. The predicates provide precise documentation on the $E2T$ message format and are used within the spec to verify that steps adhere to the $E2T$ protocol specification.

LOCAL $ValidSubscribeRequest(m) \triangleq$ TRUE

LOCAL $ValidSubscribeResponse(m) \triangleq$ TRUE

LOCAL $ValidUnsubscribeRequest(m) \triangleq$ TRUE

LOCAL $ValidUnsubscribeResponse(m) \triangleq$ TRUE

LOCAL $ValidControlRequest(m) \triangleq$ TRUE

LOCAL $ValidControlResponse(m) \triangleq$ TRUE

This section defines operators for constructing $E2T$ messages.

LOCAL $SetType(m, t) \triangleq [m \text{ EXCEPT } !.type = t]$

$WithSubscribeRequest(m) \triangleq$
  IF $Assert(ValidSubscribeRequest(m),$ "Invalid SubscribeRequest")
   THEN $SetType(m, SubscribeRequest)$
   ELSE Nil

$WithSubscribeResponse(m) \triangleq$
  IF $Assert(ValidSubscribeResponse(m),$ "Invalid SubscribeResponse")
   THEN $SetType(m, SubscribeResponse)$
   ELSE Nil

$WithUnsubscribeRequest(m) \triangleq$
  IF $Assert(ValidUnsubscribeRequest(m),$ "Invalid UnsubscribeRequest")
   THEN $SetType(m, UnsubscribeRequest)$
   ELSE Nil

$WithUnsubscribeResponse(m) \triangleq$
  IF $Assert(ValidUnsubscribeResponse(m),$ "Invalid UnsubscribeResponse")
   THEN $SetType(m, UnsubscribeResponse)$
   ELSE Nil

$WithControlRequest(m) \triangleq$

IF $Assert(ValidControlRequest(m),$ "Invalid ControlRequest")
  THEN $SetType(m,\ ControlRequest)$
  ELSE Nil

$WithControlResponse(m) \triangleq$
  IF $Assert(ValidControlResponse(m),$ "Invalid ControlResponse")
  THEN $SetType(m,\ ControlResponse)$
  ELSE Nil

The *Messages* module is instantiated locally to avoid access from outside
the module.

LOCAL $Messages \triangleq$ INSTANCE $Messages$ WITH
  $SubscribeRequest \leftarrow$ "SubscribeRequest",
  $SubscribeResponse \leftarrow$ "SubscribeResponse",
  $UnsubscribeRequest \leftarrow$ "UnsubscribeRequest",
  $UnsubscribeResponse \leftarrow$ "UnsubscribeResponse",
  $ControlRequest \leftarrow$ "ControlRequest",
  $ControlResponse \leftarrow$ "ControlResponse"

—————————————————— MODULE *Client* ——————————————————

The *Client* module provides operators for managing and operating on $E2T$ client connections
and specifies the message types supported for the client.

—————————————————— MODULE *Send* ——————————————————

This module provides message type operators for the message types that can be send by the
$E2T$ client.

  $SubscribeRequest(c,\ m) \triangleq$
    $\wedge\ gRPC\,!\,Client\,!\,Send(c,\ Messages\,!\,WithSubscribeRequest(m))$

  $UnsubscribeRequest(c,\ m) \triangleq$
    $\wedge\ gRPC\,!\,Client\,!\,Send(c,\ Messages\,!\,WithUnsubscribeRequest(m))$

  $ControlRequest(c,\ m) \triangleq$
    $\wedge\ gRPC\,!\,Client\,!\,Send(c,\ Messages\,!\,WithControlRequest(m))$

Instantiate the $E2T\,!\,Client\,!\,Requests$ module

$Send \triangleq$ INSTANCE $Send$

—————————————————— MODULE *Receive* ——————————————————

This module provides predicates for the types of messages that can be received by an $E2T$
client.

  $SubscribeResponse(c,\ h(\_)) \triangleq$
    $gRPC\,!\,Client\,!\,Handle(c,$ LAMBDA $x,\ m:$

3

$\qquad\qquad \land Messages\,!\,IsSubscribeResponse(m)$
$\qquad\qquad \land gRPC\,!\,Client\,!\,Receive(c)$
$\qquad\qquad \land h(m))$

$\quad UnsubscribeResponse(c,\ h(\_)) \;\triangleq$
$\qquad gRPC\,!\,Client\,!\,Handle(c,\ \text{LAMBDA}\ x,\ m:$
$\qquad\qquad \land Messages\,!\,IsUnsubscribeResponse(m)$
$\qquad\qquad \land gRPC\,!\,Client\,!\,Receive(c)$
$\qquad\qquad \land h(m))$

$\quad ControlResponse(c,\ h(\_)) \;\triangleq$
$\qquad gRPC\,!\,Client\,!\,Handle(c,\ \text{LAMBDA}\ x,\ m:$
$\qquad\qquad \land Messages\,!\,IsControlResponse(m)$
$\qquad\qquad \land gRPC\,!\,Client\,!\,Receive(c)$
$\qquad\qquad \land h(m))$

Instantiate the $E2T\,!\,Client\,!\,Responses$ module
$Handle \;\triangleq\; \text{INSTANCE}\ Receive$

$Connect(s,\ d) \;\triangleq\; gRPC\,!\,Client\,!\,Connect(s,\ d)$

$Disconnect(c) \;\triangleq\; gRPC\,!\,Client\,!\,Disconnect(c)$

Provides operators for the $E2T$ client
$Client \;\triangleq\; \text{INSTANCE}\ Client$

───── MODULE $Server$ ─────

The $Server$ module provides operators for managing and operating on $E2T$ servers and specifies the message types supported for the server.

───── MODULE $Send$ ─────

This module provides message type operators for the message types that can be send by the $E2T$ server.

$\quad SubscribeResponse(c,\ m) \;\triangleq$
$\qquad \land gRPC\,!\,Server\,!\,Send(c,\ Messages\,!\,WithSubscribeResponse(m))$

$\quad UnsubscribeResponse(c,\ m) \;\triangleq$
$\qquad \land gRPC\,!\,Server\,!\,Send(c,\ Messages\,!\,WithUnsubscribeResponse(m))$

$\quad ControlResponse(c,\ m) \;\triangleq$
$\qquad \land gRPC\,!\,Server\,!\,Send(c,\ Messages\,!\,WithControlResponse(m))$

Instantiate the $E2T\,!\,Server\,!\,Responses$ module

4

$Send \;\triangleq\; \text{INSTANCE } Send$

─────────────────── MODULE $Reply$ ───────────────────

This module provides message type operators for the message types that can be send by the $E2T$ server.

$SubscribeResponse(c,\, m) \;\triangleq\;$
    $\wedge\, gRPC\,!\,Server\,!\,Reply(c,\, Messages\,!\,WithSubscribeResponse(m))$

$UnsubscribeResponse(c,\, m) \;\triangleq\;$
    $\wedge\, gRPC\,!\,Server\,!\,Reply(c,\, Messages\,!\,WithUnsubscribeResponse(m))$

$ControlResponse(c,\, m) \;\triangleq\;$
    $\wedge\, gRPC\,!\,Server\,!\,Reply(c,\, Messages\,!\,WithControlResponse(m))$

───

Instantiate the $E2T\,!\,Server\,!\,Reply$ module
$Reply \;\triangleq\; \text{INSTANCE } Reply$

─────────────────── MODULE $Receive$ ───────────────────

This module provides predicates for the types of messages that can be received by an $E2T$ server.

$SubscribeRequest(c,\, h(\_)) \;\triangleq\;$
    $gRPC\,!\,Server\,!\,Handle(c,\, \text{LAMBDA } x,\, m :$
        $\wedge\, Messages\,!\,IsSubscribeRequest(m)$
        $\wedge\, gRPC\,!\,Server\,!\,Receive(c)$
        $\wedge\, h(m))$

$UnsubscribeRequest(c,\, h(\_)) \;\triangleq\;$
    $gRPC\,!\,Server\,!\,Handle(c,\, \text{LAMBDA } x,\, m :$
        $\wedge\, Messages\,!\,IsUnsubscribeRequest(m)$
        $\wedge\, gRPC\,!\,Server\,!\,Receive(c)$
        $\wedge\, h(m))$

$ControlRequest(c,\, h(\_)) \;\triangleq\;$
    $gRPC\,!\,Server\,!\,Handle(c,\, \text{LAMBDA } x,\, m :$
        $\wedge\, Messages\,!\,IsControlRequest(m)$
        $\wedge\, gRPC\,!\,Server\,!\,Receive(c)$
        $\wedge\, h(m))$

───

Instantiate the $E2T\,!\,Server\,!\,Requests$ module
$Handle \;\triangleq\; \text{INSTANCE } Receive$

───

Provides operators for the $E2T$ server

$Server \triangleq \text{INSTANCE } Server$

The set of all open $E2T$ connections

$Connections \triangleq gRPC!Connections$

$Init \triangleq$
    $\wedge gRPC!Init$

$Next \triangleq$
    $\wedge gRPC!Next$

* Modification History
* Last modified $Mon\ Sep$ 13 19:20:57 PDT 2021 by $jordanhalterman$
* $Created\ Mon\ Sep$ 13 16:23:16 PDT 2021 by $jordanhalterman$