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

Internet Draft
Document: draft-ietf-forces-tcptml-01.txt
Expires: January 2006
Working Group: ForCES

July 2005

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

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.

```
+------------------------------------------------
|               CE PL layer                      |
+------------------------------------------------
|              CE TML layer                      |
+------------------------------------------------
                           ^
                           |
             ForCES        |   (i.e. Forces data + control
             PL            |    packets )
             messages      |
             over          |
             specific      |
             TML           |
             encaps        |
             and           |
             transport     |
                           |
                           v
+------------------------------------------------
|              FE TML layer                      |
+------------------------------------------------
|               FE PL layer                      |
+------------------------------------------------
```

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

[TBD: Update this section based on the protocol selected for the data channel.]
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. As an example, the communication channels for support of High Availability with redundant CEs are also included. The setup of these channels would be dependent on the High Availability model used in the NE.

```
                           ACTIVE CE              STANDBY CE
                    +-------------------+  +-------------------+
                    | CE: PL            |  | CE: PL            |
                    +-------------------+  +-------------------+
                    | CE: TML           |  | CE: TML           |
                    +-------------------+  +-------------------+
                    | CE: TCP           |  | CE: TCP           |
                    +-------------------+  +-------------------+
                      | |   | |    | |       |  |     |  |
                      | .   | .    | .       |  .     |  .
                      | |   | |    | |       |  |     |  |
                      | .   | .    | .       |  .     |  .
                      | |   | |    | |    Cc1Æ Cd1Æ Cc2Æ Cd2àCcnÆ CdnÆ
                      | .   | .    | .
            +-Cc1-----+ |   | |    | +-.-.-.-.-.Cdn.-+
            |  +-Cd1-.-.+   | .    +--------Ccn---+  |
            |  |            | |                   |  .
            |  .        +Cc2+ .                   |  |
            |  |        | +Cd2+                   |  .
            |  .        | |                       |  |
         +-----------+ +-----------+          +-----------+
         | FE: TCP   | | FE: TCP   |   . . .  | FE: TCP   |
         +-----------+ +-----------+          +-----------+
         | FE: TML   | | FE: TML   |          | FE: TML   |
         +-----------+ +-----------+          +-----------+
```

```
         | FE: PL    | | FE: PL    |          | FE: PL    |
         +-----------+ +-----------+          +-----------+
           FE1            FE2                    FEn
        \-------------V------------/
                 FE 1+1 Redundancy

   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. [TBD: Update based on IPSec/TLS decision.]

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 control messaging. For multicast/broadcast addressing/delivery of control messages, this TML uses multiple TCP connections between the CE and FEs.

Additionally, the TML layer maintains the mapping between the PL layer addresses and the channel descriptors assigned by the TML layer. The PL layer is unaware of these descriptors; the PL layer only uses the PL layer addresses for all communication with the TML layer.

[TBD: Add any information on addressing for data channel.]

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. Additionally it supports multiple levels of prioritization for control messages. The scheduling algorithm used at the TML layer gives preferential treatment to higher priority messages. The scheduling algorithm used in the TML layer is implementation dependent.

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.

[TBD: Remove this? Use liveliness message at TCP layer? What about for the data channel? What if the protocol doesnÆt support such a message? If a heartbeat message is required at the TML layer for the data channel, does that imply that TML messaging is required? See updates to Section 4.9 and 5.]

TML is responsible for keeping the control and data communication channels up. It however does not have the authority to decide which CE to set up the channels with. That is outside its control.

If a FE-CE communication channel goes down or connectivity is lost, the following steps are taken by the TML layer:

- FE TML attempts to reestablish the communication channel
- If the FE TML is unable to reestablish the channel (after some configured number of retries/timeout), it notifies the FE PL that the channel is down.
- CE TML waits for the channel to be reestablished (since only the FE can reestablish it) for some configured timeout prior to notifying the CE PL that the channel is down.

TBD: Since the PL layer is unaware of the number of channels etc., what if only one channel goes down, but the other is up? The TML layer notifies the PL layer of which channel (control/data) is down?

If an FE goes down and a standby FE exists for it, and it has communication channels set up with the CE, the CE PL may start to use the channels associated with the standby FE. This is not within the scope of TML itself, but falls in the scope of High Availability.

TBD: If an FE goes down and a standby FE exists for it, but it does not have communication channels set up with the CE, how should it be notified to set up the channels? This is not within the scope of TML itself, but falls in the scope of High Availability. Do we need this mentioned here?

4.9.Encapsulations Used

There is no further message encapsulation of control and data messages done at the TML layer. The PL generated control messages are transported as is by the TML layer. The ForCES protocol control messages are encapsulated with a TCP/IP header. [TBD: Encapsulation of data messages is dependent on the protocol that will be used for the data channel (TCP is only for the control channel). If DCCP is used for the data channel, then a DCCP header will be added.]

5. TML Messaging

There is no TML layer messaging. TML only transports messages from the PL layer.

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 Service Interface

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.

[TBD: TBD: Should tmlInit() be invoked by the PL Layer at all since we have now said the PL Layer is oblivious of the number of channels, their attributes etc. Currently, we had specified that

tmlInit() would specify the attributes for the channels to be setup. It seems like due to the change in the connection setup model this information should be fed to the TML Layer via some other means. The CE functions as the server, so the server should be up and running however prior to the clients (FEs) coming up and trying to connect to the server (CE). The other option is that the mechanism to specify attributes for the channels is distinct from the process that uses those attributes to start up the server. In that case, tmlInit() could be used mainly to start the server.]

6.1.2.TML Channel Open

```
status tmlOpen(
  in  elementId,
  in  channelMode,
  in  ctrlChannelAttributes,
  in  dataChannelAttributes,
  in  eventHandlerCallBack)
```

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

Output Parameters:
  none

Returns:
  status: SUCCESS
          Errors TBD

Synopsis:
tmlOpen() results in one or more communication channels for control and data messaging being established with the specified elementId. It is up to the TML layer implementation whether to setup a single channel for both control and data messaging or distinct channels for each. The channel may be specified as unicast or multicast via channelMode. This call may either trigger the establishment of the channel(s), or if the channel(s) are already established, it only results in a registration for the channel(s). In either case, if successful, status is returned to indicate successful creation/registration of the control and data channels. No descriptors are returned to the PL layer since the TML layer maintains the mapping between the PL provided elementId and the

descriptors it allocates. If this call triggers the establishment of the control and data channels, the channels are established using the ctrlChannelAttributes and dataChannelAttributes parameters respectively, specified to the call. Once the channel(s) are 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 for both control and data channels 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, TCP TML has no need to set up distinct channels for unicast and multicast communication; they are both mapped to the same TCP connection.

6.1.3. TML Channel Close

```
status tmlClose(
  in  elementId,
  in  mode)
```

Input Parameters:
  elementId: address of element with which communication channel is
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 communication channels connecting to the specified elementId. This API closes both control and data channels

(regardless of whether they are implemented as a single channel or distinct channels in the TML layer); it is not possible to close just one of them.   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 mode, 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, it specifies a CE ID for the elementId. If the CE PL invokes it, it specifies an FE ID for the elementId.

6.1.4.TML Channel Write

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

Input Parameters:
  elementId: address of element to be written to; may be a unicast, multicast or broadcast address
  msgType: control versus data message
  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 to the address specified by elementId. If the specified elementId is associated with a multicast group, the message will be sent to all members of the group. Similarly, if the elementId specified is a broadcast address, the message is sent to all elements associated with the broadcast address. The msgType

parameter is used to specify whether the message is a control or data type of message. Based on the message type, the TML will send the message over the appropriate channel. The TML layer uses the address specified by elementId and the msgType to map to the appropriate channel to be used for sending the message. 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. If the FE PL invokes it, it specifies a CE ID for the elementId. If the CE PL invokes it, it specifies an (unicast/multicast/broadcast) FE ID for the elementId.
In the case of TCP TML since there is a single channel used for unicast, multicast and broadcast messaging, the same channel is used for sending messages regardless of the address specified. In other cases where there are distinct channels for unicast versus multicast, the channel to be written to will differ based on the address specified.

6.1.5.TML Channel Read

```
status tmlRead(
  in  elementId,
  in  msgType,
  in  msgBuf,
  in  timeout,
  out bytesRead)
```

Input Parameters:
  elementId: address of element to be read from; may be a unicast, multicast or broadcast address
  msgType: control versus data message
  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 from the specified address. The msgType parameter is used to specify whether the message to be read is a control or data type of message. The TML layer uses the address specified by elementId and the msgType to map to the appropriate channel to be used for reading the message. 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.

ForCES Usage Model:
This API may be invoked by either the CE or the FE. If the FE PL invokes it, it specifies a CE ID for the elementId. If the CE PL invokes it, it specifies an (unicast/multicast/broadcast) FE ID for the elementId.

In the case of TCP TML since there is a single channel used for unicast, multicast and broadcast messaging, the same channel is used for reading messages regardless of the address specified. In other cases where there are distinct channels for unicast versus multicast, the channel to be read from will differ based on the address specified.

6.1.6.TML Multicast Group Join

```
status tmlMulticastGroupJoin(
  in  groupId,
  in  groupAttributes)
```

Input Parameters:
  groupId: address of multicast group to join
  groupAttributes: attributes associated with the multicast group to be joined

Output Parameters:
  none

Returns:
```
  status: SUCCESS
          Errors TBD
```

Synopsis:

Joins the multicast group specified by groupId as leaf node in the group. Once a member of this group, the entity calling this API will be capable of receiving messages addressed to this multicast group. The TML layer on each end (CE/FE) maintains the mapping between the PL layer multicast address and the descriptors. The TML layer on the element which is the root of the multicast updates the set of elements that are members of the group specified by groupId.

ForCES Usage Model:
This API would be invoked on both the CE and the FE. Initially, the intent is to only support FE multicast. In such a case. on the FE the API is invoked once the PL layer on the FE receives a request from the PL layer on the CE to join a specified multicast group. On the CE it is invoked after the FE has successfully joined the multicast group.

6.1.7.TML Multicast Group Leave

```
status tmlMulticastGroupLeave(
  in  groupId)
```

Input Parameters:
  groupId: address of multicast group to leave

Output Parameters:
  none

Returns:
```
  status: SUCCESS
          Errors TBD
```

Synopsis:
Leaves the multicast group specified by groupId 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.   The TML layer on each end (CE/FE) updates the mapping between the PL layer multicast address and the descriptors. The TML layer on the element which is the root of the multicast updates the set of elements that are members of the group specified by groupId.

ForCES Usage Model:
This API would be invoked on both the CE and the FE. Initially, the intent is to only support FE multicast. In such a case, on the FE the API is invoked once the PL layer on the FE receives a request from the PL layer on the CE to leave a specified multicast group. On the CE it is invoked after the FE has successfully left the multicast group.

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. The following are the significant aspects associated with channel setup:

- A single call by the PL layer sets up the communication channels for both control and data messaging to a specific FE. The call specifies Unicast CE Id and attributes for control and data channels.
- It is up to the TML layer whether to set up a single channel for both control and data or distinct channels for control and data
- TML sets up the appropriate channels and allocates required descriptors for the channels. TML layer maintains a mapping between the Unicast FE/CE Id and the channel descriptors and channel type (control versus data) it creates.
- There is no need for channel descriptors to be returned to the PL layer at either the FE or the CE. PL Layer only uses the Unicast FE/CE Id for read/write calls and specifies the type of message (control versus data) to be read/written.
- If only one of the channels is setup successfully, the TML layer will have to return appropriate status that specifies which channel is setup successfully and which isnÆt.

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.

```
     FE1 PL           FE1 TML                  CE TML          CE PL
           |               |                       |               | \
        /  |               |                       | TBD:tmlInit() | |
   FE   |  |               |                       |<--------------| > CE Init/
 Init/  <  |               |                       |               | | Bootup
 Bootup |  |               |                       |               | /
        \  |               |                       |               |
           | tmlOpen(CeId) |                       |               |
```

```
           |-------------->|                        |               | \
           |               |CtrlChan(Cc) Setup      |               | | Setup
control
           |               |~~~~~~~~~~~~~~~~~~~~~~>|               | | channel
if not
           |               |               FeId . [CcDes<ctrl>]  | | setup.
TML
           |               |                        |               | > has
mapping
           |               |CtrlChan(Cc) Setup Rsp |               | | from PL
Layer
           |               |<~~~~~~~~~~~~~~~~~~~~~~|               | | Id to
channel
           |        CeId . [CcDes<ctrl>]          |               | |
descriptor and
           |                                        |               | / channel
type.
           |               |                        |               |
           |               |DataChan(Cd) Setup      |               | | Setup
data
           |               |~~~~~~~~~~~~~~~~~~~~~~>|               | | channel
if not
           |               |               FeId . [CcDes<ctrl>,  | | setup.
TML
           |               |                        CdDes<data>]  | | updates
           |               |                        |               | > mapping
from
           |               |DataChan(Cd) Setup Rsp |               | | PL Layer
           |               |<~~~~~~~~~~~~~~~~~~~~~~|               | | Id to
channel
           |        CeId . [CdDes<data>]          |               | |
descriptor and
           |               |                        |               | / channel
type.
           |               |                        |               |
           |  <-- status   |                        |               |
           |               |                        |               |
           |tmlEvent(ChUp) |                        |tmlEvent(ChUp) |
           |<--.--.--.--.--|                        |--.--.--.--.-->|
           |               |                        |               |
           |               |   Asso Setup Req       |               |
           |---------------|------------------------|-------------->|
           |               |   Asso Setup Rsp       |               |
           |<--------------|------------------------|---------------|
           |               |                        |               |
           |               |    Capability Query    |               |
           |<--------------|------------------------|---------------|
           |               | Capability Query Rsp   |               |
           |---------------|------------------------|-------------->|
```

```
           |               |                        |               |
           |               |   Topology Query       |               |
           |<--------------|------------------------|---------------|
           |               | Topology Query Rsp     |               |
           |---------------|------------------------|-------------->|
           |               |                        |               |
           |               |STEADY STATE OPERATION  |               |
           |<--------------|------------------------|-------------->|
           |               |                        |               |
Legend:
 PL  --------> PL : Protocol layer messaging
 PL  --------> TML: TML API
 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 shutdown control and data channels via invocation of the tmlClose() API. When the PL layer shuts down the channels, the channels are torn down; hence ForCES messaging between the CE and FE is no longer possible over those channels. A tmlClose() results in both control and data channels (regardless of whether they are implemented as a single channel or distinct channels in the TML layer) being shutdown; it is not possible to close just one of them. A subsequent tmlOpen() triggers establishment of the channel. The channel(s) may be shutdown by either the FE or the CE. If an FE initiates the shutdown, it specifies the CE Id associated with the channel(s) to be shutdown. If a CE initiates the shutdown, it specifies the FE Id associated with the channel(s) to be shutdown. A channel shutdown by the FE is illustrated in Figure 5 and a channel shutdown by the CE is illustrated in Figure 6.

```
        FE PL           FE TML                 CE TML          CE PL
           |               |                       |               |
           |               |STEADY STATE OPERATION |               |
           |<--------------|-----------------------|-------------->|
           |               | Config Request        |               |
           |<--------------|-----------------------|---------------|
           |               | Config Response       |               |
           |---------------|-----------------------|-------------->|
           |               |                       |               |
           |               | Association Teardown  |               |
           |<--------------|-----------------------|---------------|
           |               |                       |               |
           |               |                       |               | \
           |tmlClose(CeId) |                       |               | | FE
initiated:
           |-------------->|                       |               | > FE
specifies CE
           | <-- status    |                       |               | | Id
associated
           |               |                       |               | / with
channel.

Legend:
 PL  --------> PL : Protocol layer messaging
 PL  --------> TML: TML API
 TML --.--.--> PL : Events/Notifications/Upcalls
 TML ~~~~~~~~> TML: Internal protocol communication
```

Figure 5: Protocol Shutdown: FE Initiated

```
        FE PL            FE TML                  CE TML          CE PL
           |                |                       |               |
           |                |STEADY STATE OPERATION |               |
           |<--------------|-----------------------|-------------->|
           |               | Config Request        |               |
           |<--------------|-----------------------|---------------|
           |               | Config Response       |               |
           |---------------|-----------------------|-------------->|
           |               |                       |               |
           |               | Association Teardown  |               |
           |<--------------|-----------------------|---------------|
           |               |                       |               |
           |               |                       |               | \
           |               |                       |tmlClose(FeId) | | CE
initiated:
           |               |                       |<--------------| > FE
specifies CE
           | <-- status    |                       | status -->    | | Id
associated
           |               |                       |               | / with
channel.

Legend:
 PL  --------> PL : Protocol layer messaging
 PL  --------> TML: TML API
 TML --.--.--> PL : Events/Notifications/Upcalls
 TML ~~~~~~~~> TML: Internal protocol communication
```

Figure 6: Protocol Shutdown: CE Initiated

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, a CE may request FEs to join and leave specified multicast groups. Multicast support is CE-initiated. FEs can join a multicast group only if the CE requests them to join the group. TML maintains mapping between PL layer IDs and channel descriptors for multicast. The following is the significant steps for adding or removing members from a multicast group:

- CE PL communicates with FE PL for requesting the FE to join or leave a multicast group.
- FE PL informs FE TML regarding the join or leave request.

- FE TML updates the multicast group information. It updates the mapping between the FE Multicast Id and the channel descriptor to be used for multicast for that FE. This mapping may be from [Multicast FE Id] . [FE Id] . [Channel descriptor] or directly

     from [Multicast FE Id] . [Channel descriptor]. This is
     implementation dependent.
   - FE PL responds to CE PL informing it of the status of the join or
     leave request.
   - If the join or leave request was successful, CE PL informs CETML
     regarding the update to the multicast group membership.
   - CE TML updates the multicast group membership. It updates the
     mapping between the FE Multicast Id and the set of channel
     descriptors to be used for multicast to the FEs that are members
     of this group. This mapping may be from [Multicast FE Id] . [Set
     of FE Ids] . [Set of channel descriptors] or directly from
     [Multicast FE Id] . [Set of channel descriptors]. This is
     implementation dependent.
   - There is no need for any descriptors to be returned to the PL
     layer at either the FE or the CE. PL Layer only uses the
     Multicast FE Id for write calls and specifies the type of message
     (control versus data) to be written.

A tmlWrite() on a unicast FE Id results in a unicast message being sent to the FE associated with the channel. A tmlWrite() on a multicast FE Id results in multicast messaging. Figures 7 and 8 illustrate multicast scenarios with 2 FEs, FE1 and FE2. In Figure 7, the CE requests FE1 to join a multicast group. Although not shown as a separate figure, if FE2 were to join the same group, the join procedure would be the same as in Figure 7; it would result in the multicast group membership being updated at the TML layer on the CE to include FE2 in the group. In Figure 8, the CE requests FE1 to leave the multicast group, thus resulting in only FE2 being a member of the multicast group.

Multicast Scenario with FE1 joining group: New group created

```
        FE1 PL        FE1 TML              CE TML           CE PL
          |               |                   |               |
          |               |                   |               | \
          |               MC Grp Join Req (McId)              | |
          |<--------------|-------------------|---------------| | CE:PL Level
multicast group
[TML      | tmlJoin(McId) |                   |               | | join request
sent to each
updates  |-------------->|                   |               | | FE:PL that
needs to be part
MC grp    |        McId = {FE1_ChDes}         |               | > of a multicast
group, McId,
info]     |               |                   |               | | where McId
specifies a
          |  <-- status   |                   |               | | multicast
group Id at the
          |               |                   |               | | PL layer.
```

```
         |              MC Grp Join Rsp (status)             | |
         |---------------|-------------------|-------------->| /
         |               |                   |               |
         |               |                   |               | \
         |               |                   |tmlJoin(McId)  | | TML updates
multicast
         |               |                   |<--------------| | group
membership.  PL is
         |               |              McId = {FE1_ChDes}   | > only aware of
PL layer
         |               |                   |               | | multicast
group Id, that is,
         |               |                   |  status -->   | | McId]
```

```
          |               |                   |               | /
```

Figure 7: Multicast Support: FE1 Joins Group

Multicast Scenario with FE1 leaving group: Group membership updated to exclude FE1

```
        FE1 PL        FE1 TML              CE TML           CE PL
         |               |                   |               |
         |               |                   |               | \
         |              MC Grp Leave Req (McId, FE1)        | |
         |<--------------|-------------------|---------------| | CE:PL Level
multicast group
[TML     | tmlLeave(McId)|                   |               | | leave request
sent to FE1:PL
removes  |-------------->|                   |               | | that needs to
be removed
MC grp   |        McId = {}                  |               | > from multicast
group, McId,
info]    |               |                   |               | | where McId
specifies a
         |  <-- status   |                   |               | | multicast
group Id at the
         |               |                   |               | | PL layer.
         |             MC Grp Leave Rsp (status)             | |
         |---------------|-------------------|-------------->| /
         |               |                   |               |
         |               |                   |               | \
         |               |                   |tmlLeave(McId) | | TML removes
FE1 from
         |               |                   |<--------------| | multicast
group McId.
         |               |              McId = {FE2_ChDes}   | > That leaves
only FE2
         |               |                   |               | | in the group.
         |               |                   |  status -->   | |
         |               |                   |               | /
```

Figure 8: Multicast Support: FE1 Leaves Group

6.4.Broadcast Model

The TML layer provides support for broadcast. In the ForCES model, support is required to broadcast to the FEs from a CE. The broadcast model is just a special case of multicast, where all FEs are included. TBD: Is there anything else to be added/discussed here. Join/Leave messaging also used for broadcast? ForCES draft also talks of CE broadcast and NE broadcast.

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

[TBD: Update based on inclusion of IPSec for security.]

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 one well-defined TCP port number, which needs to be assigned by IANA. The control port is referred to as the TCP_TML_CONTROL_PORT. [TBD: Add information for data channel port if required based on the protocol for the data channel.]

9. Manageability

TBD

10. References

10.1.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.

10.2.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]

11.   Acknowledgments

12.   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