Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 27, 2007

J. Arkko
Ericsson
I. van Beijnum

September 23, 2006

Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming

draft-ietf-shim6-failure-detection-06

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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

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

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

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

This Internet-Draft will expire on March 27, 2007.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document specifies how the level 3 multihoming shim protocol (SHIM6) detects failures between two communicating hosts. It also specifies an exploration protocol for switching to another pair of interfaces and/or addresses between the same hosts if a failure occurs and an operational pair can be found.

Table of Contents

1. Introduction 3
2. Requirements language 4
3. Definitions 5
3.1. Available Addresses 5
3.2. Locally Operational Addresses 6
3.3. Operational Address Pairs 6
3.4. Primary Address Pair 8
3.5. Current Address Pair 8
4. Protocol Overview 9
4.1. Failure Detection 9
4.2. Alternative Address Pair Exploration 11
4.3. Exploration Order 12
5. Protocol Definition 14
5.1. Keepalive Message 14
5.2. Probe Message 15
6. Behaviour 20
7. Example Protocol Runs 25
8. Protocol Constants 30
9. Security Considerations 31
10. IANA Considerations 33
11. References 34
11.1. Normative References 34
11.2. Informative References 34
Appendix A. Contributors 36
Appendix B. Acknowledgements 37
Authors´ Addresses 38
Intellectual Property and Copyright Statements 39

1. Introduction

The SHIM6 protocol [I-D.ietf-shim6-proto] extends IPv6 to support multihoming. It is an IP layer mechanism that hides multihoming from applications. A part of the SHIM6 solution involves detecting when a currently used pair of addresses (or interfaces) between two communication hosts has failed, and picking another pair when this occurs. We call the former failure detection, and the latter locator pair exploration.

This document specifies the mechanisms and protocol messages to achieve both failure detection and locator pair exploration. This part of the SHIM6 protocol is called the REAchability Protocol (REAP).

The document is structured as follows: Section 3 defines a set of useful terms, Section 4 gives an overview of REAP, and Section 5 specifies the message formats and behaviour in detail. Section 9 discusses the security considerations of REAP.

In this specification, we consider an address to be synonymous with a locator. Other parts of the SHIM6 protocol ensure that the different locators used by a node actually belong together. That is, REAP is not responsible for ensuring that it ends up with a legitimate locator.

2. Requirements language

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].

3. Definitions

This section defines terms useful for discussing failure detection and locator pair exploration.

3.1. Available Addresses

SHIM6 nodes need to be aware of what addresses they themselves have. If a node loses the address it is currently using for communications, another address must replace this address. And if a node loses an address that the node´s peer knows about, the peer must be informed. Similarly, when a node acquires a new address it may generally wish the peer to know about it.

Definition. Available address. An address is said to be available if the following conditions are fulfilled:

- The address has been assigned to an interface of the node.
- The address is valid in the sense of RFC 2461 [RFC2461].
- The address is not tentative in the sense of RFC 2462 [RFC2462]. In other words, the address assignment is complete so that communications can be started.

  Note that this explicitly allows an address to be optimistic in the sense of Optimistic DAD [RFC4429] even though implementations may prefer using other addresses as long as there is an alternative.
- The address is a global unicast, unique local address [RFC4193], or an unambiguous IPv6 link-local address. That is, it is not an IPv6 site-local address.

  Where IPv6 link-local addresses are used, their use needs to be unambiguous as follows. At most one link-local address may be used per node within the same connection between two peers.
- The address and interface is acceptable for use according to a local policy.

Available addresses are discovered and monitored through mechanisms outside the scope of SHIM6. SHIM6 implementations MUST be able to employ information provided by IPv6 Neighbor Discovery [RFC2461], Address Autoconfiguration [RFC2462], and DHCP [RFC3315] (when DHCP is implemented). This information includes the availability of a new address and status changes of existing addresses (such as when an address becomes invalid).

3.2. Locally Operational Addresses

Two different granularity levels are needed for failure detection. The coarser granularity is for individual addresses:

Definition. Locally Operational Address. An available address is said to be locally operational when its use is known to be possible locally: the interface is up, a default router (if needed) suitable for this address is known to be reachable, and no other local information points to the address being unusable.

Locally operational addresses are discovered and monitored through mechanisms outside the SHIM6 protocol. SHIM6 implementations MUST be able to employ information provided from Neighbor Unreachability Detection [RFC2461]. Implementations MAY also employ additional, link layer specific mechanisms.

> Note 1: A part of the problem in ensuring that an address is operational is making sure that after a change in link layer connectivity we are still connected to the same IP subnet. Mechanisms such as DNA CPL [I-D.ietf-dna-cpl] or DNAv6 [I-D.ietf-dna-protocol] can be used to ensure this.
>
> Note 2: In theory, it would also be possible for hosts to learn about routing failures for a particular selected source prefix, if only suitable protocols for this purpose existed. Some proposals in this space have been made, see, for instance [I-D.bagnulo-shim6-addr-selection] and [I-D.huitema-multi6-addr-selection], but none have been standardized to date.

3.3. Operational Address Pairs

The existence of locally operational addresses are not, however, a guarantee that communications can be established with the peer. A failure in the routing infrastructure can prevent packets from reaching their destination. For this reason we need the definition of a second level of granularity, for pairs of addresses:

Definition. Bidirectionally operational address pair. A pair of locally operational addresses are said to be an operational address pair when bidirectional connectivity can be shown between the addresses. That is, a packet sent with one of the addresses in the source field and the other in the destination field reaches the destination, and vice versa.

Unfortunately, there are scenarios where bidirectionally operational address pairs do not exist. For instance, ingress filtering or

network failures may result in one address pair being operational in one direction while another one is operational from the other direction. The following definition captures this general situation:

Definition. Unidirectionally operational address pair. A pair of locally operational addresses are said to be an unidirectionally operational address pair when packets sent with the first address as the source and the second address as the destination can be shown to reach the destination.

SHIM6 implementations MUST support the discovery of operational address pairs through the use of explicit rechability tests and Forced Bidirectional Communication (FBD), described later in this specification. In addition, implementations MAY employ the following additional mechanisms:

- Positive feedback from upper layer protocols. For instance, TCP can indicate to the IP layer that it is making progress. This is similar to how IPv6 Neighbor Unreachability Detection can in some cases be avoided when upper layers provide information about bidirectional connectivity [RFC2461].

  In the case of unidirectional connectivity, the upper layer protocol responses come back using another address pair, but show that the messages sent using the first address pair have been received.

- Negative feedback from upper layer protocols. It is conceivable that upper layer protocols give an indication of a problem to the multihoming layer. For instance, TCP could indicate that there´s either congestion or lack of connectivity in the path because it is not getting ACKs.

- ICMP error messages. Given the ease of spoofing ICMP messages, one should be careful to not trust these blindly, however. Our suggestion is to use ICMP error messages only as a hint to perform an explicit reachability test or move an address pair to a lower place in the list of address pairs to be probed, but not as a reason to disrupt ongoing communications without other indications of problems. The situation may be different when certain verifications of the ICMP messages are being performed, as explained by Gont in [I-D.ietf-tcpm-icmp-attacks]. These verifications can ensure that (practically) only on-path attackers can spoof the messages.

Note SHIM6 needs to perform a return routability test of an address before it is taken into use. The purpose of this test is to ensure that fraudulent peers do not trick others into redirecting traffic

streams onto innocent victims. For a discussion of such attacks, see Aura et al [AURA02]. The test can at the same time work as a means to ensure that an address pair is operational, as discussed in Section 4.2.

3.4. Primary Address Pair

The primary address pair consists of the ULID addresses that upper layer protocols use in their interaction with the SHIM6 layer. Use of the primary address pair means that the communication is compatible with regular non-SHIM6 communication and no context ID needs to be present.

3.5. Current Address Pair

SHIM6 needs to avoid sending packets concurrently over multiple paths, because congestion control in commonly used transport protocols is based upon a notion of a single path. While routing can introduce path changes as well and transport protocols have means to deal with this, frequent changes will cause problems. Efficient congestion control over multible paths is a considered research at the time this specification is written.

For these reasons it is necessary to choose a particular pair of addresses as the current address pair which is used until problems occur, at least for the same session.

A current address pair need not be operational at all times. If there is no traffic to send, we may not know if the primary address pair is operational. Nevertheless, it makes sense to assume that the address pair that worked in some time ago continues to be operational for new communications as well.

4. Protocol Overview

This section discusses the design of the reachability detection and address pair exploration mechanisms, and gives on overview of the REAP protocol.

Exploring the full set of communication options between two hosts that both have two or more addresses is an expensive operation as the number of combinations to be explored increases very quickly with the number of addresses. For instance, with two addresses on both sides, there are four possible address pairs. Since we can´t assume that reachability in one direction automatically means reachability for the complement pair in the other direction, the total number of two- way combinations is eight. (Combinations = nA * nB * 2.)

An important observation in multihoming is that failures are relatively infrequent, so that an operational pair that worked a few seconds ago is very likely to be still operational. So it makes sense to have a light-weight protocol that confirms existing reachability, and only invoke heavier exploration when a there is a suspected failure.

4.1. Failure Detection

Failure detection consists of three parts: tracking local information, tracking remote peer status, and finally verifying reachability. Tracking local information consists of using, for instance, reachability information about the local router as an input. Nodes SHOULD employ techniques listed in Section 3.1 and Section 3.2 to be track the local situation. It is also necessary to track remote address information from the peer. For instance, if the peer´s currently used address is no longer in use, mechanism to relay that information is needed. The Update message in the SHIM6 protocol is used for this purpose [I-D.ietf-shim6-proto]. Finally, when the local and remote information indicates that communication should be possible and there are upper layer packets to be sent, reachability verification is necessary to ensure that the peers actually have an operational pair.

A technique called Forced Bidirectional Detection (FBD, originally defined in an earlier SHIM6 document [I-D.ietf-shim6-reach-detect]) is employed for the reachability verification. Reachability for the currently used address pair in a shim context is determined by making sure that whenever there is data traffic in one direction, there is also traffic in the other direction. This can be data traffic as well, but also transport layer acknowledgments or a REAP reachability keepalive if there is no other traffic. This way, it is no longer possible to have traffic in only one direction, so whenever there is

data traffi c going out, but there are no return packets, there must be a failure, so the full exploration mechanism is started.

A more detailed description of the current pair reachability evaluation mechanism:

1. To avoid the other side from concluding there is a reachability failure, it´ s necessary for a host implementing the failure detection mechanism to generate periodic keepalives when there is no other traffi c.

   FBD works by generating REAP keepalives if the node is receiving packets from its peer but not sending any of its own. The keepalives are sent at certain intervals so that the other side knows there is a reachability problem when it doesn´ t receive any incoming packets for 10 seconds, the Keepalive Timeout. (Mechanisms to negotiate an alternative Keepalive Timeout may be provided in the future.)

   The interval after which keepalives are sent is named Keepalive Interval. This document doesn´ t specify a value for Keepalive Interval, but recognizes that an often used approach is sending keepalives at three times the timeout interval, which would be 3 seconds here, and suggest a possible alternative of 4 seconds so that two keepalives are generated and have time to reach the correspondent. An upper bound would be 8 seconds, so that one keepalive has time to reach the other side, assuming a maximum one-way delay of 2 seconds.

2. Whenever outgoing data packets are generated, a timer is started to reflect the requirement that the peer should generate return traffi c from data packets.

   For the purposes of this specifi cation, "data packet" refers to any packet that is part of a shim context, including both upper layer protocol packets and SHIM6 protocol messages except those defi ned in this specifi cation.

3. Whenever incoming data packets are received, the timer associated with the return traffi c from the peer is stopped, and another timer is started to reflect the requirement for this node to generate return traffi c.

4. The reception of a REAP keepalive packet leads to stopping the timer associated with the return traffi c from the peer.

5. Keepalive Interval seconds after the last data packet has been received for a context, and if no other packet has been sent

within this context since the data packet has been received, a REAP keepalive packet is generated for the context in question and transmitted to the correspondent. A host may send the keepalive sooner than Keepalive Interval seconds if implementation considerations warrant this, but should take care to avoid sending keepalives at an excessive rate. After sending a single keepalive message, no additional keepalive messages are sent until a data packet is received within this shim context. Keepalives are not sent at all when a data packet was sent since the last received data packet.

6. Send Timeout seconds (10 seconds; see Section 8) after the transmission of a data packet with no return traffic on this context, a full reachability exploration is started.

Note that the above timeout values are suggestions to be used as defaults. Experience from the deployment of the SHIM6 protocol is needed in order to determine what values are most suitable. The setting of these values is also related to various parameters in transport protocols, such as TCP keepalive interval.

4.2. Alternative Address Pair Exploration

As explained in previous section, the currently used address pair may become invalid either through one of the addresses being becoming unavailable or inoperational, or the pair itself being declared inoperational. An exploration process attempts to find another operational pair so that communications can resume.

What makes this process hard is the requirement to support unidirectionally operational address pairs. It is insufficient to probe address pairs by a simple request - response protocol. Instead, the party that first detects the problem starts a process where it tries each of the different address pairs in turn by sending a message to its peer. These messages carry information about the state of connectivity between the peers, such as whether the sender has seen any traffic from the peer recently. When the peer receives a message that indicates a problem, it assists the process by starting its own parallel exploration to the other direction, again sending information about the recently received payload traffic or signaling messages.

Specifically, when A decides that it needs to explore for an alternative address pair to B, it will initiate a set of Probe messages, in sequence, until it gets an Probe message from B indicating that (a) B has received one of A´ s messages and, obviously, (b) that B´ s Probe message gets back to A. B uses the same algorithm, but starts the process from the reception of the first

Probe message from A.

Upon changing to a new address pair, the network path traversed most likely has changed, so that the ULP SHOULD be informed. This can be a signal for the ULP to adapt due to the change in path so that, for example, TCP could initiate a slow start procedure, although it´ s likely that the circumstances that led to the selection of a new path already caused enough packet loss to trigger slow start.

Similarly, one can also envision that applications would be able to tell the IP or transport layer that the current connection in unsatisfactory and an exploration for a better one would be desirable. This would require an inter-layer communication mechanism to be developed, however. In any case, this is another issue that we treat as being outside the scope of pure address exploration.

REAP is designed to support failure recovery even in the case of having only unidirectionally operational address pairs. However, due to security concerns discussed in Section 9, the exploration process can typically be run only for a session that has already been established. Specifically, while REAP would in theory be capable of exploration even during connection establishment, its use within the SHIM6 protocol does not allow this.

4.3. Exploration Order

The exploration process assumes an ability to choose address pairs for testing, in some sequence. This process may result in a combinatorial explosion when there are many addresses on both sides, but a back-off procedure is employed to avoid a "signaling storm".

Nodes first consult the RFC 3484 default address selection rules [RFC3484] Section 4 rules to determine what combinations of addresses are allowed from a local point of view, as this reduces the search space. RFC 3484 also provides a priority ordering among different address pairs, making the search possibly faster. (Additional mechanisms may be defined in the future for arriving at an initial ordering of address pairs before testing starts [I-D.ietf-shim6-locator-pair-selection].) Nodes may also use local information, such as known quality of service parameters or interface types to determine what addresses are preferred over others, and try pairs containing such addresses first. The SHIM6 protocol also carries preference information in its messages.

> Discussion note: The preferences may either be learned dynamically or be configured. It is believed, however, that dynamic learning based purely on the multihoming protocol is too hard and not the

task this layer should do. Solutions where multiple protocols share their information in a common pool of locators could provide this information from transport protocols, however.

Out of the set of possible candidate address pairs, nodes SHOULD attempt to test through all of them until an operational pair is found, and retrying the process as is necessary. However, all nodes MUST perform this process sequentially and with exponential back-off. This sequential process is necessary in order to avoid a "signaling storm" when an outage occurs (particularly for a complete site). However, it also limits the number of addresses that can in practice be used for multihoming, considering that transport and application layer protocols will fail if the switch to a new address pair takes too long.

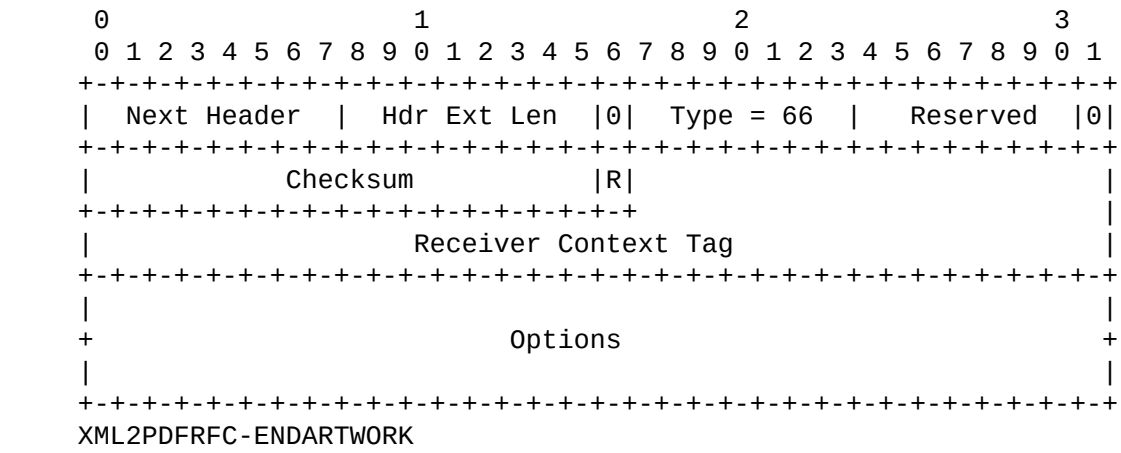
Section 8 suggests default values for the timers associated with the exploration process. The value Initial Probe Timeout (0.5 seconds) specifies the interval between initial attempts to send probes; Number of Initial Probes (4) specifies how many initial probes can be sent before the exponential backoff procedure needs to be employed. This process increases the time between every probe if there is no response. Typically, each increase doubles the time but this specification does not mandate a particular increase.

Finally, Max Probe Timeout (60 seconds) specifies a limit beyond which the probe interval may not grow. If the exploration process reaches this interval, it will continue sending at this rate until a suitable response is triggered or the SHIM6 context is garbage collected, because upper layer protocols using the SHIM6 context in question are no longer attempting to send packets. Reaching the Max Probe Timeout may also serve as a hint to the garbage collection process that the context is no longer usable.

5. Protocol Definition

5.1. Keepalive Message

The format of the keepalive message is as follows:



XML2PDFRFC-ENDARTWORK

Next Header, Hdr Ext Len, 0, 0, Checksum

These are as specified in Section 5.3 of the SHIM6 protocol description [I-D.ietf-shim6-proto].

Type

This field identifies the Probe message and MUST be set to 66 (Keepalive).

Reserved

This is a 7-bit field reserved for future use. It is set to zero on transmit, and MUST be ignored on receipt.

R

This is a 1-bit field reserved for future use. It is set to zero on transmit, and MUST be ignored on receipt.

Receiver Context Tag

This is a 47-bit field for the Context Tag the receiver has allocated for the context.

Options

This MAY contain one or more SHIM6 options.The inclusion of the latter options is not necessary, however, as there are currently no defined options that are useful in a Keepalive message. These options are provided only for future extensibility reasons.

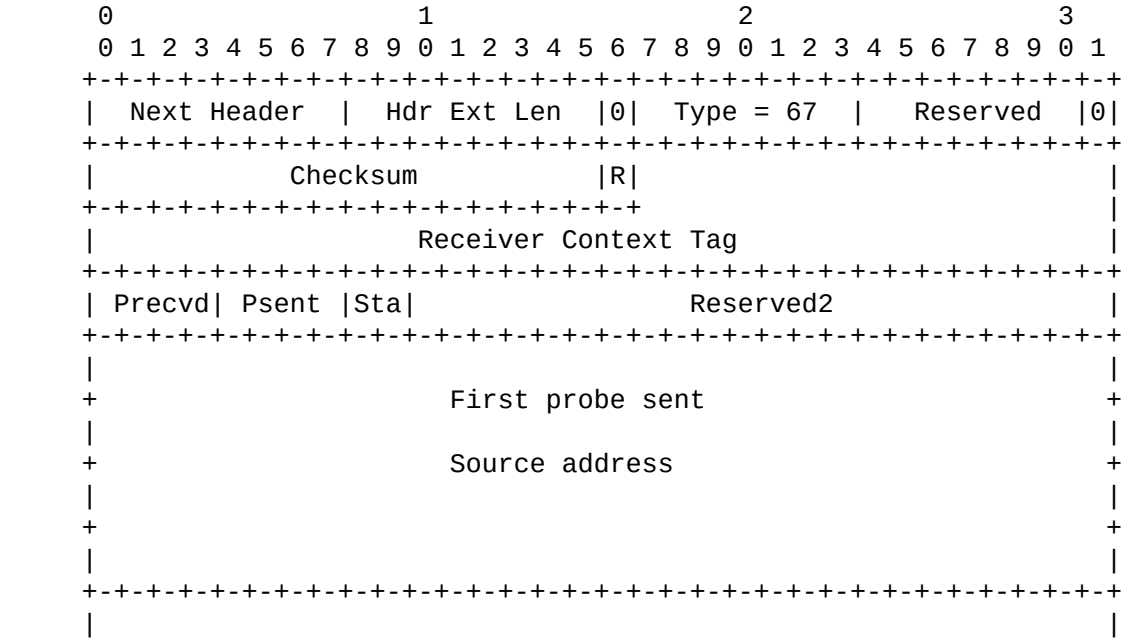
A valid message conforms to the format above, has a Receiver Context Tag that matches to context known by the receiver, is valid shim control message as defined in Section 12.2 of the SHIM6 protocol description [I-D.ietf-shim6-proto], and its shim context state is ESTABLISHED. The receiver processes a valid message by inspecting its options, and executing any actions specified for such options.

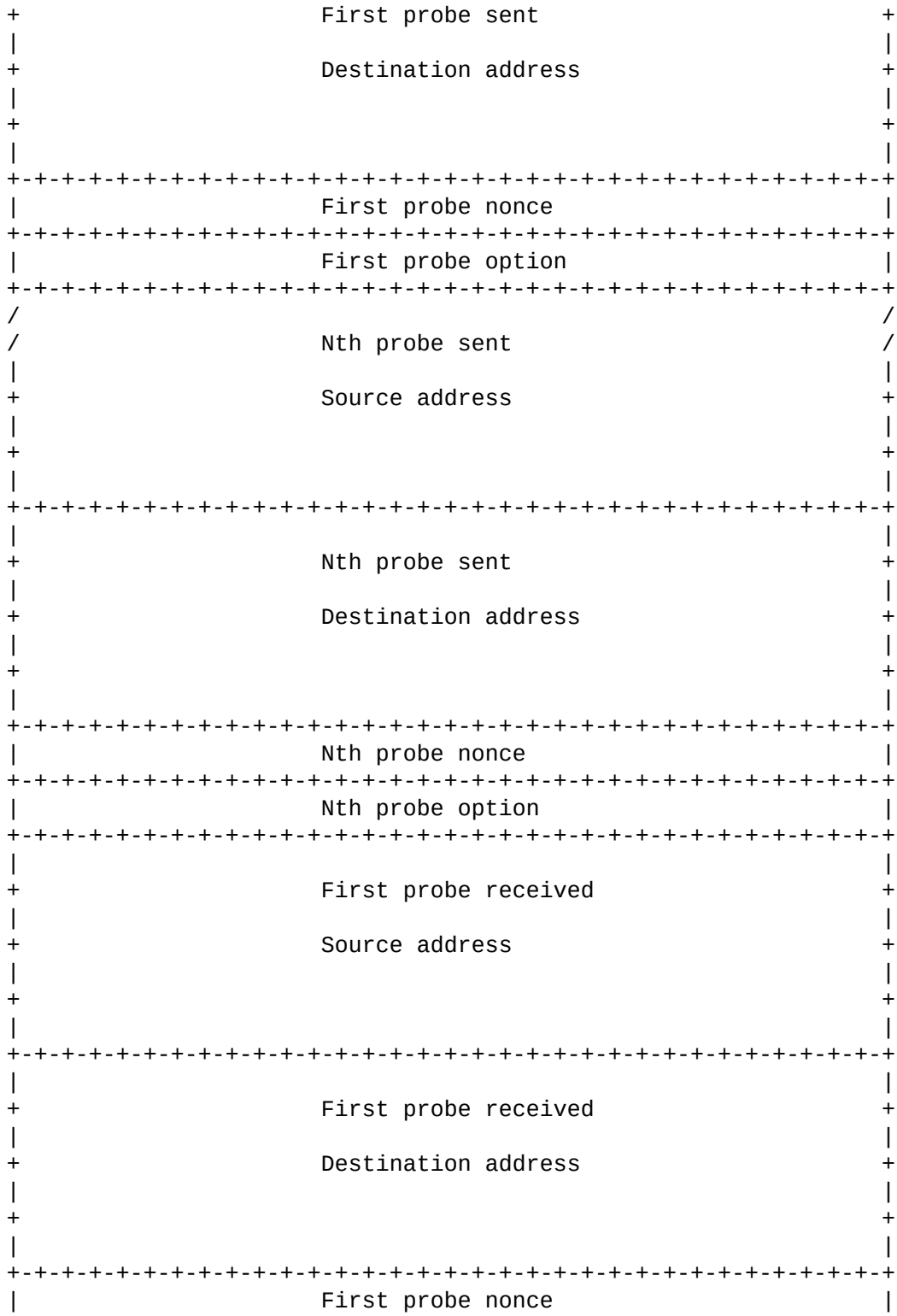
Discussion: It may appear prudent to include additional fields that would provide at least a basic level of security, but since data packets also indicate ongoing reachability, just as keepalives, and those packets don´t have such fields, there is little or no reason to include them in a keepalive.

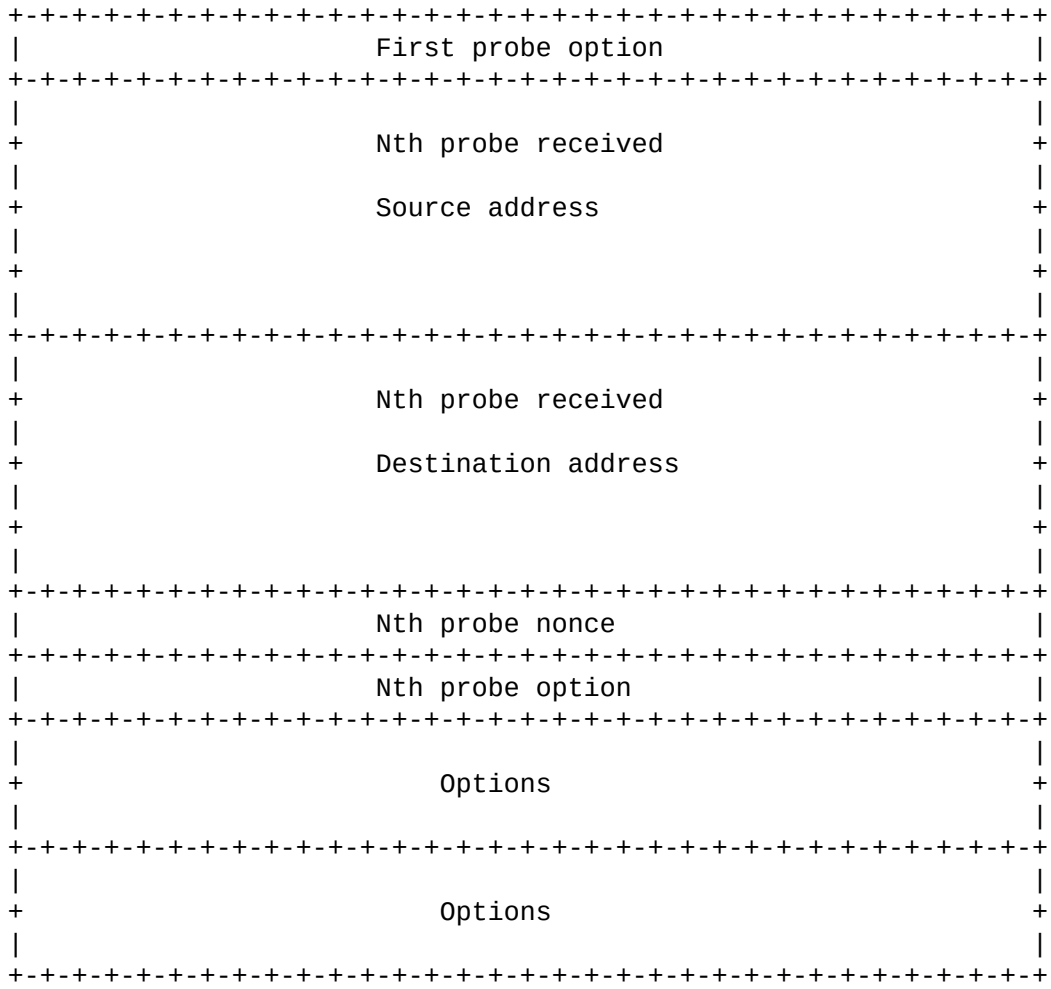
The processing rules for this message are the given in more detail in Section 6.

5.2. Probe Message

This message performs REAP exploration. Its format is as follows:




First probe sent
Destination address
First probe nonce
First probe option
Nth probe sent
Source address
Nth probe sent
Destination address
Nth probe nonce
Nth probe option
First probe received
Source address
First probe received
Destination address
First probe nonce



XML2PDFRFC-ENDARTWORK

Next Header, Hdr Ext Len, 0, 0, Checksum

These are as specified in Section 5.3 of the SHIM6 protocol description [I-D.ietf-shim6-proto].

Type

This field identifies the Probe message and MUST be set to 67 (Probe).

Reserved

This is a 7-bit field reserved for future use. It is set to zero on transmit, and MUST be ignored on receipt.

R

This is a 1-bit field reserved for future use. It is set to zero on transmit, and MUST be ignored on receipt.

Receiver Context Tag

This is a 47-bit field for the Context Tag the receiver has allocated for the context.

Psent

This is a 4-bit field that indicates the number of sent probes included in this probe message. The first set of probe fields pertains to the current message and MUST be present, so the minimum value for this field is 1. Additional sent probe fields are copies of the same fields sent in (recent) earlier probes and may be included or omitted as per any logic employed by the implementation.

Precvd

This is a 4-bit field that indicates the number of received probes included in this probe messsage. Received probe fields are copies of the same fields received in (recent) earlier probes and may be included or omitted as per any logic employed by the implementation.

The fields probe source, probe destination, probe nonce and probe option may be repeated, depending on the value of Psent and Preceived.

Sta (State)

This 2-bit State field is used to inform the peer about the state of the sender. It has three legal values:

0 (Operational) implies that the sender both (a) believes it has no problem communicating and (b) believes that the recipient also has no problem communicating.

1 (Exploring) implies that the sender has a problem communicating with the recipient, e.g., it has not seen any traffic from the recipient even when it expected some.

2 (ExploringOk) implies that the sender believes it has no problem communicating, but that the recipient either has a problem or has not yet confirmed the sender that the problem has been solved.

Reserved2

MUST be set to 0 upon transmission and MUST be ignored upon reception.

Probe source

This 128-bit field contains the source IPv6 address used to send the probe.

Probe destination

This 128-bit field contains the destination IPv6 address used to send the probe.

Probe nonce

This is a 32-bit field that is initialized by the sender with a value that allows it to determine which sent probes a received probe correlates with. It is highly recommeded that the nonce field is at least moderately hard to guess so that even on-path attackers can´t deduce the next nonce value that will be used. This value SHOULD be generated using a random number generator that is known to have good randomness properties as outlined in RFC 1750 [RFC1750].

Probe option

This is a 32-bit field with no fixed meaning. The probe option field is copied back with no changes. Future flags may define a use for this field.

Discussion: One potential use of this field relates to communicating delays between reception of a probe and transmission of a reply to it.

Options

For future extensions.

6. Behaviour

The required behaviour of REAP nodes is specified below in the form of a state machine. The externally observable behaviour of an implementation MUST conform to this state machine, but there is no requirement that the implementation actually employs a state machine. Intermixed with the following description we also provide a state machine description in a tabular form. That form is only informational, however.

On a given context with a given peer, the node can be in one of three states: Operational, Exploring, or ExploringOK. In the Operational state the underlying address pairs are assumed to be operational. In the Exploring state this node has observed a problem and has currently not seen any traffic from the peer. Finally, in the ExploringOK state this node sees traffic from the peer, but peer may not yet see any traffic from this node so that the exploration process needs to continue.

The node maintains also the Send timer (Send Timeout seconds) and Keepalive timer (Keepalive Timeout seconds). The Send timer reflects the requirement that when this node sends a payload packet there should be some return traffic (either payload packets or Keepalive messages) within Send Timeout seconds. The Keepalive timer reflects the requirement that when this node receives a payload packet there should a similar response towards the peer. The Keepalive timer is only used within the Operational state, and the Send timer in the Operational and ExploringOK states. No timer is running in the Exploring state.

Upon the reception of a payload packet in the Operational state, the node starts the Keepalive timer if it is not yet running, and stops the Send timer if it was running. If the node is in the Exploring state it transitions to the ExploringOK state, sends a Probe message, and starts the Send timer. In the ExploringOK state the node stops the Send timer if it was running, but does not do anything else. The reception of SHIM6 control messages other than the Keepalive and Probe messages are treated similarly with payload packets.

When sending a Probe message, the State field MUST be set to a value that matches the conceptual state of the sender after sending the Probe.

```
  1. EVENT: Incoming payload packet
  ================================

  Operational           Exploring               ExploringOk
  ------------------------------------------------------------
  STOP Send;            SEND Probe ExploringOk; STOP Send
  START Keepalive       START Send;
                        GOTO ExploringOk
XML2PDFRFC-ENDARTWORK
```

Upon sending a payload packet in the Operational state, the node stops the Keepalive timer if it was running and starts the Send timer if it was not running. In the Exploring state there is no effect, and in the ExploringOK state the node simply starts the Send timer if it was not yet running. (The sending of SHIM6 control messages is again treated similarly here.)

```
  2. EVENT: Outgoing payload packet
  =================================

  Operational             Exploring             ExploringOk
  ------------------------------------------------------------
  START Send;             -                     START Send
  STOP Keepalive
XML2PDFRFC-ENDARTWORK
```

Upon a timeout on the Keepalive timer the node sends a Keepalive message. This can only happen in the Operational state.

```
  3. EVENT: Keepalive timeout

  Operational             Exploring             ExploringOk
  ------------------------------------------------------------
  SEND Keepalive          -                     -
XML2PDFRFC-ENDARTWORK
```

Upon a timeout on the Send timer, the node enters the Exploring state, sends a Probe, and stops the Keepalive timer if it was running.

```
  4. EVENT: Send timeout
  ======================

  Operational              Exploring              ExploringOk
  ------------------------------------------------------------
  SEND Probe Exploring;    -                      SEND Probe Exploring;
  STOP Keepalive;                                 GOTO Exploring
  GOTO Exploring
XML2PDFRFC-ENDARTWORK
```

While in the Exploring state the node keeps retransmitting its Probe messages to different (or same) addresses as defined in Section 4.3. A similar process is employed in the ExploringOk state, except that upon such retransmission the Send timer is started if it was not running already.

```
  5. EVENT: Retransmission
  ========================

  Operational             Exploring              ExploringOk
  ------------------------------------------------------------
  -                       SEND Probe Exploring   SEND Probe ExploringOk
                                                 START Send
XML2PDFRFC-ENDARTWORK
```

Upon the reception of a Keepalive message in the Operational state, the node stops the Send timer, if it was running. If the node is in the Exploring state it transitions to the ExploringOK state, sends a Probe message, and starts the Send timer. In the ExploringOK state the Send timer is stopped, if it was running.

```
    6. EVENT: Reception of the Keepalive message
    ============================================

    Operational            Exploring               ExploringOk
    ------------------------------------------------------------
    STOP Send              SEND Probe ExploringOk; STOP Send
                           START Send;
                           GOTO ExploringOk
  XML2PDFRFC-ENDARTWORK
```

Upon receiving a Probe with State set to Exploring, the node enters the ExploringOK state, sends a Probe, stops the Keepalive timer if it was running, and restarts the Send timer.

```
  7. EVENT: Reception of the Probe message State=Exploring
  ========================================================

  Operational             Exploring               ExploringOk
  ---------------------------------------------------------
  SEND Probe ExploringOk; SEND Probe ExploringOk; SEND Probe
  STOP Keepalive;         START Send;                  ExploringOk;
  RESTART Send;           GOTO ExploringOk        RESTART Send
  GOTO ExploringOk
XML2PDFRFC-ENDARTWORK
```

Upon the reception of a Probe message with State set to ExploringOk, the node sends a Probe message, restarts the Send timer, stops the Keepalive timer if it was running, and transitions to the Operational state.

```
  8. EVENT: Reception of the Probe message State=ExploringOk
  ==========================================================

  Operational             Exploring               ExploringOk
  -----------------------------------------------------------
  SEND Probe Operational; SEND Probe Operational; SEND Probe
  RESTART Send;           RESTART Send;                Operational;
  STOP Keepalive          GOTO Operational        RESTART Send;
                                                  GOTO Operational
XML2PDFRFC-ENDARTWORK
```

Upon the reception of a Probe message with State set to Operational, the node stops the Send timer if it was running, starts the Keepalive timer if it was not yet running, and transitions to the Operational state.

> Note: This terminates the exploration process when both parties are happy and know that their peer is happy as well.

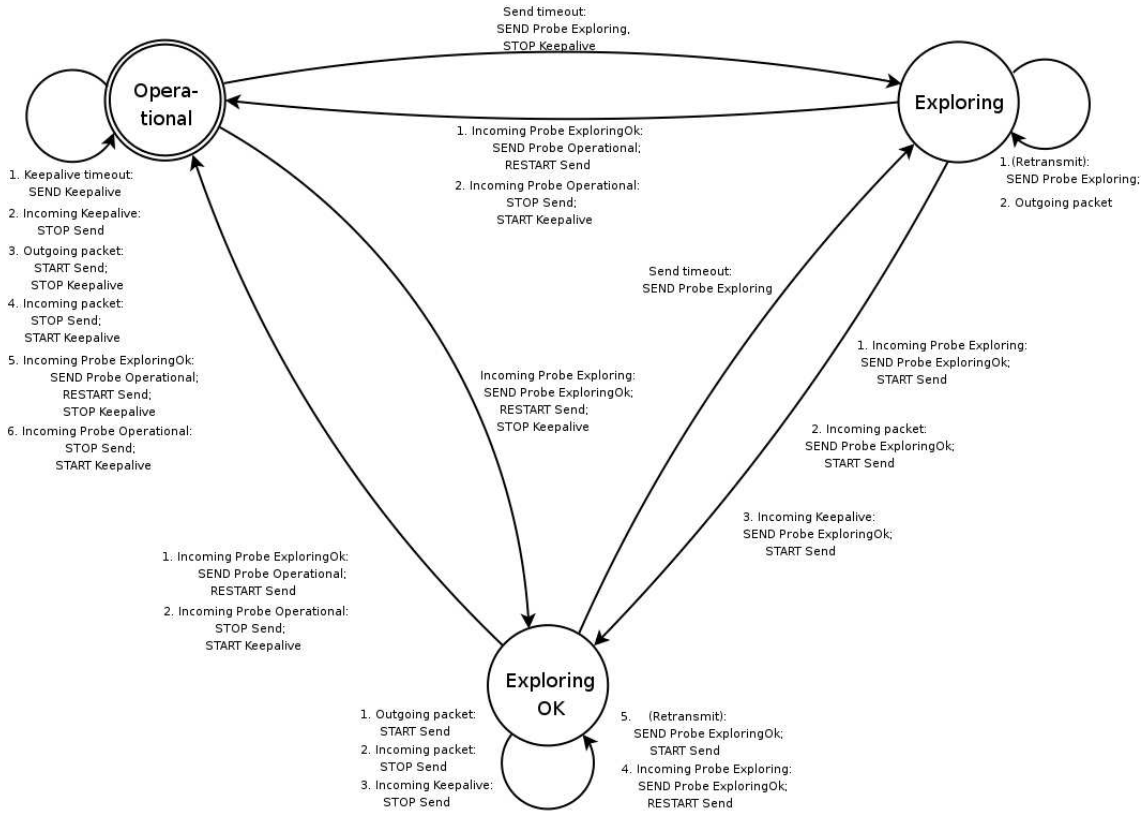
```
  9. EVENT: Reception of the Probe message State=Operational
  ==========================================================

  Operational              Exploring              ExploringOk
  -----------------------------------------------------------
  STOP Send                STOP Send;             STOP Send;
  START Keepalive          START Keepalive        START Keepalive
                           GOTO Operational       GOTO Operational
XML2PDFRFC-ENDARTWORK
```

The reachability detection and exploration process has no effect on

payload communications until a new operational address pairs have actually been confirmed. Prior to that the payload packets continue to be sent to the previously used addresses.

In the PDF version of this specification, an informational drawing illustrates the state machine. Where the text and the drawing differ, the text takes precedence.



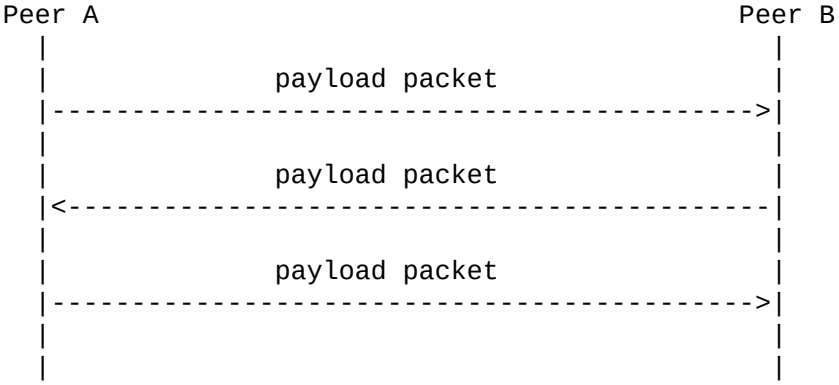
EXAMPLE 1: No communications

XML2PDFRFC-ENDARTWORK

Our second example involves an active connection with bidirectional

payload packet flows. Here the reception of data from the peer is taken as an indication of reachability, so again there are no extra packes:

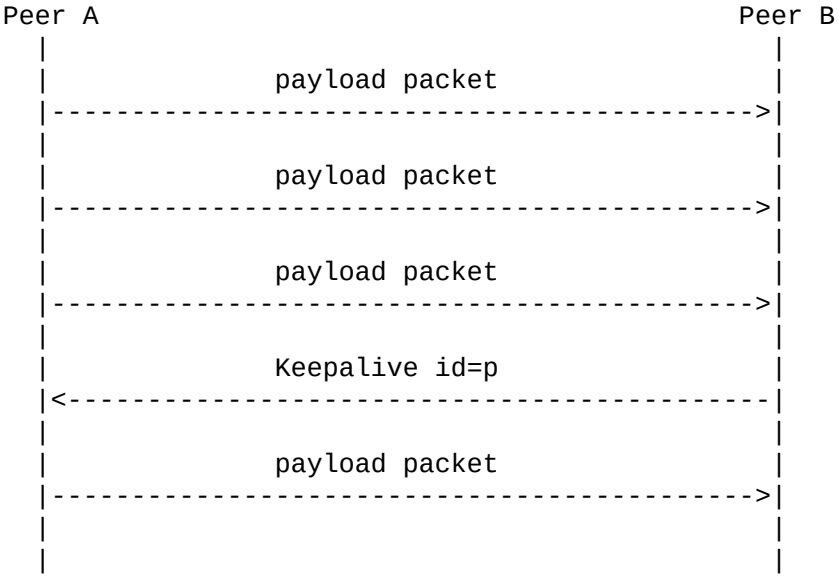
EXAMPLE 2: Bidirectional communications



XML2PDFRFC-ENDARTWORK

The third example is the first one that involves an actual REAP message. Here the hosts communicate in just one direction, so REAP messages are needed to indicate to the peer that sends payload packets that its packets are getting through:

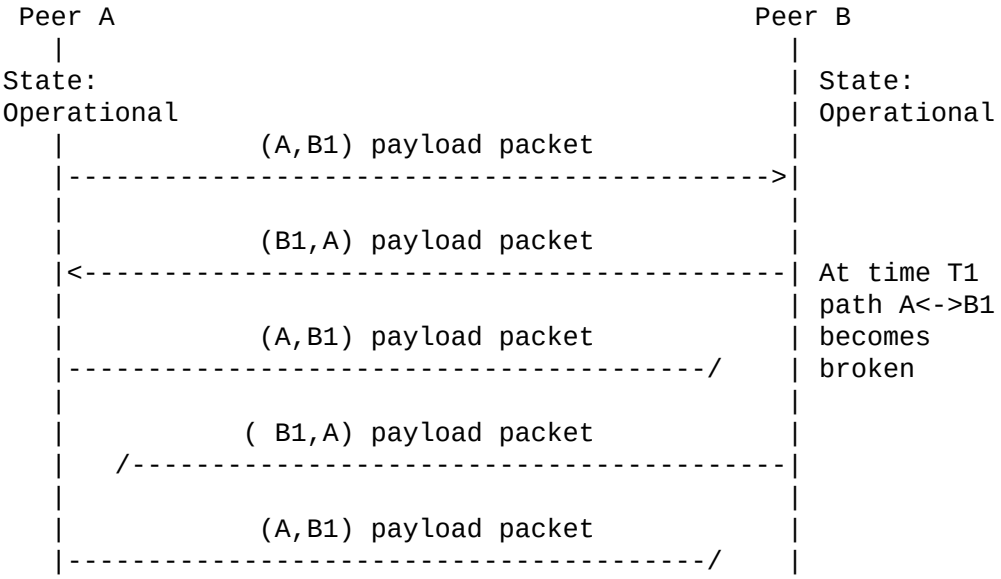
EXAMPLE 3: Unidirectional communications

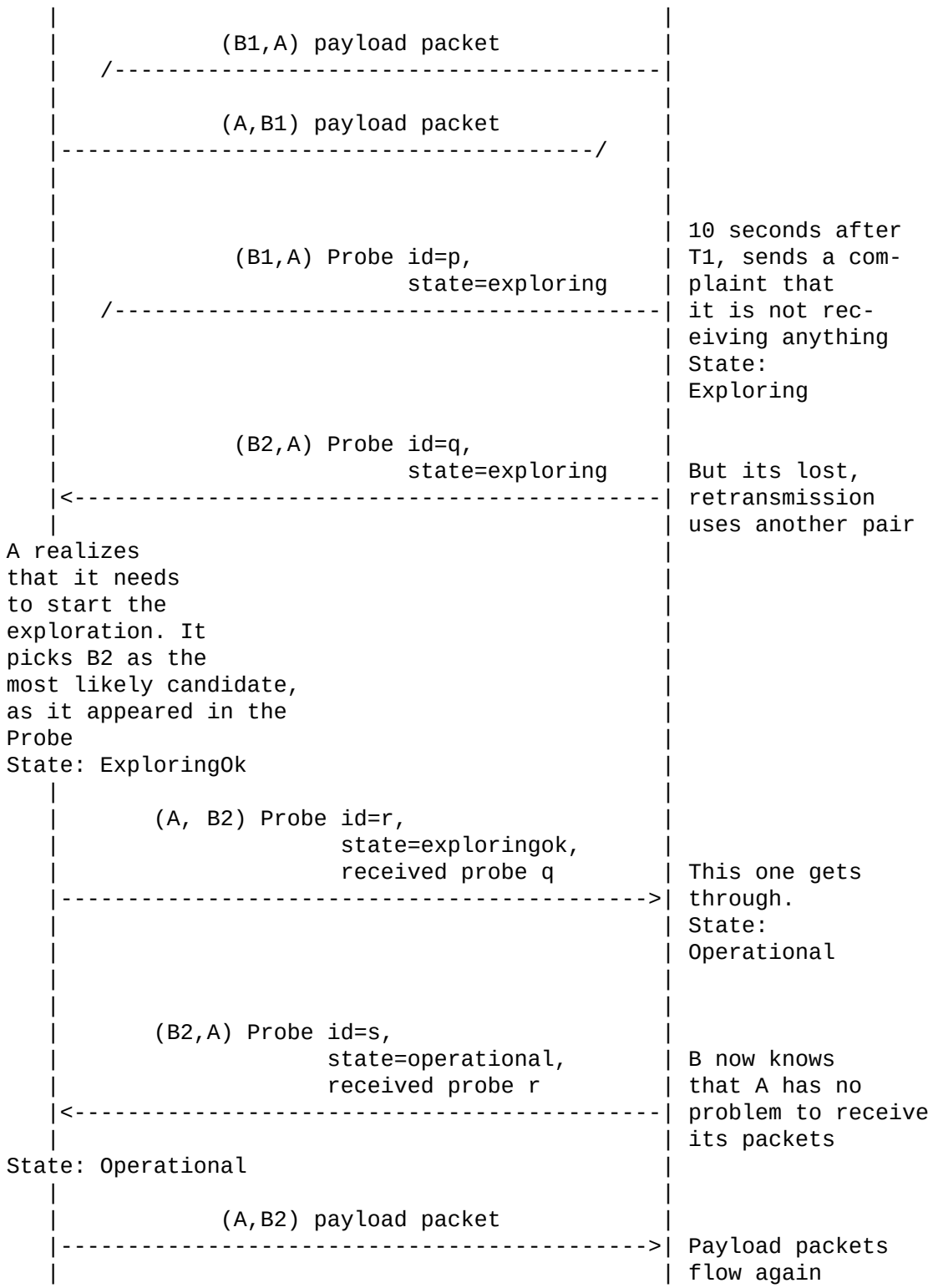


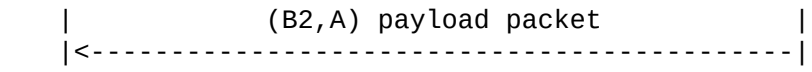
XML2PDFRFC-ENDARTWORK

The next example involves a failure scenario. Here A has addresses A and B has addresses B1 and B2. The currently used address pairs are (A, B1) and (B1, A). All connections via B1 become broken, which leads to an exploration process:

EXAMPLE 4: Failure scenario



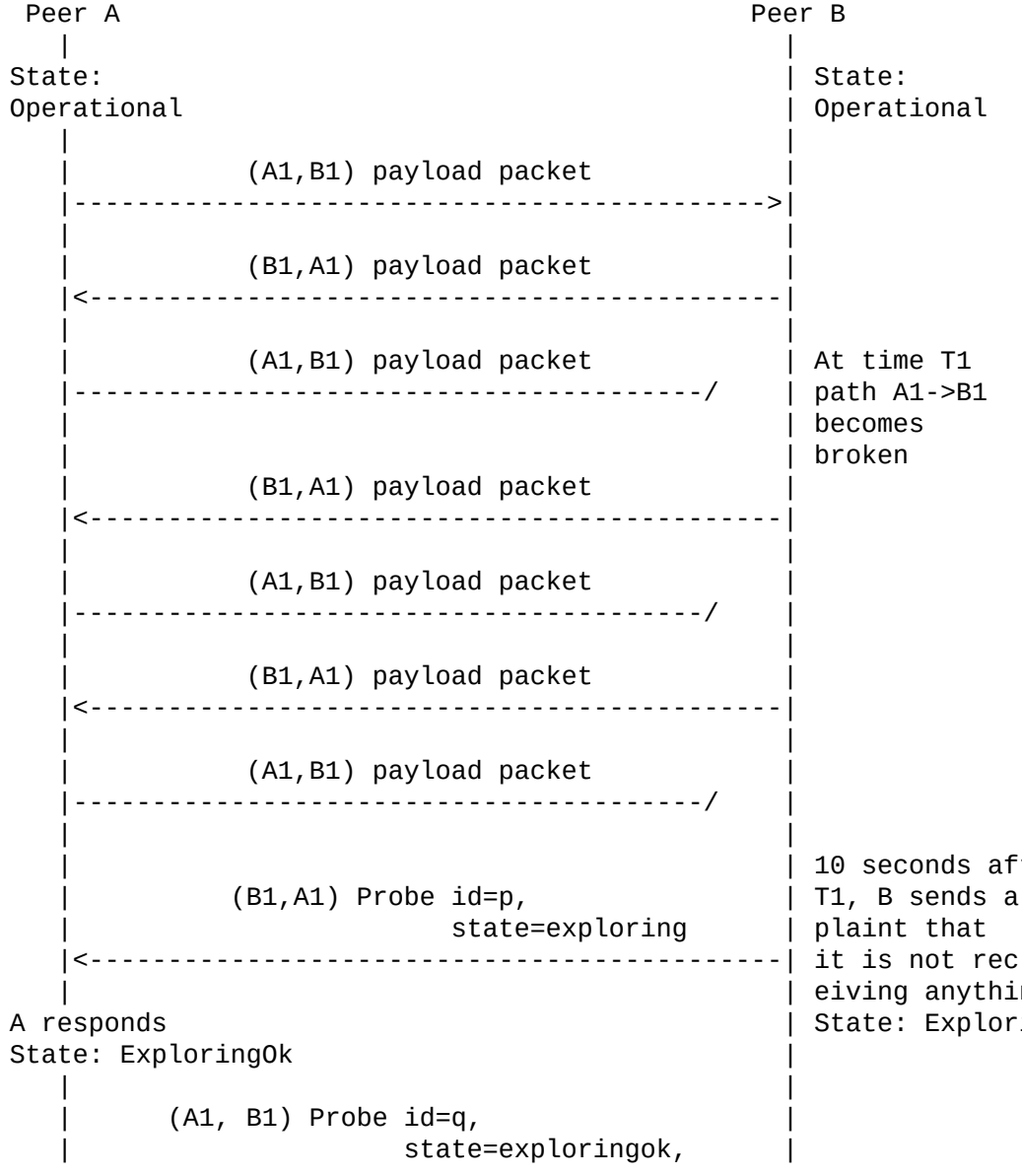

(B1,A) payload packet
(A,B1) payload packet
10 seconds after T1, sends a com-plaint that it is not rec-eiving anything
State: Exploring
(B1,A) Probe id=p, state=exploring
(B2,A) Probe id=q, state=exploring
But its lost, retransmission uses another pair
A realizes that it needs to start the exploration. It picks B2 as the most likely candidate, as it appeared in the Probe
State: ExploringOk
(A, B2) Probe id=r, state=exploringok, received probe q
This one gets through.
State: Operational
(B2,A) Probe id=s, state=operational, received probe r
B now knows that A has no problem to receive its packets
State: Operational
(A,B2) payload packet
Payload packets flow again

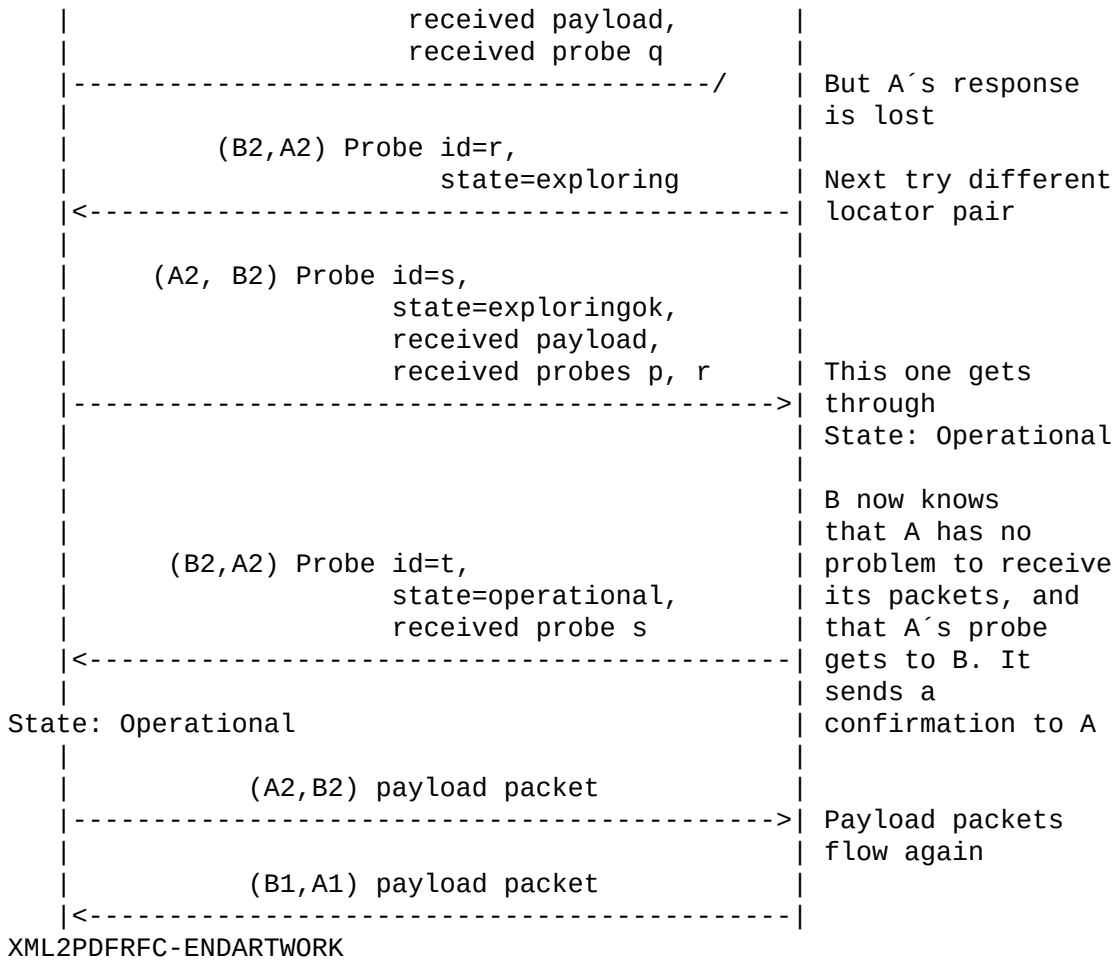


XML2PDFRFC-ENDARTWORK

The next example shows when the failure for the current locator pair is in the other direction only. A has addresses A1 and A2, and B has addresses B1 and B2. The current communication is between A1 and B1, but A´s packets no longer reach B using this pair.

EXAMPLE 5: One-way failure





XML2PDFRFC-ENDARTWORK

8. Protocol Constants

The following protocol constants are defined:

Send Timeout	10 seconds
Keepalive Interval	Not specified here
Initial Probe Timeout	0.5 seconds
Number of Initial Probes	4 probes
Max Probe Timeout	60 seconds

XML2PDFRFC-ENDARTWORK

9. Security Considerations

Attackers may spoof various indications from lower layers and the network in an effort to confuse the peers about which addresses are or are not operational. For example, attackers may spoof ICMP error messages in an effort to cause the parties to move their traffic elsewhere or even to disconnect. Attackers may also spoof information related to network attachments, router discovery, and address assignments in an effort to make the parties believe they have Internet connectivity when in reality they do not.

This may cause use of non-preferred addresses or even denial-of- service.

This protocol does not provide any protection of its own for indications from other parts of the protocol stack. Unprotected indications SHOULD NOT be taken as a proof of connectivity problems. However, REAP has weak resistance against incorrect information even from unprotected indications in the sense that it performs its own tests prior to picking a new address pair. Denial-of- service vulnerabilities remain, however, as do vulnerabilities against on path attackers.

Some aspects of these vulnerabilities can be mitigated through the use of techniques specific to the other parts of the stack, such as properly dealing with ICMP errors [I-D.ietf-tcpm-icmp-attacks], link layer security, or the use of SEND [RFC3971] to protect IPv6 Router and Neighbor Discovery.

Other parts of the SHIM6 protocol ensure that the set of addresses we are switching between actually belong together. REAP itself provides no such assurances. Similarly, REAP provides only minimal protection against third party flooding attacks; when REAP is run its Probe identifiers can be used as a return routability check that the claimed address is indeed willing to receive traffic. However, this needs to be complemented with another mechanism to ensure that the claimed address is also the correct host. SHIM6 does this by performing binding of all operations to context tags.

The keepalive mechanism in this specification is vulnerable to spoofing. On path-attackers that can see a SHIM6 context tag can send spoofed Keepalive messages once per Send Timeout interval, to prevent two SHIM6 nodes from sending Keepalives themselves. This vulnerability is only relevant to nodes involved in a one-way communication. The result of the attack is that the nodes enter the exploration phase needlessly, but they should be able to confirm connectivity unless, of course, the attacker is able to prevent the exploration phase from completing. Off-path attackers may not be

able to generate spoofed results, given that the context tags are 47- bit random numbers.

The exploration phase is vulnerable to attackers that are on the path. Off-path attackers would find it hard to guess either the context tag or the correct probe identifiers. Given that IPsec operates above the shim layer, it is not possible to protect the exploration phase against on-path attackers. This is similar to the ability to protect other Shim6 control exchanges. There are mechanisms in place to prevent the redirection of communications to wrong addresses, but on-path attackers can cause denial-of-service, move communications to less-preferred address pairs, and so on.

Finally, the exploration itself can cause a number of packets to be sent. As a result it may be used as a tool for packet amplification in flooding attacks. In order to prevent this it is required that the protocol employing REAP has built-in mechanisms to prevent this. For instance, in SHIM6 contexts are created only after a relatively large number of packets has been exchanged, a cost which reduces the attractiveness of using SHIM6 and REAP for amplification attacks. However, such protections are typically not present at connection establishment time. When exploration would be needed for connection establishment to succeed, its usage would result in an amplification vulnerability. As a result, SHIM6 does not support the use of REAP in connection establishment stage.

10. IANA Considerations

No IANA actions are required.

11. References

11.1. Normative References

[RFC1750] Eastlake, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 2006.

11.2. Informative References

[AURA02] Aura, T., Roe, M., and J. Arkko, "Security of Internet Location Management", In Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA., December 2002.

[I-D.bagnulo-shim6-addr-selection] Bagnulo, M., "Address selection in multihomed environments", draft-bagnulo-shim6-addr-selection-00 (work in progress), October 2005.

[I-D.huitema-multi6-addr-selection] Huitema, C., "Address selection in multihomed environments", draft-huitema-multi6-addr-selection-00 (work in progress), October 2004.

[I-D.ietf-dna-cpl]

Nordmark, E. and J. Choi, "DNA with unmodified routers: Prefix list based approach", draft-ietf-dna-cpl-02 (work in progress), January 2006.

[I-D.ietf-dna-protocol]
Kempf, J., "Detecting Network Attachment in IPv6 Networks (DNAv6)", draft-ietf-dna-protocol-01 (work in progress), June 2006.

[I-D.ietf-hip-mm]
Nikander, P., "End-Host Mobility and Multihoming with the Host Identity Protocol", draft-ietf-hip-mm-04 (work in progress), June 2006.

[I-D.ietf-shim6-locator-pair-selection]
Bagnulo, M., "Default Locator-pair selection algorithm for the SHIM6 protocol", draft-ietf-shim6-locator-pair-selection-00 (work in progress), May 2006.

[I-D.ietf-shim6-proto]
Bagnulo, M. and E. Nordmark, "Level 3 multihoming shim protocol", draft-ietf-shim6-proto-05 (work in progress), May 2006.

[I-D.ietf-shim6-reach-detect]
Beijnum, I., "Shim6 Reachability Detection", draft-ietf-shim6-reach-detect-01 (work in progress), October 2005.

[I-D.ietf-tcpm-icmp-attacks]
Gont, F., "ICMP attacks against TCP", draft-ietf-tcpm-icmp-attacks-00 (work in progress), February 2006.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

Appendix A. Contributors

This draft attempts to summarize the thoughts and unpublished contributions of many people, including the MULTI6 WG design team members Marcelo Bagnulo Braun, Erik Nordmark, Geoff Huston, Kurtis Lindqvist, Margaret Wasserman, and Jukka Ylitalo, the MOBIKE WG contributors Pasi Eronen, Tero Kivinen, Francis Dupont, Spencer Dawkins, and James Kempf, and HIP WG contributors such as Pekka Nikander. This draft is also in debt to work done in the context of SCTP [RFC2960] and HIP multihoming and mobility extension [I-D.ietf-hip-mm].

Appendix B. Acknowledgements

The authors would also like to thank Christian Huitema, Pekka Savola, John Loughney, Sam Xia, and Hannes Tschofenig for interesting discussions in this problem space, and for review of this specification.

Authors´ Addresses

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@ericsson.com

Iljitsch van Beijnum

Email: iljitsch@muada.com

Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

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.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).