Network Working Group
Internet-Draft
Expires: April 27, 2006

J. Arkko
Ericsson
I. Beijnum
Muada

October 24, 2005

# Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming
draft-ietf-shim6-failure-detection-02

## Status of this Memo

## Copyright Notice

## Abstract

This document defines a mechanism for the detection of communication failures between two communicating hosts at IP layer, and an exploration protocol for switching to another pair of interfaces and/or addresses between the same hosts if a working pair can be found. The draft also discusses the roles of a multihoming protocol versus network attachment functions at IP and link layers.

Table of Contents

## 1. Introduction

The SHIM6 protocol extends IPv6 to support multihoming. This protocol is an IP layer mechanism that hides multihoming from applications [18]. 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 draft defines the mechanism and protocol to achieve both failure detection and locator pair exploration. This protocol is called REAchability Protocol (REAP). It designed to be carried within the SHIM6 protocol, but may also be used in other contexts.

The draft is structured as follows: Section 3 discusses prior work in this space, Section 4 defines a set of useful terms, Section 5 gives an overview of REAP, and Section 6 specifies the message formats and behaviour in detail. Section 7 discusses the security considerations of REAP.

For the purposes of this draft, we consider an address to be synonymous with a locator. We assume that there are other, higher level identifiers such as CGA public keys or HBA bindings that tie the different locators used by a node together [17].

## 2. Requirements language

In this document, the key words "MAY", "MUST, "MUST NOT", "OPTIONAL", "RECOMMENDED", "SHOULD", and "SHOULD NOT", are to be interpreted as described in [2].

## 3. Related Work

In SCTP [10], the addresses of the endpoints are learned in the connection setup phase either through listing them explicitly or via giving a DNS name that points to them. In order to provide a failover mechanism between multihomed hosts, SCTP selects one of the peer´s addresses as the primary address by the application running on top of SCTP. All data packets are sent to this address until there is a reason to choose another address, such as the failure of the primary address.

SCTP also tests the reachability of the peer endpoint´s addresses. This is done both via observing the data packets sent to the peer or via a periodic heartbeat when there is no data packets to send. Each time data packet retransmission is initiated (or when a heartbeat is not answered within the estimated round-trip time) an error counter is incremented. When a configured error limit is reached, the particular destination address is marked as inactive. The reception of an acknowledgement or heartbeat response clears the counter. Retransmission: When retransmitting the endpoint attempts pick the most "divergent" source-destination pair from the original source- destination pair to which the packet was transmitted. Rules for such selection are, however, left as implementation decisions in SCTP.

SCTP does not define how local knowledge (such as information learned from the link layer) should be used. SCTP also has no mechanism to deal with dynamic changes to the set of available addresses, although mechanisms for that are being developed [20].

The MOBIKE protocol [15] provides multihoming and mobility for VPN connections. Its failure detection and locator pair exploration is designed to work across mixed IPv4/IPv6 environments and NATs, as long as a path that allows bidirectional communication can be found.

Existing mechanisms at lower layers or in IKEv2 are used to detect failures, and upon failure MOBIKE attempts to explore all combinations of addresses to find a working pair. Such exploration is necessary when a problem affects both nodes. For instance, two nodes connected by two separate point-to-point links will be unable to switch to the other link if a failure occurs on the first one. While both communicating hosts are aware of each others´ addresses, only one end of the communication is in charge of deciding what address pair to use, however.

The mobility and multihoming specification for the HIP protocol [14] leaves the determination of when address updates are sent to a local policy, but suggests the use of local information and ICMP error messages.

Network attachment procedures are also relevant for multihoming. The IPv6 and MIP6 working groups have standardized mechanisms to learn about networks that a node has attached to. Basic IPv6 Neighbor Discovery was, however, designed primarily for static situations. The fully dynamic detection procedure has turned out to be a relatively complex procedure for mobile hosts, and it was not fully anticipated at the time IPv6 Neighbor Discovery or DHCP were being designed. As a result, enhanced or optimized mechanisms are being designed in the DHC and DNA working groups [13] [7].

ICE [16], STUN [11], and TURN [25] are also related mechanisms. They are primarily used for NAT detection and communication through NATs in IPv4 environment, for application such as as voice over IP. STUN uses a server in the Internet to discover the presence and type of NATs and the client´s public IP addresses and ports. TURN makes it possible to receive incoming connections in hosts behind NATs. ICE makes use of these protocols in peer-to-peer cooperative fashion, allowing participants to discover, create and verify mutual connectivity, and then use this connectivity for multimedia streams. While these mechanisms are not designed for dynamic and failure situations, they have many of the same requirements for the exploration of connectivity, as well as the requirement to deal with middleboxes.

Related work in the IPv6 area includes RFC 3484 [6] which defines source and destination address selection rules for IPv6 in situations where multiple candidate address pairs exist. RFC 3484 considers only a static situation, however, and does not take into account the effect of failures. In the MULTI6 working group [24] considers how applications can re-initiate connections after failures in the best way. This work differs from the shim-layer approach selected for further development in the working group with respect to the timing of the address selection. In the shim-layer approach failure detection and the selection of new addresses happens at any time, while [24] considers only the case when an application re-establishes connections.

An earlier SHIM6 document [19] discussed what kind of mechanisms can be used to detect whether the peer is still reachable at the currently used address. Two proposed mechanisms, Correspondent Unreachability Detection (CUD) and Forced Bidirectional Communication (FBD) were presented. CUD is based on getting upper layer positive feedback, and IPv6 NUD-like probing if there is no feedback. FBD is based on forcing bidirectional communication by adding keepalive messages when there is no other, payload traffic. FBD is the chosen mechanism in this document.

## 4. Definitions

This section defines terms useful in discussing the problem space.

### 4.1. Available Addresses

Multihoming 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:

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

o If the address is an IPv6 address, we additionally require that (a) the address is valid in the sense of RFC 2461 [3], and that (b) the address is not tentative in the sense of RFC 2462 [4]. 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 [8] even though implementations are probably better off using other addresses as long as there is an alternative.

o The address is a global unicast, unique local address [9], 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.

> IPv4 compatibility note: If this protocol were defined to handle IPv4, then RFC 1918 addresses would also need to be allowed.

o 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 the protocol described here. These mechanisms include IPv6 Neighbor Discovery and Address Autoconfiguration [3] [4], DHCP [5], and DNA mechanisms [7].

> IPv4 compatibility note: If IPv4 was supported in this protocol, then also mechanisms defined in [13] would need to be supported.

### 4.2. 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, at least one default router (if applicable) that could be used to send a packet with this address as a source 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 protocol described here. These mechanisms include IPv6 Neighbor Discovery [3] and link layer specific mechanisms.

> IPv4 compatibility note: In IPv4, mechanisms such as those defined in [13] could be used.

It is also possible for hosts to learn about routing failures for a particular selected source prefix, if suitable protocols for this purpose exist. Some proposals in this space have been made, see, for instance [21] and [24]. Potential approaches include overloading information in current IPv6 Router Advertisement or adding some new information in them. Similarly, hosts could learn information from servers that query the BGP routing tables.

### 4.3. 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 the sent 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, iff 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. Undirectionally operational address pair. A pair of locally operational addresses are said to be an unidirectionally operational address pair, iff packets sent with the first address as the source and the second address as the destination can be shown to reach the destination.

Both types of operational pairs could be discovered and monitored through the following 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 [3]. 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.

- Explicit reachability tests, such as keepalives or probes added when there´s only unidirectional payload traffic.

- 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, 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 [23]. These verifications can ensure that (practically) only on-path attackers can spoof the messages.

Note a multihoming protocol 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 [26]. This test can at the same time work as a means to ensure that an address pair is operational, as discussed in Section 5.2.

## 4.4. Current Address Pair

IP-layer solutions need to avoid sending packets concurrently over multiple paths; TCP behaves rather poorly in such circumstances. For this reason 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 work for new communications as well.

## 4.5. Miscellaneous

Addresses can become deprecated [3]. When other operational addresses exist, nodes generally wish to move their communications away from the deprecated addresses.

Similarly, IPv6 source address selection [6] may guide the selection of a particular source address - destination address pair.

## 5. Protocol Overview

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

### 5.1. Failure Detection

This process consists of three tasks. First, it is necessary to track local information from lower and upper layers. For instance, when link layer informs that we have no connection then we know there is a failure. Nodes SHOULD employ techniques listed in Section 4.1 and Section 4.2 to be aware of the local situation.

Similarly, it is necessary to track remote address information from the peer. For instance, the peer may inform that its currently used address is no longer in use. Techniques outside the scope of this document are used for this, for further information see [18].

The third task is to ensure verify reachability with the peer when the local and remote information indicates that communication should be possible. This needs to be performed only if there´s upper layer packets to be sent, however.

This document defines the protocol mechanisms only for the third task. We employ a technique called Forced Bidirectional Detection (FBD) to ensure reachability. This technique tests reachability only when there´s payload traffic. When there is no payload traffic, no tests will be performed, and no failure are assumed to exist.

Similarly, when there is bidirectional payload traffic, there is no need for FBD to do anything, as packets are already flowing as expected. However, if one of the peers is only receiving but not sending any other traffic, then FBD sends occasional keepalives to the other peer in order to let the peer know that its payload traffic is getting through. As a result, a node that is sending something to the peer but receives nothing in return can assume that there´s a failure.

### 5.2. 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 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 Event messages, in sequence, until it gets an Event message from B indicating that (a) B has received one of A´s messages and, obviously, (b) that B´s Event message gets back to A. B uses the same algorithm, but starts the process from the reception of the first Event message from A.

Upon changing to a new address pair, transport layer protocol needs to be informed so that it can perform a slow start, or some other form of adaptation to the possibly changed conditions. However, this functionality is outside the scope of REAP and is rather seen as a general multihoming issue.

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 API to be developed, however. In any case, this is another issue that we treat as being outside the scope of pure address exploration.

### 5.3. Exploration Order

The exploration process assumes an ability to pick current and alternative address pairs. This process results in a combinatorial explosion when there are many addresses on both sides. However, not all combinations are legal. In order to avoid congestion, we also can not explore all pairs without performing some kind of a back-off procedure.

Nodes MUST first consult RFC 3484 [6] Section 4 rules to determine what combinations of addresses are legal 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. 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 multihoming protocol also carries preference information in its messages [18].

> 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 [22].

One suggested good implementation strategy is to record the reachability test result (an on/off value) and multiply this by the age of the information. This allows recently tested address pairs to be chosen before old ones.

> IPv4 compatibility note: As has been noted in the context of MOBIKE, the existence of NATs can require that peers continuously monitor the operational status of address pairs, as otherwise NAT state related to a particular communication is lost, and the peer on the outer side of the NAT can no longer reach the peer inside the NAT.

Out of the set of possible candidate address pairs, nodes SHOULD attempt a test through all of them until a working pair is found. However, all nodes MUST do this sequentially and using an exponential back-off procedure. 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.

### 5.4. Protocol Design

REAP is designed as a modular part of SHIM6 in the hopes that it may also be useful in other contexts. This document defines how it is carried within SHIM6, but the actual protocol messages are self- contained so that it could be carried by other protocols as well.

Similarly, while this document defines a SHIM6 message that carries REAP, this message is only used when no other SHIM6 message is about to be sent that could be used to carry the REAP information. For instance, a Locator List Update message can be used to carry a REAP message that conveys reachability information.

The REAP design allows performing both failure detection and address pair exploration in the same sequence of messages, without a need to designate a specific point when the current address pair is declared inoperational and the search for a new pair begins. This is useful, as the loss of a small number of packets is not a proof that a problem exists. Integrated failure detection and exploration allows us to test multiple address pairs simultaneously, including the current pair in case it starts working again.

REAP also integrates a return routability function, making it unnecessary to perform another roundtrip before a newly discovered address can be taken into use.

This document defines a minimal set of parameters that are carried by the messages of the protocol. Specifically, we have limited the parameters to those that are necessary to find a working path. We note there may be extensions that are needed in the future for various reasons, such as the desire to support load balancing or finding best paths. An option format has been specified to allow this.

## 5.5. Example Protocol Runs

This section has examples of REAP protocol runs in typical scenarios. We start with the simplest scenario of two hosts, A and B, that have a SHIM6 connection with each other but are not currently sending any data. As neither side sends anything, they also do not expect anything back, so there are no messages at all:

```
Peer A                                              Peer B
   |                                                   |
   |                                                   |
   |                                                   |
   |                                                   |
   |                                                   |
   |                                                   |
   |                                                   |
   |                                                   |
```

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:

```
Peer A                                        Peer B
  |                                             |
  |              payload packet                 |
  |-------------------------------------------->|
  |                                             |
  |              payload packet                 |
  |<--------------------------------------------|
  |                                             |
  |              payload packet                 |
  |-------------------------------------------->|
  |                                             |
  |                                             |
```

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:

```
Peer A                                        Peer B
  |                                             |
  |              payload packet                 |
  |-------------------------------------------->|
  |                                             |
  |              payload packet                 |
  |-------------------------------------------->|
  |                                             |
  |              payload packet                 |
  |-------------------------------------------->|
  |                                             |
  |      REAP Event id=p,                       |
  |                 iseeyou=yes,                |
  |                 payload reception report    |
  |<--------------------------------------------|
  |                                             |
  |              payload packet                 |
  |-------------------------------------------->|
  |                                             |
  |                                             |
```

Finally, our last example involves a failure scenario. Here A has addresses A1 and A2 and B has addresses B1 and B2. The currently used address pairs are (A1, B1) and (B1, A1). The first of these becomes broken, which leads to an exploration process:

```
Peer A                                        Peer B
```

```
        |                                             |
        |           (A1,B1) payload packet            |
        |-------------------------------------------->|
        |                                             |
        |           (B1,A1) payload packet            |
        |<--------------------------------------------|
        |                                             | Path A1->B1
        |           (A1,B1) payload packet            | is now
        |----------------------------------------/    | broken
        |                                             |
        |                                             | Eventually, B
        |                                             | sends a com-
        |       (B1,A1) REAP Event id=p,              | plaint that
        |                          iseeyou=no         | it is not rec-
        |<--------------------------------------------| eiving anything
        |                                             |
     A realizes                                       |
     that it needs                                    |
     to start the                                     |
     exploration                                      |
        |                                             |
        |   (A1, B1) REAP Event id=q,                 |
        |                       iseeyou=yes           |
        |                       payload reception rep |
        |                       event reception rep(p)| But it gets lost
        |-------------------------------------/       | due to broken path
        |                                             |
     Retransmission                                   |
     to a different                                   |
     address                                          |
        |                                             |
        |   (A1, B2) REAP Event id=r,                 |
        |                       iseeyou=yes           |
        |                       payload reception rep |
        |                       event reception rep(p)| This one gets
        |-------------------------------------------->| through
        |                                             |
        |                                             |
        |                                             | B now knows
        |                                             | that A has no
        |    (B1,A1) REAP Event id=p,                 | problem to receive
        |                       iseeyou=yes,          | its packets and
        |                       event reception rep(r)| This one gets
        |<--------------------------------------------| that A has found
        |                                             | a new path to B
        |                                             |
        |           (A1,B2) payload packet            |
        |-------------------------------------------->| Payload packets
        |                                             | flow again
        |           (B1,A1) payload packet            |
        |<--------------------------------------------|
        |                                             |
```

### 5.6. Limitations

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 7, 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.

REAP does not support IPv4, but could be extended to do so. We have noted IPv4 compatibility issues where they exist.

# 6. Protocol Definition

## 6.1. SHIM6 Probe Message

The SHIM6 Probe message carries REAP messages when no other SHIM6 message needs to be sent. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       59      |  Hdr Ext Len  |0|  Type = TBD |   Reserved1 |0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Checksum           |           Reserved2           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Receiver Context Tag                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         Options                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Header

This value MUST be set to NO_NXT_HDR (59).

Type

This field identifies the Probe message and MUST be set to < TBD by IANA > (Probe).

Reserved1

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

Reserved2

This is a 16-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 32-bit field for the Context Tag the receiver has allocated for the context.

Options

This MUST contain at least the SHIM6 Event option and MAY contain other options.

A valid SHIM6 Probe message conforms to the format above and has a Receiver Context Tag that matches to context known by the receiver. The receiver processes a valid message by inspecting its options, and executing any actions specified for the SHIM6 Event option found within the options.

## 6.2. SHIM6 Event Option

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type = TBD         |0|            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                           REAP Event                          ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

This value MUST be set to < TBD by IANA > (Event Option).

0

This value MUST be set to 0, as in other SHIM6 options.

Length

This is the length of the option and MUST be calculated as specified in Section 5.14 of [18].

The processing rules for this option are the ones defined for the REAP Event field in Section 6.3.

## 6.3. REAP Event Message

REAP Event messages are the only messages in the REAP protocol. Their 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Message Type = 1           |            Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Y| Res |                   Identifier                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                          REAP Options                         ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type

This value identifies the REAP message, and MUST be set to 1 (Event).

Length

This is the length of the message excluding the Message Type and Length fields, expressed in bytes.

Y (The "I See You" flag)

This flag is set to 1 if the sender receives either payload packets or REAP messages from the peer, and 0 otherwise. The determination of when the sender receives something is made during the last Exploration Init Timeout seconds (see Section 6.6) when traffic was expected, i.e., when there was either payload traffic or REAP messages.

Upon reception, a value of 1 indicates that the receiver does not need to change its behaviour as the sender is already seeing its packets. A value of 0 indicates that the receiver MUST explore different outgoing address pairs.

Res

This 3-bit reserved field MUST be set to zero when sending, and ignored on receipt.

Identifier

This identifies this particular instance of an Event message. This value SHOULD be generated using a random number generator that is known to have good randomness properties [1]. Upon reception, Identifier values are copied onto Event Reception Report options. This allows them to be used for both identifying which Events were received as well as for performing a return routability test.

REAP Options

This field contains zero or REAP options. Unrecognized options MUST be ignored upon receipt. All implementations MUST support the options defined in Section 6.4, however.

### 6.4. REAP Options

The general option 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Option Type          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                          Option Data                          ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

This value identifies the option.

Length

This is the length of the option excluding the Option Type and Length fields, expressed in bytes.

Option Data

Option-specific content.

#### 6.4.1. Payload Reception Report

This option SHOULD be included in any Event message when the sender has recently (within the last Exploration Init Timeout seconds) received payload packets from the peer. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Option Type = 1         |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                           Suboptions                          ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

This value identifies the option and MUST be set to 1 (Payload Reception Report).

Length

This is the length of the option excluding the Option Type and Length fields, expressed in bytes.

Suboptions

This field is reserved for possible future REAP Options that are carried (recursively) within this option. Unrecognized options MUST be ignored upon receipt. Currently there are no defined options that can be carried here.

> IPv4 compatibility note: If IPv4 and NATs would need to be supported, then it might be necessary to indicate what addresses and port numbers were used in the received payload packets.

### 6.4.2. Event Reception Report

This option MUST be included in any Event message when the sender has recently (within the last Exploration Init Timeout seconds) received Event messages from the peer. Depending on MTU and timing considerations, the sender MAY, however, include options for only some of the received Event messages. All implementations MUST support sending of at least five such options, however.

The format of this option 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Option Type = 2         |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|                          Identifier                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                          Suboptions                           ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

This value identifies the option and MUST be set to 1 (Payload Reception Report).

Length

This is the length of the option excluding the Option Type and Length fields, expressed in bytes.

R

This is a 1 bit reserved field, MUST be set to zero and ignored on receipt.

Identifier

This 31 bit field carries the identifier of the Event message that was recently received.

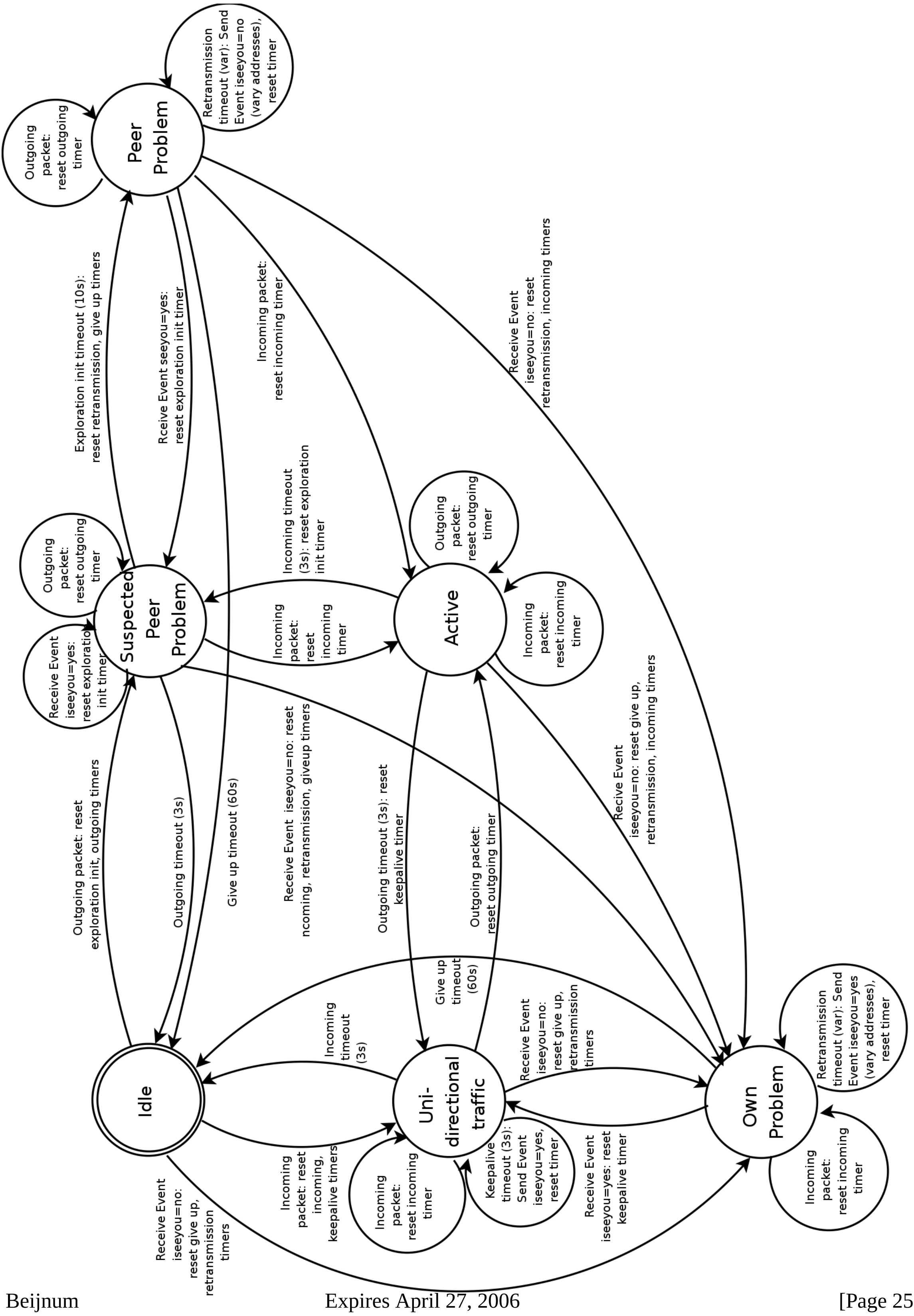
Suboptions

This field is reserved for possible future REAP Options that are carried (recursively) within this option. Unrecognized options MUST be ignored upon receipt. Currently there are no defined options that can be carried here.

> IPv4 compatibility note: If IPv4 and NATs would need to be supported, then it might be necessary to indicate what addresses and port numbers were used in the received payload packets.

### 6.5. State Machine

A suggested state machine to implement REAP is shown below.


Peer Problem
Suspected Peer Problem
Active
Idle
Uni-directional traffic
Own Problem
Outgoing packet: reset outgoing timer
Retransmission timeout (var): Send Event iseeyou=no (vary addresses), reset timer
Exploration init timeout (10s): reset retransmission, give up timers
Rceive Event seeyou=yes: reset exploration init timer
Incoming packet: reset incoming timer
Receive Event iseeyou=no: reset retransmission, incoming timers
Outgoing packet: reset outgoing timer
Receive Event iseeyou=yes: reset exploration init timer
Incoming timeout (3s): reset exploration init timer
Incoming packet: reset incoming timer
Outgoing packet: reset outgoing timer
Incoming packet: reset incoming timer
Outgoing packet: reset exploration init, outgoing timers
Outgoing timeout (3s)
Give up timeout (60s)
Receive Event iseeyou=no: reset ncoming, retransmission, giveup timers
Outgoing timeout (3s): reset keepalive timer
Outgoing packet: reset outgoing timer
Recive Event iseeyou=no: reset give up, retransmission, incoming timers
Give up timeout (60s)
Incoming timeout (3s)
Receive Event iseeyou=no: reset give up, retransmission timers
Retransmission timeout (var): Send Event iseeyou=yes (vary addresses), reset timer
Incoming packet: reset incoming timer
Receive Event iseeyou=yes: reset keepalive timer
Keepalive timeout (3s): Send Event iseeyou=yes, reset timer
Incoming packet: reset incoming timer
Incoming packet: reset incoming, keepalive timers
Receive Event iseeyou=no: reset give up, retransmission timers

This state machine still has some known open issues. One issue is that it does not represent other events than those present in a static situation. For instance, the loss of an address, or peer telling us of its new addresses should also affect the state machine. A more serious issue is that the machine treats all flows of under 3 seconds as something that do not need to be acknowledged. This can be easily corrected, but we are struggling with the support of this while simultaneously not having to perform extra exploration when the traffic flow legitimately ends.

## 6.6. Protocol Constants

The following protocol constants are defined:

| | |
|---|---|
| Exploration Init Timeout | 10 seconds |
| Incoming Timeout | 3 seconds |
| Outgoing Timeout | 3 seconds |
| Give Up Timeout | 60 seconds |
| Keepalive Timeout | 3 seconds |

## 7. 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 working. 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. However, this protocol has weak resistance against incorrect information from these sources 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 [23], link layer security, or the use of [12] to protect IPv6 Router and Neighbor Discovery.

This protocol is designed to be used in situations where other parts of the stack have ensured that a set of addresses belong together, such as via SHIM6 HBAs [17]. That is, REAP itself provides no assurance that a set of addresses belongs to the same host. Similarly, REAP provides only minimal protection against third party flooding attacks; when REAP is run its Event 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. In SHIM6 this is performed by binding all operations to context tags.

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.

## 8. IANA Considerations

This document requires the allocation of a SHIM6 message Type code for the SHIM6 Probe message (Section 6.1).

This document requires also the allocation of a SHIM6 option Type code for the SHIM6 Event option (Section 6.2).

This document creates two new name spaces under the new SHIM6 REAP repository. The first name space is for REAP Message Type (Section 6.3) and it has one reserved value (0) and one defined value, 1 (Event). Further allocations within this 16-bit field can be made through Standards Action or IESG Approval. The range from 65000 to 65535 is reserved for experimental use.

The second name space is for REAP Option Type (Section 6.4) and it has one reserved value (0) and two defined values, 1 (Payload Reception Report defined in Section 6.4.1) and 2 (Event Reception Report defined in Section 6.4.2). Further allocations within this 16-bit field can be made through Specification Required. The range from 65535 to 65535 is reserved for experimental use.

## 9. References

### 9.1. Normative References

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

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

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

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

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

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

[7] Choi, J., "Detecting Network Attachment in IPv6 Goals", draft-ietf-dna-goals-00 (work in progress), June 2004.

[8] Moore, N., "Optimistic Duplicate Address Detection for IPv6", draft-ietf-ipv6-optimistic-dad-01 (work in progress), June 2004.

[9] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", draft-ietf-ipv6-unique-local-addr-05 (work in progress), June 2004.

### 9.2. Informative References

[10] 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.

[11] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

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

[13] Aboba, B., "Detection of Network Attachment (DNA) in IPv4",

draft-ietf-dhc-dna-ipv4-08 (work in progress), July 2004.

[14] Nikander, P., "End-Host Mobility and Multi-Homing with Host Identity Protocol", draft-ietf-hip-mm-00 (work in progress), October 2004.

[15] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", draft-ietf-mobike-protocol-03 (work in progress), September 2005.

[16] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Multimedia Session Establishment Protocols", draft-ietf-mmusic-ice-02 (work in progress), July 2004.

[17] Bagnulo, M., "Hash Based Addresses (HBA)", draft-ietf-shim6-hba-00 (work in progress), July 2005.

[18] Nordmark, E., "Level 3 multihoming shim protocol", draft-ietf-shim6-proto-00 (work in progress), October 2005.

[19] Beijnum, I., "Shim6 Reachability Detection", draft-ietf-shim6-reach-detect-00 (work in progress), July 2005.

[20] Stewart, R., "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", draft-ietf-tsvwg-addip-sctp-10 (work in progress), January 2005.

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

[22] Crocker, D., "Framework for Common Endpoint Locator Pools", draft-crocker-celp-00 (work in progress), February 2004.

[23] Gont, F., "ICMP attacks against TCP", draft-gont-tcpm-icmp-attacks-00 (work in progress), August 2004.

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

[25] Rosenberg, J., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-05 (work in progress), July 2004.

[26] 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.

## 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, Iljitsch van Beijnum, Erik Nordmark, Geoff Huston, Margaret Wasserman, and Jukka Ylitalo, the MOBIKE WG contributors Pasi Eronen, Tero Kivinen, Francis Dupont, Spencer Dawkins, and James Kempf, and my colleague Pekka Nikander at Ericsson. This draft is also in debt to work done in the context of SCTP [10] and HIP [14].

## Appendix B. Acknowledgements

The author would also like to thank Christian Huitema, Pekka Savola, and Hannes Tschofenig for interesting discussions in this problem space, and for their comments on earlier versions of this draft.

## Authors´ Addresses

Jari Arkko
Ericsson
Jorvas  02420
Finland

Email: jari.arkko@ericsson.com

Iljitsch van Beijnum
Muada
The Netherlands

Email: iljitsch@muada.com

## Intellectual Property Statement

## Disclaimer of Validity

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.

## Copyright Statement

## Acknowledgment

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