Internet Draft
Document: draft-khosravi-forces-tcptml-01.txt
Expires: August 2005
Working Group: ForCES

Hormuzd Khosravi
Shuchi Chawla
Intel Corp.
Furquan Ansari
Lucent Tech.
Jon Maloy
Ericsson

February 2005

TCP/IP based TML (Transport Mapping Layer) for ForCES protocol

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [2].

Abstract

This document defines the TCP/IP based TML (Transport Mapping Layer) for the ForCES protocol. It explains the rationale for choosing the transport protocols and also describes how this TML addresses all

the requirements described in the Forces [3] requirements and ForCES protocol [7] document.

Table of Contents

1. Definitions

The following definitions are taken from [3], [5]

ForCES Protocol - While there may be multiple protocols used within the overall ForCES architecture, the term "ForCES protocol" refers only to the protocol used at the Fp reference point in the ForCES Framework in RFC3746 [RFC3746]. This protocol does not apply to CE-to-CE communication, FE-to-FE communication, or to communication between FE and CE managers. Basically, the ForCES protocol works in a master-slave mode in which FEs are slaves and CEs are masters.

ForCES Protocol Layer (ForCES PL) -- A layer in ForCES protocol architecture that defines the ForCES protocol messages, the protocol state transfer scheme, as well as the ForCES protocol architecture itself (including requirements of ForCES TML (see below)). Specifications of ForCES PL are defined by this document.

ForCES Protocol Transport Mapping Layer (ForCES TML) -- A layer in ForCES protocol architecture that specifically addresses the protocol message transportation issues, such as how the protocol messages are mapped to different transport media (like TCP, IP, ATM, Ethernet, etc), and how to achieve and implement reliability, multicast, ordering, etc. This document defines a TCP/IP based ForCES TML.

2. Introduction

The ForCES (Forwarding and Control Element Separation) working group in the IETF is defining the architecture and protocol for separation of control and forwarding elements in network elements such as routers. [3], [4] define both architectural and protocol requirements for the communication between CE and FE. The ForCES protocol layer [7] describes the protocol specification. It is envisioned that the ForCES protocol would be independent of the interconnect technology between the CE and FE and can run over

multiple transport technologies and protocol. Thus a Transport Mapping Layer (TML) has been defined in the protocol framework that will take care of mapping the protocol messages to specific transports. This document defines the TCP/IP based TML for the ForCES protocol layer. It also addresses all the requirements for the TML including security, reliability, etc.

3. Protocol Framework Overview

The reader is referred to the Framework document [4], and in particular sections 3 and 4, for architectural overview and where and how the ForCES protocol fits in. There may be some content overlap between the ForCES protocol draft [7] and this section in order to provide clarity.

The ForCES protocol constitutes two pieces: the PL and TML layer. This is depicted in Figure 1 below.

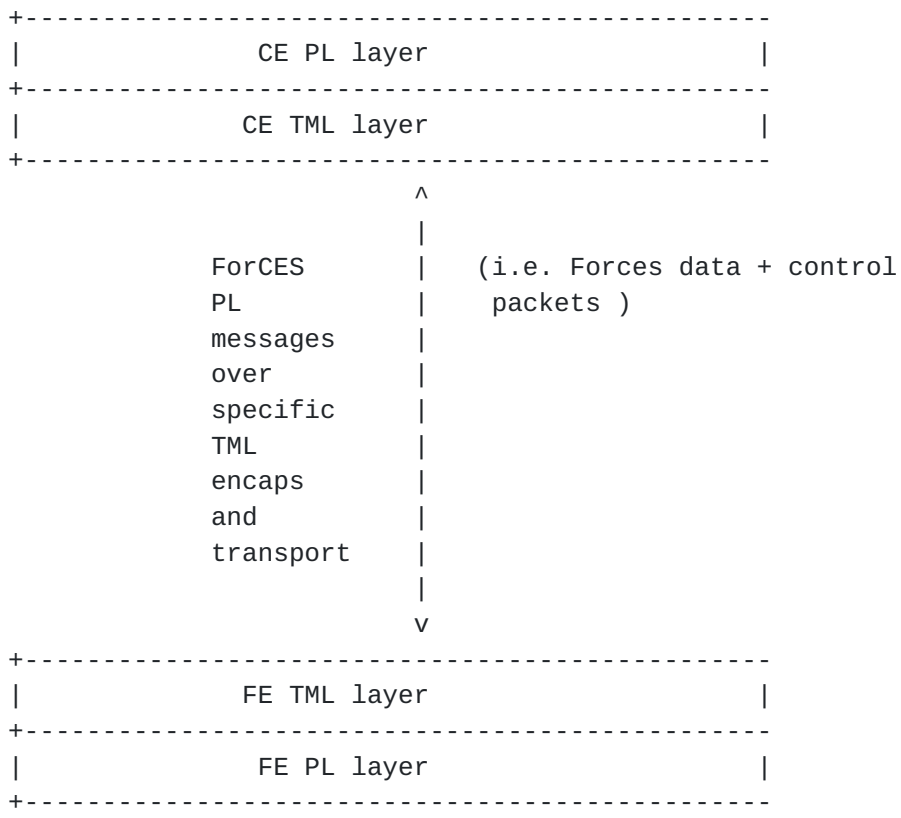


Figure 1: ForCES Interface

The PL layer is in fact the ForCES protocol. Its semantics and message layout are defined in [7]. The TML Layer is necessary to connect two ForCES PL layers as shown in Figure 1 above.

Both the PL and TML layers are standardized by the IETF. While only one PL layer is defined, different TMLs are expected to be standardized. To interoperate the TML layer at the CE and FE are expected to be of the same definition.

On transmit, the PL layer delivers its messages to the TML layer. The TML layer delivers the message to the destination TML layer(s). On reception, the TML delivers the message to its destination PL layer(s).

3.1.1.The PL layer

The PL is common to all implementations of ForCES and is standardized by the IETF [7]. The PL layer is responsible for associating an FE or CE to an NE. It is also responsible for tearing down such associations. An FE uses the PL layer to throw various subscribed-to events to the CE PL layer as well as respond to various status requests issued from the CE PL. The CE configures both the FE and associated LFBs attributes using the PL layer. In addition the CE may send various requests to the FE to activate or deactivate it, reconfigure itÆs HA parameterization, subscribe to specific events etc.

3.1.2.The TML layer

The TML layer is essentially responsible for transport of the PL layer messages. The TML is where the issues of how to achieve transport level reliability, congestion control, multicast, ordering, etc. are handled. It is expected more than one TML will be standardized. The different TMLs each could implement things differently based on capabilities of underlying media and transport. However, since each TML is standardized, interoperability is guaranteed as long as both endpoints support the same TML. All ForCES Protocol Layer implementations should be portable across all TMLs, because all TMLs have the same top edge semantics.

4. TCP/IP TML Overview

The TCP/IP TML consists of two TCP connections between the CE and FE over which the protocol messages are exchanged. One of the connections is called the control channel, over which control messages are exchanged, the other is called data channel over which external protocol packets, such as routing packets will be exchanged. The TCP connections will use unique server port numbers for each of the channels. In addition to this, this TML will provide mechanisms to prioritize the messages over the different channels.

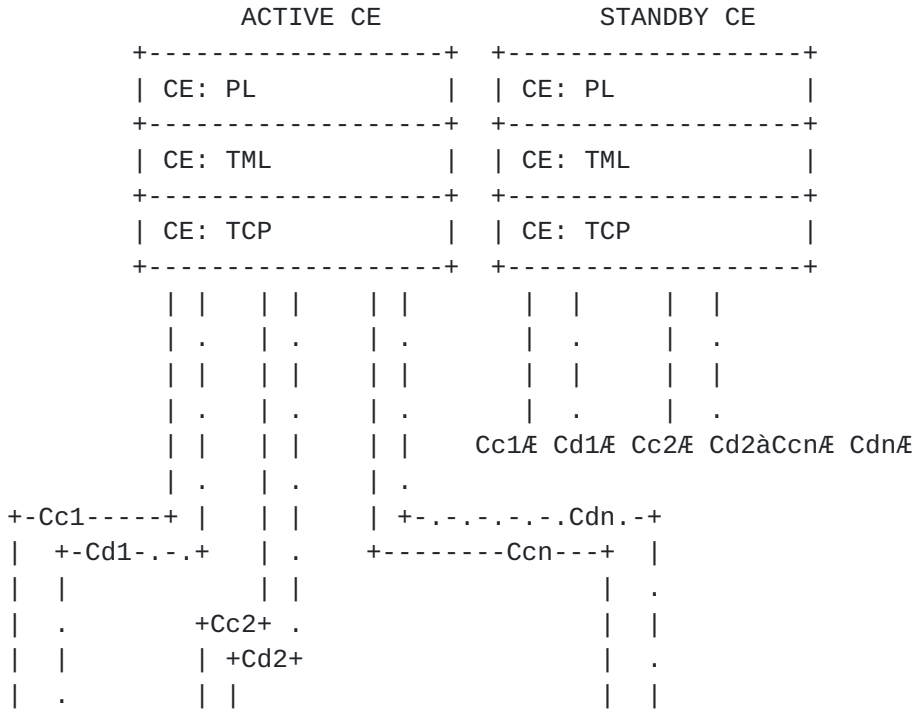
Some of the rationale for choosing this transport mechanism as well as explanation of how it meets the TML requirements is explained below.

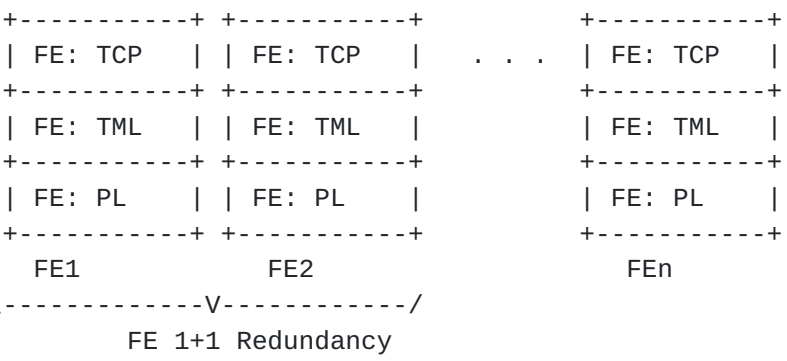
4.1.Rationale for using TCP/IP

TCP meets all the reliability requirements (no losses, no data corruption, no re-ordering of data) for the ForCES protocol/TML and also provides congestion control mechanism, which is important to meet the scalability requirement. In addition, it helps with interoperability since TCP is a well-understood, widely deployed transport protocol. Using TCP also enables this TML and the protocol to work seamlessly in single hop and multihop scenarios.

4.2.Separate Control and Data channels

The ForCES NEs are subject to Denial of Service (DoS) attacks [Requirements Section 7 #15]. A malicious system in the network can flood a ForCES NE with bogus control packets such as spurious RIP or OSPF packets in an attempt to disrupt the operation of and the communication between the CEs and FEs. In order to protect against this situation, the TML uses separate control and data channels for communication between the CEs and FEs. Figure 2 below illustrates the different communication channels between the CEs and the FEs; the communication channels for support of High Availability with redundant CEs are also included.





Legend:

---- Cc# : Unicast Control Channel between Active CE and FE#

-.-. Cd# : Unicast Data Channel between Active CE and FE#

---- Cc#Æ: Unicast Control Channel between Standby CE and FE#

-.-. Cd#Æ: Unicast Data Channel between Standby CE and FE#

Figure 2: CE-FE Communication Channels

The data channel carries the control protocol packets such as RIP, OSPF messages as outlined in Requirements [3] Section 7 #10, which are carried in ForCES Packet Redirect messages [7], between the CEs and FEs. All the other ForCES messages, which are used for configuration/capability exchanges, event notification, etc, are carried over the control channel. The data channel is set up only after the control channel is set up. By default, the data channel is established on the CE control channel port number +1.

The reliability requirements for the data channel messages are different from that of the control messages [Reqs] i.e. they donÆt require strict reliability in terms of retransmission, etc. However congestion control is important for the data channel because in case of DoS attacks, if an unreliable transport such as UDP is used for the data traffic, it can more easily overflow the physical connection, overwhelming the control traffic with congestion. Thus we need a transport protocol that provides congestion control but does not necessarily provide full reliability. Datagram Congestion Control Protocol (DCCP) [11], which is currently being defined, is a transport protocol that exactly meets this requirement. However since it is currently not an IETF standard RFC, and the authors are unaware of any existing implementations, this TML uses TCP as transport protocol for the data channel (for IP interconnect). TCP provides the congestion control mechanism required for the data channel and its wide deployment eases interoperability.

4.3.Reliability

TCP provides the reliability (no losses, no data corruption, no re-ordering of data) required for ForCES protocol control messages.

4.4.Congestion Control

TCP provides congestion control needed to satisfy this requirement.

4.5.Security

This TML uses TLS [8] to provide security in insecure environments. Please see section 7 on security considerations for more details.

4.6.Addressing

This TML uses addressing provided by IP layer. For unicast addressing/delivery, it uses the TCP connection between the CE and FE. For multicast/broadcast addressing/delivery, this TML uses multiple TCP connections between the CE and FEs.

4.7.Prioritization

This TML provides prioritization of messages sent over control channel as compared to the data channel. This has also been found to be useful in face of DoS attacks on the protocol. The details of this are TBD.

4.8.HA Decisions

The TML layer supports heartbeat messages between peer TML layers to indicate liveness of the entity generating it. The frequency of the heartbeat message may be specified at protocol initialization time.

4.9.Encapsulations Used

TML adds its own header on all ForCES protocol messages that it receives, and additionally on messages it generates. The ForCES protocol and TML messages are further encapsulated with a TCP/IP header. The format of the TML header is specified in Section 5.

5. TML Messaging

TML adds on a TML header to all messages that it either receives from the PL layer or those messages that it generates. The TML header is followed by the message body. The message body may comprise a PL message or it may be a message generated by the TML layer for communicating with its peer. A flag in the TML header

specifies whether the message is associated with the PL layer. The next section details the TML header.

5.1.TML Message Header

Figure 3 below shows the format of the TCP TML message header. NOTE: The header may undergo some modifications in the next revision of this draft.

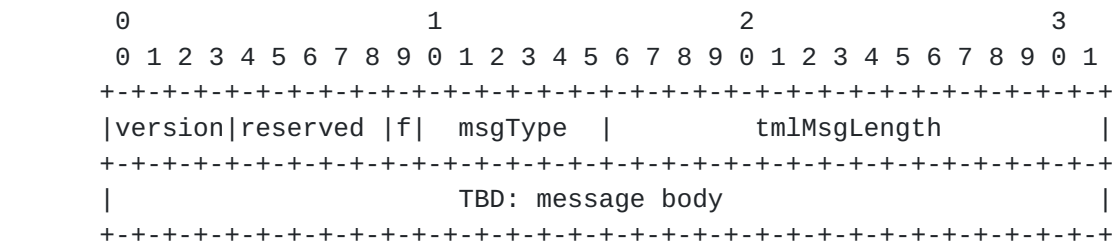


Figure 3: TCP TML Message Header Format

5.1.1.Version (version)

The version field (4 bits) specifies the version of the TML layer protocol supported by the implementation. An entity implementing the TML layer protocol should be backward compatible with previous versions of the protocol.

5.1.2.Upper Layer Protocol Flag (f)

The upper layer protocol flag (1 bit) specifies if the TML layer is carrying a message sent by the PL Layer. If the protocol flag is set, the message body comprises of a PL message which is not interpreted by the peer TML layer; the rest of the TML header and message is ignored by the peer TML layer. If the protocol flag is not set, the message body carries a TML message; hence, the rest of the TML header needs to be read by the peer TML layer and the message body appropriately processed.

5.1.3.Message Type (msgType)

The message type (6 bits) field is applicable only if the protocol flag field is not set. It specifies the TML message being sent between peer TML layers.

5.1.4.TML Message Length (tmlMsgLength)

The TML message length field is applicable only if the protocol flag field is not set. It specifies the length of the TML message in octets inclusive of the TML header.

5.1.5.TML Message Body

Format of this is TBD.

5.2.TML Messages

5.2.1.Channel Close

The channel close message is generated by the TML layer in response to a request to close a specified channel. This message is sent to the peer TML layer to notify it of the closure. This provides information to the peer TML layer so it can either close its end of the channel also or drop any messages received by the upper layer protocol.

ForCES protocol usage: This message may be sent from either the CE or the FE TML layer depending on which entity initiated the close.

5.2.2.Heartbeat

A heartbeat message is exchanged between peer TML layers to communicate liveness of the entity generating the message.

ForCES protocol usage: This message may be sent from either the CE to an FE or vice-versa.

5.2.3.Multicast Group Join Request

The multicast group join request message is triggered by a request from the PL layer to join a specific multicast group. The peer TML layer on receiving this request creates the specified multicast group if it didnÆt previously exist. It updates the membership of the group to include the entity requesting the join.

ForCES protocol usage: Since only FEs may be the leaves in a multicast group, this message is sent by the FE TML to the CE TML, due to a join request from the FE PL.

5.2.4.Multicast Group Join Response

The multicast group join response message is generated by the TML Layer in response to a join request message.

ForCES protocol usage: If only FEs may be the leaves in a multicast group, this message is generated by the CE TML and sent to the FE TML, in response to the join request message received from the FE TML.

5.2.5.Multicast Group Leave Request

The multicast group leave request message is triggered by a request from the PL layer to leave a specific multicast group. The peer TML layer on receiving this request removes the specified multicast group if the entity requesting to leave is the only member of the group. Else, it updates the membership of the group to exclude the entity requesting the leave.

ForCES protocol usage: If only FEs may be the leaves in a multicast group, this message is sent by the FE TML to the CE TML, due to a leave request from the FE PL.

5.2.6.Multicast Group Leave Response

The multicast group leave response message is generated by the TML Layer in response to a leave request message.

ForCES protocol usage: If only FEs may be the leaves in a multicast group, this message is generated by the CE TML and sent to the FE TML, in response to the leave request message received from the FE TML.

6. TML Interface to Upper layer Protocol

ForCES TML interfaces with an upper layer protocol, the PL Layer and a lower layer protocol, TCP (in the case of TCP TML). This section defines the interface to the upper layer protocol. This interface should be used only as a guideline in implementing the API. Additionally, although the current interface is defined mainly as a synchronous interface, the interface may be implemented to be asynchronous if desired.

6.1.TML API

6.1.1.TML Initialize

```
status tmlInit(
  in  channelType,
  in  initAttributes)
```

Input Parameters:
  channelType: control versus data channel
  initAttributes: initialization parameters

Output Parameters:
  none

```
   Returns:
     status: SUCCESS
             Errors TBD

   Synopsis:
   tmlInit() enables establishment of communication channels on the
   entity that this API is invoked.  Optionally specifies attributes if
   any, for initialization. This call does not however result in the
   setup of any channels.

   ForCES Usage Model:
   In the case of ForCES which follows a client-server model, this API
   would be invoked on the CE, which functions as the server. It is
   invoked once for every class of TML channels on a per channel type
   basis (control channel versus data channel).  For example, say for
   control messaging, the CE communicates with five FEs using TCP TML
   and with another two FEs, using UDP TML.  tmlInit() will need to be
   invoked twice, once for the TCP TML attributes and once for the UDP
   TML attributes for the control channel setup with all of the FEs.
   The same holds true for the data channel setup in the above case.
```

6.1.2.TML Channel Open

```
   status tmlOpen(
     in  channelType,
     in  elementId,
     in  channelMode,
     in  channelAttributes,
     in  eventHandlerCallBack,
     out channelDescriptor)

   Input Parameters:
     channelType: control channel or data channel
     elementId: Specific CE for which channel needs to be setup
     channelMode: unicast versus multicast
     channelAttributes: channel establishment parameters
     eventHandlerCallback: Callback function to be invoked on event
   generation

   Output Parameters:
     channelDescriptor: handle to communication channel

   Returns:
     status: SUCCESS
             Errors TBD
```

Synopsis:
tmlOpen() results in a communication channel of type channelType (control versus data), being established with the specified elementId. The channel may be specified as unicast or multicast via channelMode. This call may either trigger the establishment of the channel, or if the channel is already established, it only results in a registration for that channel. In either case, if successful, a handle to the open channel is returned via a channelDescriptor. If this call triggers the establishment of the channel, the channel is established using the channelAttributes parameter specified to the call. Once the channel is setup (or if already setup prior to this call), the caller of this API is also capable of receiving TML events via the specified event handling callback function. If this call is invoked multiple times on a channel that has already been opened and registered, a return status of ALREADY_REGISTERED is returned, with no change to registration.

ForCES Usage Model:
In the case of ForCES which follows a client-server model, this API would be invoked on the FE by FE PL, which functions as the client. On each FE, it is invoked once per channel that the FE wishes to setup with the CE.

Notes:
In the case of TCP TML, since there is no inherent support for multicast, regardless of the channelMode specified, the specified channel would be setup as a unicast channel; however, the unicast channel would be able to support pseudo multicast. Hence, there is no need to set up a distinct channel for unicast and a distinct channel for multicast in the case of TCP TML (as may be the case for UDP TML).

6.1.3. TML Channel Close

```
status tmlClose(
  in  channelDescriptor
  in  mode)
```

Input Parameters:
channelDescriptor: handle to communication channel to be terminated
mode: mode of operation for the close û forced versus controlled

Output Parameters:
none

Returns:
  status: SUCCESS
          Errors TBD

Synopsis:
Tears down/terminates specified communication channel. No further CE PL û FE PL messaging is possible after this. If the mode is specified as controlled, current messages that are pending in the TML layer shall be sent, but no new messages shall be accepted by the TML layer on this channel. In the forced model, messages pending in the TML layer shall be discarded. Since the channel was terminated, a subsequent tmlOpen() will trigger establishment of the channel.

ForCES Usage Model:
This API may be invoked by either the CE or the FE. If the FE PL invokes it, the FE TML sends a message to the CE TML informing it that the channel has been shutdown, which results in an upcall to the CE PL. This will result in the CE PL also shutting down its end of the channel.

6.1.4.TML Channel Write

```
status tmlWrite(
  in  channelDescriptor,
  in  msg,
  in  msgSize,
  in  timeout,
  out bytesWritten)
```

Input Parameters:
  channelDescriptor: handle to communication channel to be written to
  msg: message to be sent
  msgSize: size of message to be sent
  timeout: specifies blocking or non-blocking write. Value of -1 implies blocking write (wait forever), value of 0 implies non-blocking write

Output Parameters:
  bytesWritten: number of bytes actually transmitted

Returns:
  status: SUCCESS
          Errors TBD

Synopsis:

Sends message over the specified channel. If the specified channelDescriptor is associated with a multicast group, the message will be sent to all members of the group. This does not imply that the message has actually been transmitted. The message is queued in the appropriate queue for transmission. Once this call returns, the message buffer may be freed. If TMLÆs message queues are full, the timeout will be used to determine how long to wait prior to returning; if the specified timeout expires, and no message buffer becomes available, the API returns with an error.

ForCES Usage Model:
This API may be invoked by either the FE PL or the CE PL.

6.1.5.TML Channel Read

```
status tmlRead(
  in  channelDescriptor,
  in  msgBuf,
  in  timeout,
  out bytesRead)
```

Input Parameters:
  channelDescriptor: handle to communication channel to be read from
  msgBuf: buffer into which message is to be read
  timeout: specifies blocking or non-blocking read. Value of -1 implies blocking read (wait forever), value of 0 implies non-blocking read

Output Parameters:
  bytesRead: number of bytes actually read

Returns:
  status: SUCCESS
          Errors TBD

Synopsis:
Reads message on the specified channel. Once the message is copied into msgBuf specified by the call, the TML message buffer may be freed. If TMLÆs message queues are empty (no message is available), the timeout will be used to determine how long to wait prior to returning; if the specified timeout expires, and no message becomes available, the API returns with an error.
If a non-blocking read is executed, the caller of the API is notified via an upcall when a message becomes available.
TBD: If the channelDescriptor specified is associated with a multicast group, it should be considered invalid since multicast is

only supported on a write; a read is always associated with reading from a single channel.

6.1.6.TML Multicast Group Join

```
status tmlMulticastGroupJoin(
  in  groupAttributes)
```

Input Parameters:
groupAttributes: Attributes associated with the multicast group to be joined

Output Parameters:
none

Returns:
status: SUCCESS
Errors TBD

Synopsis:
Joins the specified multicast group as leaf node in the group. If this is the first join request being received for this group, it results in the creation of the group and the allocation of a groupDescriptor (which is equivalent to a channelDescriptor). Once a member of this group, the entity calling this API will be capable of receiving messages addressed to this multicast group.

ForCES Usage Model:
If the intent is that only FEs can be members of a multicast group and only a CE can be the source of a multicast, this API would be invoked on the FE that wishes to join a multicast group.

6.1.7.TML Multicast Group Leave

```
status tmlMulticastGroupLeave(
  in  groupAttributes)
```

Input Parameters:
groupAttributes: Attributes associated with the multicast group to leave

Output Parameters:
none

Returns:
status: SUCCESS
Errors TBD

Synopsis:
Leaves the specified multicast group it had previously joined. Once an entity is not a member of the multicast group, it is no longer capable of receiving messages addressed to group. If this leave request is associated with the only member of the group, the multicast group is removed, and its associated groupDescriptor invalidated.

ForCES Usage Model:
If the intention is that only FEs can be members of a multicast group and only a CE can be the source of a multicast, this API would be invoked on the FE that wishes to leave a group it had previously joined.

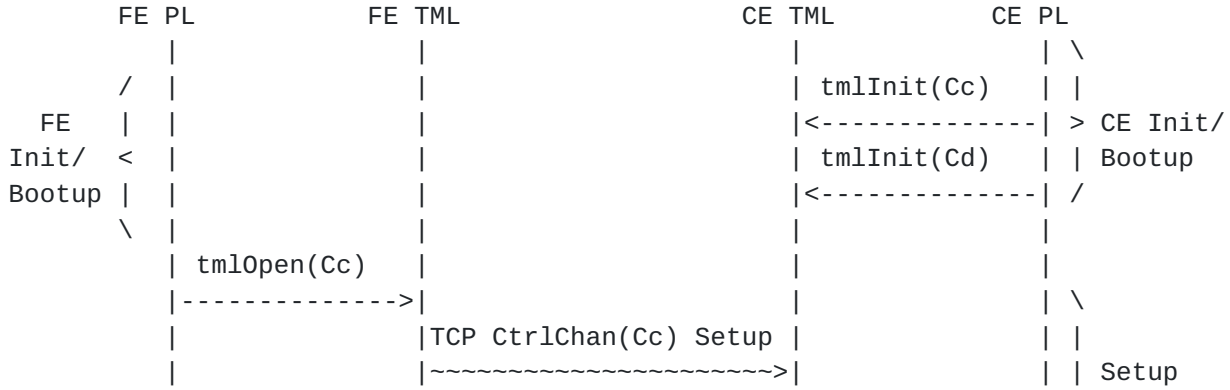
6.2.Protocol Initialization and Shutdown Model

In order for the peer PL Layers to communicate, the control and data channels must be setup. This section defines a model for the setup of the channels, using the TML interface defined above. In this model, the peer TML Layers may establish the control and data channels between the FE and the CE without the involvement of the PL Layers, or if desired, the PL Layer may trigger the setup of the channels; this is left as an implementation decision. Both modes may also be supported within an implementation.

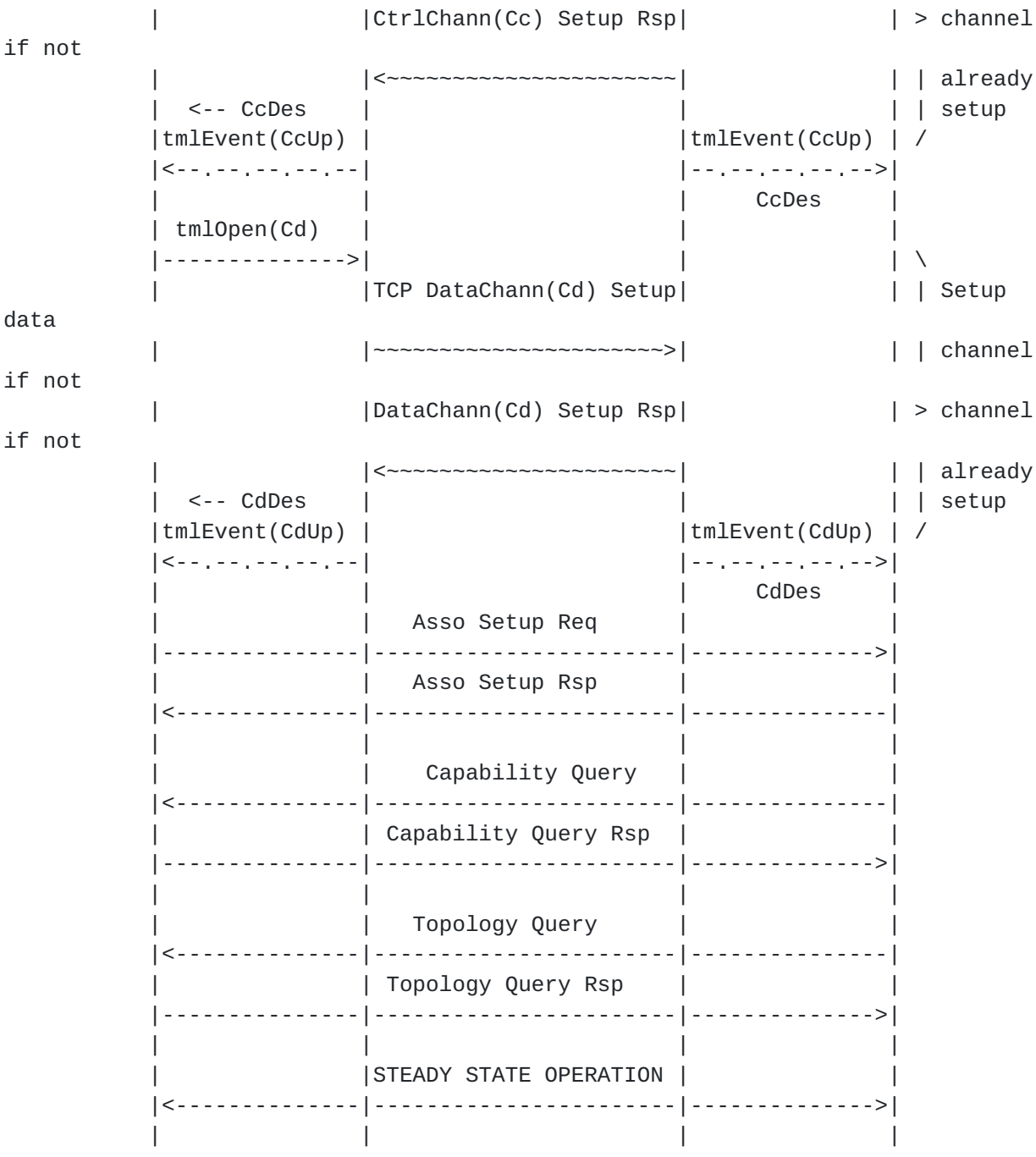
6.2.1.Protocol Initialization

The control channel must be established between the FE TML and the CE TML for establishment of association to proceed. This channel will be used for messages related to the association setup and capability query. The data channel must be established no later than the response from the FE to the CE Topology query message.

Figure 4 illustrates the initialization model where the PL layer via an interface provided by the TML Layer, triggers the setup of the control and data channels.



control



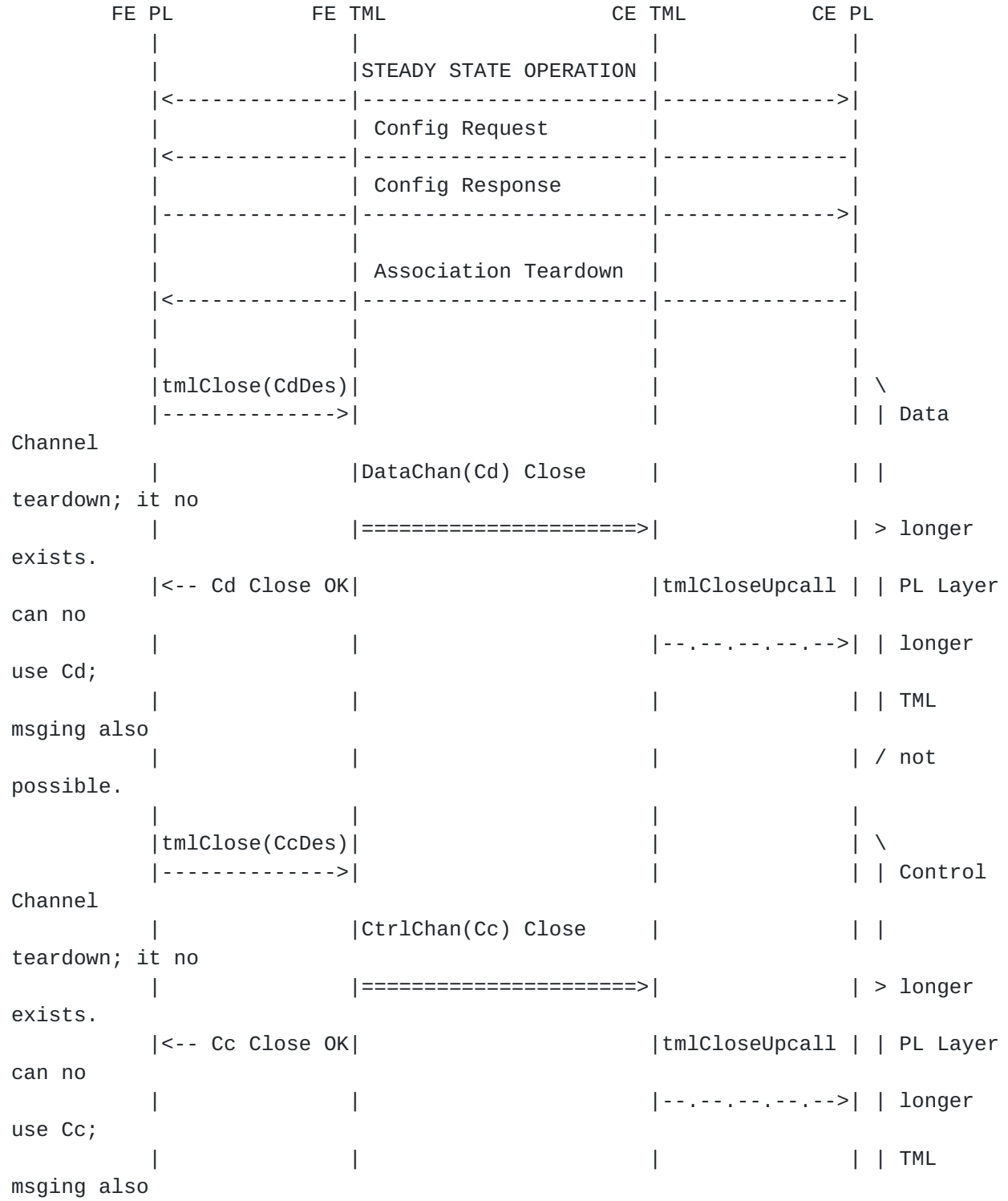
Legend:
 PL --------> PL : Protocol layer messaging
 PL --------> TML: TML API
 TML ========> TML: Messaging between TML Layers
 TML --.--.--> PL : Events/Notifications/Upcalls
 TML ~~~~~~~~> TML: Internal protocol communication

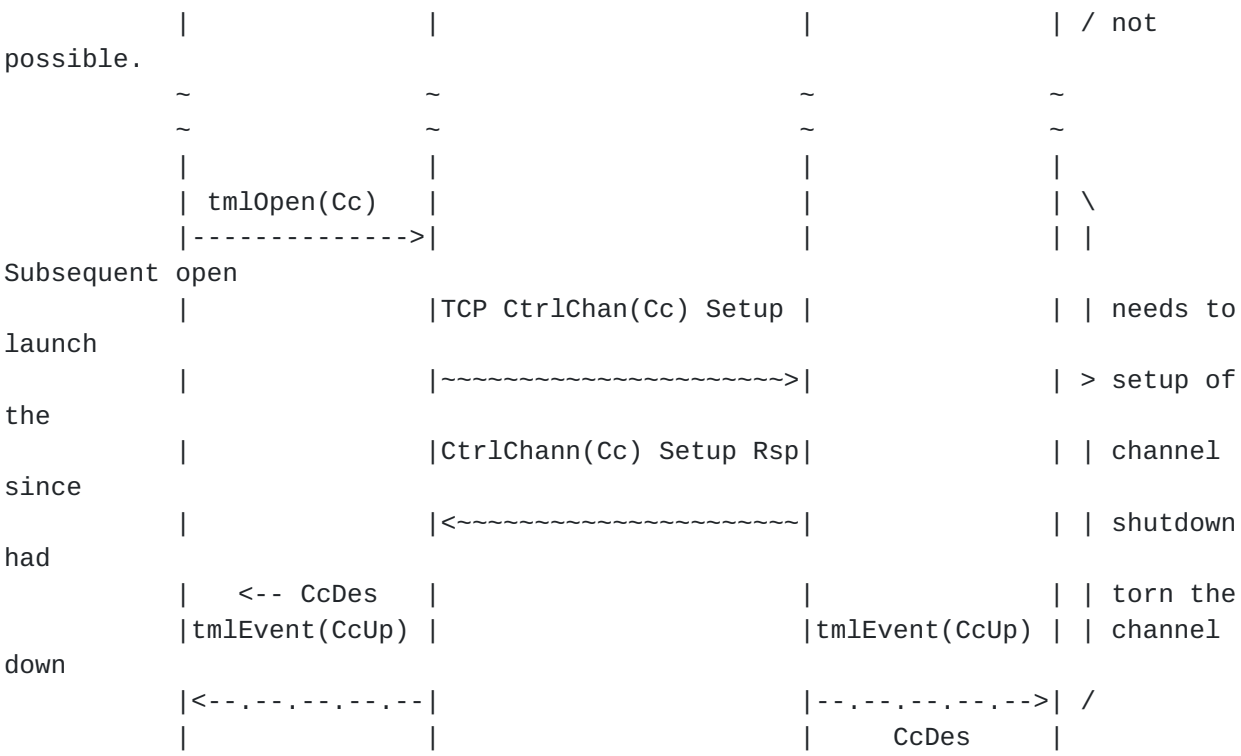
Figure 4: Protocol Initialization (Channel Setup)

6.2.2.Protocol Shutdown

The control channel teardown must occur only after the association teardown has occurred. The data channel teardown may occur no earlier than the association teardown.

The PL Layer may completely shutdown a channel via invocation of the tmlClose() API. When the PL layer shuts down a channel, the channel is torn down; hence ForCES messaging between the CE and FE is no longer possible over that channel. A subsequent tmlOpen() triggers establishment of the channel. This scenario is illustrated in Figure 5.




| | | | | / not
possible.
~ ~ ~ ~
~ ~ ~ ~
| | | | |
| tmlOpen(Cc) | | | | \
|--------------->| | | | |
Subsequent open
| |TCP CtrlChan(Cc) Setup | | | needs to
launch
| |~~~~~~~~~~~~~~~~~~~~~~>| | > setup of
the
| |CtrlChann(Cc) Setup Rsp| | | channel
since
| |<~~~~~~~~~~~~~~~~~~~~~~| | | shutdown
had
| <-- CcDes | | | | torn the
|tmlEvent(CcUp) | |tmlEvent(CcUp) | | channel
down
|<--.--.--.--.--| |--.--.--.--.-->| /
| | | CcDes |

Legend:
 PL --------> PL : Protocol layer messaging
 PL --------> TML: TML API
 TML ========> TML: Messaging between TML Layers
 TML --.--.--> PL : Events/Notifications/Upcalls
 TML ~~~~~~~~> TML: Internal protocol communication

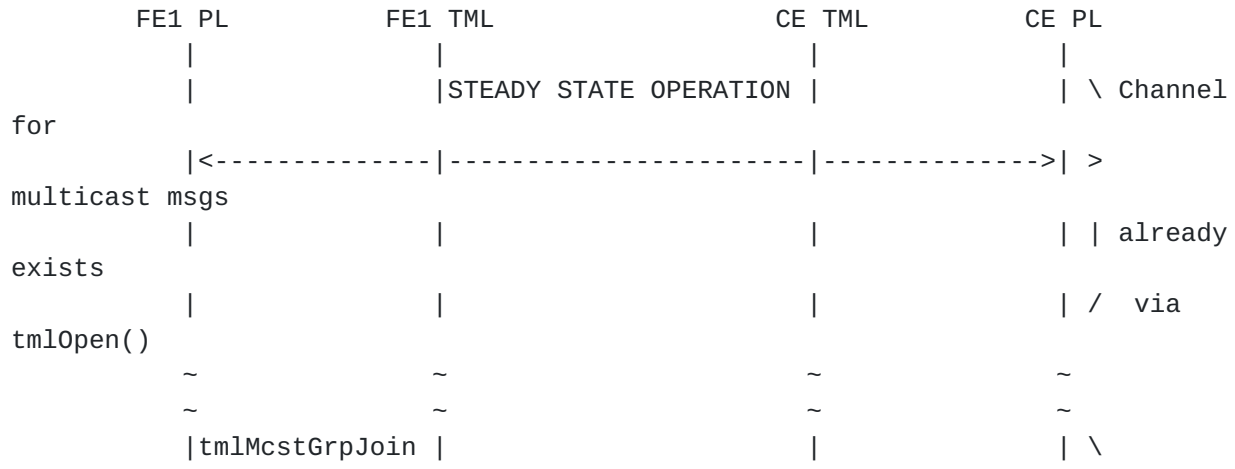
Figure 5: Protocol Shutdown

6.3.Multicast Model

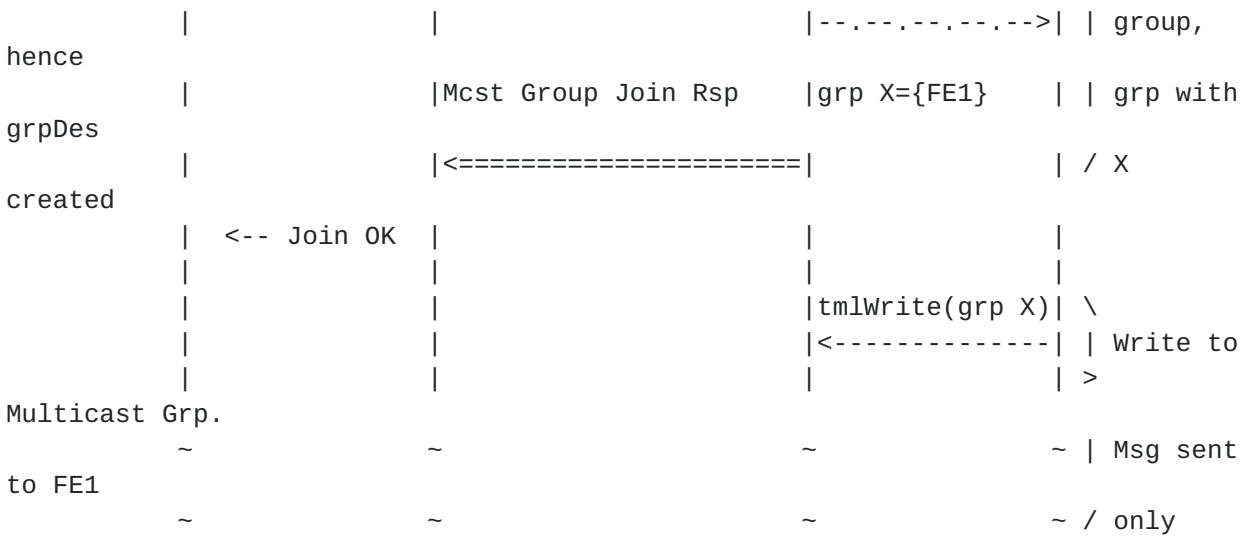
The TML layer provides support for multicast. In the ForCES model, support is required to multicast to the FEs from a CE; in this case, the CE is the source or root of the multicast and the FEs are the leaves.

Support for multicast requires that a channel for supporting multicast be opened between an FE and the CE. In the case of TCP TML, the same channel is used for both unicast and multicast messaging since multicast mode is simulated using unicast channels in this case. Once the channel is open, FEs may join and leave specified multicast groups. The first ômulticast group joinö request from an FE to a CE for a specific multicast group results in the creation of the group and an allocation of a group descriptor (which is similar to a channel descriptor) for the group. The ômulticast group leaveö request from an FE to the CE on a multicast group with just that one member results in the group being removed and the descriptor deallocated. A tmlWrite() on a unicast channel descriptor results in a unicast message being sent to the FE associated with the channel. A tmlWrite() on a group descriptor results in multicast messaging. Figure 6 illustrates a multicast scenario with 2 FEs, FE1 and FE2. In the first case, FE1 joins a multicast group. In the second case, FE2 joins the same multicast group, and leaves the group some time later.

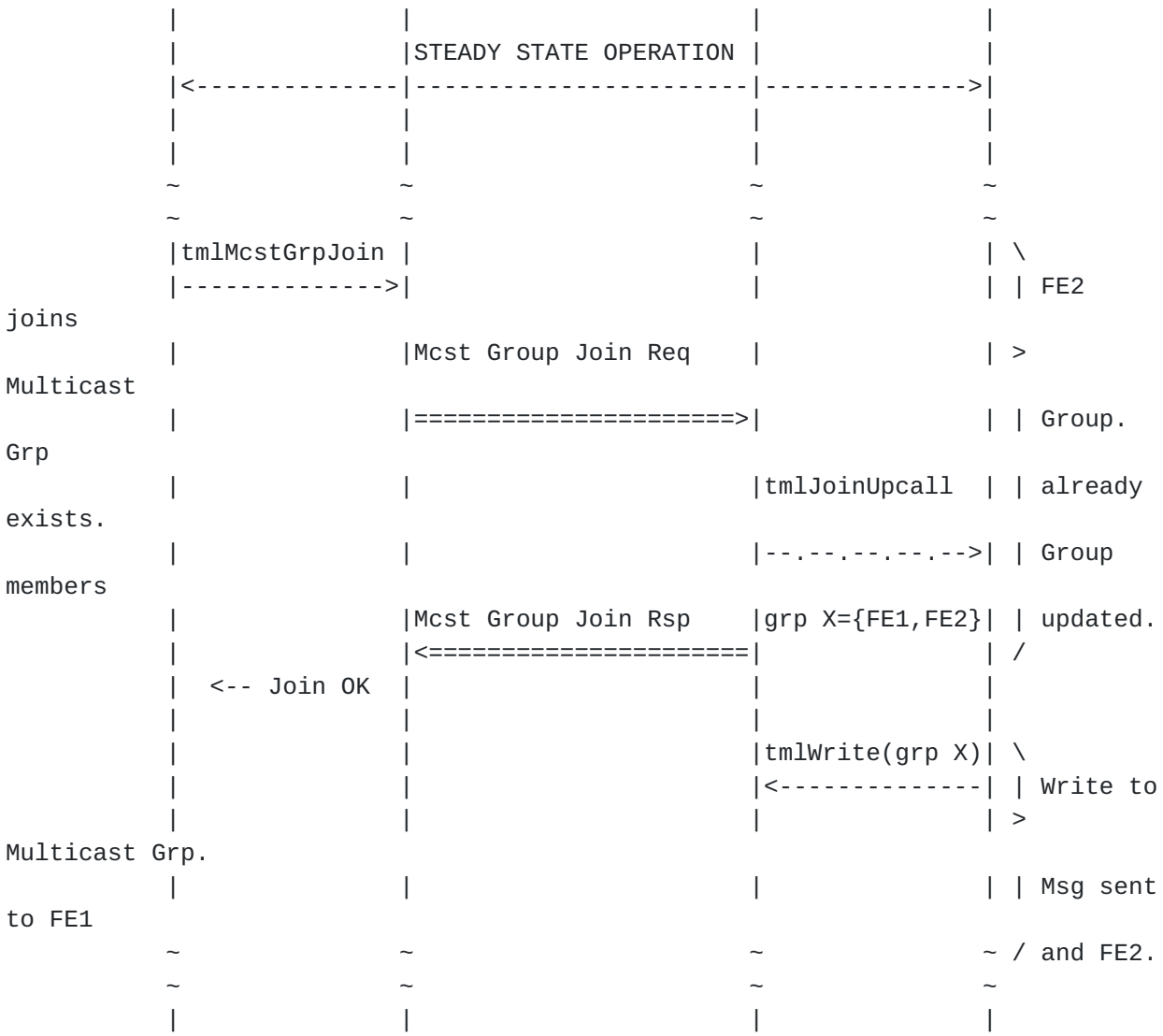
Multicast Scenario with FE1:

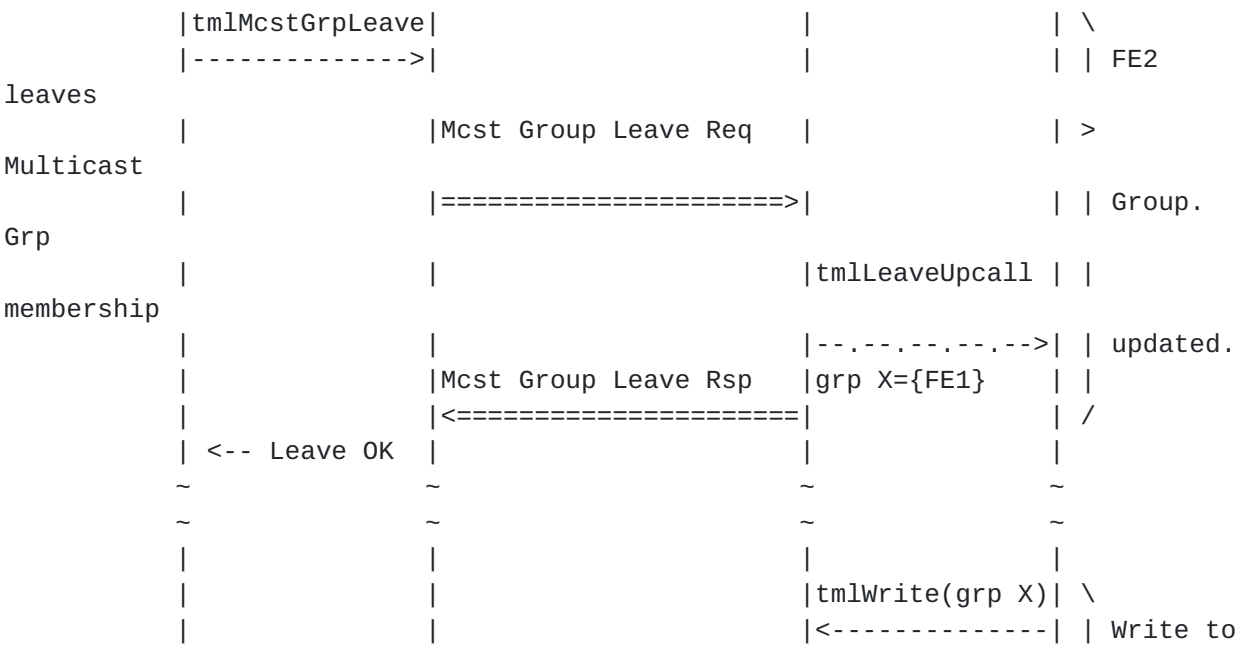



|-------------->|
FE1 joins
|
|Mcst Group Join Req
|
Multicast
|
|=======================>|
> Group.
|
|
|tmlJoinUpcall
|
request for



Multicast Scenario with FE2:




|tmlMcstGrpLeave|
|-------------->|
| \
| FE2 leaves
|
|Mcst Group Leave Req
|
| >
Multicast
|
|=======================>|
| | Group.
Grp
|
|tmlLeaveUpcall | |
membership
|--.--.--.--.-->| | updated.
|Mcst Group Leave Rsp
|grp X={FE1}
| |
|<=======================|
| /
| <-- Leave OK
|
|
~
~
~
~
|tmlWrite(grp X)| \
|<--------------| | Write to

Legend:
PL --------> PL : Protocol layer messaging
PL --------> TML: TML API
TML ========> TML: Messaging between TML Layers
TML --.--.--> PL : Events/Notifications/Upcalls
TML ~~~~~~~~> TML: Internal protocol communication

Figure 6: Support for Multicast

7. Security Considerations

If the CE or FE are in a single box and network operator is running under a secured environment then it is up to the network administrator to turn off all the security functions. This is configured during the pre-association phase of the protocol.

When the CEs, FEs are running over IP networks or in an insecure environment, this TML uses TLS [8] to provide security. The security association between the CEs and FEs MUST be established before any ForCES protocol messages are exchanged between the CEs and FEs.

7.1.TLS Usage for this TML

This section is applicable for CE or FE endpoints that use the TML with TLS [8] to secure communication.

Since CE is master and FEs are slaves, the FEs are TLS clients and CEs are TLS server. The endpoints that implement TLS MUST perform mutual authentication during TLS session establishment process. CE must request certificate from FE and FE needs to pass the requested information.

We recommend ôTLS-RSA-with-AES-128-CBC-SHAö cipher suite, but CE or FE may negotiate other TLS cipher suites. TLS must be used for all control channel messages. TLS is optional for the data channel since data channel packets are not encrypted externally to the NE.

This TML uses TLS to provide security when the NE is in an insecure environment. This is because IPsec provides less flexibility when configuring trust anchors since it is transparent to the application and use of Port identifiers is prohibited within IKE Phase 1. This provides restriction for IPsec to configure trust anchors for each

application separately and policy configuration is common for all applications.

8. IANA Considerations

The TCP/IP TML needs to have a two well-defined TCP port numbers, which needs to be assigned by IANA. The control port is referred to as the TCP_TML_CONTROL_PORT. The data port is referred to as the TCP TML DATA PORT.

9. References

9.1.Normative References

1. S. Bradner, "The Internet Standards Process -Revision 3", RFC 2026, October 1996.

2. S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels", RFC2119 (BCP), IETF, March 1997.

3. Khosravi, et al., öRequirements for Separation of IP Control and Forwardingö, RFC 3654, November 2003.

4. L. Yang, et al., ö ForCES Architectural Frameworkö, RFC 3746, April 2004.

5. L. Yang, et al., ö ForCES Forwarding Element Functional Modelö, work in progressö, July 2004,<draft-ietf-forces-model-03.txt>

6. A. Audu, et al., ô Forwarding and Control Element protocol (FACT)" draft-gopal-forces-fact-06.txt, February 2004.

7. A. Doria, et al., öForCES protocol specificationö, draft-ietf-forces-protocol-00.txt, September 2004.

9.2.Informative References

8. Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A. and P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

9. Jungmaier, A., Rescorla, E. and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002.

10. R. Stewart, et al., ôStream Control Transmission Protocol (SCTP)ö, RFC 2960, October 2000.

11. E. Kohler, M. Handley, S. Floyd, J. Padhye, ôDatagram Congestion Control Protocol (DCCP)ö, draft-ietf-dccp-spec-04.txt, June 2003.

12. Floyd, S., ôCongestion Control Principlesö, RFC 2914, September 2000.

13. A. Doria, F. Hellstrand, K. Sundell, T. Worster, ôGeneral Switch Management Protocol (GSMP) V3ö, RFC 3292, June 2002.

14. H. Balakrishnan, et al. ôThe Congestion Managerö, RFC 3124, June 2001.

15. H. Khosravi, S. Lakkavali, ôAnalysis of protocol design issues for open standards based programmable routers and switchesö [SoftCOM 2004]

16. S. Lakkavali, H. Khosravi, ôForCES protocol design analysis for protection against DoS attacksö [ICCCN 2004]

10. Acknowledgments

11. Authors' Addresses

Hormuzd Khosravi
Intel
2111 NE 25th Avenue
Hillsboro, OR 97124
Phone: 1-503-264-0334
Email: hormuzd.m.khosravi@intel.com

Furquan Ansari
101 Crawfords Corner Road
Holmdel, NJ 07733
USA
Phone: +1 732-949-5249
Email: furquan@lucent.com

Jon Maloy
Ericsson Research Canada
8400 Boul Decarie
Ville Mont-Royal, Quebec H4P 2N2

Canada
Phone: 1-514-345-7900
Email: jon.maloy@ericsson.com

Shuchi Chawla
Intel
2111 NE 25th Avenue
Hillsboro, OR 97124
Phone: 1-503-712-4539
Email: shuchi.chawla@intel.com

Copyright Statement