ForCES Working Group
Internet-Draft
Expires: October, 2006

W. M. Wang
Zhejiang Gongshang Univ.
J. Hadi Salim
Znyx Networks

April 2006

ForCES Transport Mapping Layer (TML) Service Primitives

draft-ietf-forces-tmlsp-00.txt

Status of this Memo

Copyright Notice

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 [RFC2119].

Abstract

This document proposes Service Primitives of Transport Mapping Layer (TML) in a Forwarding and Control Element Separation (ForCES) network

element, to standardize the operations of ForCES Protocol Layer (PL) to a variety of TMLs.

Table of Contents

1. Introduction

The Forwarding and Control Element Separation (ForCES) is a proposed architecture for network elements like routers, documents of which include RFC3654, RFC3746, and the ForCES protocol[ForCES-PL] and the ForCES FE model[ForCES-Model] that are work in progress. RFC3654 defines the ForCES requirements, RFC3746 defines the ForCES framework, and the ForCES protocol defines the message exchange protocol between the Forwarding Element (FE) and the Control Element (CE) in the ForCES NE (see RFC3654 for the terminology definitions).

The ForCES protocol infrastructure consists of two components:

1. The Protocol Layer (PL), which is responsible for generating ForCES protocol messages and also processing protocol messages that come from peering protocol layers in the same ForCES NE.

2. The Transport Mapping Layer (TML), which is responsible for ForCES protocol message transports over variant transport media like IP, Ethernet, ATM, etc.

The IETF ForCES protocol mainly defines the PL. TMLs according to various transport media are also to be individually defined by IETF. A ForCES PL implementation must be portable across all TMLs. It is feasible that the implementers of TML and PL may be from different organizations. As a result, there must be an interoperable method to interconnect the PL and TML. A private method would make the PL and TML implementations also private.

The purpose of this document is to present the method for an interoperable interconnection of the PL and a variety of TMLs. Although there might be other choices like using PL-TML messages, as a more efficient way for data transmission and processing, a method based on service primitives are recommended for interconnection of PL and TML. In this document, a set of TML Service Primitives are presented and related TML parameters are defined. Also presented in the document is the theory of operation of PL-TML based on the service primitives and the parameters.

2. Definitions

This document follows the terminology used by RFC3654, RFC3746, and the ForCES protocol[ForCES-PL]. Some are just copied here:

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 [ForCES-PL].

ForCES Protocol Transport Mapping Layer (ForCES TML) -- A layer in ForCES protocol architecture that uses the capabilities of existing transport protocols to specifically address 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. The ForCES TML specifications are detailed in separate ForCES documents, one for each TML.

3. Overview

3.1. ForCES Protocol Framework

The ForCES protocol has presented the protocol framework as in

Figure 1. The framework shows the relationship between PL and TML. According to this framework, TML lies under PL and provides services to the PL. CE PL communicates with FE PL via CE TML and FE TML. On

transmit, the PL delivers its ForCES messages to the TML. The TML further delivers the message to the destination TML(s). On receive, the TML delivers the ForCES messages it received to the PL.

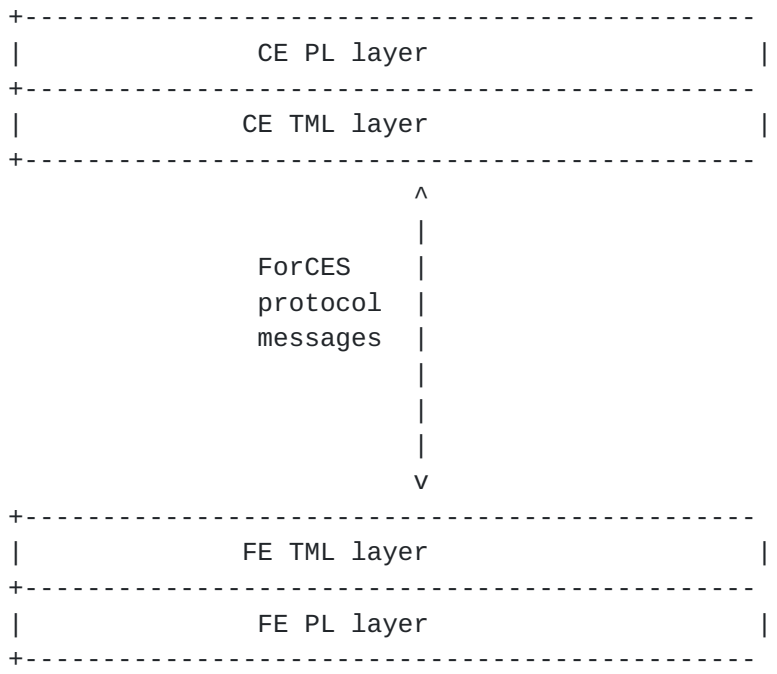

Figure 1

3.2. TML Requirements

The ForCES protocol [ForCES-PL] also presents TML requirements. We list the requirements as below. These requirements are expected to be delivered by TML using any kind of transport media, though the text does not define how such mechanisms are delivered. Each TML must describe how it contributes to achieving the requirements. If for any reason a TML does not provide a service listed below a justification needs to be provided.

The TML requirements are:

1. Reliability
As defined by RFC 3654, section 6 #6.

2. Security
TML provides security services to the ForCES PL. TML layer should

support the following security services and describe how they are achieved.

* Endpoint authentication of FE and CE.

* Message Authentication

* Confidentiality service

3. Congestion Control
The congestion control scheme used needs to be defined. The congestion control mechanism defined by the TML should prevent the FE from being overloaded by the CE or the CE from being overwhelmed by traffic from the FE. Additionally, the circumstances under which notification is sent to the PL to notify it of congestion must be defined.

4.Uni/multi/broadcast addressing/delivery if any
If there is any mapping between PL and TML level Uni/Multi/Broadcast addressing it needs to be defined.

5. HA decisions
It is expected that availability of transport links is the TML's responsibility. However, on config basis, the PL layer may wish to participate in link failover schemes and therefore the TML must support this capability.

6. Encapsulations used.
Different types of TMLs will encapsulate the PL messages on different types of headers. The TML needs to specify the encapsulation used.

7. Prioritization
It is expected that the TML will be able to handle up to 8 priority levels needed by the PL layer and will provide preferential treatment. TML needs to define how this is achieved. The requirement for supporting up to 8 priority levels does not mean that the underlying TML MUST be capable of handling up to 8 priority levels. In such an event the priority levels should be divided between the available TML priority levels. For example, if the TML only supports 2 priority levels, the 0-3 could go in one TML priority level, while 4-7 could go in the other.

8. Protection against DoS attacks
As described in the Requirements RFC 3654, section 6

4. TML Modeling

To make PL capable of interconnection with variant TMLs in a standardized way, the first work to do is to model the variant TMLs in a uniform way from the perspective of a uniform PL.

Identical to modeling an LFB in an FE, from the perspective of PL, TML properties can be abstracted by the following entities:

TML Events:

The TML events that PL requires to know when the events happen in the TML.

TML attributes:
The TML attributes that are required to be configured by PL according to PL requirements. The attributes can also be queried by PL.

TML capabilities:
The TML capabilities that PL are interested to know.

Note that, not all TML properties should be made perceivable by PL from PL point of view. PL only cares those TML properties that are common to all TMLs and that PL should interact with via service primitives so that TML can work according to PL's requirements.

Via TML Service Primitives, PL should be able to access above properties of variant TMLs.

4.1.TML events

There might be several TML events, but only some of them are PL must sense via PL-TML interface.

A callback mechanism is defined for PL to sense the TML events by PL-TML service primitives. PL first provides every event with a callback function for the event processing in the PL. PL then tells the callback handle to TML by service primitives. The callback handle is usually stored in TML as a parameter. Whenever the relavent event happens in TML, the TML triggers the handle to notify PL of the event. PL executes the callback function, and asynchronously begins to process the event.

After a TML event happened and PL is notified and a procedure to process the event is executed, PL may be further interested in knowing if the event status in the TML still exists or not. To meet this requirement, an 'eventState' parameter is used in callback functions to indicate if the reported is the event happened or the

event released. Whereas, not all events need to notify PL of their release state.

The following TML events must be made to be notified by PL. If for any reason a TML does not provide the service, a justification needs to be provided in the TML specification.

1) TML failure event

This event happens when there is a TML failure. It is up to individual TML specifications and even individual implementations to

specify the detailed event triggering conditions, but usually, TML failure should include the following cases:
. local TML link failure
. peer TML unavailable
. peer TML left

The different cases that cause the failure will be stated by a failure code in the event callback function.

Callback handle for TML failure event is then defined as:

```
  status =              -- SUCCESS or errorIndication
           callbackTMLFailureEvent(
                     IN   eventState
                     IN   failCode
                                  )
```

Where,
eventState = EventHAPPENED or EventRELEASED
failCode indicates the failure case

2) Message arrival event

TML can make it as an event when a PL ForCES protocol message coming from peering TML arrives at the TML and the TML has made it ready for PL to receive the message. In this way, an asynchronous message receive mode can be realized in PL. In addition to this asynchronous mode, PL can also use a specific TML receive service primitive (defined below) for synchronous reception of PL messages.

Callback handle for message arrival event is defined as:

```
  status =              -- SUCCESS or errorIndication
           callbackTMLMsgArrivalEvent(
                        IN   msglen
                        IN   msgPDU
```

)

Where, msglen indicates the ForCES message length, and msgPDU is the ForCES message body in its Protocol Data Unit format. There is no need for message arrival event to notify a release state.

3) Control message congestion event

ForCES control messages are defined as all ForCES protocol messages except ForCES redirect messages. ForCES redirect messages are identified by the ForCES message types being marked as 'PacketRedirect'.

This event happens when the TML comes to a congestion state for the control message transmission. It is up to individual TML

specifications and even individual implementations to specify the detailed event triggering conditions.

This event can be used by PL layer to monitor the control message transmission. In FEs, this event may also be used to help monitoring possible DoS attacks from redirected packets.

Callback handle for TML control message congestion event is defined as:

```
  status =              -- SUCCESS or errorIndication
           callbackTMLCtrMsgCongestEvent(
                          IN   eventState
                                        )
```

Where,
eventState = EventHAPPENED or EventRELEASED

4)Redirect message congestion event

This event happens when the TML comes to a congestion state for the PL redirect message transmission. It is up to individual TML specifications and even individual implementations to specify the detailed event triggering conditions.

This event can be used by PL layer to monitor the redirect message transmission. In FEs, this may also be used to help monitoring possible DoS attacks from redirect packets.

Callback handle for TML redirect message congestion event is defined as:

```
  status =              -- SUCCESS or errorIndication
```

```
            callbackTMLRedMsgCongestEvent(
                          IN   eventState
                                        )
```

Where,

eventState = EventHAPPENED or EventRELEASED

5)DoS attack alert event

This event happens when the TML comes to a state that it feels there are abnormal amount of PL redirect messages and there has made it quite hard to transport PL control messages. Whereas, it is up to individual TML specifications and even individual implementations to specify the detailed and precise triggering conditions for the event.

This event is used by PL to monitor the security state for TML message transmission. In FEs, when this event happens, usually FE PL should trigger a DoS attack alert event to inform CE of the event. CE may take further effects trying to prevent the attack. Note that, the

event is an alert, when it happens, usually it does not mean the CE-FE communication is totally lost.

Callback handle for TML DoS attack alert event is defined as:

```
  status =              -- SUCCESS or errorIndication
            callbackTMLDoSAttackAlertEvent(
                          IN   eventState
                                        )
```

Where,

eventState = EventHAPPENED or EventRELEASED

4.2.TML attributes

1) Event handles

Event handles are handles for PL to access events. An event callback function handle is used as the purpose. Defining the handles as an attribute of the TML, then, for PL to set the handle to TML is for the PL to subscribe to the TML for the event notification, to delete the handle from the TML is to unsubscribe the event from the TML.

We define the event handle TML attribute as below:

```
<dataTypeDef>
  <name>Eventhandle</name>
  <synopsis>Event callback handle</synopsis>
  <struct>
    <element elementID="1">
```

```
        <name>eventType</name>
        <synopsis>event type represented by an ID </synopsis>
        <typeRef>uint16</typeRef>
        <specialValues>
          <specialValue Value="1">
            <name>TMLFailureEvent</name>
            <synopsis>TML failure event</synopsis>
          </specialValue>
          <specialValue Value="2">
            <name>TMLMsgArrivalEvent</name>
            <synopsis>TML ForCES message arrival event</synopsis>
          </specialValue>
          <specialValue Value="3">
            <name>TMLCtrMsgCongestEvent</name>
            <synopsis>
            TML ForCES congtrol message transmit congestion event
            </synopsis>
          </specialValue>
          <specialValue Value="4">
            <name>TMLRedMsgCongestEvent</name>
            <synopsis>
```



```
            TML ForCES redirect message transmit congestion event
            </synopsis>
          </specialValue>
          <specialValue Value="5">
            <name>TMLDoSAttackAlertEvent</name>
            <synopsis>TML DoS attack alert event</synopsis>
          </specialValue>
        </specialValues>
      </element>
      <element elementID="2">
        <name>handle</name>
        <synopsis>callback function handle of the event</synopsis>
        <typeRef>uint64</typeRef>
      </element>
    </struct>
  </dataTypeDef>

  <attribute access="read-write" elementID="1">
    <name>EventHandles</name>
    <synopsis>event handle table in the TML</synopsis>
    <array>
       <typeRef>EventHandle</typeRef>
    </array>
  </attribute>
```

2)multicast lists

This attribute is used for TML to multicast ForCES messages. The multicast list should be configured by PL, but individual TML specifications should define how such multicast list maps to TML transport level multicast mechanisms.

A PL level multicast list includes a group ID for the multicast and a number of members of the multicast. The members are represented by PL level protocol src/destIDs (include FE ID and CE ID).

Note that, there might be more than one multicast group for multicast applications. The multiple multicast lists form a multicast list table in TML.

The attribute for the multicast lists is defined as below:

```
                    }
<dataTypeDef>
   <name>McastList</name>
   <synopsis>
   a PL level multicast list for ForCES multicast transport
   </synopsis>
   <struct>
       <element elementID="1">
          <name>groupID</name>
```



```
          <synopsis>32bits group ID of the multicast</synopsis>
          <typeRef>uint32</typeRef>
       </element>
       <element elementID="2">
          <name>members</name>
          <synopsis>
          members of the multicast represented by FE ID or CE ID
          </synopsis>
          <array>
             <typeRef>uint32</typeRef>
          </array>
       </element>
    </struct>
</dataTypeDef>

<attribute access="read-write" elementID="2">
  <name>McastLists</name>
  <synopsis>
  a table representing several multicast lists in the TML
  </synopsis>
  <array>
```

```
      <typeRef>McastList</typeRef>
    </array>
  </attribute>
```

3) Working TML Type

A TML implementation may be capable of several TML transport ways. For example, a TML with IP transport media may be able to support several transport protocols, like TCP+UDP, TCP+DCCP, SCTP, etc. In this case, there should be a TML attribute describing the different types and make PL able to switch among the types under the control of CE.

The TML type is represented by an ID, defined as below:

```
<dataTypeDef>
  <name>TMLType</name>
  <synopsis>TML Type</synopsis>
  <atomic>
    <typeRef>uint16</typeRef>
      <specialValues>
        <specialValue Value="1">
          <name>TmlTcpUdp</name>
          <synopsis>TML uses TCP+UDP</synopsis>
        </specialValue>
        <specialValue Value="2">
          <name>TmlTcpDccp</name>
          <synopsis>TML uses TCP+DCCP</synopsis>
```



```
        </specialValue>
        <specialValue Value="3">
          <name>TmlSctp</name>
          <synopsis> TML uses SCTP</synopsis>
        </specialValue>
        <specialValue Value="4">
          <name>TmlEth</name>
          <synopsis>TML uses Ethernet</synopsis>
        </specialValue>
        <specialValue Value="5">
          <name>TmlAtm</name>
          <synopsis>TML uses ATM</synopsis>
        </specialValue>
      </specialValues>
  </atomic>
</dataTypeDef>
```

The TML attribute for the working TML type is defined as below:

```
<attribute access="read-write read-only" elementID="3">
  <name>WorkingTMLType</name>
  <synopsis>current working TML type assigned by PL </synopsis>
  <typeRef>TMLType</typeRef>
</attribute>
```

Note that, the working TML type attribute may be configurable or may only be readable depending on implementations. TML capability on the TML type (defined below) will tell if it is configurable or not. To query the TML type capability before configuring the working TML type attribute will help to correctly configure it.

4) Media specific TML parameters

Individual TML may require configuring some TML parameters specific to its TML media. There leave a space in TML service primitives for such requirement. Individual TML specifications should provide detailed definitions for such parameters.

5) Vendor specific TML parameters

Vendors of individual implementations of TML may require configuring some TML parameters specific to its implementation. There leave a space in TML service primitives for such requirement. Individual implementation should provide detailed definitions for such parameters.

4.3. TML capabilities

1) Supported TML type

A TML implementation may be capable of several TML transport ways. This capability indicates PL of the supported types.

The TML capability for the supported TML type is defined as below:

```
<capability elementID="4">
  <name>SupportedTMLType</name>
  <synopsis>supported TML types in the TML mechanism</synopsis>
  <array type="variabl-size">
     <typeRef>TMLType</typeRef>
  </array>
</capability>
```

2) TML type configuration capability

```
<capability elementID="5">
  <name>TMLTypeConfigurable</name>
  <synopsis>TML Type configurable or not by PL </synopsis>
  <typeRef>boolean</typeRef>
</capability>
```

5. Service Primitives

5.1.Design Principles

Two principles are applied to this PL-TML service primitives design:

1.PL-TML service primitives should hide implementation details regarding reliability, security, multicast, congestion control, etc from PL.

2.PL-TML service primitives should avoid leading TML to read ForCES protocol message PDU to get information, so as to immunize the TML from the possible change of ForCES protocol PDU (like the protocol update).

5.2.Objectives

There are several basic design objectives:

1. Support for unicast, multicast and broadcast PL level mechanisms.
2. support for both reliable and unreliable delivery.
3. Support for in-order or agnostic delivery.
4. Support for timeliness requirements.
5. Support for both synchronous and asynchronous operations.
6. Support for event notifications from TML to PL.

5.3. TML Open

Syntax:


```
  status =              -- SUCCESS or errorIndication
    TMLopen( )
```

Parameters:
none

Service Description:
The PL connects to the TML by invoking the TML open call. It highly depends on the individual TML specifications what a TML should do after receiving this call. For some TMLs, this primitive call may only act as a signal to inform TML that PL is going to use the TML

for sending or receiving PL messages, while for some other TMLs, a TML may have to do some TML level operation to prepare for PL usage when receiving this primitive call. For example, For a connectionless TML, this open primitive may does not have to do anything, while for a connection-oriented TML, this open call may be a signal for the TML to setup TML level connection(s) to peering TML(this actually means the peer-to-peer TMLs do not have to always be connected after the TML is initialized and during the post-association phase).

Another important point is, to better synchronize the operations between peering PLs, the TML will have to discard all the PL messages received from peering PL before the local TML has not yet been opened by the local PL,

5.4. TML close

Syntax:

```
  status =              -- SUCCESS or errorIndication
    TMLclose( )
```

Parameters:

```
     none
```

Service Description:
In this call, the PL disconnects from the TML. It highly depends on the individual TML specifications what a TML should do after receiving this call. For some TMLs, this primitive call may only act as a signal to inform TML that PL is not going to use the TML for sending or receiving PL messages anymore, while for some other TMLs, a TML may have to do some extra TML level operations to disconnect it to peering TMLs. For example, for a connectionless TML, this primitive may do not have to do anything, while for a connection-oriented TML, this primitive call may be a signal for the TML to disconnect all TML level connections to peering TML.

Another important point is, to better synchronize the operations between peering PLs, a TML will have to discard all PL messages received by the TML after the TML has been closed by local PL.

5.5.TML Configuration

Syntax:

```
  status =              -- SUCCESS or errorIndication
          TMLconfig(
              IN  operation
              IN  path
              IN  data
```

```
                    )
```

Parameters:

operation ?the operation type for the configuration. Two operations are defined:

```
       operation = ADD ?to add parameter
                 = DELETE ?to delete parameter
```

path ?a path composed of element ID(s) and (or) array index pointing to the element to be configured.
(TBD)

data ?the data to be configured to the element.
(TBD)

Service Description:
This primitive is used by the PL to control the behavior of the TML by the PL layer configuring the related property elements in the TML. The element is indicated by the 'path'. The value to be configured to the element is accessed by the 'data'.

5.6. TML Query

Syntax:

```
  status =              -- SUCCESS or errorIndication
          TMLquery(
              IN  path
              OUT data
                   )
```

Parameters:

path ?a path composed of element ID(s) and (or) array index pointing to the element to be queried.
(TBD)

data ?the data returned by the query.
(TBD)

Service Description:

This primitive is used by the PL to query the behavior of the TML. The querying element is indicated by the 'path'. The value queried is stored at the 'data' field. The TML executes the primitive by filling out the data field with the queried value of the element.

5.7. TML send

```
   Syntax:
     status =              -- SUCCESS or errorIndication
              TMLsend(
                  IN  msgDestID,
                  IN  msgType,
                  IN  msgPrio,
                  IN  msgLength,
                  IN  msgPDU,
                      )
```

   Parameters:
   msgDestID: the destination ID for the PL message to be sent
   msgType: the message type for the PL message to be sent
   msgPrio: the message priority for the PL message to be sent
   msgLen: the message length to be sent
   msgPDU: the ForCES protocol message to be sent in its PDU

   Service Description:
   In this service, the PL sends a message to one (unicast) or more (multicast) peer PLs via the TML. Note that this primitive includes all parameters that are necessary for TML to manage transmission of the PL message, therefore, there is no need for the TML to read in the PL message body to retrieve this parameters. In this way, we may decouple changes in ForCES protocol PDU (e.g., by the protocol update) from TML level.

   The msgDestID is used for the TML to find out TML layer transport addresses for the message transmission. It also includes the processing for PL message multicast transports, for the destination ID may also be a multicast group ID.

   The msgType is used for the TML to infer the requirements from PL level for the manage sending, regarding its reliability, timeliness, security, and congestion control. With this message type, it is easy to recognize PL redirect messages from PL control messages. Individual TML specifications should define how the msgTypes are used to map into the TML requirements.

   The msgPrio is used for the TML to meet the PL requirement for the message transmission priority; it may also be used for TML to meet the TML requirements for reliability, timeliness, security, and congestion control. Individual TML specifications should define how

   the priority is mapped into the TML transport mechanisms for prioritized transmission. Individual TML specifications may also

define how the priority is used for other TML requirements.

5.8. TML receive

```
Syntax:
  status =              -- SUCCESS or errorIndication
           TMLreceive(
               IN  msgLength,
               IN  msgPDU,
                      )
```

Parameters:
msgLen: the length of the message received.
msgPDU: the received ForCES protocol message body in its PDU format

Service Description:
This service is used for PL to synchronously receive PL messages from peering TML.

Note that a PL message arrival event described before can also be used for PL to receive PL messages from TML. The difference is that this TML receive primitive makes PL to synchronously receive messages, while a PL message arrival event receives messages in an asynchronous way. Usually, an asynchronous method is more efficient in terms of CPU cycles.

6. Theory of Operation

(TBD)

7. References

[RFC3654] Khosravi, et al., Requirements for Separation of IP Control and Forwarding, RFC 3654, November 2003.

[ForCES-PL] A. Doria, et al., ForCES protocol specifications, draft-ietf-forces-protocol-08.txt, work-in-progress, Mar. 2006.

[ForCES-Model] Yang, L., ForCES Forwarding Element Model, Aug. 2003, http://www.ietf.org/internet-drafts/draft-ietf-forces-model-05.txt.

8. Author's Address

Weiming Wang
Zhejiang Gongshang University
149 Jiaogong Road
Hangzhou 310035

P.R.China
Phone: +86-571-88071024
EMail: wmwang@mail.zjgsu.edu.cn

Jamal Hadi Salim
Znyx Networks
195 Stafford Rd. West
Ottawa, Ontario
Canada
Email: hadi@znyx.com

Appendix A. TML Attributes XML file

(TBD)

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.