Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 9, 2010

F. Templin, Ed.
Boeing Research & Technology

June 7, 2010

The Subnetwork Encapsulation and Adaptation Layer (SEAL) draft-templin-intarea-seal-15.txt

Abstract

For the purpose of this document, a subnetwork is defined as a virtual topology configured over a connected IP network routing region and bounded by encapsulating border nodes. These virtual topologies may span multiple IP and/or sub-IP layer forwarding hops, and can introduce failure modes due to packet duplication and/or links with diverse Maximum Transmission Units (MTUs). This document specifies a Subnetwork Encapsulation and Adaptation Layer (SEAL) that accommodates such virtual topologies over diverse underlying link technologies.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on December 9, 2010.

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

As Internet technology and communication has grown and matured, many techniques have developed that use virtual topologies (including tunnels of one form or another) over an actual network that supports the Internet Protocol (IP) [RFC0791][RFC2460]. Those virtual topologies have elements that appear as one hop in the virtual topology, but are actually multiple IP or sub-IP layer hops. These multiple hops often have quite diverse properties that are often not even visible to the endpoints of the virtual hop. This introduces failure modes that are not dealt with well in current approaches.

The use of IP encapsulation has long been considered as the means for creating such virtual topologies. However, the insertion of an outer IP header reduces the effective path MTU visible to the inner network layer. When IPv4 is used, this reduced MTU can be accommodated through the use of IPv4 fragmentation, but unmitigated in-the-network fragmentation has been found to be harmful through operational experience and studies conducted over the course of many years [FRAG][FOLK][RFC4963]. Additionally, classical path MTU discovery [RFC1191] has known operational issues that are exacerbated by in-the-network tunnels [RFC2923][RFC4459]. The following subsections present further details on the motivation and approach for addressing these issues.

1.1. Motivation

Before discussing the approach, it is necessary to first understand the problems. In both the Internet and private-use networks today, IPv4 is ubiquitously deployed as the Layer 3 protocol. The two primary functions of IPv4 are to provide for 1) addressing, and 2) a fragmentation and reassembly capability used to accommodate links with diverse MTUs. While it is well known that the IPv4 address space is rapidly becoming depleted, there is a lesser-known but growing consensus that other IPv4 protocol limitations have already or may soon become problematic.

First, the IPv4 header Identification field is only 16 bits in length, meaning that at most 2^16 unique packets with the same (source, destination, protocol)-tuple may be active in the Internet at a given time [I-D.ietf-intarea-ipv4-id-update]. Due to the escalating deployment of high-speed links (e.g., 1Gbps Ethernet), however, this number may soon become too small by several orders of magnitude for high data rate packet sources such as tunnel endpoints [RFC4963]. Furthermore, there are many well-known limitations pertaining to IPv4 fragmentation and reassembly - even to the point that it has been deemed "harmful" in both classic and modern-day studies (cited above). In particular, IPv4 fragmentation raises

issues ranging from minor annoyances (e.g., in-the-network router fragmentation) to the potential for major integrity issues (e.g., mis-association of the fragments of multiple IP packets during reassembly [RFC4963]).

As a result of these perceived limitations, a fragmentation-avoiding technique for discovering the MTU of the forward path from a source to a destination node was devised through the deliberations of the Path MTU Discovery Working Group (PMTUDWG) during the late 1980's through early 1990's (see Appendix D). In this method, the source node provides explicit instructions to routers in the path to discard the packet and return an ICMP error message if an MTU restriction is encountered. However, this approach has several serious shortcomings that lead to an overall "brittleness" [RFC2923].

In particular, site border routers in the Internet are being configured more and more to discard ICMP error messages coming from the outside world. This is due in large part to the fact that malicious spoofing of error messages in the Internet is made simple since there is no way to authenticate the source of the messages [I-D.ietf-tcpm-icmp-attacks]. Furthermore, when a source node that requires ICMP error message feedback when a packet is dropped due to an MTU restriction does not receive the messages, a path MTU-related black hole occurs. This means that the source will continue to send packets that are too large and never receive an indication from the network that they are being discarded. This behavior has been confirmed through documented studies showing clear evidence of path MTU discovery failures in the Internet today [TBIT][WAND].

The issues with both IPv4 fragmentation and this "classical" method of path MTU discovery are exacerbated further when IP tunneling is used [RFC4459]. For example, ingress tunnel endpoints (ITEs) may be required to forward encapsulated packets into the subnetwork on behalf of hundreds, thousands, or even more original sources in the end site. If the ITE allows IPv4 fragmentation on the encapsulated packets, persistent fragmentation could lead to undetected data corruption due to Identification field wrapping. If the ITE instead uses classical IPv4 path MTU discovery, it may be inconvenienced by excessive ICMP error messages coming from the subnetwork that may be either suspect or contain insufficient information for translation into error messages to be returned to the original sources.

Although recent works have led to the development of a robust end-to-end MTU determination scheme [RFC4821], this approach requires tunnels to present a consistent MTU the same as for ordinary links on the end-to-end path. Moreover, in current practice existing tunneling protocols mask the MTU issues by selecting a "lowest common denominator" MTU that may be much smaller than necessary for most

paths and difficult to change at a later date. Due to these many consideration, a new approach to accommodate tunnels over links with diverse MTUs is necessary.

1.2. Approach

For the purpose of this document, a subnetwork is defined as a virtual topology configured over a connected network routing region and bounded by encapsulating border nodes. Examples include the global Internet interdomain routing core, Mobile Ad hoc Networks (MANETs) and enterprise networks. Subnetwork border nodes forward unicast and multicast packets over the virtual topology across multiple IP and/or sub-IP layer forwarding hops that may introduce packet duplication and/or traverse links with diverse Maximum Transmission Units (MTUs).

This document introduces a Subnetwork Encapsulation and Adaptation Layer (SEAL) for tunneling network layer protocols (e.g., IP, OSI, etc.) over IP subnetworks that connect Ingress and Egress Tunnel Endpoints (ITEs/ETEs) of border nodes. It provides a modular specification designed to be tailored to specific associated tunneling protocols. A transport-mode of operation is also possible, and described in Appendix C. SEAL accommodates links with diverse MTUs, protects against off-path denial-of-service attacks, and supports efficient duplicate packet detection through the use of a minimal mid-layer encapsulation.

SEAL specifically treats tunnels that traverse the subnetwork as unidirectional links that must support network layer services. As for any link, tunnels that use SEAL must provide suitable networking services including best-effort datagram delivery, integrity and consistent handling of packets of various sizes. As for any link whose media cannot provide suitable services natively, tunnels that use SEAL employ link-level adaptation functions to meet the legitimate expectations of the network layer service. As this is essentially a link level adaptation, SEAL is therefore permitted to alter packets within the subnetwork as long as it restores them to their original form when they exit the subnetwork. The mechanisms described within this document are designed precisely for this purpose.

SEAL encapsulation introduces an extended Identification field for packet identification and a mid-layer segmentation and reassembly capability that allows simplified cutting and pasting of packets. Moreover, SEAL senses in-the-network fragmentation as a "noise" indication that packet sizing parameters are "out of tune" with respect to the network path. As a result, SEAL can naturally tune its packet sizing parameters to eliminate the in-the-network

fragmentation. This approach is in contrast to existing tunneling protocol practices which seek to avoid MTU issues by selecting a "lowest common denominator" MTU that may be overly conservative for many tunnels and difficult to change even when larger MTUs become available.

The following sections provide the SEAL normative specifications, while the appendices present non-normative additional considerations.

2. Terminology and Requirements

The following terms are defined within the scope of this document:

subnetwork
: a virtual topology configured over a connected network routing region and bounded by encapsulating border nodes.

Ingress Tunnel Endpoint
: a virtual interface over which an encapsulating border node (host or router) sends encapsulated packets into the subnetwork.

Egress Tunnel Endpoint
: a virtual interface over which an encapsulating border node (host or router) receives encapsulated packets from the subnetwork.

inner packet
: an unencapsulated network layer protocol packet (e.g., IPv6 [RFC2460], IPv4 [RFC0791], OSI/CLNP [RFC1070], etc.) before any mid-layer or outer encapsulations are added. Internet protocol numbers that identify inner packets are found in the IANA Internet Protocol registry [RFC3232].

mid-layer packet
: a packet resulting from adding mid-layer encapsulating headers to an inner packet.

outer IP packet
: a packet resulting from adding an outer IP header to a mid-layer packet.

packet-in-error
: the leading portion of an invoking data packet encapsulated in the body of an error control message (e.g., an ICMPv4 [RFC0792] error message, an ICMPv6 [RFC4443] error message, etc.).

IP, IPvX, IPvY
used to generically refer to either IP protocol version, i.e., IPv4 or IPv6.

The following abbreviations correspond to terms used within this document and elsewhere in common Internetworking nomenclature:

DF - the IPv4 header "Don't Fragment" flag [RFC0791]

ETE - Egress Tunnel Endpoint

HLEN - the sum of MHLEN and OHLEN

ITE - Ingress Tunnel Endpoint

MHLEN - the length of any mid-layer headers and trailers

OHLEN - the length of the outer encapsulating headers and trailers, including the outer IP header, the SEAL header and any other outer headers and trailers.

PTB - a Packet Too Big message recognized by the inner network layer, e.g., an ICMPv6 "Packet Too Big" message [RFC4443], an ICMPv4 "Fragmentation Needed" message [RFC0792], etc.

S_MRU - the SEAL Maximum Reassembly Unit

S_MSS - the SEAL Maximum Segment Size

SCMP - the SEAL Control Message Protocol

SEAL_ID - an Identification value, randomly initialized and monotonically incremented for each SEAL protocol packet

SEAL_PORT - a TCP/UDP service port number used for SEAL

SEAL_PROTO - an IPv4 protocol number used for SEAL

TE - Tunnel Endpoint (i.e., either ingress or egress)

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]. When used in lower case (e.g., must, must not, etc.), these words MUST NOT be interpreted as described in [RFC2119], but are rather interpreted as they would be in common English.

3. Applicability Statement

SEAL was originally motivated by the specific case of subnetwork abstraction for Mobile Ad hoc Networks (MANETs), however it soon became apparent that the domain of applicability also extends to subnetwork abstractions of enterprise networks, ISP networks, SOHO networks, the interdomain routing core, and any other networking scenario involving IP encapsulation. SEAL and its associated technologies (including Virtual Enterprise Traversal (VET) [I-D.templin-intarea-vet]) are functional building blocks for a new Internetworking architecture based on Routing and Addressing in Networks with Global Enterprise Recursion (RANGER) [RFC5720][I-D.russert-rangers] and the Internet Routing Overlay Network (IRON) [I-D.templin-iron].

SEAL provides a network sublayer for encapsulation of an inner network layer packet within outer encapsulating headers. For example, for IPvX in IPvY encapsulation (e.g., as IPv4/SEAL/IPv6), the SEAL header appears as a subnetwork encapsulation as seen by the inner IP layer. SEAL can also be used as a sublayer within a UDP data payload (e.g., as IPv4/UDP/SEAL/IPv6 similar to Teredo [RFC4380]), where UDP encapsulation is typically used for operation over subnetworks that give preferential treatment to the "core" Internet protocols (i.e., TCP and UDP). The SEAL header is processed the same as for IPv6 extension headers, i.e., it is not part of the outer IP header but rather allows for the creation of an arbitrarily extensible chain of headers in the same way that IPv6 does.

SEAL supports a segmentation and reassembly capability for adapting the network layer to the underlying subnetwork characteristics, where the Egress Tunnel Endpoint (ETE) determines how much or how little reassembly it is willing to support. In the limiting case, the ETE acts as a passive observer that simply informs the Ingress Tunnel Endpoint (ITE) of any MTU limitations and otherwise discards all packets that arrive as multiple fragments. This mode is useful for determining an appropriate MTU for tunnels between performance-critical routers connected to high data rate subnetworks such as the Internet DFZ, as well as for other uses in which reassembly would present too great of a burden for the routers or end systems.

When the ETE supports reassembly, the tunnel can be used to transport packets that are too large to traverse the path without fragmentation. In this mode, the ITE determines the tunnel MTU based on the largest packet the ETE is capable of reassembling rather than on the MTU of the smallest link in the path. Therefore, tunnel endpoints that use SEAL can transport packets that are much larger than the underlying subnetwork links themselves can carry in a single piece.

SEAL tunnels may be configured over paths that include not only ordinary physical links, but also virtual links that may include other SEAL tunnels. An example application would be linking two geographically remote supercomputer centers with large MTU links by configuring a SEAL tunnel across the Internet. A second example would be support for sub-IP segmentation over low-end links, i.e., especially over wireless transmission media such as IEEE 802.15.4, broadcast radio links in Mobile Ad-hoc Networks (MANETs), Very High Frequency (VHF) civil aviation data links, etc.

Many other use case examples are anticipated, and will be identified as further experience is gained.

4. SEAL Protocol Specification

The following sections specify the operation of the SEAL protocol.

4.1. Model of Operation

SEAL is an encapsulation sublayer that supports a multi-level segmentation and reassembly capability for the transmission of unicast and multicast packets across an underlying IP subnetwork with heterogeneous links. First, the ITE can use IPv4 fragmentation to fragment inner IPv4 packets before SEAL encapsulation if necessary. Secondly, the SEAL layer itself provides a simple cutting-and-pasting capability for mid-layer packets to avoid IP fragmentation on the outer packet. Finally, ordinary IP fragmentation is permitted on the outer packet after SEAL encapsulation and is used to detect and tune out any in-the-network fragmentation.

SEAL-enabled ITEs encapsulate each inner packet in any mid-layer headers and trailers, segment the resulting mid-layer packet into multiple segments if necessary, then append a SEAL header and any outer encapsulations to each segment. As an example, for IPv6-in-IPv4 encapsulation a single-segment inner IPv6 packet encapsulated in any mid-layer headers and trailers, followed by the SEAL header, followed by any outer headers and trailers, followed by an outer IPv4 header would appear as shown in Figure 1:

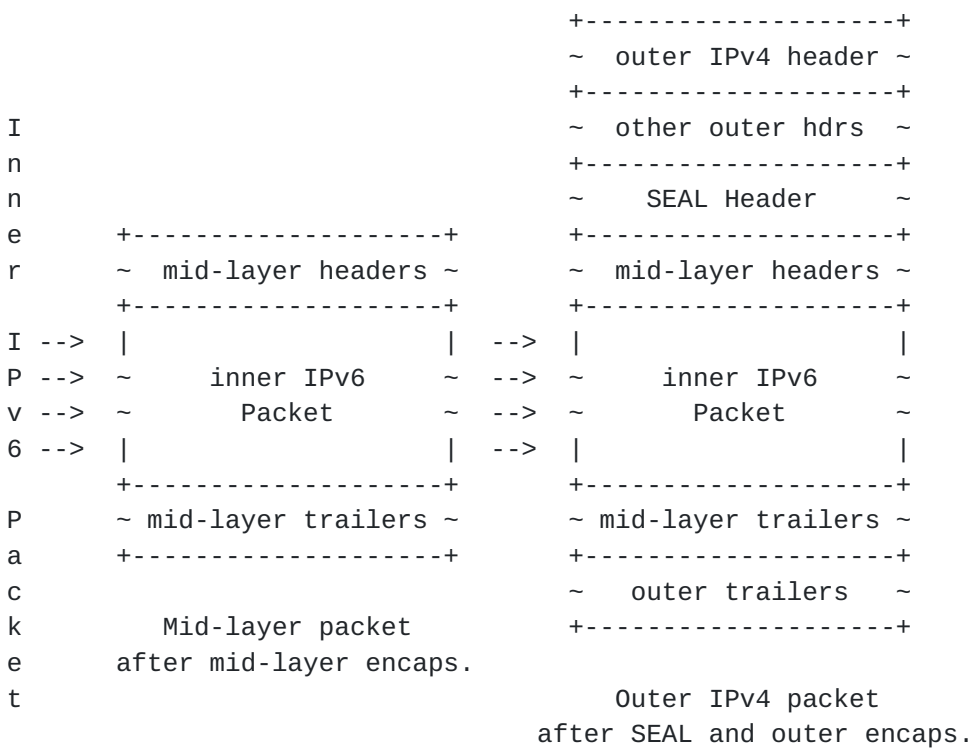

Figure 1: SEAL Encapsulation - Single Segment

As a second example, for IPv4-in-IPv6 encapsulation an inner IPv4 packet requiring three SEAL segments would appear as three separate outer IPv6 packets, where the mid-layer headers are carried only in segment 0 and the mid-layer trailers are carried in segment 2 as shown in Figure 2:

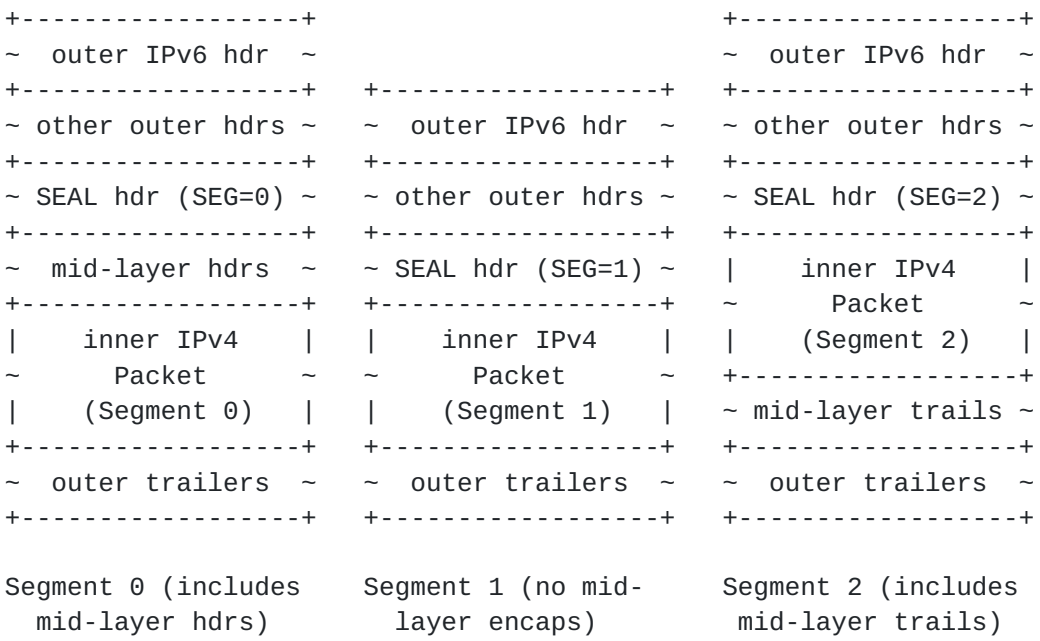

Figure 2: SEAL Encapsulation - Multiple Segments

The SEAL header itself is inserted according to the specific tunneling protocol. Examples include the following:

- For simple encapsulation of an inner network layer packet within an outer IPvX header (e.g., [RFC1070][RFC2003][RFC2473][RFC4213], etc.), the SEAL header is inserted between the inner packet and outer IPvX headers as: IPvX/SEAL/{inner packet}.

- For encapsulations over transports such as UDP (e.g., [RFC4380]), the SEAL header is inserted between the outer transport layer header and the mid-layer packet, e.g., as IPvX/UDP/SEAL/{mid-layer packet}. Here, the UDP header is seen as an "other outer header".

SEAL-encapsulated packets include a SEAL_ID to uniquely identify each packet. Routers within the subnetwork use the SEAL_ID for duplicate packet detection, and TEs use the SEAL_ID for SEAL segmentation/ reassembly and protection against off-path attacks. The following sections specify the SEAL header format and SEAL-related operations of the ITE and ETE, respectively.

4.2. SEAL Header Format

The SEAL header is formatted as follows:

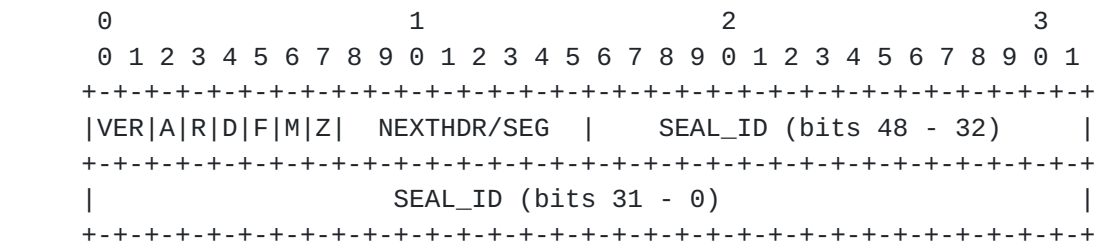

Figure 3: SEAL Header Format

where the header fields are defined as:

VER (2)
: a 2-bit version field. This document specifies Version 0 of the SEAL protocol, i.e., the VER field encodes the value 0.

A (1)
: the "Acknowledgement Requested" bit. Set to 1 by the ITE in data packets if it wishes to receive an explicit acknowledgement from the ETE.

R (1)
: the "Redirect" bit. Set to 0 unless otherwise specified in other documents.

D (1)
: the "Discard" bit. Set to 0 unless otherwise specified in other documents.

F (1)
: the "First Segment" bit. Set to 1 if this SEAL protocol packet contains the first segment (i.e., Segment #0) of a mid-layer packet.

M (1)
: the "More Segments" bit. Set to 1 if this SEAL protocol packet contains a non-final segment of a multi-segment mid-layer packet.

Z (1)
: a 1-bit "Reserved" field. Set to zero by the ITE and ignored by the ETE.

NEXTHDR/SEG (8) an 8-bit field. When 'F'=1, encodes the next header Internet Protocol number the same as for the IPv4 protocol and IPv6 next header fields. When 'F'=0, encodes a segment number of a multi-segment mid-layer packet. (The segment number 0 is reserved.)

SEAL_ID (48)
a 48-bit Identification field.

Setting of the various bits and fields of the SEAL header is specified in the following sections.

4.3. ITE Specification

4.3.1. Tunnel Interface MTU

The ITE configures a tunnel virtual interface over one or more underlying links that connect the border node to the subnetwork. The tunnel interface must present a fixed MTU to Layer 3 as the size for admission of inner packets into the tunnel. Since the tunnel interface may support a large set of ETEs that accept widely varying maximum packet sizes, however, a number of factors should be taken into consideration when selecting a tunnel interface MTU.

Due to the ubiquitous deployment of standard Ethernet and similar networking gear, the nominal Internet cell size has become 1500 bytes; this is the de facto size that end systems have come to expect will either be delivered by the network without loss due to an MTU restriction on the path or a suitable ICMP Packet Too Big (PTB) message returned. When the 1500 byte packets sent by end systems incur additional encapsulation at an ITE, however, they may be dropped silently since the network may not always deliver the necessary PTBs [RFC2923].

The ITE should therefore set a tunnel virtual interface MTU of at least 1500 bytes plus extra room to accommodate any additional encapsulations that may occur on the path from the original source. The ITE can set larger MTU values still, but should select a value that is not so large as to cause excessive PTBs coming from within the tunnel interface. The ITE can also set smaller MTU values; however, care must be taken not to set so small a value that original sources would experience an MTU underflow. In particular, IPv6 sources must see a minimum path MTU of 1280 bytes, and IPv4 sources should see a minimum path MTU of 576 bytes.

The ITE can alternatively set an indefinite MTU on the tunnel virtual interface such that all inner packets are admitted into the interface without regard to size. For ITEs that host applications, this option must be carefully coordinated with protocol stack upper layers, since some upper layer protocols (e.g., TCP) derive their packet sizing parameters from the MTU of the outgoing interface and as such may select too large an initial size. This is not a problem for upper layers that use conservative initial maximum segment size estimates and/or when the tunnel interface can reduce the upper layer's maximum

segment size (e.g., the size advertised in the TCP MSS option) based on the per-neighbor MTU.

The inner network layer protocol consults the tunnel interface MTU when admitting a packet into the interface. For inner IPv4 packets with the IPv4 Don't Fragment (DF) bit set to 0, if the packet is larger than the tunnel interface MTU the inner IPv4 layer uses IPv4 fragmentation to break the packet into fragments no larger than the tunnel interface MTU. The ITE then admits each fragment into the tunnel as an independent packet.

For all other inner packets, the ITE admits the packet if it is no larger than the tunnel interface MTU; otherwise, it drops the packet and sends a PTB error message to the source with the MTU value set to the tunnel interface MTU. The message must contain as much of the invoking packet as possible without the entire message exceeding the network layer minimum MTU (e.g., 576 bytes for IPv4, 1280 bytes for IPv6, etc.).

Note that when the tunnel interface sets an indefinite MTU the ITE unconditionally admits all packets into the interface without fragmentation. In light of the above considerations, it is RECOMMENDED that the ITE configure an indefinite MTU on the tunnel virtual interface and adapt to any per-neighbor MTU limitations within the tunnel virtual interface as described in the following sections.

4.3.2. Tunnel Interface Soft State

For each ETE, the ITE maintains soft state within the tunnel interface (e.g., in a neighbor cache) used to support inner fragmentation and SEAL segmentation for packets admitted into the tunnel interface. The soft state includes the following:

- a Mid-layer Header Length (MHLEN); set to the length of any mid-layer encapsulation headers and trailers that must be added before SEAL segmentation.

- an Outer Header Length (OHLEN); set to the length of the outer IP, SEAL and other outer encapsulation headers and trailers.

- a total Header Length (HLEN); set to MHLEN plus OHLEN.

- a SEAL Maximum Segment Size (S_MSS). The ITE initializes S_MSS to the underlying interface MTU if the underlying interface MTU can be determined (otherwise, the ITE initializes S_MSS to "infinity"). The ITE decreases or increased S_MSS based on any SCMP "MTU Report" messages received (see Section 4.5).

- a SEAL Maximum Reassembly Unit (S_MRU). The ITE initializes S_MRU to "infinity" and decreases or increases S_MRU based on any SCMP MTU Report messages received (see Section 4.5). When (S_MRU>(S_MSS*256)), the ITE uses (S_MSS*256) as the effective S_MRU value.

Note that S_MSS and S_MRU include the length of the outer and mid-layer encapsulating headers and trailers (i.e., HLEN), since the ETE must retain the headers and trailers during reassembly. Note also that the ITE maintains S_MSS and S_MRU as 32-bit values such that inner packets larger than 64KB (e.g., IPv6 jumbograms [RFC2675]) can be accommodated when appropriate for a given subnetwork.

4.3.3. Admitting Packets into the Tunnel

After the ITE admits an inner packet/fragment into the tunnel interface, it uses the following algorithm to determine whether the packet can be accommodated and (if so) whether (further) inner IP fragmentation is needed:

- if the inner packet is unfragmentable (e.g., an IPv6 packet, an IPv4 packet with DF=1, etc.), and the packet is larger than (MAX(S_MRU, S_MSS) - HLEN), the ITE drops the packet and sends a PTB message to the original source with an MTU value of (MAX(S_MRU, S_MSS) - HLEN); else,

- if the inner packet is fragmentable (e.g., an IPv4 packet with DF=0), and the packet is larger than (foo) bytes, the ITE uses inner fragmentation to break the packet into fragments no larger than (foo) bytes; else,

- the ITE processes the packet without inner fragmentation.

In the above, the ITE must track whether the tunnel interface is using header compression. If so, the ITE must include the length of the uncompressed headers and trailers when calculating HLEN. Note also in the above that the ITE is permitted to admit inner packets into the tunnel that can be accommodated in a single SEAL segment (i.e., no larger than S_MSS) even if they are larger than the ETE would be willing to reassemble if fragmented (i.e., larger than S_MRU) - see: Section 4.4.1.

When the ITE uses inner fragmentation, it should use a "safe" fragment size of (foo) bytes that would be highly unlikely to incur an outer IP MTU restriction within the tunnel. If the ITE can determine a larger fragment size (e.g., via probing), it should use the larger size for inner fragmentation. In the absence of deterministic information, it is RECOMMENDED that the ITE set (foo)

to 1280.

4.3.4. Mid-Layer Encapsulation

After inner IP fragmentation (if necessary), the ITE next encapsulates each inner packet/fragment in the MHLEN bytes of mid-layer headers and trailers. The ITE then presents the mid-layer packet for SEAL segmentation and outer encapsulation.

4.3.5. SEAL Segmentation

After mid-layer encapsulation, if the length of the resulting mid-layer packet plus OHLEN is greater than S_MSS the ITE must additionally perform SEAL segmentation. To do so, it breaks the mid-layer packet into N segments (N <= 256) that are no larger than (S_MSS - OHLEN) bytes each. Each segment, except the final one, MUST be of equal length. The first byte of each segment MUST begin immediately after the final byte of the previous segment, i.e., the segments MUST NOT overlap. The ITE SHOULD generate the smallest number of segments possible, e.g., it SHOULD NOT generate 6 smaller segments when the packet could be accommodated with 4 larger segments.

Note that this SEAL segmentation ignores the fact that the mid-layer packet may be unfragmentable outside of the subnetwork. This segmentation process is a mid-layer (not an IP layer) operation employed by the ITE to adapt the mid-layer packet to the subnetwork path characteristics, and the ETE will restore the packet to its original form during reassembly. Therefore, the fact that the packet may have been segmented within the subnetwork is not observable outside of the subnetwork.

4.3.6. Outer Encapsulation

Following SEAL segmentation, the ITE next encapsulates each segment in a SEAL header formatted as specified in Section 4.2. For the first segment, the ITE sets F=1, then sets NEXTHDR to the Internet Protocol number of the encapsulated inner packet, and finally sets M=1 if there are more segments or sets M=0 otherwise. For each non-initial segment of an N-segment mid-layer packet (N <= 256), the ITE sets (F=0; M=1; SEG=1) in the SEAL header of the first non-initial segment, sets (F=0; M=1; SEG=2) in the next non-initial segment, etc., and sets (F=0; M=0; SEG=N-1) in the final segment. (Note that the value SEG=0 is not used, since the initial segment encodes a NEXTHDR value and not a SEG value.)

The ITE next encapsulates each segment in the requisite outer headers and trailers according to the specific encapsulation format (e.g.,

[RFC1070], [RFC2003], [RFC2473], [RFC4213], etc.), except that it writes 'SEAL_PROTO' in the protocol field of the outer IP header (when simple IP encapsulation is used) or writes 'SEAL_PORT' in the outer destination service port field (e.g., when IP/UDP encapsulation is used). The ITE finally sets A=1 if probing is necessary as specified in Section 4.3.7, sets the packet identification values as specified in Section 4.3.8 and sends the packets as specified in Section 4.3.9.

4.3.7. Probing Strategy

All SEAL encapsulated packets sent by the ITE are considered implicit probes. SEAL encapsulated packets that use IPv4 as the outer layer of encapsulation will elicit SCMP PTB messages from the ETE (see: Section 4.5) if any IPv4 fragmentation occurs in the path. SEAL encapsulated packets that use IPv6 as the outer layer of encapsulation may be dropped by an IPv6 router on the path to the ETE which will also return an ICMPv6 PTB message to the ITE. The ITE can then use the SEAL_ID within the packet-in-error to determine whether the PTB message corresponds to one of its recent packet transmissions.

The ITE should also send explicit probes, periodically, to verify that the ETE is still reachable. The ITE sets A=1 in the SEAL header of a segment to be used as an explicit probe, where the probe can be either an ordinary data packet or a NULL packet created by setting the NEXTHDR field to a value of "No Next Header" (see Section 4.7 of [RFC2460]). The probe will elicit a solicited SCMP Neighbor Advertisement (NA) message from the ETE as an acknowledgement (see Section 4.5.1).

Finally, the ITE MAY send "expendable" outer IP probe packets (see Section 4.3.9) as explicit probes in order to detect increases in the path MTU to the ETE. One possible strategy is to send expendable packets with A=1 in the SEAL header and DF=1 in the IP header. In all cases, the ITE MUST be conservative in its use of the A bit in order to limit the resultant control message overhead.

4.3.8. Packet Identification

The ITE maintains a randomly-initialized SEAL_ID value as per-ETE soft state (e.g., in the neighbor cache) and monotonically increments it for each successive SEAL protocol packet it sends to the ETE. For each successive SEAL protocol packet, the ITE writes the current SEAL_ID value into the header field of the same name in the SEAL header.

4.3.9. Sending SEAL Protocol Packets

Following SEAL segmentation and encapsulation, the ITE sets DF=0 in the header of each outer IPv4 packet to ensure that they will be delivered to the ETE even if they are fragmented within the subnetwork. (The ITE can instead set DF=1 for "expendable" outer IPv4 packets (e.g., for NULL packets used as probes -- see Section 4.3.7), but these may be lost due to an MTU restriction). For outer IPv6 packets, the "DF" bit is always implicitly set to 1; hence, they will not be fragmented within the subnetwork.

The ITE sends each outer packet that encapsulates a segment of the same mid-layer packet into the tunnel in canonical order, i.e., segment 0 first, followed by segment 1, etc., and finally segment N-1.

4.3.10. Processing Raw ICMP Messages

The ITE may receive "raw" ICMP error messages [RFC0792][RFC4443] from either the ETE or routers within the subnetwork that comprise an outer IP header, followed by an ICMP header, followed by a portion of the SEAL packet that generated the error (also known as the "packet-in-error"). The ITE can use the SEAL_ID encoded in the packet-in-error as a nonce to confirm that the ICMP message came from either the ETE or an on-path router, and can use any additional information to determine whether to accept or discard the message.

The ITE should specifically process raw ICMPv4 Protocol Unreachable messages and ICMPv6 Parameter Problem messages with Code "Unrecognized Next Header type encountered" as a hint that the ETE does not implement the SEAL protocol; specific actions that the ITE may take in this case are out of scope.

4.4. ETE Specification

4.4.1. Reassembly Buffer Requirements

The ETE SHOULD support IP-layer and SEAL-layer reassembly for inner packets of at least 1280 bytes in length and MAY support reassembly for larger inner packets; the ETE may instead support only a minimum-sized reassembly buffer, but this may cause MTU underruns in some environments. The ETE must retain the outer IP, SEAL and other outer headers and trailers during both IP-layer and SEAL-layer reassembly for the purpose of associating the fragments/segments of the same packet, and must also configure a SEAL-layer reassembly buffer that is no smaller than the IP-layer reassembly buffer. Hence, the ETE:

- SHOULD configure an outer IP-layer reassembly buffer size of at least (1280 + HELN) bytes.

- MUST configure a SEAL-layer reassembly buffer size (i.e., S_MRU) that is no smaller than the IP-layer reassembly buffer size.

- MUST be capable of discarding inner packets that require IP-layer or SEAL-layer reassembly and that are larger than (S_MRU - HLEN).

The ETE can maintain S_MRU either as a single value to be applied for all ITEs, or as a per-ITE value. In that case, the ETE can manage each per-ITE S_MRU value separately (e.g., to reduce congestion caused by excessive segmentation from specific ITEs) but should seek to maintain as stable a value as possible for each ITE.

Note that the ETE is permitted to accept inner packets that did not undergo IP-layer and/or SEAL-layer reassembly even if they are larger than (S_MRU - HELN) bytes. Hence, S_MRU is a maximum *reassembly* size, and may be less than the ETE is able to receive without reassembly.

4.4.2. IP-Layer Reassembly

The ETE submits unfragmented SEAL protocol IP packets for SEAL-layer reassembly as specified in Section 4.4.3. The ETE instead performs standard IP-layer reassembly for multi-fragment SEAL protocol IP packets as follows.

The ETE should maintain conservative IP-layer reassembly cache high- and low-water marks. When the size of the reassembly cache exceeds this high-water mark, the ETE should actively discard incomplete reassemblies (e.g., using an Active Queue Management (AQM) strategy) until the size falls below the low-water mark. The ETE should also actively discard any pending reassemblies that clearly have no opportunity for completion, e.g., when a considerable number of new fragments have been received before a fragment that completes a pending reassembly has arrived. Following successful IP-layer reassembly, the ETE submits the reassembled packet for SEAL-layer reassembly as specified in Section 4.4.3.

When the ETE processes the IP first fragment (i.e., one with MF=1 and Offset=0 in the IP header) of a fragmented SEAL packet, it sends an SCMP PTB message back to the ITE (see Section 4.5.1). When the ETE processes an IP fragment that would cause the reassembled outer packet to be larger than the IP-layer reassembly buffer following reassembly, it discontinues the reassembly and discards any further fragments of the same packet.

4.4.3. SEAL-Layer Reassembly

Following IP reassembly (if necessary), if the mid-layer packet has an incorrect value in the SEAL header the ETE discards the packet and returns an SCMP "Parameter Problem" message (see Section 4.5.1). Next, if the SEAL header has A=1, the ETE sends a solicited SCMP Neighbor Advertisement (NA) message back to the ITE (see Section 4.5.1). The ETE next submits single-segment mid-layer packets for decapsulation and delivery to upper layers as specified in Section 4.4.4. The ETE instead performs SEAL-layer reassembly for multi-segment mid-layer packets as follows.

The ETE adds each segment of a multi-segment mid-layer packet to a SEAL-layer pending-reassembly queue according to the (Source, Destination, SEAL_ID)-tuple found in the outer IP and SEAL headers. The ETE performs SEAL-layer reassembly through simple in-order concatenation of the encapsulated segments of the same mid-layer packet from N consecutive SEAL segments. SEAL-layer reassembly requires the ETE to maintain a cache of recently received segments for a hold time that would allow for nominal inter-segment delays. When a SEAL reassembly times out, the ETE discards the incomplete reassembly and returns an SCMP "Time Exceeded" message to the ITE (see Section 4.5.1). As for IP-layer reassembly, the ETE should also maintain a conservative reassembly cache high- and low-water mark and should actively discard any pending reassemblies that clearly have no opportunity for completion, e.g., when a considerable number of new SEAL packets have been received before a packet that completes a pending reassembly has arrived.

If the ETE receives a SEAL packet for which a segment with the same (Source, Destination, SEAL_ID)-tuple is already in the queue, it must determine whether to accept the new segment and release the old, or drop the new segment. If accepting the new segment would cause an inconsistency with other segments already in the queue (e.g., differing segment lengths), the ETE drops the segment that is least likely to complete the reassembly. If the ETE accepts a new SEAL segment that would cause the reassembled outer packet to be larger than S_MRU following reassembly, it schedules the reassembly resources for garbage collection and sends an SCMP PTB message back to the ITE (see Section 4.5.1).

After all segments are gathered, the ETE reassembles the packet by concatenating the segments encapsulated in the N consecutive SEAL packets beginning with the initial segment (i.e., SEG=0) and followed by any non-initial segments 1 through N-1. That is, for an N-segment mid-layer packet, reassembly entails the concatenation of the SEAL-encapsulated packet segments with (F=1, M=1, SEAL_ID=j) in the first SEAL header, followed by (F=0, M=1, SEG=1, SEAL_ID=(j+1)) in the next

SEAL header, followed by (F=0, M=1, SEG=2, SEAL_ID=(j+2)), etc., up to (F=0, M=0, SEG=(N-1), SEAL_ID=(j + N-1)) in the final SEAL header. (Note that modulo arithmetic based on the length of the SEAL_ID field is used). Following successful SEAL-layer reassembly, the ETE submits the reassembled mid-layer packet for decapsulation and delivery to upper layers as specified in Section 4.4.4.

4.4.4. Decapsulation and Delivery to Upper Layers

Following any necessary IP- and SEAL-layer reassembly, the ETE discards the outer headers and trailers and performs any mid-layer transformations on the mid-layer packet. The ETE next discards the mid-layer headers and trailers, and delivers the inner packet to the upper-layer protocol indicated either in the SEAL NEXTHDR field or the next header field of the mid-layer packet (i.e., if the packet included mid-layer encapsulations). The ETE instead silently discards the inner packet if it was a NULL packet (see Section 4.3.9).

4.5. The SEAL Control Message Protocol (SCMP)

SEAL uses a companion SEAL Control Message Protocol (SCMP) based on the same message format as the Internet Control Message Protocol for IPv6 (ICMPv6) [RFC4443]. SCMP messages are further identified by the NEXTHDR value '58' the same as for ICMPv6 messages, however the SCMP message is *not* immediately preceded by an inner IPv6 header. Instead, SCMP messages appear immediately following the SEAL header which allows TEs to differentiate them from ordinary ICMPv6 messages. Unlike ICMPv6 messages, SCMP messages are used only for the purpose of conveying information between TEs, i.e., they are used only for sharing control information within the tunnel and not beyond the tunnel.

The following sections specify the generation and processing of SCMP messages:

4.5.1. Generating SCMP Messages

SCMP messages may be generated by either ITEs or ETEs (i.e., by any TE) using use the same message Type and Code values specified for ordinary ICMPv6 messages in [RFC4443]. SCMP can also be used to carry other message types and their associated options as specified in other documents (e.g., [RFC4191][RFC4861]). The general format for SCMP messages is shown in Figure 4:

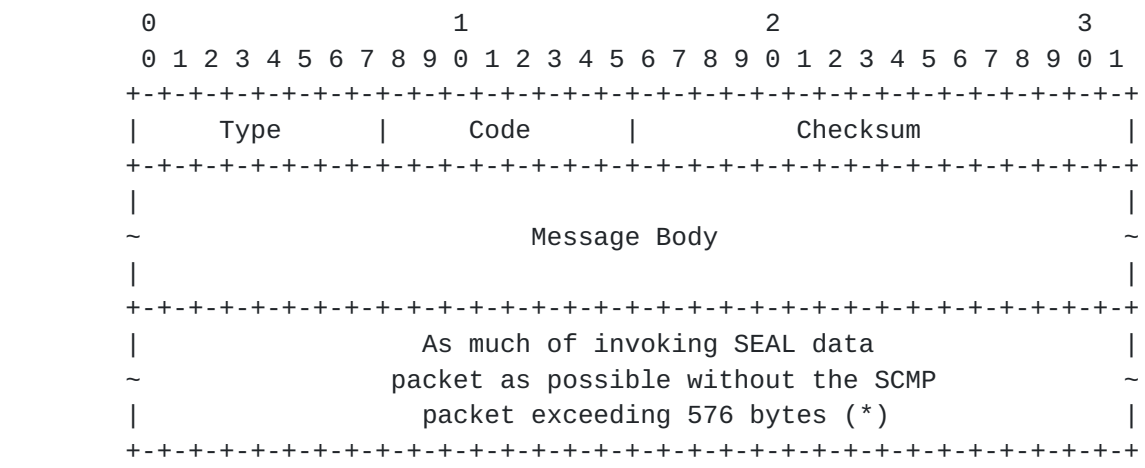

Figure 4: SCMP Message Format

As for ordinary ICMPv6 messages, the SCMP message begins with a 4 byte header that includes 8-bit Type and Code fields followed by a 16-bit Checksum field followed by a variable-length Message Body. The Message Body is followed by the leading portion of the invoking SEAL data packet (i.e., the "packet-in-error") IFF the packet-in-error would also be included in the corresponding ICMPv6 message. The TE sets the Type and Code fields to the same values that would appear in the corresponding ICMPv6 message and also formats the message body the same as for the corresponding ICMPv6 message except as otherwise specified.

If the SCMP message will include a packet-in-error, the TE then includes as much of the leading portion of the invoking SEAL data packet as possible beginning with the outer IP header and extending to a length that would not cause the entire SCMP message following encapsulation to exceed 576 bytes. The ITE finally calculates the Checksum the same as specified for ICMPv4 messages [RFC0792] and does not include a pseudo-header of the outer IP header since the SEAL_ID gives sufficient assurance against mis-delivery. The TE then encapsulates the SCMP message in the outer headers as shown in Figure 5:

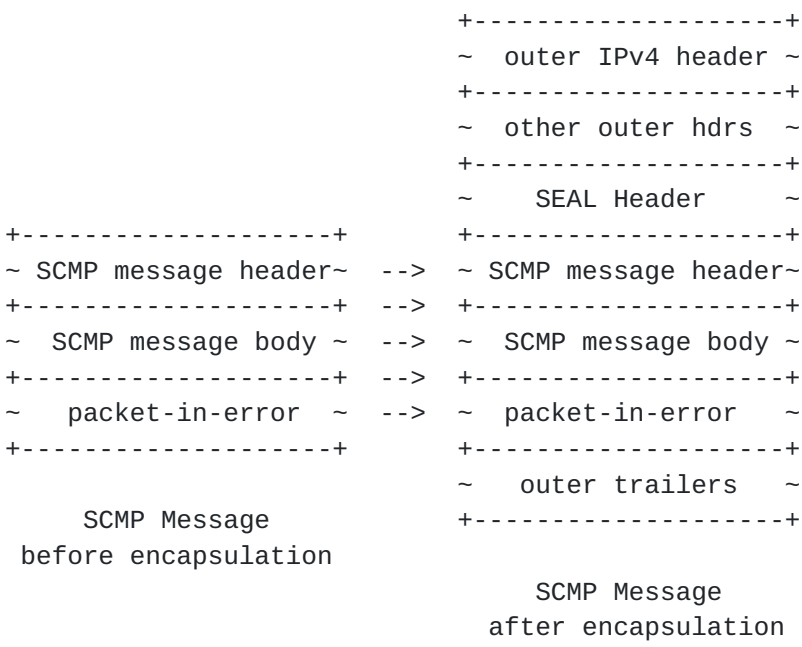

Figure 5: SCMP Message Encapsulation

When a TE generates an SCMP message in response to a packet-in-error, it sets the outer IP destination and source addresses of the SCMP packet to the packet-in-error's source and destination addresses (respectively). (If the destination address in the packet-in-error was multicast, the TE instead sets the outer IP source address of the SCMP packet to an address assigned to the underlying IP interface.) When a TE generates an SCMP message that is not due to a packet-in-error, it sets the outer IP destination and source addresses of the SCMP packet the same as for ordinary data packets. The TE finally sets the NEXTHDR field in the SEAL header to the value '58' (i.e., the official IANA protocol number for the ICMPv6 protocol) and sends the SCMP message to the tunnel far end.

4.5.1.1. Generating SCMP Packet Too Big (PTB) Messages

An ETE generates an SCMP Packet Too Big (PTB) message when it receives the IP first fragment (i.e., one with MF=1 and Offset=0 in the outer IP header) of a SEAL protocol packet that arrived as multiple IP fragments, or when it discontinues reassembly of a SEAL protocol packet that arrived as multiple IP fragments and/or multiple SEAL segments and would exceed S_MRU following reassembly.

The ETE prepares an SCMP PTB message the same as for the corresponding ICMPv6 PTB message, except that it writes the value 0 in the MTU field of the message if the PTB is generated as a result of receiving an IP first fragment and writes the S_MRU value for this ITE in the MTU field otherwise.

4.5.1.2. Generating SCMP Neighbor Discovery Messages

An ITE generates an SCMP "Neighbor Solicitation" (NS) or "Router Solicitation" (RS) message when it needs to solicit a response from an ETE. An ETE generates a solicited SCMP "Neighbor Advertisement" (NA) or "Router Advertisement" (RA) message when it receives an NS/RS message, and also generates a solicited NA message when it receives a SEAL protocol packet with A=1 in the SEAL header. Any TE may also generate unsolicited NA/RA messages that are not triggered by a specific solicitation event.

The TE generates NS/RS and NA/RA messages the same as described for the corresponding IPv6 Neighbor Discovery (ND) messages (see: [RFC4861]), except that for solicited NA/RA messages it also includes a Redirected Header option formatted the same as for an IPv6 ND Redirect message. The messages may also be used in conjunction with the tunnel endpoint synchronization procedure specified in Section 4.6.

4.5.1.3. Generating Other SCMP Messages

An ETE generates an SCMP "Destination Unreachable - Communication with Destination Administratively Prohibited" message when it is operating in synchronized mode and receives a SEAL packet with a SEAL_ID that is outside of the current window for this ITE (see: Section 4.6). An ETE also generates an SCMP "Destination Unreachable" message with an appropriate code under the same circumstances that an IPv6 system would generate an ICMPv6 Destination Unreachable message using the same code. The SCMP Destination Unreachable message is formatted the same as for ICMPv6 Destination Unreachable messages.

An ETE generates an SCMP "Parameter Problem" message when it receives a SEAL packet with an incorrect value in the SEAL header, and generates an SCMP "Time Exceeded" message when it garbage collects an incomplete SEAL data packet reassembly. The message formats used are the same as for the corresponding ICMPv6 messages.

Generation of all other SCMP message types is outside the scope of this document.

4.5.2. Processing SCMP Messages

An ITE processes any SCMP messages it receives as long as it can verify that the message was sent from an on-path ETE. The ITE can verify that the SCMP message came from an on-path ETE by checking that the SEAL_ID in the encapsulated packet-in-error corresponds to one of its recently-sent SEAL data packets.

An ITE maintains a window of SEAL_IDs of packets that it has recently sent to each ETE. For each SCMP message it receives, the ITE first verifies that the SEAL_ID encoded in the packet-in-error is within the window of packets that it has recently sent to the ETE. The ITE then verifies that the Checksum in the SCMP message header is correct. If the SEAL_ID is outside of the window and/or the checksum is incorrect, the ITE discards the message; otherwise, it processes the message the same as for ordinary ICMPv6 messages.

Any TE may also receive unsolicited SCMP messages from the tunnel far end. When the TEs are synchronized, they can also check that the SEAL_ID in the SEAL header of an SCMP message is within the window of recently received packets from this tunnel far end (see Section 4.6).

Finally, TEs process SCMP messages as an indication that the tunnel far end is responsive, i.e., in the same manner implied for IPv6 Neighbor Unreachability Detection "hints of forward progress" (see: [RFC4861]).

4.5.2.1. Processing SCMP PTB Messages

An ITE may receive an SCMP PTB message after it sends a SEAL data packet (see: Section 4.5.1). When the ITE receives an SCMP PTB message, it examines the MTU field in the message. If the MTU field is non-zero, the PTB was the result of a reassembly buffer limitation; in that case, the ITE records the value in the MTU field as the new S_MRU value for this ETE then (optionally) sends a translated PTB message of the inner network layer protocol to the original source with MTU set to (MAX(S_MRU, S_MSS) - HLEN). If the MTU field is zero, however, the PTB was the result of an IP fragmentation event; in that case, the ITE does not send back a translated PTB message but determines a new S_MSS value according to the length recorded in the IP header of the packet-in-error as follows:

- If the length is no less than 1280, the ITE records the length as the new S_MSS value.

- If the length is less than the current S_MSS value and also less than 1280, the ITE can discern that IP fragmentation is occurring but it cannot determine the true MTU of the restricting link due to the possibility that a router on the path is generating runt first fragments.

In this latter case, the ITE must search for a reduced S_MSS value through an iterative searching strategy that parallels (Section 5 of [RFC1191]). This searching strategy may require multiple iterations in which the ITE sends SEAL data packets using a reduced S_MSS and

receives additional SCMP MTU Report messages. During this process, it is essential that the ITE reduce S_MSS based on the first SCMP MTU Report message received under the current S_MSS size, and refrain from further reducing S_MSS until SCMP MTU Report messages pertaining to packets sent under the new S_MSS are received.

4.5.2.2. Processing SCMP Neighbor Discovery Messages

An ETE may received NS/RS messages from an ITE as an the initial leg in a neighbor discovery exchange. An ITE may receive both solicited and unsolicited NA/RA messages from an ETE, where solicited NA/RA messages are distinguished by their inclusion of a Redirected header option (see: Section 4.5.1).

The TE processes NS/RS and NA/RA messages the same as described for the corresponding IPv6 Neighbor Discovery (ND) messages (see: [RFC4861]). The messages may also be used in conjunction with the tunnel endpoint synchronization procedure specified in Section 4.6.

4.5.2.3. Processing Other SCMP Messages

An ITE may receive an SCMP "Destination Unreachable - Communication with Destination Administratively Prohibited" message after it sends a SEAL data packet. The ITE processes this message as an indication that it needs to (re)synchronize with the ETE (see: Section 4.6). An ITE may also receive an SCMP "Destination Unreachable" message with an appropriate code under the same circumstances that an IPv6 host would receive an ICMPv6 Destination Unreachable message.

An ITE may receive an SCMP "Parameter Problem" message when the ETE receives a SEAL packet with an incorrect value in the SEAL header. The ITE should examine the incorrect SEAL header field setting to determine whether a different setting should be used in subsequent packets.

.An ITE may receive an SCMP "Time Exceeded" message when the ETE garbage collects an incomplete SEAL data packet reassembly. The ITE should consider the message as an indication of congestion.

Processing of all other SCMP message types is outside the scope of this document.

4.6. Tunnel Endpoint Synchronization

The SEAL ITE maintains a per-ETE window of SEAL_IDs of its recently-sent packets, but by default the SEAL ETE does not retain inter-packet state. When closer synchronization is required, SEAL Tunnel Endpoints (TEs) can exchange initial sequence numbers in a procedure

that parallels IPv6 neighbor discovery and the TCP 3-way handshake. When the TEs are synchronized, the ETE can also maintain a per-ITE window of SEAL_IDs of its recently-received packets.

When an initiating TE ("TE(A)") needs to synchronize with a new tunnel far end ("TE(B)"), it first chooses a randomly-initialized 48-bit SEAL_ID value that it would like TE(B) to use (i.e., "SEAL_ID(B)"). TE(A) then creates a neighbor cache entry for TE(B) and records SEAL_ID(B) in the neighbor cache entry. Next, TE(A) creates an SCMP NS or RS message that includes a Nonce option (see: [RFC3971], Section 5.3). TE(A) then writes the value SEAL_ID(B) in the Nonce option, writes the value 0 in the SEAL_ID field of the SEAL header and sends the NS/RS message to TE(B).

When TE(B) receives an NS/RS message with a Nonce option and with the value 0 in the SEAL_ID of the SEAL header, it considers the message as a potential synchronization request. TE(B) first extracts the value SEAL_ID(B) from the Nonce option then chooses a randomly-initialized 48-bit SEAL_ID value that it would like TE(A) to use (i.e., "SEAL_ID(A)"). TE(B) then stores the tuple (ip_src, SEAL_ID(A), SEAL_ID(B)) in a minimal temporary fast path data structure, where "ip_src" is the outer IP source address of the SCMP message. (For efficiency and security purposes, the data structure should be indexed, e.g., by a secret hash of the -tuple). TE(B) then creates a solicited SCMP NA or RA message that includes a Nonce option. It then writes the value SEAL_ID(A) in the Nonce option, writes the value SEAL_ID(B) in the SEAL_ID field of the SEAL header and sends the NA/RA message back to TE(A).

When TE(A) receives the NA/RA, it considers the message as a potential synchronization acknowledgement. TE(A) first verifies that the value encoded in the SEAL_ID of the SEAL header matches the SEAL_ID(B) in the neighbor cache entry. If the values match, TE(A) extracts SEAL_ID(A) from the nonce option and records it in the neighbor cache entry; otherwise, it drops the packet. If instead TE(A) does not receive a timely NA/RA response, it retransmits the initial NS/RS message for a total of 3 tries before giving up the same as for ordinary IPv6 neighbor discovery.

After TE(A) receives the synchronization acknowledgement, it begins sending either unsolicited NA/RA messages or ordinary data packets back to TE(B) using SEAL_ID(A) as the initial sequence number. When TE(B) receives these packets, it first checks its neighbor cache to see if there is a matching neighbor cache entry. If there is a neighbor cache entry, and the SEAL_ID in the header of the packet is within the window of the SEAL_ID recorded in the neighbor cache entry, TE(B) accepts the packet. If the SEAL_ID in the packet is newer than the SEAL_ID in the neighbor cache entry, TE(B) also

updates the neighbor cache value. If there is no neighbor cache entry, TE(B) instead checks the fast path cache to see if the packet is a match for an in-progress synchronization event. If there is a fast path cache entry with a SEAL_ID(A) that is within the window of the SEAL_ID in the packet header, TE(B) accepts the packet and also creates a new neighbor cache entry with the tuple (ip_src, SEAL_ID(A), SEAL_ID(B)). If there is no matching fast path cache entry, TE(B) instead simply discards the packet.

By maintaining the fast path cache, each TE is able to mitigate buffer exhaustion attacks that may be launched by off-path attackers [RFC4987]. The TE will receive positive confirmation that the synchronization request came from an on-path tunnel far end after it receives a stream of in-window packets as the "third leg" of this three-way handshake as described above. The TEs should maintain neighbor cache entries as long as they receive hints of forward progress from the tunnel far end, but should delete the neighbor cache entries after a nominal stale time (e.g., 30 seconds). The TEs should also purge fast-path cache entries for which no window synchronization messages are received within a nominal stale time (e.g., 5 seconds).

After synchronization is complete, when a TE receives a SEAL packet it checks in its neighbor cache to determine whether the SEAL_ID is within the current window, and discards any packets that are outside the window. Since packets may be lost or reordered, and since SEAL presents only a best effort (i.e., and not reliable) link model, the TE should set a coarse-grained window size (e.g., 32768) and accept any packet with a SEAL_ID that is within the window.

5. Link Requirements

Subnetwork designers are expected to follow the recommendations in Section 2 of [RFC3819] when configuring link MTUs.

6. End System Requirements

SEAL provides robust mechanisms for returning PTB messages; however, end systems that send unfragmentable IP packets larger than 1500 bytes are strongly encouraged to implement their own end-to-end MTU assurance, e.g., using Packetization Layer Path MTU Discovery per [RFC4821].

7. Router Requirements

IPv4 routers within the subnetwork are strongly encouraged to implement IPv4 fragmentation such that the first fragment is the largest and approximately the size of the underlying link MTU, i.e., they should avoid generating runt first fragments.

IPv6 routers within the subnetwork are required to generate the necessary PTB messages when they drop outer IPv6 packets due to an MTU restriction.

8. IANA Considerations

The IANA is instructed to allocate an IP protocol number for 'SEAL_PROTO' in the 'protocol-numbers' registry.

The IANA is instructed to allocate a Well-Known Port number for 'SEAL_PORT' in the 'port-numbers' registry.

The IANA is instructed to establish a "SEAL Protocol" registry to record SEAL Version values. This registry should be initialized to include the initial SEAL Version number, i.e., Version 0.

9. Security Considerations

Unlike IPv4 fragmentation, overlapping fragment attacks are not possible due to the requirement that SEAL segments be non-overlapping. This condition is naturally enforced due to the fact that each consecutive SEAL segment begins at offset 0 with respect to the previous SEAL segment.

An amplification/reflection attack is possible when an attacker sends IP first fragments with spoofed source addresses to an ETE, resulting in a stream of SCMP messages returned to a victim ITE. The SEAL_ID in the encapsulated segment of the spoofed IP first fragment provides mitigation for the ITE to detect and discard spurious SCMP messages.

The SEAL header is sent in-the-clear (outside of any IPsec/ESP encapsulations) the same as for the outer IP and other outer headers. In this respect, the threat model is no different than for IPv6 extension headers. As for IPv6 extension headers, the SEAL header is protected only by L2 integrity checks and is not covered under any L3 integrity checks.

SCMP messages carry the SEAL_ID of the packet-in-error. Therefore, when an ITE receives an SCMP message it can unambiguously associate

it with the SEAL data packet that triggered the error. When the TEs are synchronized, the ETE can also detect off-path spoofing attacks.

Security issues that apply to tunneling in general are discussed in [I-D.ietf-v6ops-tunnel-security-concerns].

10. Related Work

Section 3.1.7 of [RFC2764] provides a high-level sketch for supporting large tunnel MTUs via a tunnel-level segmentation and reassembly capability to avoid IP level fragmentation, which is in part the same approach used by SEAL. SEAL could therefore be considered as a fully functioned manifestation of the method postulated by that informational reference.

Section 3 of [RFC4459] describes inner and outer fragmentation at the tunnel endpoints as alternatives for accommodating the tunnel MTU; however, the SEAL protocol specifies a mid-layer segmentation and reassembly capability that is distinct from both inner and outer fragmentation.

Section 4 of [RFC2460] specifies a method for inserting and processing extension headers between the base IPv6 header and transport layer protocol data. The SEAL header is inserted and processed in exactly the same manner.

The concepts of path MTU determination through the report of fragmentation and extending the IP Identification field were first proposed in deliberations of the TCP-IP mailing list and the Path MTU Discovery Working Group (MTUDWG) during the late 1980's and early 1990's. SEAL supports a report fragmentation capability using bits in an extension header (the original proposal used a spare bit in the IP header) and supports ID extension through a 16-bit field in an extension header (the original proposal used a new IP option). A historical analysis of the evolution of these concepts, as well as the development of the eventual path MTU discovery mechanism for IP, appears in Appendix D of this document.

11. SEAL Advantages over Classical Methods

The SEAL approach offers a number of distinct advantages over the classical path MTU discovery methods [RFC1191] [RFC1981]:

1. Classical path MTU discovery always results in packet loss when an MTU restriction is encountered. Using SEAL, IP fragmentation provides a short-term interim mechanism for ensuring that packets

are delivered while SEAL adjusts its packet sizing parameters.

2. Classical path MTU may require several iterations of dropping packets and returning PTB messages until an acceptable path MTU value is determined. Under normal circumstances, SEAL determines the correct packet sizing parameters in a single iteration.

3. Using SEAL, ordinary packets serve as implicit probes without exposing data to unnecessary loss. SEAL also provides an explicit probing mode not available in the classic methods.

4. Using SEAL, ETEs encapsulate SCMP error messages in outer and mid-layer headers such that packet-filtering network middleboxes will not filter them the same as for "raw" ICMP messages that may be generated by an attacker.

5. The SEAL approach ensures that the tunnel either delivers or deterministically drops packets according to their size, which is a required characteristic of any IP link.

6. Most importantly, all SEAL packets have an Identification field that is sufficiently long to be used for duplicate packet detection purposes and to associate ICMP error messages with actual packets sent without requiring per-packet state; hence, SEAL avoids certain denial-of-service attack vectors open to the classical methods.

12. Acknowledgments

The following individuals are acknowledged for helpful comments and suggestions: Jari Arkko, Fred Baker, Iljitsch van Beijnum, Oliver Bonaventure, Teco Boot, Bob Braden, Brian Carpenter, Steve Casner, Ian Chakeres, Noel Chiappa, Remi Denis-Courmont, Remi Despres, Ralph Droms, Aurnaud Ebalard, Gorry Fairhurst, Dino Farinacci, Joel Halpern, Sam Hartman, John Heffner, Thomas Henderson, Bob Hinden, Christian Huitema, Eliot Lear, Darrel Lewis, Joe Macker, Matt Mathis, Erik Nordmark, Dan Romascanu, Dave Thaler, Joe Touch, Mark Townsley, Ole Troan, Margaret Wasserman, Magnus Westerlund, Robin Whittle, James Woodyatt, and members of the Boeing Research & Technology NST DC&NT group.

Path MTU determination through the report of fragmentation was first proposed by Charles Lynn on the TCP-IP mailing list in 1987. Extending the IP identification field was first proposed by Steve Deering on the MTUDWG mailing list in 1989.

13. References

13.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

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

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

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

13.2. Informative References

[FOLK] Shannon, C., Moore, D., and k. claffy, "Beyond Folklore: Observations on Fragmented Traffic", December 2002.

[FRAG] Kent, C. and J. Mogul, "Fragmentation Considered Harmful", October 1987.

[I-D.ietf-intarea-ipv4-id-update]
Touch, J., "Updated Specification of the IPv4 ID Field", draft-ietf-intarea-ipv4-id-update-00 (work in progress), March 2010.

[I-D.ietf-tcpm-icmp-attacks]
Gont, F., "ICMP attacks against TCP", draft-ietf-tcpm-icmp-attacks-12 (work in progress), March 2010.

[I-D.ietf-v6ops-tunnel-security-concerns]
Hoagland, J., Krishnan, S., and D. Thaler, "Security Concerns With IP Tunneling",

draft-ietf-v6ops-tunnel-security-concerns-02 (work in progress), March 2010.

[I-D.russert-rangers]
Russert, S., Fleischman, E., and F. Templin, "Operational Scenarios for IRON and RANGER", draft-russert-rangers-03 (work in progress), June 2010.

[I-D.templin-intarea-vet]
Templin, F., "Virtual Enterprise Traversal (VET)", draft-templin-intarea-vet-13 (work in progress), June 2010.

[I-D.templin-iron]
Templin, F., "The Internet Routing Overlay Network (IRON)", draft-templin-iron-02 (work in progress), June 2010.

[MTUDWG] "IETF MTU Discovery Working Group mailing list, gatekeeper.dec.com/pub/DEC/WRL/mogul/mtudwg-log, November 1989 - February 1995.".

[RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, July 1988.

[RFC1070] Hagens, R., Hall, N., and M. Rose, "Use of the Internet as a subnetwork for experimentation with the OSI network layer", RFC 1070, February 1989.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, August 1999.

[RFC2764] Gleeson, B., Heinanen, J., Lin, A., Armitage, G., and A. Malis, "A Framework for IP Based Virtual Private Networks", RFC 2764, February 2000.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[RFC3232] Reynolds, J., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.

[RFC3366] Fairhurst, G. and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)", BCP 62, RFC 3366, August 2002.

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, April 2006.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.

[RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, March 2009.

[RFC5720] Templin, F., "Routing and Addressing in Networks with Global Enterprise Recursion (RANGER)", RFC 5720, February 2010.

[TBIT] Medina, A., Allman, M., and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", October 2004.

[TCP-IP] "Archive/Hypermail of Early TCP-IP Mail List, http://www-mice.cs.ucl.ac.uk/multimedia/misc/tcp_ip/, May 1987 - May 1990.".

[WAND] Luckie, M., Cho, K., and B. Owens, "Inferring and Debugging Path MTU Discovery Failures", October 2005.

Appendix A. Reliability

Although a SEAL tunnel may span an arbitrarily-large subnetwork expanse, the IP layer sees the tunnel as a simple link that supports the IP service model. Since SEAL supports segmentation at a layer below IP, SEAL therefore presents a case in which the link unit of loss (i.e., a SEAL segment) is smaller than the end-to-end retransmission unit (e.g., a TCP segment).

Links with high bit error rates (BERs) (e.g., IEEE 802.11) use Automatic Repeat-ReQuest (ARQ) mechanisms [RFC3366] to increase packet delivery ratios, while links with much lower BERs typically omit such mechanisms. Since SEAL tunnels may traverse arbitrarily-long paths over links of various types that are already either performing or omitting ARQ as appropriate, it would therefore often be inefficient to also require the tunnel to perform ARQ.

When the SEAL ITE has knowledge that the tunnel will traverse a subnetwork with non-negligible loss due to, e.g., interference, link errors, congestion, etc., it can solicit Segment Reports from the ETE periodically to discover missing segments for retransmission within a single round-trip time. However, retransmission of missing segments may require the ITE to maintain considerable state and may also result in considerable delay variance and packet reordering.

SEAL may also use alternate reliability mechanisms such as Forward Error Correction (FEC). A simple FEC mechanism may merely entail gratuitous retransmissions of duplicate data, however more efficient alternatives are also possible. Basic FEC schemes are discussed in [RFC5445].

The use of ARQ and FEC mechanisms for improved reliability are for further study.

Appendix B. Integrity

Each link in the path over which a SEAL tunnel is configured is responsible for link layer integrity verification for packets that traverse the link. As such, when a multi-segment SEAL packet with N

segments is reassembled, its segments will have been inspected by N independent link layer integrity check streams instead of a single stream that a single segment SEAL packet of the same size would have received. Intuitively, a reassembled packet subjected to N independent integrity check streams of shorter-length segments would seem to have integrity assurance that is no worse than a single-segment packet subjected to only a single integrity check steam, since the integrity check strength diminishes in inverse proportion with segment length. In any case, the link-layer integrity assurance for a multi-segment SEAL packet is no different than for a multi-fragment IPv6 packet.

Fragmentation and reassembly schemes must also consider packet-splicing errors, e.g., when two segments from the same packet are concatenated incorrectly, when a segment from packet X is reassembled with segments from packet Y, etc. The primary sources of such errors include implementation bugs and wrapping IP ID fields. In terms of implementation bugs, the SEAL segmentation and reassembly algorithm is much simpler than IP fragmentation resulting in simplified implementations. In terms of wrapping ID fields, when IPv4 is used as the outer IP protocol, the 16-bit IP ID field can wrap with only 64K packets with the same (src, dst, protocol)-tuple alive in the system at a given time [RFC4963] increasing the likelihood of reassembly mis-associations. However, SEAL ensures that any outer IPv4 fragmentation and reassembly will be short-lived and tuned out as soon as the ITE receives a Reassembly Repot, and SEAL segmentation and reassembly uses a much longer ID field. Therefore, reassembly mis-associations of IP fragments nor of SEAL segments should be prohibitively rare.

Appendix C. Transport Mode

SEAL can also be used in "transport-mode", e.g., when the inner layer comprises upper-layer protocol data rather than an encapsulated IP packet. For instance, TCP peers can negotiate the use of SEAL for the carriage of protocol data encapsulated as IPv4/SEAL/TCP. In this sense, the "subnetwork" becomes the entire end-to-end path between the TCP peers and may potentially span the entire Internet.

Section specifies the operation of SEAL in "tunnel mode", i.e., when there are both an inner and outer IP layer with a SEAL encapsulation layer between. However, the SEAL protocol can also be used in a "transport mode" of operation within a subnetwork region in which the inner-layer corresponds to a transport layer protocol (e.g., UDP, TCP, etc.) instead of an inner IP layer.

For example, two TCP endpoints connected to the same subnetwork

region can negotiate the use of transport-mode SEAL for a connection by inserting a 'SEAL_OPTION' TCP option during the connection establishment phase. If both TCPs agree on the use of SEAL, their protocol messages will be carried as TCP/SEAL/IPv4 and the connection will be serviced by the SEAL protocol using TCP (instead of an encapsulating tunnel endpoint) as the transport layer protocol. The SEAL protocol for transport mode otherwise observes the same specifications as for Section 4.

Appendix D. Historic Evolution of PMTUD

The topic of Path MTU discovery (PMTUD) saw a flurry of discussion and numerous proposals in the late 1980's through early 1990. The initial problem was posed by Art Berggreen on May 22, 1987 in a message to the TCP-IP discussion group [TCP-IP]. The discussion that followed provided significant reference material for [FRAG]. An IETF Path MTU Discovery Working Group [MTUDWG] was formed in late 1989 with charter to produce an RFC. Several variations on a very few basic proposals were entertained, including:

1. Routers record the PMTUD estimate in ICMP-like path probe messages (proposed in [FRAG] and later [RFC1063])

2. The destination reports any fragmentation that occurs for packets received with the "RF" (Report Fragmentation) bit set (Steve Deering's 1989 adaptation of Charles Lynn's Nov. 1987 proposal)

3. A hybrid combination of 1) and Charles Lynn's Nov. 1987 (straw RFC draft by McCloughrie, Fox and Mogul on Jan 12, 1990)

4. Combination of the Lynn proposal with TCP (Fred Bohle, Jan 30, 1990)

5. Fragmentation avoidance by setting "IP_DF" flag on all packets and retransmitting if ICMPv4 "fragmentation needed" messages occur (Geof Cooper's 1987 proposal; later adapted into [RFC1191] by Mogul and Deering).

Option 1) seemed attractive to the group at the time, since it was believed that routers would migrate more quickly than hosts. Option 2) was a strong contender, but repeated attempts to secure an "RF" bit in the IPv4 header from the IESG failed and the proponents became discouraged. 3) was abandoned because it was perceived as too complicated, and 4) never received any apparent serious consideration. Proposal 5) was a late entry into the discussion from Steve Deering on Feb. 24th, 1990. The discussion group soon thereafter seemingly lost track of all other proposals and adopted

5), which eventually evolved into [RFC1191] and later [RFC1981].

In retrospect, the "RF" bit postulated in 2) is not needed if a "contract" is first established between the peers, as in proposal 4) and a message to the MTUDWG mailing list from jrd@PTT.LCS.MIT.EDU on Feb 19. 1990. These proposals saw little discussion or rebuttal, and were dismissed based on the following the assertions:

- routers upgrade their software faster than hosts
- PCs could not reassemble fragmented packets
- Proteon and Wellfleet routers did not reproduce the "RF" bit properly in fragmented packets
- Ethernet-FDDI bridges would need to perform fragmentation (i.e., "translucent" not "transparent" bridging)
- the 16-bit IP_ID field could wrap around and disrupt reassembly at high packet arrival rates

The first four assertions, although perhaps valid at the time, have been overcome by historical events. The final assertion is addressed by the mechanisms specified in SEAL.

Author's Address

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org