Network Working Group
Request for Comments: 5534
Category: Standards Track

J. Arkko
Ericsson
I. van Beijnum
IMDEA Networks

June 2009

Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming

Status of This Memo

Copyright Notice

Abstract

This document specifies how the level 3 multihoming Shim6 protocol (Shim6) detects failures between two communicating nodes. It also specifies an exploration protocol for switching to another pair of interfaces and/or addresses between the same nodes if a failure occurs and an operational pair can be found.

Table of Contents

1. Introduction

The Shim6 protocol [RFC5533] 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 nodes 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).

Failure detection is made as lightweight as possible. Payload data traffic in both directions is observed, and in the case where there is no traffic because the communication is idle, failure detection is also idle and doesn't generate any packets. When payload traffic is flowing in both directions, there is no need to send failure detection packets, either. Only when there is traffic in one direction does the failure detection mechanism generate keepalives in the other direction. As a result, whenever there is outgoing traffic and no incoming return traffic or keepalives, there must be failure, at which point the locator pair exploration is performed to find a working address pair for each direction.

This document is structured as follows: Section 3 defines a set of useful terms, Section 4 gives an overview of REAP, and Section 5 provides a detailed definition. Section 6 specifies behavior, and Section 7 discusses protocol constants. Section 8 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 said node ends up with a legitimate locator.

REAP has been designed to be used with Shim6 and is therefore tailored to an environment where it typically runs on hosts, uses widely varying types of paths, and is unaware of application context. As a result, REAP attempts to be as self-configuring and unobtrusive as possible. In particular, it avoids sending any packets except where absolutely required and employs exponential back-off to avoid congestion. The downside is that it cannot offer the same granularity of detecting problems as mechanisms that have more application context and ability to negotiate or configure parameters.

Future versions of this specification may consider extensions with such capabilities, for instance, through inheriting some mechanisms from the Bidirectional Forwarding Detection (BFD) protocol [BFD].

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 it. 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 all the following conditions are fulfilled:

- The address has been assigned to an interface of the node.

- The valid lifetime of the prefix (Section 4.6.2 of RFC 4861 [RFC4861]) associated with the address has not expired.

- The address is not tentative in the sense of RFC 4862 [RFC4862]. 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 Duplicate Address Detection (DAD) [RFC4429] even though implementations may prefer using other addresses as long as there is an alternative.

- The address is a global unicast or unique local address [RFC4193]. That is, it is not an IPv6 site-local or link-local address.

  With link-local addresses, the nodes would be unable to determine on which link the given address is usable.

- The address and interface are 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 [RFC4861], Address Autoconfiguration [RFC4862], 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. In other words, when 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 [RFC4861]. 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-SIM] can be used to ensure this.
>
> Note 2: In theory, it would also be possible for nodes 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 [ADD-SEL] and [MULTI6]), 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, which is used 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 a unidirectionally operational address pair when packets sent with the first address as the source and the second address as the destination reach the destination.

Shim6 implementations MUST support the discovery of operational address pairs through the use of explicit reachability tests and Forced Bidirectional Communication (FBD), described later in this specification. Future extensions of Shim6 may specify additional mechanisms. Some ideas of such mechanisms are listed below but are not fully specified in this document:

- 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 [RFC4861].

  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 not to trust these blindly, however. One approach would be to use ICMP error messages only as a hint to perform an explicit reachability test or to move an address pair to a lower place in the list of address pairs to be probed, but

not to use these messages 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 [GONT]. These verifications can ensure that (practically) only on-path attackers can spoof the messages.

3.4. Primary Address Pair

The primary address pair consists of the 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 that no context tag needs to be present.

3.5. Current Address Pair

Shim6 needs to avoid sending packets that belong to the same transport connection concurrently over multiple paths. This is 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. Effective congestion control over multiple paths is considered a research topic at the time of publication of this document. Shim6 does not attempt to employ multiple paths simultaneously.

Note: The Stream Control Transmission Protocol (SCTP) and future multipath transport protocols are likely to require interaction with Shim6, at least to ensure that they do not employ Shim6 unexpectedly.

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

It is theoretically possible to support multiple current address pairs for different transport sessions or Shim6 contexts. However, this is not supported in this version of the Shim6 protocol.

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

4. Protocol Overview

This section discusses the design of the reachability detection and full reachability exploration mechanisms, and gives an overview of the REAP protocol.

Exploring the full set of communication options between two nodes 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 an operational pair that worked a few seconds ago is very likely to still be operational. Thus, it makes sense to have a lightweight protocol that confirms existing reachability, and to only invoke heavier exploration mechanism when 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 Sections 3.1 and 3.2 to track the local situation. It is also necessary to track remote address information from the peer. For instance, if the peer's address in the current address pair is no longer locally operational, a mechanism to relay that information is needed. The Update Request message in the Shim6 protocol is used for this purpose [RFC5533]. 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 address pair.

A technique called Forced Bidirectional Detection (FBD) is employed for the reachability verification. Reachability for the currently used address pair in a Shim6 context is determined by making sure that whenever there is payload traffic in one direction, there is also traffic in the other direction. This can be data traffic as well, or it may be 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 payload traffic going out, but there are no return packets, there must be a failure, and the full exploration mechanism is started.

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

1. To prevent the other side from concluding that there is a reachability failure, it's necessary for a node implementing the failure-detection mechanism to generate periodic keepalives when there is no other traffic.

   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 the duration of a Send Timeout period. The node communicates its Send Timeout value to the peer as a Keepalive Timeout Option (Section 5.3) in the I2, I2bis, R2, or UPDATE messages. The peer then maps this value to its Keepalive Timeout value.

   The interval after which keepalives are sent is named the Keepalive Interval. The RECOMMENDED approach for the Keepalive Interval is to send keepalives at one-half to one-third of the Keepalive Timeout interval, so that multiple keepalives are generated and have time to reach the peer before it times out.

2. Whenever outgoing payload packets are generated, a timer is started to reflect the requirement that the peer should generate return traffic from payload packets. The timeout value is set to the value of Send Timeout.

   For the purposes of this specification, "payload packet" refers to any packet that is part of a Shim6 context, including both upper-layer protocol packets and Shim6 protocol messages, except those defined in this specification. For the latter messages, Section 6 specifies what happens to the timers when a message is transmitted or received.

3. Whenever incoming payload packets are received, the timer associated with the return traffic from the peer is stopped, and another timer is started to reflect the requirement for this node to generate return traffic. This timeout value is set to the value of Keepalive Timeout.

These two timers are mutually exclusive. In other words, either the node is expecting to see traffic from the peer based on the traffic that the node sent earlier or the node is expecting to respond to the peer based on the traffic that the peer sent earlier (otherwise, the node is in an idle state).

4. The reception of a REAP Keepalive message leads to stopping the timer associated with the return traffic from the peer.

5. Keepalive Interval seconds after the last payload packet has been received for a context, if no other packet has been sent within this context since the payload packet has been received, a REAP Keepalive message is generated for the context in question and transmitted to the peer. A node 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. REAP Keepalive messages SHOULD continue to be sent at the Keepalive Interval until either a payload packet in the Shim6 context has been received from the peer or the Keepalive Timeout expires. Keepalives are not sent at all if one or more payload packets were sent within the Keepalive Interval.

6. Send Timeout seconds after the transmission of a payload packet with no return traffic on this context, a full reachability exploration is started.

Section 7 provides some suggested defaults for these timeout values. The actual value SHOULD be randomized in order to prevent synchronization. Experience from the deployment of the Shim6 protocol is needed in order to determine what values are most suitable.

4.2. Full Reachability Exploration

As explained in previous sections, the currently used address pair may become invalid, either through one of the addresses becoming unavailable or nonoperational or through the pair itself being declared nonoperational. 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 a 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 the upper-layer protocol (ULP), SHOULD be informed. This can be a signal for the ULP to adapt, due to the change in path, so that for example, if the ULP is TCP, it could initiate a slow start procedure. However, it's likely that the circumstances that led to the selection of a new path already caused enough packet loss to trigger slow start.

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 8, 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. An overview of the choosing process used by REAP is as follows:

- As an input to start the process, the node has knowledge of its own addresses and has been told via Shim6 protocol messages what the addresses of the peer are. A list of possible pairs of addresses can be constructed by combining the two pieces of information.

- By employing standard IPv6 address selection rules, the list is pruned by removing combinations that are inappropriate, such as attempting to use a link-local address when contacting a peer that uses a global unicast address.

- Similarly, standard IPv6 address selection rules provide a basic priority order for the pairs.

- Local preferences may be applied for some additional tuning of the order in the list. The mechanisms for local preference settings are not specified but can involve, for instance, configuration that sets the preference for using one interface over another.

- As a result, the node has a prioritized list of address pairs to try. However, the list may still be long, as there may be a combinatorial explosion when there are many addresses on both sides. REAP employs these pairs sequentially, however, and uses a back-off procedure to avoid a "signaling storm". This ensures that the exploration process is relatively conservative or "safe". The tradeoff is that finding a working path may take time if there are many addresses on both sides.

In more detail, the process is as follows. Nodes first consult the RFC 3484 default address selection rules [RFC3484] 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, possibly making the search faster. (Additional mechanisms may be defined in the future for arriving at an initial ordering of address pairs before testing starts [PAIR].) 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.

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 retry the process as 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 7 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; the Number of Initial Probes (4) specifies how many initial probes can be sent before the exponential back-off 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.

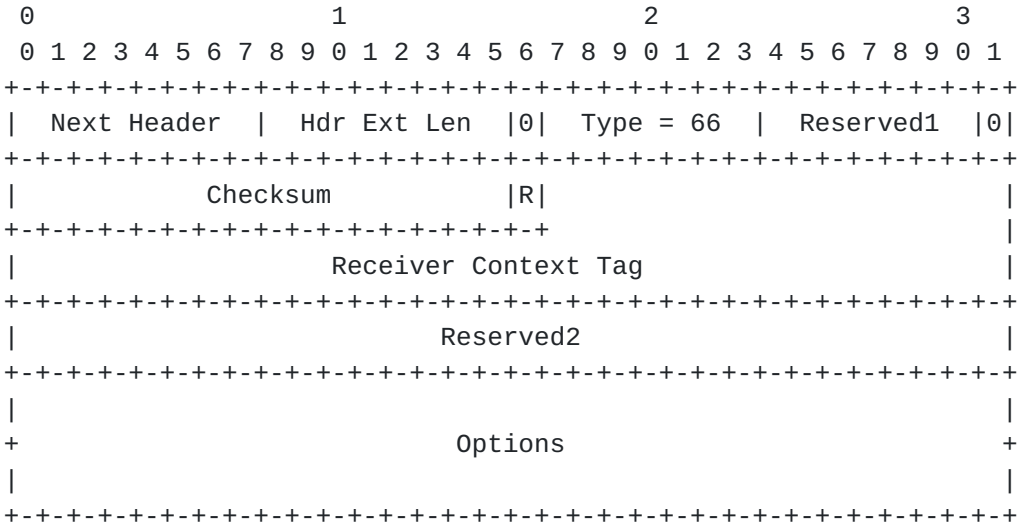
Note: The rationale for sending four packets at a fixed rate before the exponential back-off is employed is to avoid having to send these packets excessively fast. Without this, having 0.5 seconds between the third and fourth probe means that the time between the first and second probe would have to be 0.125 seconds, which gives very little time for a reply to the first packet to arrive. Also, this means that the first four packets are sent within 0.875 seconds rather than 2 seconds, increasing the potential for congestion if a large number of Shim6 contexts need to send probes at the same time after a failure.

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:



Next Header, Hdr Ext Len, 0, 0, Checksum
These are as specified in Section 5.3 of the Shim6 protocol description [RFC5533].

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

Reserved1
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 that the receiver has allocated for the context.

Reserved2
This is a 32-bit field reserved for future use. It is set to zero on transmit and MUST be ignored on receipt.

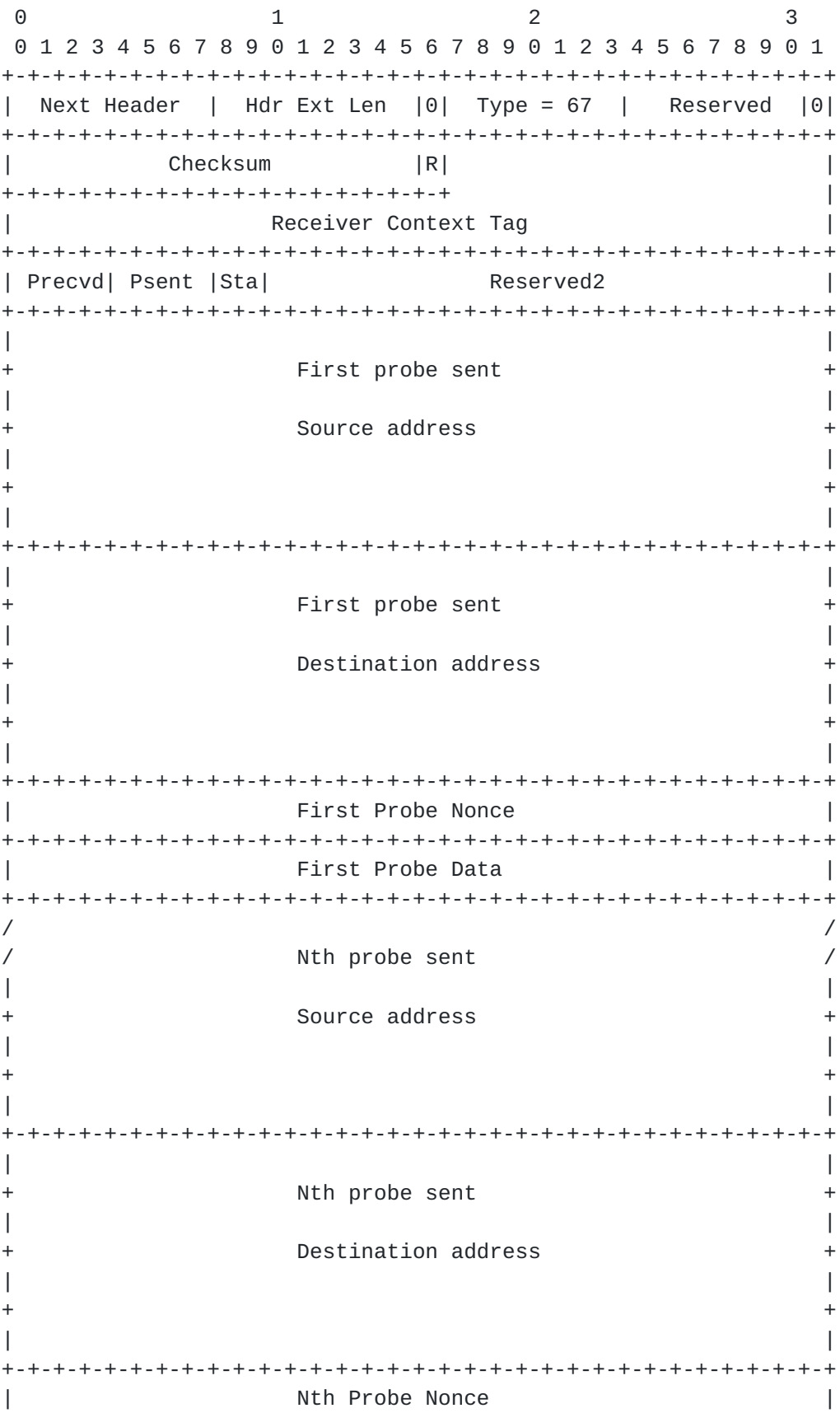
Options
This field MAY contain one or more Shim6 options. However, there are currently no defined options that are useful in a Keepalive message. The Options field is provided only for future extensibility reasons.

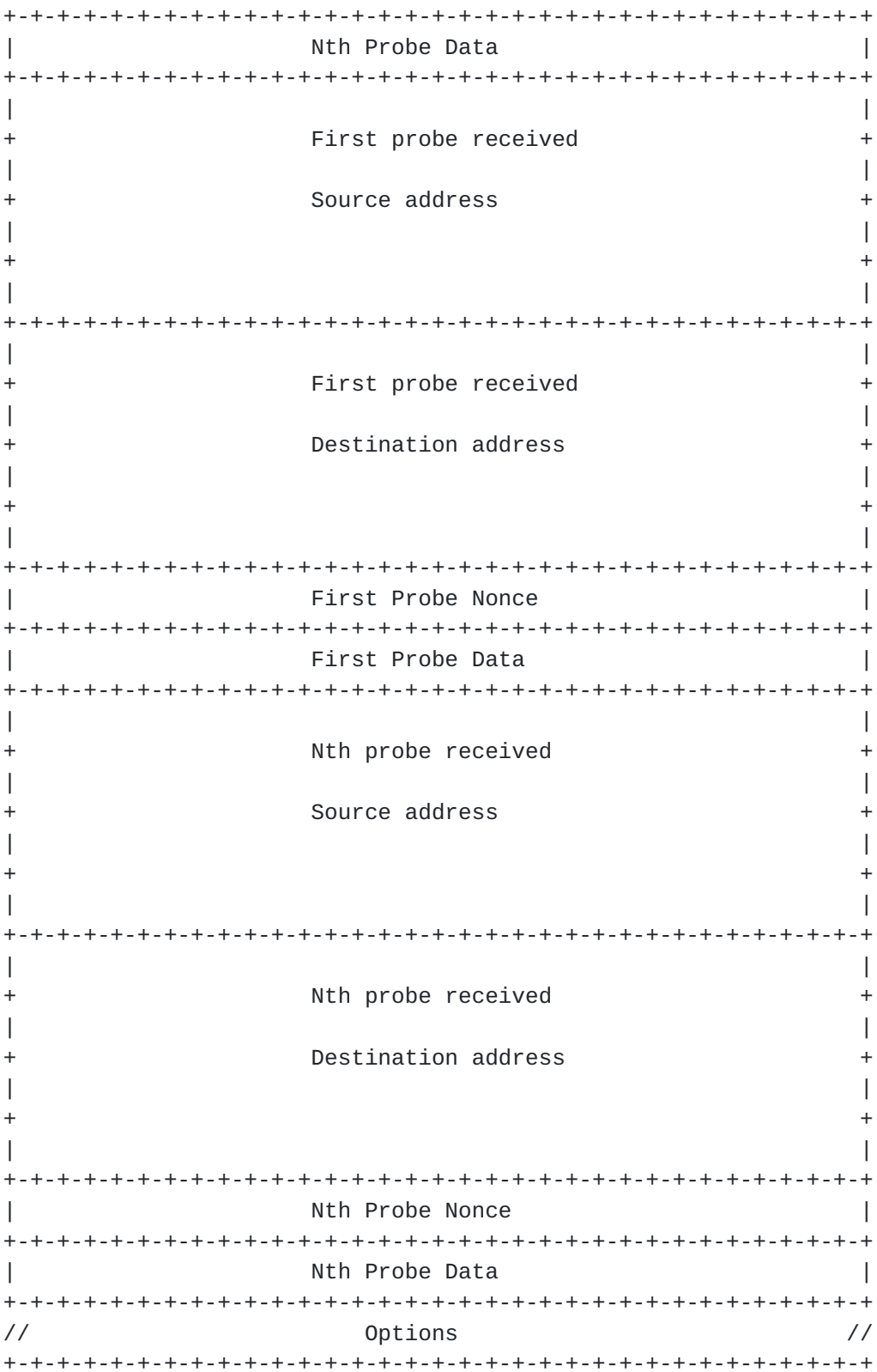
A valid message conforms to the format above, has a Receiver Context Tag that matches the context known by the receiver, is a valid Shim6 control message as defined in Section 12.3 of the Shim6 protocol description [RFC5533], and has a Shim6 context that is in state ESTABLISHED. The receiver processes a valid message by inspecting its options and executing any actions specified for such options.

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

5.2. Probe Message

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


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Next Header
Hdr Ext Len
0
Type = 67
Reserved
0
Checksum
R
Receiver Context Tag
Precvd
Psent
Sta
Reserved2
First probe sent
Source address
First probe sent
Destination address
First Probe Nonce
First Probe Data
Nth probe sent
Source address
Nth probe sent
Destination address
Nth Probe Nonce


Nth Probe Data
First probe received
Source address
First probe received
Destination address
First Probe Nonce
First Probe Data
Nth probe received
Source address
Nth probe received
Destination address
Nth Probe Nonce
Nth Probe Data
Options

Next Header, Hdr Ext Len, 0, 0, Checksum
These are as specified in Section 5.3 of the Shim6 protocol description [RFC5533].

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 that 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 message. Received Probe fields are copies of the same fields in earlier received probes that arrived since the last transition to state Exploring. When a sender is in state InboundOk it MUST include copies of the fields of at least one of the inbound probes. A sender MAY include additional sets of these received Probe fields in any state as per any logic employed by the implementation.

The fields Probe Source, Probe Destination, Probe Nonce, and Probe Data 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 (InboundOk) implies that the sender believes it has no problem communicating, i.e., it at least sees packets from the recipient but that the recipient either has a problem or has not yet confirmed to the sender that the problem has been resolved.

Reserved2
MUST be set to zero 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 with which sent probes a received probe correlates. It is highly RECOMMENDED that the Nonce field be 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 4086 [RFC4086].

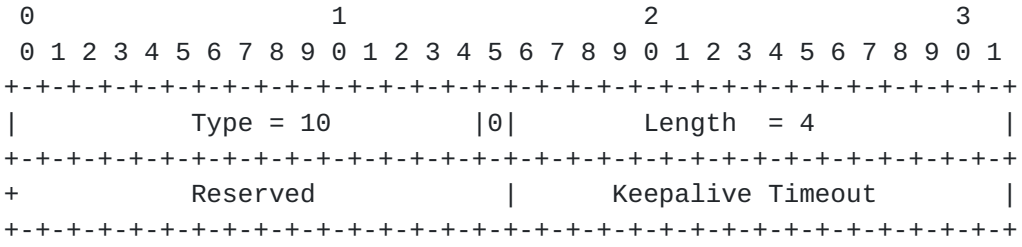
Probe Data
This is a 32-bit field with no fixed meaning. The Probe Data field is copied back with no changes. Future flags may define a use for this field.

Options
For future extensions.

5.3. Keepalive Timeout Option Format

Either side of a Shim6 context can notify the peer of the value that it would prefer the peer to use as its Keepalive Timeout value. If the node is using a non-default Send Timeout value, it MUST

communicate this value as a Keepalive Timeout value to the peer in the below option. This option MAY be sent in the I2, I2bis, R2, or UPDATE messages. The option SHOULD only need to be sent once in a given Shim6 association. If a node receives this option, it SHOULD update its Keepalive Timeout value for the peer.



Fields:

Type
This field identifies the option and MUST be set to 10 (Keepalive Timeout).

Length
This field MUST be set as specified in Section 5.1 of the Shim6 protocol description [RFC5533] -- that is, set to 4.

Reserved
A 16-bit field reserved for future use. It is set to zero upon transmit and MUST be ignored upon receipt.

Keepalive Timeout
The value in seconds corresponding to the suggested Keepalive Timeout value for the peer.

6. Behavior

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

On a given context with a given peer, the node can be in one of three states: Operational, Exploring, or InboundOK. In the Operational state, the underlying address pairs are assumed to be operational. In the Exploring state, this node hasn't seen any traffic from the peer for more than a Send Timer period. Finally, in the InboundOK

state, this node sees traffic from the peer, but the peer may not yet see any traffic from this node, so the exploration process needs to continue.

The node also maintains 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 within the Operational and InboundOK states. No timer is running in the Exploring state. As explained in Section 4.1, the two timers are mutually exclusive. That is, either the Keepalive Timer or the Send Timer is running, or neither of them is running.

Note that Appendix A gives some examples of typical protocol runs in order to illustrate the behavior.

6.1. Incoming Payload Packet

Upon the reception of a payload packet in the Operational state, the node starts the Keepalive Timer if it was not yet running, and stops the Send Timer if it was running.

If the node is in the Exploring state, it transitions to the InboundOK state, sends a Probe message, and starts the Send Timer. It fills the Psent and corresponding Probe Source Address, Probe Destination Address, Probe Nonce, and Probe Data fields with information about recent Probe messages that have not yet been reported as seen by the peer. It also fills the Precvd and corresponding Probe Source Address, Probe Destination Address, Probe Nonce, and Probe Data fields with information about recent Probe messages it has seen from the peer. 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. In this case, the node therefore sets the State field to 2 (InboundOk). The IP source and destination addresses for sending the Probe message are selected as discussed in Section 4.3.

In the InboundOK 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 the same as the reception of payload packets.

While the Keepalive Timer is running, the node SHOULD send Keepalive messages to the peer with an interval of Keepalive Interval seconds. Conceptually, a separate timer is used to distinguish between the interval between Keepalive messages and the overall Keepalive Timeout interval. However, this separate timer is not modelled in the tabular or graphical state machines. When sent, the Keepalive message is constructed as described in Section 5.1. It is sent using the current address pair.

In the below tables, "START", "RESTART", and "STOP" refer to starting, restarting, and stopping the Keepalive Timer or the Send Timer, respectively. "GOTO" refers to transitioning to another state. "SEND" refers to sending a message, and "-" refers to taking no action.

Operational	Exploring	InboundOk
STOP Send START Keepalive	SEND Probe InboundOk START Send GOTO InboundOk	STOP Send

6.2. Outgoing Payload Packet

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 InboundOK state the node simply starts the Send Timer if it was not yet running. (The sending of Shim6 control messages is again treated the same.)

Operational	Exploring	InboundOk
START Send STOP Keepalive	-	START Send

6.3. Keepalive Timeout

Upon a timeout on the Keepalive Timer, the node sends one last Keepalive message. This can only happen in the Operational state.

The Keepalive message is constructed as described in Section 5.1. It is sent using the current address pair.

Operational	Exploring	InboundOk
SEND Keepalive	-	-

6.4. Send Timeout

Upon a timeout on the Send Timer, the node enters the Exploring state and sends a Probe message. The Probe message is constructed as explained in Section 6.1, except that the State field is set to 1 (Exploring).

Operational	Exploring	InboundOk
SEND Probe Exploring GOTO Exploring	-	SEND Probe Exploring GOTO Exploring

6.5. Retransmission

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

The Probe messages are constructed as explained in Section 6.1, except that the State field is set to 1 (Exploring) or 2 (InboundOk), depending on which state the sender is in.

Operational	Exploring	InboundOk
-	SEND Probe Exploring	SEND Probe InboundOk START Send

6.6. Reception of the Keepalive Message

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 InboundOK state, sends a Probe message, and starts the Send Timer. The Probe message is constructed as explained in Section 6.1.

In the InboundOK state, the Send Timer is stopped if it was running.

Operational	Exploring	InboundOk
STOP Send	SEND Probe InboundOk START Send GOTO InboundOk	STOP Send

6.7. Reception of the Probe Message State=Exploring

Upon receiving a Probe message with State set to Exploring, the node enters the InboundOK state, sends a Probe message as described in Section 6.1, stops the Keepalive Timer if it was running, and restarts the Send Timer.

Operational	Exploring	InboundOk
SEND Probe InboundOk	SEND Probe InboundOk	SEND Probe InboundOk
STOP Keepalive	START Send	RESTART Send
RESTART Send	GOTO InboundOk	
GOTO InboundOk		

6.8. Reception of the Probe Message State=InboundOk

Upon the reception of a Probe message with State set to InboundOk, the node sends a Probe message, restarts the Send Timer, stops the Keepalive Timer if it was running, and transitions to the Operational state. A new current address pair is chosen for the connection, based on the reports of received probes in the message that we just received. If no received probes have been reported, the current address pair is unchanged.

The Probe message is constructed as explained in Section 6.1, except that the State field is set to zero (Operational).

Operational	Exploring	InboundOk
SEND Probe Operational	SEND Probe Operational	SEND Probe Operational
RESTART Send	RESTART Send	RESTART Send
STOP Keepalive	GOTO Operational	GOTO Operational

6.9. Reception of the Probe Message State=Operational

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. The Probe message is constructed as explained in Section 6.1, except that the State field is set to zero (Operational).

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

Operational	Exploring	InboundOk
STOP Send	STOP Send	STOP Send
START Keepalive	START Keepalive	START Keepalive
	GOTO Operational	GOTO Operational

The reachability detection and exploration process has no effect on payload communications until a new operational address pair has actually been confirmed. Prior to that, the payload packets continue to be sent to the previously used addresses.

6.10. Graphical Representation of the State Machine

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.

7. Protocol Constants and Variables

The following protocol constants are defined:

Initial Probe Timeout	0.5 seconds
Number of Initial Probes	4 probes

And these variables have the following default values:

Send Timeout	15 seconds
Keepalive Timeout	X seconds, where X is the peer's Send Timeout as communicated in the Keepalive Timeout Option
	15 seconds if the peer didn't send a Keepalive Timeout option
Keepalive Interval	Y seconds, where Y is one-third to one-half of the Keepalive Timeout value (see Section 4.1)

Alternate values of the Send Timeout may be selected by a node and communicated to the peer in the Keepalive Timeout Option. A very small value of the Send Timeout may affect the ability to exchange keepalives over a path that has a long roundtrip delay. Similarly, it may cause Shim6 to react to temporary failures more often than necessary. As a result, it is RECOMMENDED that an alternate Send Timeout value not be under 10 seconds. Choosing a higher value than the one recommended above is also possible, but there is a relationship between Send Timeout and the ability of REAP to discover and correct errors in the communication path. In any case, in order for Shim6 to be useful, it should detect and repair communication

problems long before upper layers give up. For this reason, it is RECOMMENDED that Send Timeout be at most 100 seconds (default TCP R2 timeout [RFC1122]).

> Note: It is not expected that the Send Timeout or other values will be estimated based on experienced roundtrip times. Signaling exchanges are performed based on exponential back-off. The keepalive processes send packets only in the relatively rare condition that all traffic is unidirectional.

8. Security Considerations

Attackers may spoof various indications from lower layers and from 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 [GONT], 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 some protection against third-party flooding attacks [AURA02]; when REAP is run, its Probe Nonces 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 node. 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 in order 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.

To protect against spoofed Keepalive messages, a node implementing both Shim6 and IPsec MAY ignore incoming REAP keepalives if it has good reason to assume that the other side will be sending IPsec-protected return traffic. In other words, if a node is sending TCP payload data, it can reasonably expect to receive TCP ACKs in return. If no IPsec-protected ACKs come back but unprotected keepalives do, this could be the result of an attacker trying to hide broken connectivity.

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 Shim6 layer, it is not possible to protect the exploration phase against on-path attackers with IPsec. This is similar to the issues with protecting 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. It is required that the protocol employing REAP has built-in mechanisms to prevent this. For instance, Shim6 contexts are created only after a relatively large number of packets have been exchanged, a cost that 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 the connection-establishment stage.

9. Operational Considerations

When there are no failures, the failure-detection mechanism (and Shim6 in general) are lightweight: keepalives are not sent when a Shim6 context is idle or when there is traffic in both directions. So in normal TCP or TCP-like operations, there would only be one or two keepalives when a session transitions from active to idle.

Only when there are failures is there significant failure-detection traffic, especially in the case where a link goes down that is shared by many active sessions and by multiple nodes. When this happens, one keepalive is sent and then a series of probes. This happens per active (traffic-generating) context, all of which will time out within 15 seconds after the failure. This makes the peak traffic that Shim6 generates after a failure around one packet per second per context. Presumably, the sessions that run over those contexts were sending at least that much traffic and most likely more, but if the backup path is significantly lower bandwidth than the failed path, this could lead to temporary congestion.

> However, note that in the case of multihoming using BGP, if the failover is fast enough that TCP doesn't go into slow start, the full payload data traffic that flows over the failed path is switched over to the backup path, and if this backup path is of a lower capacity, there will be even more congestion.

Although the failure detection probing does not perform congestion control as such, the exponential back-off makes sure that the number of packets sent quickly goes down and eventually reaches one per context per minute, which should be sufficiently conservative even on the lowest bandwidth links.

Section 7 specifies a number of protocol parameters. Possible tuning of these parameters and others that are not mandated in this specification may affect these properties. It is expected that further revisions of this specification provide additional information after sufficient deployment experience has been obtained from different environments.

Implementations may provide means to monitor their performance and send alarms about problems. Their standardization is, however, the subject of future specifications. In general, Shim6 is most applicable for small sites and nodes, and it is expected that monitoring requirements on such deployments are relatively modest. In any case, where the node is associated with a management system, it is RECOMMENDED that detected failures and failover events are

reported via asynchronous notifications to the management system. Similarly, where logging mechanisms are available on the node, these events should be recorded in event logs.

Shim6 uses the same header for both signaling and the encapsulation of payload packets after a rehoming event. This way, fate is shared between the two types of packets, so the situation where reachability probes or keepalives can be transmitted successfully but payload packets cannot, is largely avoided: either all Shim6 packets make it through, so Shim6 functions as intended, or none do, and no Shim6 state is negotiated. Even in the situation where some packets make it through and others do not, Shim6 will generally either work as intended or provide a service that is no worse than in the absence of Shim6, apart from the possible generation of a small amount of signaling traffic.

Sometimes payload packets (and possibly payload packets encapsulated in the Shim6 header) do not make it through, but signaling and keepalives do. This situation can occur when there is a path MTU discovery black hole on one of the paths. If only large packets are sent at some point, then reachability exploration will be turned on and REAP will likely select another path, which may or may not be affected by the PMTUD black hole.

10. References

10.1. Normative References

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

[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.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[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.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.

10.2. Informative References

[ADD-SEL] Bagnulo, M., "Address selection in multihomed environments", Work in Progress, October 2005.

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

[BFD] Katz, D. and D. Ward, "Bidirectional Forwarding Detection", Work in Progress, February 2009.

[DNA-SIM] Krishnan, S. and G. Daley, "Simple procedures for Detecting Network Attachment in IPv6", Work in Progress, February 2009.

[GONT] Gont, F., "ICMP attacks against TCP", Work in Progress, October 2008.

[MULTI6] Huitema, C., "Address selection in multihomed environments", Work in Progress, October 2004.

[PAIR] Bagnulo, M., "Default Locator-pair selection algorithm for the Shim6 protocol", Work in Progress, October 2008.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

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

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5206] Nikander, P., Henderson, T., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008.

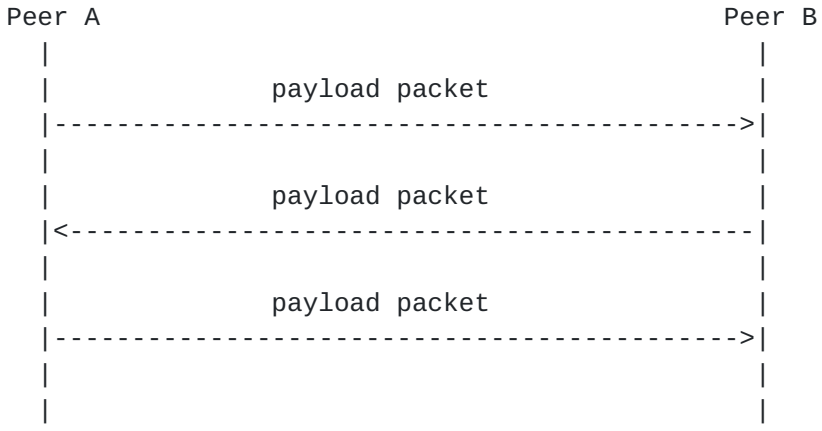
Appendix A. Example Protocol Runs

This appendix has examples of REAP protocol runs in typical scenarios. We start with the simplest scenario of two nodes, A and B, that have a Shim6 connection with each other but are not currently sending any payload data. As neither side sends anything, they also do not expect anything back, so there are no messages at all:

EXAMPLE 1: No Communications

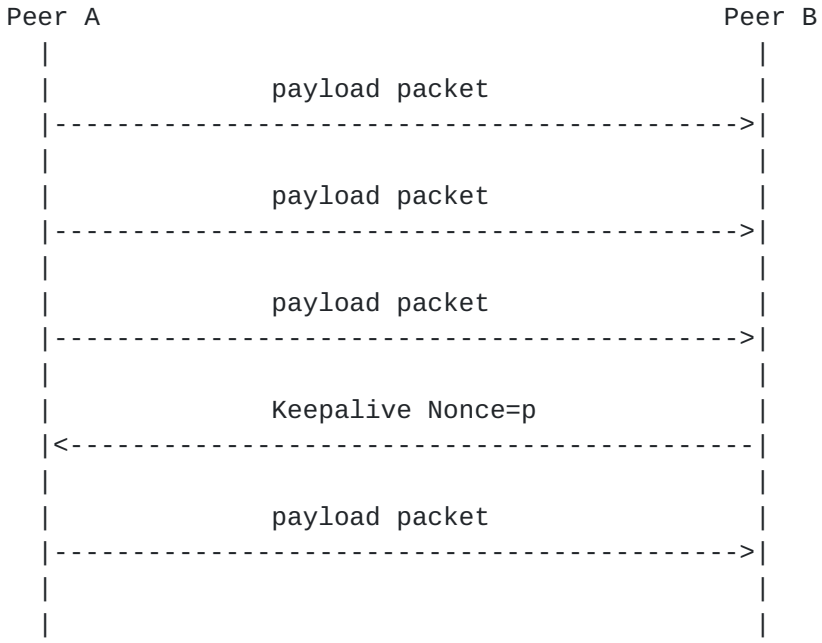
Our second example involves an active connection with bidirectional payload packet flows. Here, the reception of payload data from the peer is taken as an indication of reachability, so again there are no extra packets:

EXAMPLE 2: Bidirectional Communications



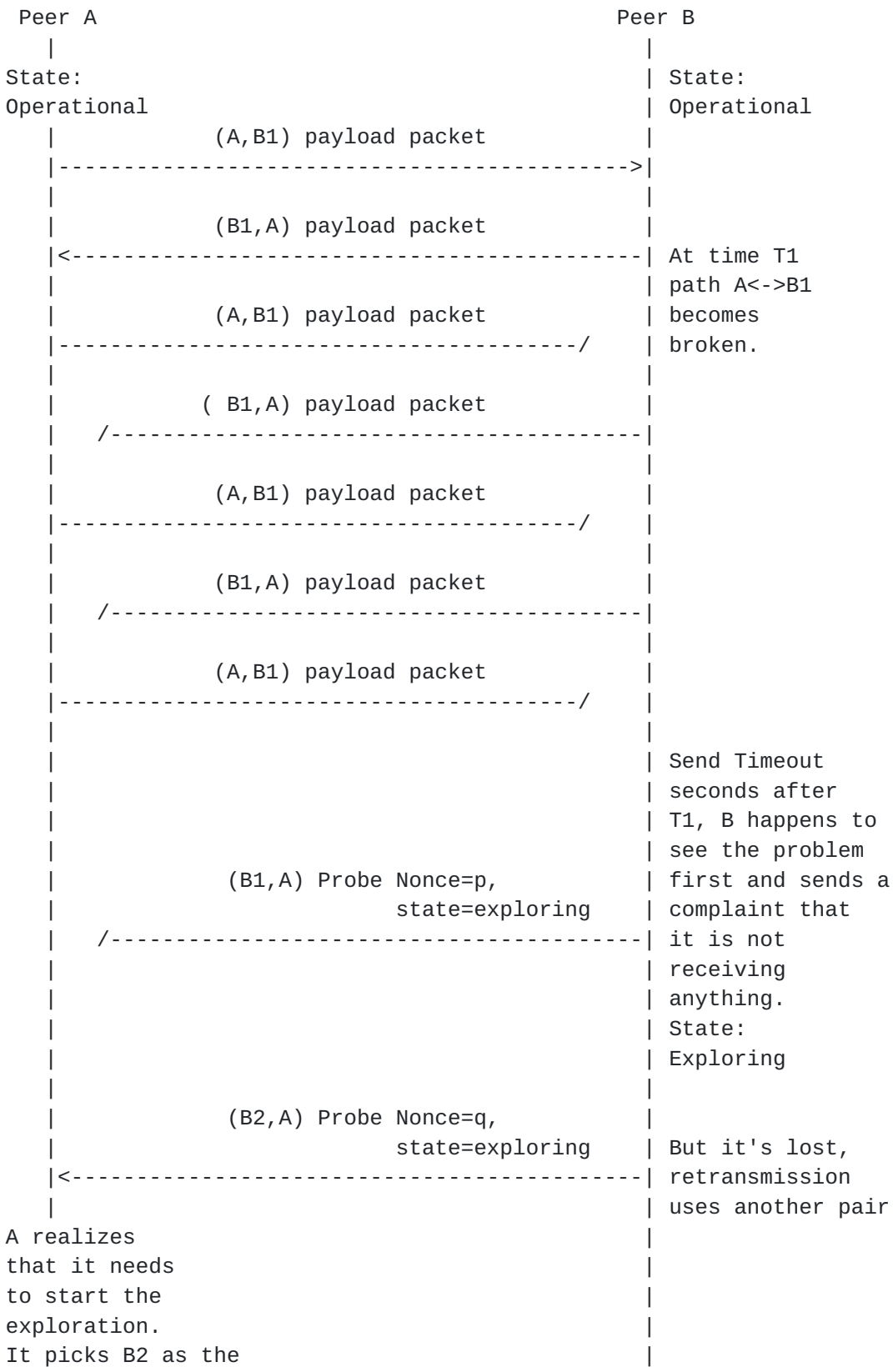
The third example is the first one that involves an actual REAP message. Here, the nodes communicate in just one direction, so REAP messages are needed to indicate to the peer that sends payload packets that its packets are getting through:

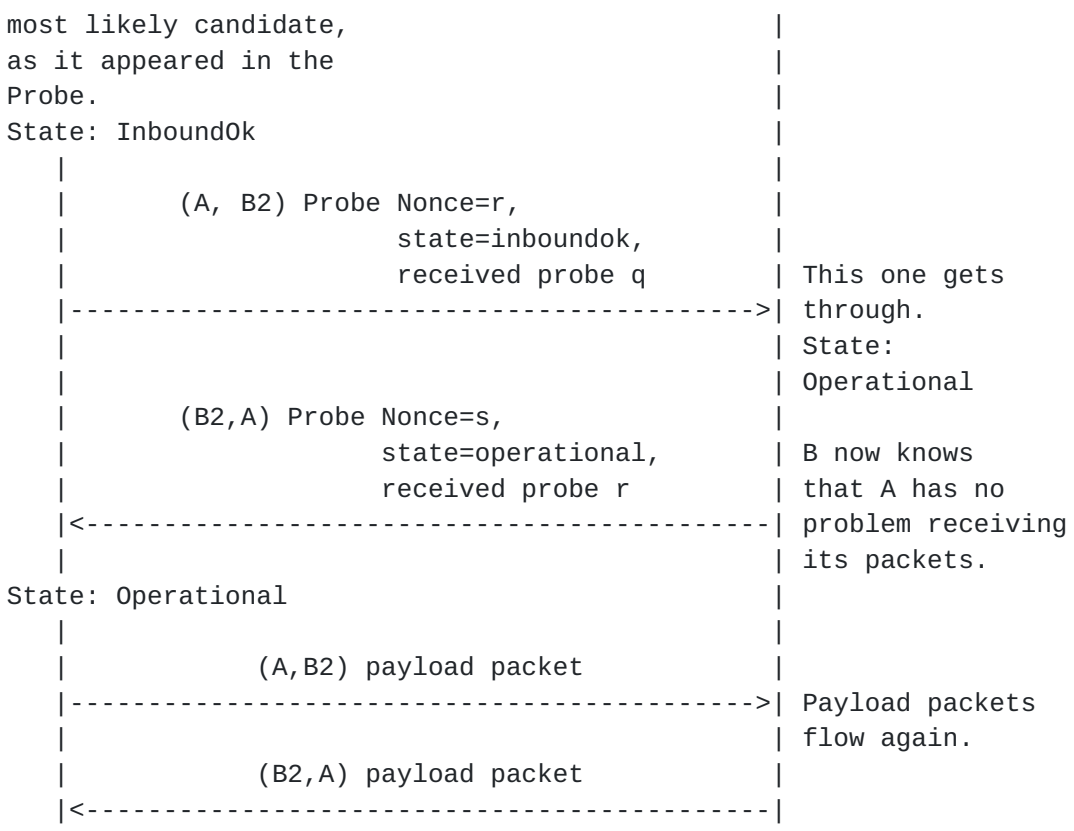
EXAMPLE 3: Unidirectional Communications



The next example involves a failure scenario. Here, A has address 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:

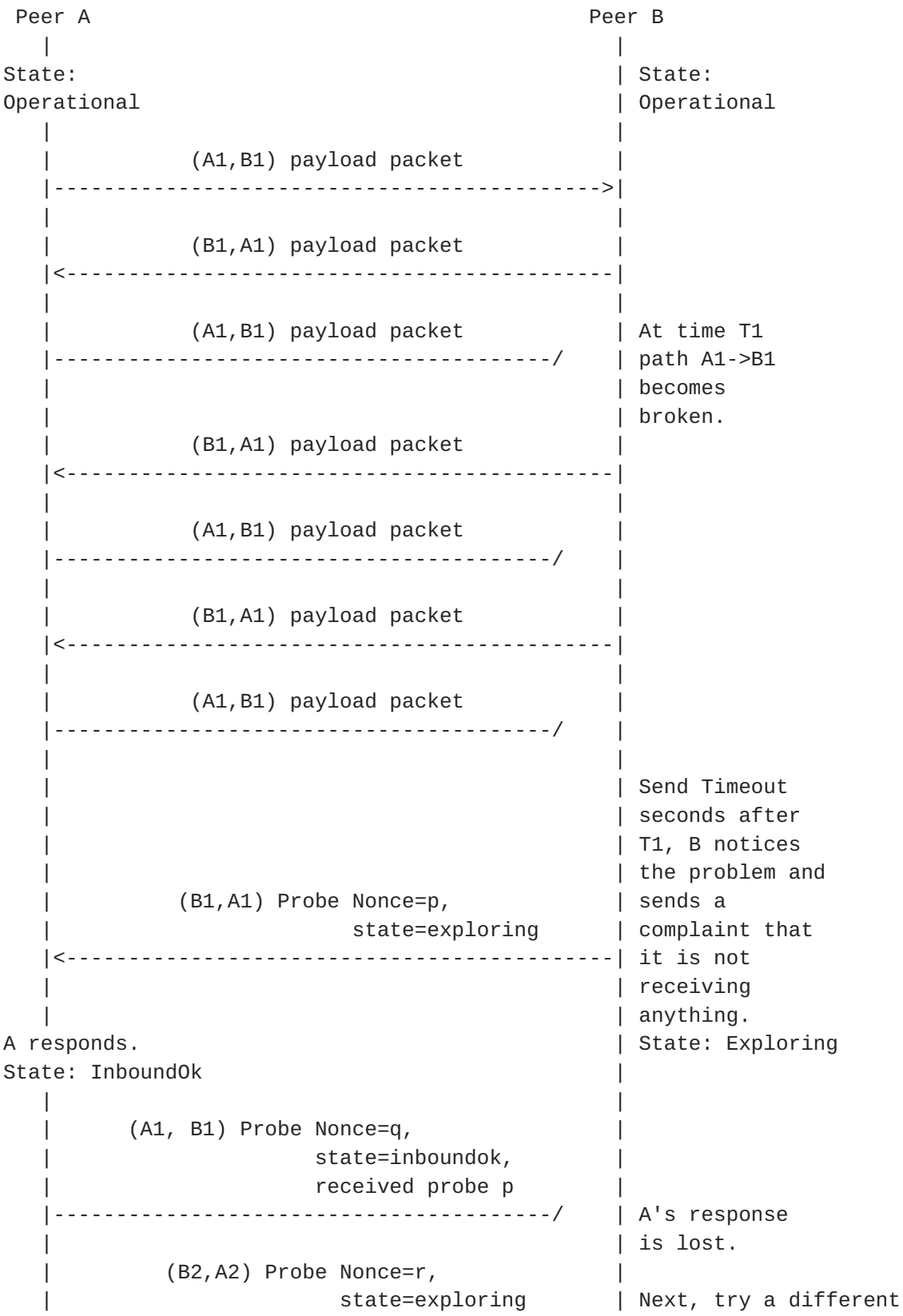
EXAMPLE 4: Failure Scenario

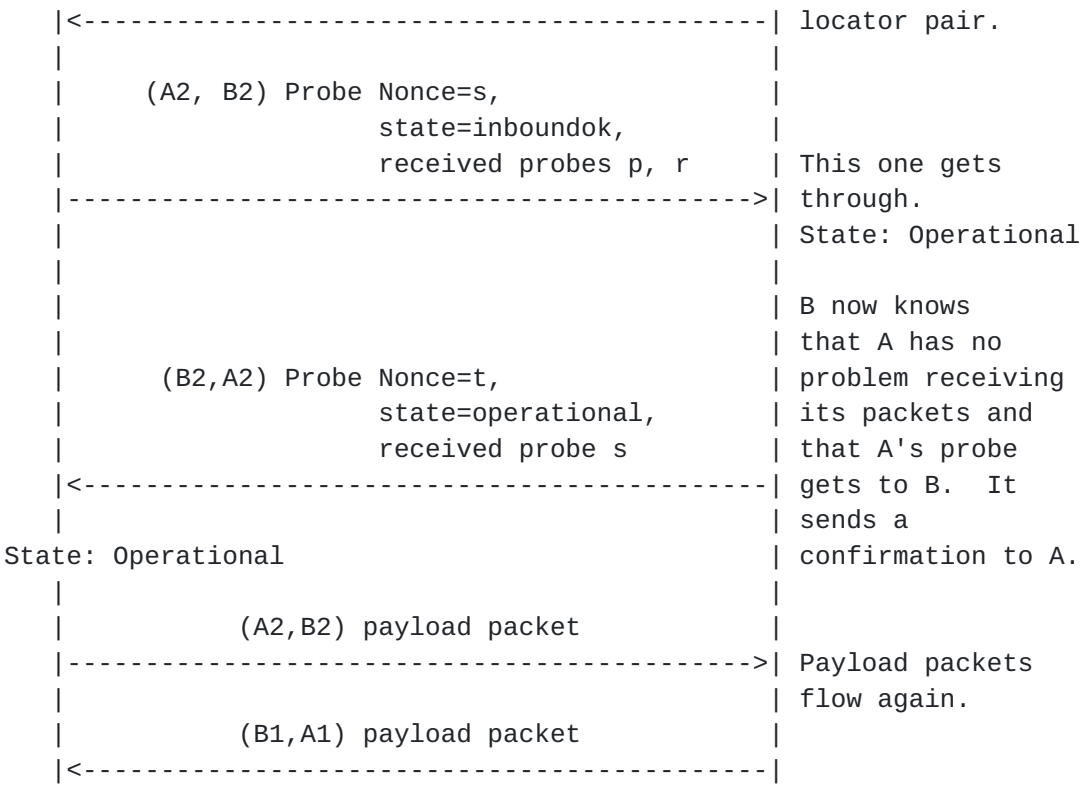




The next example shows when the failure for the current locator pair is in the other direction only. A has addresses A1 and A2, and B has addresses B1 and B2. The current communication is between A1 and B1, but A's packets no longer reach B using this pair.

EXAMPLE 5: One-Way Failure





Appendix B. Contributors

This document attempts to summarize the thoughts and unpublished contributions of many people, including MULTI6 WG design team members Marcelo Bagnulo Braun, Erik Nordmark, Geoff Huston, Kurtis Lindqvist, Margaret Wasserman, and Jukka Ylitalo; MOBIKE WG contributors Pasi Eronen, Tero Kivinen, Francis Dupont, Spencer Dawkins, and James Kempf; and HIP WG contributors such as Pekka Nikander. This document is also in debt to work done in the context of SCTP [RFC4960] and the Host Identity Protocol (HIP) multihoming and mobility extension [RFC5206].

Appendix C. Acknowledgements

The authors would also like to thank Christian Huitema, Pekka Savola, John Loughney, Sam Xia, Hannes Tschofenig, Sebastien Barre, Thomas Henderson, Matthijs Mekking, Deguang Le, Eric Gray, Dan Romascanu, Stephen Kent, Alberto Garcia, Bernard Aboba, Lars Eggert, Dave Ward, and Tim Polk 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
IMDEA Networks
Avda. del Mar Mediterraneo, 22
Leganes, Madrid 28918
Spain

EMail: iljitsch@muada.com