INTERNET-DRAFT
Oct 15, 2004

Erik Nordmark
Sun Microsystems
Marcelo Bagnulo
UC3M

Multihoming L3 Shim Approach

<draft-nordmark-multi6dt-shim-00.txt>

Status of this Memo

Abstract

This document outlines an approach to solving IPv6 multihoming in order to stimulate discussion.

The approach is based on using a multi6 shim placed between the IP endpoint sublayer and the IP routing sublayer, and, at least initially, using routable IP locators as the identifiers visible above the shim layer. The approach does not introduce a "stack name" type of identifier, instead it ensures that all upper layer protocols can operate unmodified in a multihomed setting while still seeing a stable IPv6 address.

This document does not specify the mechanism for authenticating and authorizing the "rehoming" - this is specified in the HBA document. Nor does it specify the messages used to establish multihoming state.

The document does not even specify the packet format used for the data packets. Instead it discusses the issue of receive side demultiplexing and the different tradeoffs. The resolution of this issue will effect the packet format for the data packets.

Contents

1. Introduction

The goal of the IPv6 multihoming work is to allow a site to take advantage of multiple attachments to the global Internet without having a specific entry for the site visible in the global routing table. Specifically, a solution should allow users to use multiple attachments in parallel, or to switch between these attachment points dynamically in the case of failures, without an impact on the upper layer protocols.

The goals for this approach is to:

- o Have no impact on upper layer protocols in general and on transport protocols in particular.

- o Address the security threats in [M6THREATS] through a separate document [HBA]

- o No extra roundtrip for setup; deferred setup.

- o Take advantage of multiple locators/addresses for load spreading so that different sets of communication to a host (e.g., different connections) might use different locators of the host.

1.1. Non-Goals

The assumption is that the problem we are trying to solve is site multihoming, with the ability to have the set of site locator prefixes change over time due to site renumbering. Further, we assume that such changes to the set of locator prefixes can be relatively slow and managed; slow enough to allow updates to the DNS to propagate. This proposal does not attempt to solve, perhaps related, problems such as host multihoming or host mobility.

This proposal also does not try to provide an IP identifier. Even though such a concept would be useful to ULPs and applications, especially if the management burden for such a name space was zero and there was an efficient yet secure mechanism to map from identifiers to locators, such a name space isn't necessary (and furthermore doesn't seem to help) to solve the multihoming problem.

2. Terminology

upper layer protocol (ULP)
- a protocol layer immediately above IP. Examples are transport protocols such as TCP and UDP, control protocols such as ICMP, routing protocols such as OSPF, and internet or lower-layer protocols being "tunneled" over (i.e., encapsulated in) IP such as IPX, AppleTalk, or IP itself.

interface - a node's attachment to a link.

address - an IP layer name that contains both topological significance and acts as a unique identifier for an interface. 128 bits.

locator - an IP layer topological name for an interface or a set of interfaces. 128 bits. The locators are carried in the IP address fields as the packets traverse the network.

identifier - an IP layer identifier for an IP layer endpoint (stack name in [NSRG]). The transport endpoint is a function of the transport protocol and would typically include the IP identifier plus a port number. NOTE: This proposal does not contain any IP layer identifiers.

upper-layer identifier (ULID)
- an IP locator which has been selected for communication with a peer to be used by the upper layer protocol. 128 bits. This is used for pseudo-header checksum computation and connection identification in the ULP. Different sets of communication to a host (e.g., different connections) might use different ULIDs in order to enable load spreading.

address field
- the source and destination address fields in the IPv6 header. As IPv6 is currently specified this fields carry "addresses". If identifiers and locators are separated these fields will contain locators.

FQDN - Fully Qualified Domain Name

Host-pair context
- the state that the multi6 shim maintains for a particular peer. The peer is identified by one or more ULIDs.

2.1. Notational Conventions

A, B, and C are hosts. X is a potentially malicious host.

FQDN(A) is the domain name for A.

Ls(A) is the locator set for A, which consists of L1(A), L2(A), ... Ln(A).

ULID(A) is an upper-layer ID for A. In this proposal, ULID(A) is always one member of A's locator set.

3. Overview

This document specifies certain aspects of the approach, yet leaves other aspects open.

The main points are about using locators as the ULIDs, and the exact placement of the multi6 shim in the protocol stack.

The draft also discusses issues about receive side demultiplexing, which affects the packet format for data packets.

The approach assumes that there are mechanisms (specified in other drafts) which:

- can prevent redirection attacks [HBA]

- can prevent 3rd party DoS attacks [HBA, M6FUNC]

- can detect whether or not a peer supports the multi6 protocol [M6FUNC, M6DET]

- can explore all the locator pairs to find a working pair when the initial pair does not work [M6DET]

4. Locators as Upper-layer Identifiers

Central to this approach is to not introduce a new identifier name space but instead use one of the locators as the upper-layer ID, while allowing the locators used in the address fields to change over time in response to failures of using the original locator.

This implies that the ULID section is performed as today's default address selection as specified in [RFC 3484]. Underneath, and transparently, the multi6 shim selects working locator pairs with the initial locator pair being the ULID pair. When communication fails the shim can test and select alternate locators. A subsequent section discusses the issues when the selected ULID is not initially working hence there is a need to switch locators up front.

Using one of the locators as the ULID has certain benefits for applications which have long-lived session state, or performs callbacks or referrals, because both the FQDN and the 128-bit ULID work as handles for the applications. However, using a single 128-bit ULID doesn't provide seamless communication when that locator is unreachable. See [M6REFER] for further discussion of the application implications.

There has been some discussion of using non-routable locators, such as unique-local addresses, as ULIDs in a multihoming solution. While this approach doesn't currently specify all aspects of this, it is believed that the approach can be extended to handle such a case. For example, the protocol probably needs to handle ULIDs that are not initially reachable. Thus the same mechanism can handle ULIDs that are permanently unreachable. Note that the hard issues with ULIDs is how to perform the mappings between them and the locator sets. With routable ULIDs the AAAA resource record set provides this "mapping". Non-routable ULIDs would need similar mechanisms, which is probably feasible for unique local addresses based on prefixes that are centrally assigned.

5. Placement of the multi6 shim


```
                           -----------------------
                           | Transport Protocols |
                           -----------------------

              ------ ------- -------------- -------------    IP endpoint
              | AH | | ESP | | Frag/reass | | Dest opts |    sub-layer
              ------ ------- -------------- -------------

                        ---------------------
                        | multi6 shim layer |
                        ---------------------

                               ------                       IP routing
                               | IP |                       sub-layer
                               ------
```

Figure 1: Protocol stack

The proposal uses an multi6 shim layer between IP and the ULPs as shown in figure 1, in order to provide ULP independence.

Conceptually the multi6 shim layer behaves as if it is associated with an extension header, which would be ordered immediately after any hop-by-hop options in the packet. However, the amount of data that needs to be carried in an actual multi6 extension header is close to zero, thus it might not be necessary to add bytes to each packet. See section 9.

We refer to packets that at least conceptually have this extension header, i.e., packets that should be processed by the multi6 shim on the receiver, as "multi6 packets" (analogous to "ESP packets" or "TCP packets").

Layering AH and ESP above the multi6 shim means that IPsec can be made to be unaware of locator changes the same way that transport protocols can be unaware. Thus the IPsec security associations remain stable even though the locators are changing. Layering the fragmentation header above the multi6 shim makes reassembly robust in the case that there is broken multi-path routing which results in using different paths, hence potentially different source locators, for different fragments. Thus, effectively the multi6 shim layer is placed between the IP endpoint sublayer, which handles fragmentation, reassembly, and IPsec, and the IP routing sublayer, which on a host selects which default router to use etc.

Applications and upper layer protocols use ULIDs which the multi6 layer will map to/from different locators. The multi6 layer maintains state, called host-pair context, in order to perform this mapping. The mapping is performed consistently at the sender and the

receiver, thus from the perspective of the upper layer protocols, packets appear to be sent using ULIDs from end to end, even though the packets travel through the network containing locators in the IP address fields, and even though those locators might be changed by the transmitting multi6 shim layer.

The context state in this approach is maintained per remote ULID i.e.

approximately per peer host, and not at any finer granularity. It might make sense to merge the context state for multiple ULIDs assigned to the same peer host, but this is for further study.

```
   ----------------------------           ----------------------------
   | Sender A                 |           | Receiver B               |
   |                          |           |                          |
   |     ULP                  |           |     ULP                  |
   |      | src ULID(A)=L1(A) |           |      ^                   |
   |      | dst ULID(B)=L1(B) |           |      | src ULID(A)=L1(A) |
   |      v                   |           |      | dst ULID(B)=L1(B) |
   |   multi6 shim            |           |   multi6 shim            |
   |      | src L2(A)         |           |      ^                   |
   |      | dst L3(B)         |           |      | src L2(A)         |
   |      v                   |           |      | dst L3(B)         |
   |      IP                  |           |      IP                  |
   ----------------------------           ----------------------------
          |                                      ^
          -------- cloud with some routers -------
```

Figure 2: Mapping with changed locators.

The result of this consistent mapping is that there is no impact on the ULPs. In particular, there is no impact on pseudo-header checksums and connection identification.

Conceptually one could view this approach as if both ULIDs and locators are being present in every packet, but with a header compression mechanism applied that removes the need for the ULIDs once the state has been established. In order for the receiver to recreate a packet with the correct ULIDs there might be a need to include some "compression tag" in the data packets. This would serve to indicate the correct context to use for decompression when the locator pair in the packet is insufficient to uniquely identify the context.

6. Deferred Context Establishment

The protocol will use some context establishment exchange in order to setup multi6 state at the two endpoints. Similar to MAST [MAST] this initial exchange can be performed asynchronously with data packets flowing between the two hosts; until context state has been established at both ends the packets would flow just as for unmodified IPv6 hosts i.e., without the ability for the hosts to switch locators. This approach allows the hosts to have some local policy on when to attempt to establish multi6 state with a peer; perhaps based on the transport protocols and port numbers, or perhaps based on the number of packets that have flowed to/from the peer.

Once the initial exchange has completed there is host-pair context state at both hosts, and both ends know a set of locators for the peer that are acceptable as the source in received packets. This will trigger some verification of the set of locators, which is the subject of the security scheme.

7. Assumptions about the DNS

This approach assumes that hosts in multihomed sites have multiple AAAA records under a single name, in order to allow initial communication to try all the locators. For multi6 capable hosts, the content of those records are the locators (which also serve as ULIDs).

However, the approach does not assume that all the AAAA records for a given name refer to the same host. Instead the context establishment allows each host to pass its locators to the peer. This set could be either smaller or larger (or neither) than the AAAA record set.

The approach makes no assumption about the reverse tree since the approach does not use it. However, applications might rely on the reverse tree whether multi6 is used or not.

8. Protocol Walkthrough

8.1. Initial Context Establishment

Here is the sequence of events when A starts talking to B:

1. A looks up FQDN(B) in the DNS which returns a locator set which includes some locators for B. (The set could include locators for other hosts since e.g., www.example.com might include AAAA records for multiple hosts.) The application would typically try to connect using the first locator in the set i.e., ULID(B) = L1(B). The application is prepared to try the other locators should the first one fail.

2. The ULP creates "connection" state between ULID(A)=L1(A) and ULID(B) and sends the first packet down to the IP/multi6 shim layer on A. L1(A) was picked using regular source address selection mechanisms.

3. The packet passes through the multi6 layer, which has no state for ULID(B). A local policy will be used to determine when, if at all, to attempt to setup multi6 state with the peer. Until this state triggers packets pass back and forth between A and B as they do in unmodified IPv6 today.

   When the policy is triggers, which could be on either A or B, an initial context establishment takes place. This exchange might fail should the peer not support the multi6 protocol. If it succeeds it results in both ends receiving the locator sets from their respective peer, and the security mechanism provides some way to verify these sets.

At this point in time it is possible for the hosts to change to a different locator in the set. But until they have exhanged the locator sets, and probably until they rehome the context to use different locators, they continue sending and receiving IPv6 packets as before.

8.2. Locator Change

When a host detects that communication is no longer working it can try to switch to a different locator pair. A host might suspect that communication isn't working due to

- lack of positive advise from the ULP (akin to the NUD advise in [RFC 2461]

- negative advise from the ULP

- failure of some explicit multi6 "heartbeat" messages

- local indications such as the local locator becoming invalid [RFC 2462] or the interface being disabled

Given that each host knows the locator set for its peer, the host can just switch to using a different locator pair. It might make sense for the host to test the locator pair before using it for ULP traffic, both to verify that the locator pair is working and to verify that it is indeed the peer that is present at the other end; the latter to prevent 3rd party DoS attacks. Such testing needs to complete before using the locator as a destination in order to prevent 3rd party DoS attacks [M6THREATS].

8.3. Concurrent Context Establishment

Should both A and B attempt to contact each other at about the same time using the same ULIDs for each other, the context establishment should create a single host-pair context.

However, if different ULIDs are used this would result in two completely independent contexts between the two hosts following the basic content establishment above.

8.4. Handling Initial Locator Failures

Should not all locators be working when the communication is initiated some extra complexity arises, because the ULP has already been told which ULIDs to use. If the locators that where selected to be ULIDs are not working and the multi6 shim does not know of alternate locators, it has to choice than to have the application try a different ULID.

Thus the simplest approach is to always punt initial locator failures up the stack to the application. However, this might imply significant delays while transport protocol times out.

It is possible to optimize this case when the multi6 shim already has alternate locators for the peer. This might be the case when the two hosts have already communicated, and it might be possible to have the DNS resolver library provide alternate locators to the shim in the speculation that they might be useful. However, those are optimizations and not required for the protocol to work.

Should the multi6 shim know alternate locators for the peer, it needs to perform the multi6 protocol before upper layer protocol packets are exchanged. This means that the context establishment can not be deferred, and that there is a rehoming event, with the necessary security checks, before the first ULP packets can be successfully exchanged.

9. Demultiplexing of data packets in multi6 communications

The mechanisms for preserving established communications through outages that reside in the M6 shim layer manage the multiple addresses available in the multihomed node so that a reachable address is used in the communication. Since reachability may vary during the communication lifetime, different addresses may have to be used in order to keep packets flowing. However, the addresses presented by the M6 shim layer to the upper layer protocols must remain constant through the locator changes, so that received packets are recognized by the upper layer protocols as belonging to the established communication. In other words, in order to preserve established communications through outages, the M6 shim layer will use different locators for exchanging packets while presenting the same identifiers for the upper layer protocols. This means that upon the reception of an incoming packet with a pair of locators, the M6 shim layer will need to translate the received locators to the identifiers that are being used by the upper layer protocols in the particular communication. This operation is called demultiplexing.

For example, if a host has address A1 and A2 and starts communicating with a peer with addresses B1 and B2, then some communication (connections) might use the pair <A1, B1> as upper-layer identifiers and others might use e.g., <A2, B2>. Initially there are no failures so these address pairs are used as locators i.e. in the IP address fields in the packets on the wire. But when there is a failure the multi6 shim on A might decide to send packets that used <A1, B1> as upper-layer identifiers using <A2, B2> as the locators. In this case B needs to be able to rewrite the IP address field for some packets and not others, but the packets all have the same locator pair.

Either we must prevent this from happening, or provide some additional information to B so that it can tell which packets need to have the IP address fields rewritten.

In this section, we will analyze different approaches to perform the demultiplexing operation. The possible approaches can be classified into two categories: First, the approaches that prevent the existence of ambiguities on the demultiplexing operation i.e. each received

locator corresponds to one and only one ULP identifier. Second, the approaches that use a context tag to provide additional information to the receiver that indicates the identifiers that correspond to the locators contained in the packets.

9.1. Approaches preventing the existence of ambiguities

9.1.1. Pre-agreed identifiers

The simplest approach of this type is to designate one of the available addresses as the identifier to be used for all the communications while the remaining addresses will only be used as locators. This means that the upper layer protocols will only be aware of a single address, the one used as identifier, and all the remaining addresses that are used as locators will remain invisible to them. Consequently, only the address that is being used as identifier can be returned by the resolver to the applications. The addresses used as locators cannot be returned to the applications by the resolver. So, if no additional information about the role of the addresses is placed in the DNS, only the identifier-address can be published in the DNS. This configuration has reduced fault tolerance capabilities during the initial contact, since the initiator will have only one address available to reach the receiver. If the identifier address placed in the DNS is not reachable, the communication will fail. It would be possible to overcome this limitation by defining a new DNS record for storing information about address that can be only used as locators. If such record is defined, the initiator can use an alternative locator, even for initial contact, while still presenting the address designated as identifier to the upper layer protocols. However, this approach requires support form the initiator node, implying that only upgraded nodes will obtain improves fault tolerance while legacy nodes that don't support the new DNS record will still obtain reduced fault tolerance capabilities.

9.1.2. N-square addresses

In order to overcome the limitations presented by the previous scheme, it is possible to create additional addresses that have a pre-determined role. In this approach, each multihomed node that has n prefixes available, will create n^2 addresses, or in other words, the node will have n sets of n addresses each. Each set will contain one address per prefix. So, in each set, one address will be designated as identifier while the remaining addresses will be designated as locators. The addresses designated as identifiers will have different prefixes in the different sets. The result is that

there will be n addresses designated as identifiers, one per available prefix, and each identifier-address will have an associated set of n-1 addresses that can only be used as locators. The addresses designated as identifiers will be published in the DNS while the addresses used as locators must not be AAAA records in the DNS to prevent them from every being used as ULIDs. The applications will only have knowledge of the first ones, and only the M6 shim layer will deal with locators. The resulting configuration has full fault tolerance capabilities since n addresses (one per prefix) will be published in the DNS, allowing the usage of different addresses to make the initial contact.

9.2. Providing additional information to the receiver

When two nodes establish a multi6 enabled communication, a context is created at the M6 shim layers of each node. The context stores information about the addresses that are used as identifiers for the upper layer protocols and also about the locator set available for each node. In this approach, data packets carry a context tag that allows the receiver determine which is the context that has to be used to perform the demultiplexing operation. There are several ways to carry the context tag within the data packets. In this section we will explore the following options: the Flow Label, and an Extension Header.

9.2.1. Flow-label

A possible approach is to carry the context tag in the Flow Label field of the IPv6 header. This means that when a multi6 context is established, a Flow Label value is associated with this context. When a packet is received, the Flow Label value is used as a key to determine the context to be used for the demultiplexing operation, hence determining the identifiers that have to be presented to the upper layers. Because this approach overloads the Flow Label field, it is necessary to have an additional information to determine whether the Flow Label field is actually being used as a context tag or not. In other words, additional information is needed to identify multi6 packets from regular IPv6 packets. This is because, the same Flow Label value that is being used as context tag in multi6 enabled communication can be used for other purposes by a non-multi6 enabled host, resulting in two communications using the same Flow Label value. The result of this situation would be that packets of the non-multi6 enabled communication would be demultiplexed using the context associated to the Flow Label value carried in the packets. A possible approach to solve this issue it to use an additional bit to identify data packets that belong to multi6 capable communications and that have to be demultiplexed using the Flow Label value.

However, there are no obvious choices for that bit, since all bits of the IPv6 header are currently in use. A possibility would be to use new Next Header values to indicate that the packet belongs to a multi6 enabled communication and that the Flow Label carries context information as proposed in [NOID].

Another approach is to extend the context tag to include additional fields of the IPv6 header. The obvious choice would be to extend the context tag to the combination of Flow Label, Source Address and Destination Address. In this case, the context tag is composed of these three values and it will be used to identify the context. The limitation imposed by this approach is that all the potential source and destination addresses have to be known beforehand by the receiver in order to be recognized. This means that before sending packets with a new address, the sender has to inform the receiver about the new address.

9.2.2. Extension Header

Another approach is to define a new Extension Header to carry the context tag. This context tag is agreed between the involved parties during the multi6 protocol initial negotiation. Following data packets will be demultiplexed using the tag carried in the Extension Header. This seems a clean approach since it does not overload existing fields. However, it introduces additional overhead in the packet due to the additional header. The additional overhead introduced is 8 octets. However, it should be noted that the context tag is only required when an address other than the one used as identifier for upper layer protocols is contained in the packet. Packets carrying the addresses that have to be used as identifier for the upper layer protocols do not require a context tag, since the address contained in the packets is the address presented to the upper layers. This approach would reduce the overhead. On the other hand, this approach would cause changes in the available MTU, since packets that include the Extension Header will have an MTU 8 octets shorter.

9.3. Host-Pair Context

The host-pair context is established on each end of the communication when one of the endpoints performs the multi6 signaling (the 4-way handshake referred to in [M6FUNC].

This context is accessed differently in the transmit and receive paths. In the transmit path when the ULP passes down a packet the key to the context state is the tuple <ULID(local), ULID(peer)>; this key must identify at most one state record. In the receive path the context must be found based on what is in the packet, be it just the locators, or the locators plus some additional "context tag" as discussed above, or just a "context tag".

10. IPSEC INTERACTIONS

As specified, all of ESP, AH, and key management is layered above the multi6 layer. Thus they benefit from the stable ULIDs provided above the multi6 layer. This means the IPsec security associations are unaffected by switching locators.

The alternative would be to layer multi6 above IPsec, but that doesn't seem to provide any benefits and it would add the need to create different IPsec SAs when the locators change due to rehoming.

A result of layering multi6 above IPsec is that the multi6 protocol can potentially be used to redirect IPsec protected traffic as a selective DoS mechanism. If we somehow could require IPsec for the multi6 protocol packets when the ULP packets between the same hosts use IPsec, then we could prevent such attacks.

However, due to the richness in IPsec policy, this would be a bit tricky. If only some protocols or port numbers/selectors are to be protected by IPsec per a host's IPsec policy, then how would one determine whether multi6 traffic needs to be protected? Should one take the conservative approach that if any packets between the hosts/ULIDs need to be protected, then the multi6 traffic should also be protected?

For this to be useful both communicating hosts would need to make the same policy decisions, so if we are to take this path there would need to some standardization in this area.

11. OPEN ISSUES

Receive side demultiplexing issue as described above.

Is it possible to facilitate transition to multi6 using some "multi6 proxy" at site boundaries until all important hosts in a site have been upgraded to support multi6? Would would be the properties of such a proxy? Would it place any additional requirements on the protocol itself?

12. ACKNOWLEDGMENTS

This document was originally produced of a MULTI6 design team consisting of (in alphabetical order): Jari Arkko, Iljitsch van Beijnum, Marcelo Bagnulo Braun, Geoff Huston, Erik Nordmark, Margaret Wasserman, and Jukka Ylitalo.

The idea to use a set of locators and not inventing a new identifier name space, as well as using the DNS for verification of the locators, was first brought up by Tony Li.

13. REFERENCES

13.1. Normative References

[M6THREATS] Nordmark, E., and T. Li, "Threats relating to IPv6 multihoming solutions", draft-ietf-multi6-multihoming-threats-00.txt, July 2004.

[ADDR-ARCH] S. Deering, R. Hinden, Editors, "IP Version 6 Addressing Architecture", RFC 3513, April 2003.

[IPv6] S. Deering, R. Hinden, Editors, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2461.

[M6FUNC] Functional decomposition of the M6 protocol, draft-dt-multi6-functional-dec-00.txt

[HBA] Hash Based Addresses (HBA), draft-bagnulo-multi6dt-hba-00.txt

[M6DET] Jari Arkko, Failure Detection and Locator Selection in Multi6, draft-multi6dt-failure-detection-00.txt

13.2. Informative References

[NSRG] Lear, E., and R. Droms, "What's In A Name: Thoughts from the NSRG", draft-irtf-nsrg-report-09.txt (work in progress), March 2003.

[ULA] R. Hinden, and B. Haberman, Centrally Assigned Unique Local IPv6 Unicast Addresses, draft-ietf-ipv6-ula-central-00.txt

[MAST] D. Crocker, "MULTIPLE ADDRESS SERVICE FOR TRANSPORT (MAST): AN EXTENDED PROPOSAL", draft-crocker-mast-protocol-01.txt, October, 2003.

[RFC3041] T. Narten, R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, January 2001.

AUTHORS' ADDRESSES

Erik Nordmark
Sun Microsystems, Inc.
17 Network Circle
Mountain View, CA
USA

phone: +1 650 786 2921
fax: +1 650 786 5896
email: erik.nordmark@sun.com

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
EMail: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Intellectual Property Statement

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at

ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

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