Independent Submission
Request for Comments: 5320
Category: Experimental
ISSN: 2070-1721

F. Templin, Ed.
Boeing Research & Technology

February 2010

The Subnetwork Encapsulation and Adaptation Layer (SEAL)

Abstract

For the purpose of this document, subnetworks are defined as virtual topologies that span connected network regions 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 document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5320.

IESG Note

This RFC is not a candidate for any level of Internet Standard. The IETF disclaims any knowledge of the fitness of this RFC for any purpose and in particular notes that the decision to publish is not based on IETF review for such things as security, congestion control, or inappropriate interaction with deployed protocols. The RFC Editor has chosen to publish this document at its discretion. Readers of this document should exercise caution in evaluating its value for implementation and deployment. See RFC 3932 for more information.

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 as-seen by the IP 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]. In the following subsections, we 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 addressing properties of IPv4 are limited (hence, the larger address space provided by IPv6), there is a lesser-known but growing consensus that other limitations may be unable to sustain continued growth.

First, the IPv4 header Identification field is only 16 bits in length, meaning that at most 2^16 packets pertaining to the same (source, destination, protocol, Identification)-tuple may be active in the Internet at a given time. 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. 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., slow-path processing in routers) to the potential for major integrity issues (e.g., mis-association of the fragments of multiple IP packets during reassembly).

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 A). 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".

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. 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.

The issues with both IPv4 fragmentation and this "classical" method of path MTU discovery are exacerbated further when IP-in-IP tunneling is used. For example, site border routers that are configured as ingress tunnel endpoints may be required to forward packets into the subnetwork on behalf of hundreds, thousands, or even more original sources located within the site. If IPv4 fragmentation were used, this would quickly wrap the 16-bit Identification field and could lead to undetected data corruption. If classical IPv4 path MTU discovery were used instead, the site border router may be bombarded by ICMP error messages coming from the subnetwork that may be either untrustworthy or insufficiently provisioned to allow translation into error message to be returned to the original sources.

The situation is exacerbated further still by IPsec tunnels, since only the first IPv4 fragment of a fragmented packet contains the transport protocol selectors (e.g., the source and destination ports) required for identifying the correct security association rendering fragmentation useless under certain circumstances. Even worse, there may be no way for a site border router that configures an IPsec tunnel to transcribe the encrypted packet fragment contained in an

ICMP error message into a suitable ICMP error message to return to the original source. Due to these many limitations, a new approach to accommodate links with diverse MTUs is necessary.

1.2. Approach

For the purpose of this document, subnetworks are defined as virtual topologies that span connected network regions 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 IP 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 tunnel-mode operation of IP over subnetworks that connect Ingress and Egress Tunnel Endpoints (ITEs/ETEs) of border nodes. Operation in transport mode is also supported when subnetwork border node upper-layer protocols negotiate the use of SEAL during connection establishment. SEAL accommodates links with diverse MTUs and supports efficient duplicate packet detection by introducing a minimal mid-layer encapsulation.

The 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 IPv4 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.

The SEAL encapsulation layer and protocol are specified in the following sections.

2. Terminology and Requirements

The terms "inner", "mid-layer", and "outer", respectively, refer to the innermost IP (layer, protocol, header, packet, etc.) before any encapsulation, the mid-layer IP (protocol, header, packet, etc.) after any mid-layer '*' encapsulation, and the outermost IP (layer, protocol, header, packet etc.) after SEAL/*/IPv4 encapsulation.

The term "IP" used throughout the document refers to either Internet Protocol version (IPv4 or IPv6). Additionally, the notation IPvX/*/SEAL/*/IPvY refers to an inner IPvX packet encapsulated in any

mid-layer '*' encapsulations, followed by the SEAL header, followed by any outer '*' encapsulations, followed by an outer IPvY header, where the notation "IPvX" means either IP protocol version (IPv4 or IPv6).

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

ITE - Ingress Tunnel Endpoint

ETE - Egress Tunnel Endpoint

PTB - an ICMPv6 "Packet Too Big" or an ICMPv4 "Fragmentation Needed" message

DF - the IPv4 header "Don't Fragment" flag

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

OHLEN - the length of the outer encapsulating SEAL/*/IPv4 headers

HLEN - the sum of MHLEN and OHLEN

S_MRU - the per-ETE SEAL Maximum Reassembly Unit

S_MSS - the SEAL Maximum Segment Size

SEAL_ID - a 32-bit Identification value, randomly initialized and monotonically incremented for each SEAL protocol packet

SEAL_PROTO - an IPv4 protocol number used for SEAL

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

SEAL_OPTION - a TCP option number used for (transport-mode) SEAL

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

3. Applicability Statement

SEAL was motivated by the specific case of subnetwork abstraction for Mobile Ad hoc Networks (MANETs); however, the domain of applicability also extends to subnetwork abstractions of enterprise networks, the interdomain routing core, etc. The domain of application therefore

also includes the map-and-encaps architecture proposals in the IRTF Routing Research Group (RRG) (see http://www3.tools.ietf.org/group/irtf/trac/wiki/RoutingResearchGroup).

SEAL introduces a minimal new sublayer for IPvX in IPvY encapsulation (e.g., as IPv6/SEAL/IPv4), and appears as a subnetwork encapsulation as seen by the inner IP layer. SEAL can also be used as a sublayer for encapsulating inner IP packets within outer UDP/IPv4 headers (e.g., as IPv6/SEAL/UDP/IPv4) such as for the Teredo domain of applicability [RFC4380]. When it appears immediately after the outer IPv4 header, the SEAL header is processed exactly as for IPv6 extension headers.

SEAL can also be used in "transport-mode", e.g., when the inner layer includes 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 TCP/SEAL/IPv4. In this sense, the "subnetwork" becomes the entire end-to-end path between the TCP peers and may potentially span the entire Internet.

The current document version is specific to the use of IPv4 as the outer encapsulation layer; however, the same principles apply when IPv6 is used as the outer layer.

4. SEAL Protocol Specification - Tunnel Mode

4.1. Model of Operation

SEAL supports the encapsulation of inner IP packets in mid-layer and outer encapsulating headers/trailers. For example, an inner IPv6 packet would appear as IPv6/*/SEAL/*/IPv4 after mid-layer and outer encapsulations, where '*' denotes zero or more additional encapsulation sublayers. Ingres Tunnel Endpoints (ITEs) add mid-layer inject into a subnetwork, where the outermost IPv4 header contains the source and destination addresses of the subnetwork entry/exit points (i.e., the ITE/ETE), respectively. SEAL uses a new Internet Protocol type and a new encapsulation sublayer for both unicast and multicast. The ITE encapsulates an inner IP packet in mid-layer and outer encapsulations as shown in Figure 1:

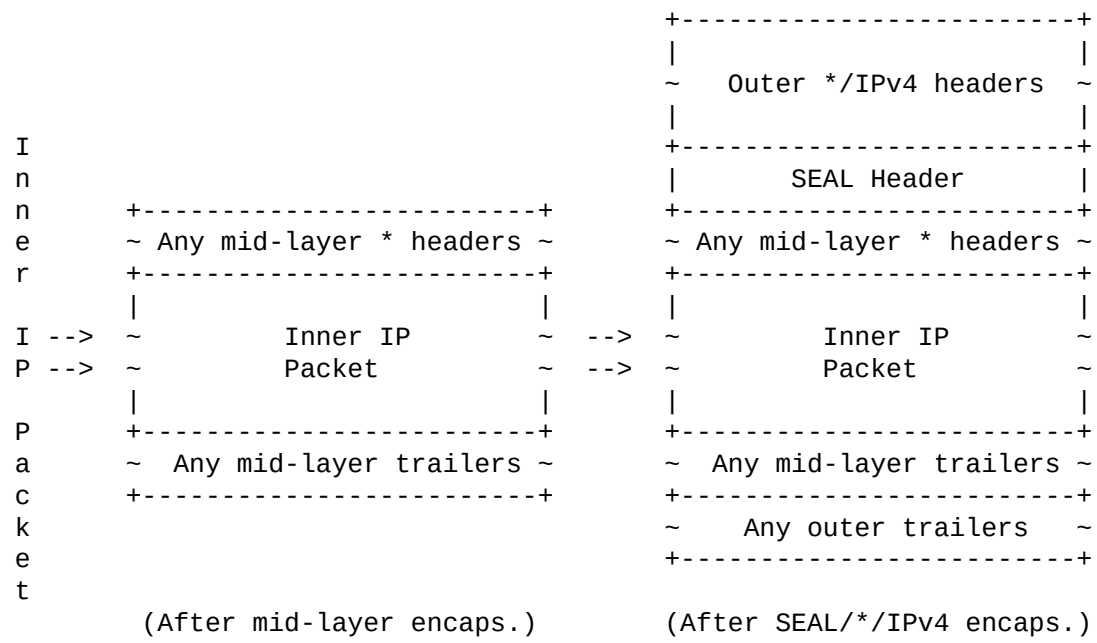

Figure 1: SEAL Encapsulation

where the SEAL header is inserted as follows:

- For simple IPvX/IPv4 encapsulations (e.g., [RFC2003][RFC2004][RFC4213]), the SEAL header is inserted between the inner IP and outer IPv4 headers as: IPvX/SEAL/IPv4.

- For tunnel-mode IPsec encapsulations over IPv4, [RFC4301], the SEAL header is inserted between the {AH,ESP} header and outer IPv4 headers as: IPvX/*/{AH,ESP}/SEAL/IPv4.

- For IP encapsulations over transports such as UDP, the SEAL header is inserted immediately after the outer transport layer header, e.g., as IPvX/*/SEAL/UDP/IPv4.

SEAL-encapsulated packets include a 32-bit SEAL_ID formed from the concatenation of the 16-bit ID Extension field in the SEAL header as the most-significant bits, and with the 16-bit Identification value in the outer IPv4 header as the least-significant bits. (For tunnels that traverse IPv4 Network Address Translators, the SEAL_ID is instead maintained only within the 16-bit ID Extension field in the SEAL header.) Routers within the subnetwork use the SEAL_ID for duplicate packet detection, and ITEs/ETEs use the SEAL_ID for SEAL segmentation and reassembly.

SEAL enables a multi-level segmentation and reassembly capability.

First, the ITE can use IPv4 fragmentation to fragment inner IPv4 packets with DF=0 before SEAL encapsulation to avoid lower-layer segmentation and reassembly. Secondly, the SEAL layer itself provides a simple cutting-and-pasting capability for mid-layer packets to avoid IPv4 fragmentation on the outer packet. Finally, ordinary IPv4 fragmentation is permitted on the outer packet after SEAL encapsulation and used to detect and dampen any in-the-network fragmentation as quickly as possible.

The following sections specify the SEAL-related operations of the ITE and ETE, respectively:

4.2. ITE Specification

4.2.1. 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 the inner IP layer (i.e., Layer 3) as the size for admission of inner IP packets into the tunnel. Since the tunnel interface may support a potentially large set of ETEs, however, care must be taken in setting a greatest-common-denominator MTU for all ETEs while still upholding end system expectations.

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 PTB message returned. However, the network may not always deliver the necessary PTBs, leading to MTU-related black holes [RFC2923]. The ITE therefore requires a means for conveying 1500 byte (or smaller) packets to the ETE without loss due to MTU restrictions and without dependence on PTB messages from within the subnetwork.

In common deployments, there may be many forwarding hops between the original source and the ITE. Within those hops, there may be additional encapsulations (IPSec, L2TP, etc.) such that a 1500 byte packet sent by the original source might grow to a larger size by the time it reaches the ITE for encapsulation as an inner IP packet. Similarly, additional encapsulations on the path from the ITE to the ETE could cause the encapsulated packet to become larger still and trigger in-the-network fragmentation. In order to preserve the end system expectations, the ITE therefore requires a means for conveying these larger packets to the ETE even though there may be links within the subnetwork that configure a smaller MTU.

The ITE should therefore set a tunnel virtual interface MTU of 1500 bytes plus extra room to accommodate any additional encapsulations that may occur on the path from the original source (i.e., even if the path to the ETE does not support an MTU of this size). 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 (see Sections 4.2.2 and 4.2.6). 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 inner IP layer consults the tunnel interface MTU when admitting a packet into the interface. For inner IPv4 packets larger than the tunnel interface MTU and with the IPv4 Don't Fragment (DF) bit set to 0, the inner IPv4 layer uses IPv4 fragmentation to break the packet into fragments no larger than the tunnel interface MTU (but, see also Section 4.2.3), then admits each fragment into the tunnel as an independent packet. For all other inner packets (IPv4 or IPv6), the ITE admits the packet if it is no larger than the tunnel interface MTU; otherwise, it drops the packet and sends an ICMP PTB message with an MTU value of the tunnel interface MTU to the source.

4.2.2. Accounting for Headers

As for any transport layer protocol, ITEs use the MTU of the underlying IPv4 interface, the length of any mid-layer '*' headers and trailers, and the length of the outer SEAL/*/IPv4 headers to determine the maximum size for a SEAL segment (see Section 4.2.3). For example, when the underlying IPv4 interface advertises an MTU of 1500 bytes and the ITE inserts a minimum-length (i.e., 20-byte) IPv4 header, the ITE sees a maximum segment size of 1480 bytes. When the ITE inserts IPv4 header options, the size is further reduced by as many as 40 additional bytes (the maximum length for IPv4 options) such that as few as 1440 bytes may be available for the upper-layer payload. When the ITE inserts additional '*' encapsulations, the maximum segment size is reduced further still.

The ITE must additionally account for the length of the SEAL header itself as an extra encapsulation that further reduces the maximum segment size. The length of the SEAL header is not incorporated in the IPv4 header length; therefore, the network does not observe the SEAL header as an IPv4 option. In this way, the SEAL header is inserted after the IPv4 options but before the upper-layer payload in exactly the same manner as for IPv6 extension headers.

4.2.3. Segmentation and Encapsulation

For each ETE, the ITE maintains the length of any mid-layer '*' encapsulation headers and trailers (e.g., for '*' = AH, ESP, NULL, etc.) in a variable 'MHLEN' and maintains the length of the outer SEAL/*/IPv4 encapsulation headers in a variable 'OHLEN'. The ITE further maintains a variable 'HLEN' set to MHLEN plus OHLEN. The ITE maintains a SEAL Maximum Reassembly Unit (S_MRU) value for each ETE as soft state within the tunnel interface (e.g., in the IPv4 destination cache). The ITE initializes S_MRU to a value no larger than 2KB and uses this value to determine the maximum-sized packet it will require the ETE to reassemble. The ITE additionally maintains a SEAL Maximum Segment Size (S_MSS) value for each ETE. The ITE initializes S_MSS to the maximum of (the underlying IPv4 interface MTU minus OHLEN) and S_MRU/8 bytes, and decreases or increases S_MSS based on any ICMPv4 Fragmentation Needed messages received (see Section 4.2.6).

The ITE performs segmentation and encapsulation on inner packets that have been admitted into the tunnel interface. For inner IPv4 packets with the DF bit set to 0, if the length of the inner packet is larger than (S_MRU - HLEN), the ITE uses IPv4 fragmentation to break the packet into IPv4 fragments no larger than (S_MRU - HLEN). For unfragmentable inner packets (e.g., IPv6 packets, IPv4 packets with DF=1, etc.), if the length of the inner packet is larger than (MAX(S_MRU, S_MSS) - HLEN), the ITE drops the packet and sends an ICMP PTB message with an MTU value of (MAX(S_MRU, S_MSS) - HLEN) back to the original source.

The ITE then encapsulates each inner packet/fragment in the MHLEN bytes of mid-layer '*' headers and trailers. For each such resulting mid-layer packet of length 'M', if (S_MRU >= (M + OHLEN) > S_MSS), the ITE must perform SEAL segmentation. To do so, it breaks the mid-layer packet into N segments (N <= 8) that are no larger than (MIN(1KB, S_MSS) - OHLEN) bytes each. Each segment, except the final one, MUST be of equal length, while the final segment MUST be no larger than the initial segment. 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. 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.

The ITE next encapsulates each segment in a SEAL header formatted as follows:

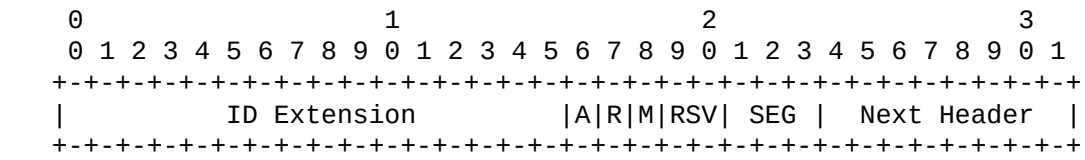

Figure 2: SEAL Header Format

where the header fields are defined as follows:

ID Extension (16)
: a 16-bit extension of the Identification field in the outer IPv4 header; encodes the most-significant 16 bits of a 32 bit SEAL_ID value.

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

R (1)
: the "Report Fragmentation" bit. Set to 1 if the ITE wishes to receive a report from the ETE if any IPv4 fragmentation occurs.

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.

RSV (2)
: a 2-bit field reserved for future use. Must be set to 0 for the purpose of this specification.

SEG (3)
: a 3-bit segment number. Encodes a segment number between 0 - 7.

Next Header (8)
: an 8-bit field that encodes an Internet Protocol number the same as for the IPv4 protocol and IPv6 next header fields.

For single-segment mid-layer packets, the ITE encapsulates the segment in a SEAL header with (M=0; SEG=0). For N-segment mid-layer packets (N <= 8), the ITE encapsulates each segment in a SEAL header with (M=1; SEG=0) for the first segment, (M=1; SEG=1) for the second segment, etc., with the final segment setting (M=0; SEG=N-1).

The ITE next sets RSV='00' and sets the A and R bits in the SEAL header of the first segment according to whether the packet is to be used as an explicit/implicit probe as specified in Section 4.2.4. The ITE then writes the Internet Protocol number corresponding to the mid-layer packet in the SEAL 'Next Header' field and encapsulates each segment in the requisite */IPv4 outer headers according to the specific encapsulation format (e.g., [RFC2003], [RFC4213], [RFC4380], etc.), except that it writes 'SEAL_PROTO' in the protocol field of the outer IPv4 header (when simple IPv4 encapsulation is used) or writes 'SEAL_PORT' in the outer destination service port field (e.g., when UDP/IPv4 encapsulation is used). The ITE finally sets packet identification values as specified in Section 4.2.5 and sends the packets as specified in Section 4.2.6.

4.2.4. Sending Probes

When S_MSS is larger than S_MRU/8 bytes, the ITE sends ordinary encapsulated data packets as implicit probes to detect in-the-network IPv4 fragmentation and to determine new values for S_MSS. The ITE sets R=1 in the SEAL header of a packet with SEG=0 to be used as an implicit probe, and will receive ICMPv4 Fragmentation Needed messages from the ETE if any IPv4 fragmentation occurs. When the ITE has already reduced S_MSS to the minimum value, it instead sets R=0 in the SEAL header to avoid generating fragmentation reports for unavoidable in-the-network fragmentation.

The ITE should send explicit probes periodically to manage a window of SEAL_IDs of outstanding probes as a means to validate any ICMPv4 messages it receives. The ITE sets A=1 in the SEAL header of a packet with SEG=0 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 'Next Header' field in the SEAL header to a value of "No Next Header" (see Section 4.7 of [RFC2460]).

The ITE should further send explicit probes, periodically, to detect increases in S_MSS by resetting S_MSS to the maximum of (the underlying IPv4 interface MTU minus OHLEN) and S_MRU/8 bytes, and/or by sending explicit probes that are larger than the current S_MSS.

Finally, the ITE MAY send "expendable" probe packets with DF=1 (see Section 4.2.6) in order to generate ICMPv4 Fragmentation Needed messages from routers on the path to the ETE.

4.2.5. Packet Identification

For the purpose of packet identification, the ITE maintains a 32-bit SEAL_ID value as per-ETE soft state, e.g., in the IPv4 destination cache. The ITE randomly initializes SEAL_ID when the soft state is created and monotonically increments it (modulo 2^32) for each successive SEAL protocol packet it sends to the ETE. For each packet, the ITE writes the least-significant 16 bits of the SEAL_ID value in the Identification field in the outer IPv4 header, and writes the most-significant 16 bits in the ID Extension field in the SEAL header.

For SEAL encapsulations specifically designed for the traversal of IPv4 Network Address Translators (NATs), e.g., for encapsulations that insert a UDP header between the SEAL header and outer IPv4 header such as IPv6/SEAL/UDP/IPv4, the ITE instead maintains SEAL_ID as a 16-bit value that it randomly initializes when the soft state is created and monotonically increments (modulo 2^16) for each successive packet. For each packet, the ITE writes SEAL_ID in the ID extension field of the SEAL header and writes a random 16-bit value in the Identification field in the outer IPv4 header. This is due to the fact that the ITE has no way to control IPv4 NATs in the path that could rewrite the Identification value in the outer IPv4 header.

4.2.6. Sending SEAL Protocol Packets

Following SEAL segmentation and encapsulation, the ITE sets DF=0 in the outer IPv4 header of every outer packet it sends. For "expendable" packets (e.g., for NULL packets used as probes -- see Section 4.2.4), the ITE may instead set DF=1.

The ITE then 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.

4.2.7. Processing Raw ICMPv4 Messages

The ITE may receive "raw" ICMPv4 error messages from either the ETE or routers within the subnetwork that comprise an outer IPv4 header, followed by an ICMPv4 header, followed by a portion of the SEAL packet that generated the error (also known as the "packet-in-error"). For such messages, the ITE can use the 32-bit 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. The ITE MAY process raw ICMPv4 messages as soft errors indicating that the path to the ETE may be failing.

The ITE should specifically process raw ICMPv4 Protocol Unreachable messages as a hint that the ETE does not implement the SEAL protocol.

4.2.8. Processing SEAL-Encapsulated ICMPv4 Messages

In addition to any raw ICMPv4 messages, the ITE may receive SEAL-encapsulated ICMPv4 messages from the ETE that comprise outer ICMPv4/*/SEAL/*/IPv4 headers followed by a portion of the SEAL-encapsulated packet-in-error. The ITE can use the 32-bit SEAL ID encoded in the packet-in-error as well as information in the outer IPv4 and SEAL headers as nonces to confirm that the ICMP message came from a legitimate ETE. The ITE then verifies that the SEAL_ID encoded in the packet-in-error is within the current window of transmitted SEAL_IDs for this ETE. If the SEAL_ID is outside of the window, the ITE discards the message; otherwise, it advances the window and processes the message.

The ITE processes SEAL-encapsulated ICMPv4 messages other than ICMPv4 Fragmentation Needed exactly as specified in [RFC0792].

For SEAL-encapsulated ICMPv4 Fragmentation Needed messages, the ITE sets a variable 'L' to the IPv4 length of the packet-in-error minus OHLEN. If (L > S_MSS), or if the packet-in-error is an IPv4 first fragment (i.e., with MF=1; Offset=0) and (L >= (576 - OHLEN)), the ITE sets (S_MSS = L).

Note that 576 in the above corresponds to the nominal minimum MTU for IPv4 links. When an ITE instead receives an IPv4 first fragment packet-in-error with (L < (576 - OHLEN)), it discovers that IPv4 fragmentation is occurring in the network but it cannot determine the true MTU of the restricting link due to a router on the path generating runt first fragments. The ITE should therefore search for a reduced S_MSS value (to a minimum of S_MRU/8) through an iterative searching strategy that parallels (Section 5 of [RFC1191]).

This searching strategy may require multiple iterations of sending SEAL packets with DF=0 using a reduced S_MSS and receiving additional Fragmentation Needed messages, but it will soon converge to a stable value. During this process, it is essential that the ITE reduce S_MSS based on the first Fragmentation Needed message received, and refrain from further reducing S_MSS until ICMPv4 Fragmentation Needed messages pertaining to packets sent under the new S_MSS are received.

As an optimization only, the ITE MAY transcribe SEAL-encapsulated Fragmentation Needed messages that contain sufficient information into corresponding PTB messages to return to the original source.

4.3. ETE Specification

4.3.1. Reassembly Buffer Requirements

ETEs MUST be capable of using IPv4-layer reassembly to reassemble SEAL protocol outer IPv4 packets up to 2KB in length, and MUST also be capable of using SEAL-layer reassembly to reassemble mid-layer packets up to (2KB - OHLEN). Note that the ITE must retain the SEAL/*/IPv4 header during both IPv4-layer and SEAL-layer reassembly for the purpose of associating the fragments/segments of the same packet.

4.3.2. IPv4-Layer Reassembly

The ETE performs IPv4 reassembly as normal, and should maintain a conservative high- and low-water mark for the number of outstanding reassemblies pending for each ITE. When the size of the reassembly buffer exceeds this high-water mark, the ETE actively discards incomplete reassemblies (e.g., using an Active Queue Management (AQM) strategy) until the size falls below the low-water mark. The ETE should also use a reduced IPv4 maximum segment lifetime value (e.g., 15 seconds), i.e., the time after which it will discard an incomplete IPv4 reassembly for a SEAL protocol packet. Finally, the ETE should also actively discard any pending reassemblies that clearly have no opportunity for completion, e.g., when a considerable number of new IPv4 fragments have been received before a fragment that completes a pending reassembly has arrived.

After reassembly, the ETE either accepts or discards the reassembled packet based on the current status of the IPv4 reassembly cache (congested versus uncongested). The SEAL_ID included in the IPv4 first fragment provides an additional level of reassembly assurance, since it can record a distinct arrival timestamp useful for associating the first fragment with its corresponding non-initial fragments. The choice of accepting/discarding a reassembly may also depend on the strength of the upper-layer integrity check if known (e.g., IPSec/ESP provides a strong upper-layer integrity check) and/or the corruption tolerance of the data (e.g., multicast streaming audio/video may be more corruption-tolerant than file transfer, etc.). In the limiting case, the ETE may choose to discard all IPv4 reassemblies and process only the IPv4 first fragment for SEAL-encapsulated error generation purposes (see the following sections).

4.3.3. Generating SEAL-Encapsulated ICMPv4 Fragmentation Needed Messages

During IPv4-layer reassembly, the ETE determines whether the packet belongs to the SEAL protocol by checking for SEAL_PROTO in the outer IPv4 header (i.e., for simple IPv4 encapsulation) or for SEAL_PORT in the outer */IPv4 header (e.g., for '*'=UDP). When the ETE processes the IPv4 first fragment (i.e, one with DF=1 and Offset=0 in the IPv4 header) of a SEAL protocol IPv4 packet with (R=1; SEG=0) in the SEAL header, it sends a SEAL-encapsulated ICMPv4 Fragmentation Needed message back to the ITE with the MTU field set to 0. (Note that setting a non-zero value in the MTU field of the ICMPv4 Fragmentation Needed message would be redundant with the length value in the IPv4 header of the first fragment, since this value is set to the correct path MTU through in-the-network fragmentation. Setting the MTU field to 0 therefore avoids the ambiguous case in which the MTU field and the IPv4 length field of the first fragment would record different non-zero values.)

When the ETE processes a SEAL protocol IPv4 packet with (A=1; SEG=0) for which no IPv4 reassembly was required, or for which IPv4 reassembly was successful and the R bit was not set, it sends a SEAL-encapsulated ICMPv4 Fragmentation Needed message back to the ITE with the MTU value set to 0. Note therefore that when both the A and R bits are set and fragmentation occurs, the ETE only sends a single ICMPv4 Fragmentation Needed message, i.e., it does not send two separate messages (one for the first fragment and a second for the reassembled whole SEAL packet).

The ETE prepares the ICMPv4 Fragmentation Needed message by encapsulating as much of the first fragment (or the non-fragmented packet) as possible in outer */SEAL/*/IPv4 headers without the length of the message exceeding 576 bytes, as shown in Figure 3:

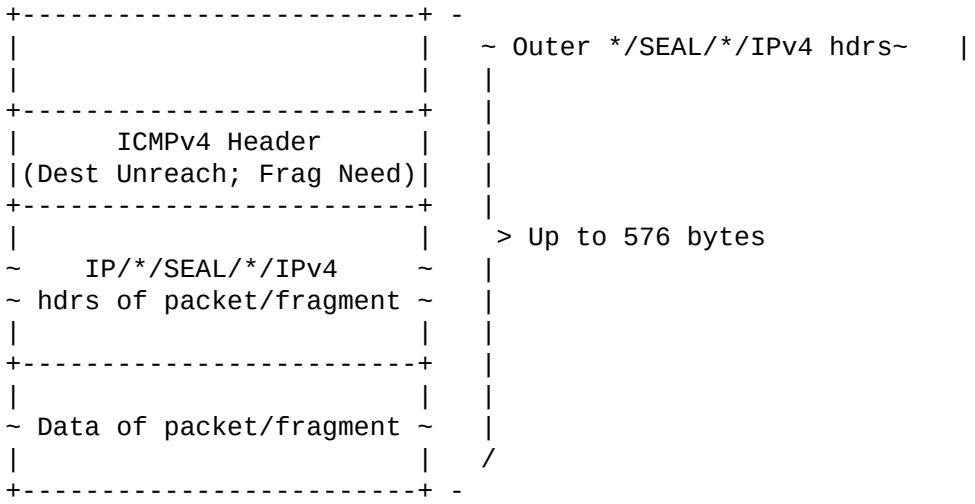

Figure 3: SEAL-Encapsulated ICMPv4 Fragmentation Needed Message

The ETE next sets A=0, R=0, and SEG=0 in the outer SEAL header, sets the SEAL_ID the same as for any SEAL packet, then sets the SEAL Next Header field and the fields of the outer */IPv4 headers the same as for ordinary SEAL encapsulation. The ETE then sets the outer IPv4 destination and source addresses to the source and destination addresses (respectively) of the packet/fragment. If the destination address in the packet/fragment was multicast, the ETE instead sets the outer IPv4 source address to an address assigned to the underlying IPv4 interface. The ETE finally sends the SEAL-encapsulated ICMPv4 message to the ITE the same as specified in Section 4.2.5, except that when the A bit in the packet/fragment is not set, the ETE sends the messages subject to rate limiting since it is not entirely critical that all fragmentation be reported to the ITE.

4.3.4. SEAL-Layer Reassembly

Following IPv4 reassembly of a SEAL packet with (RSV!=0; SEG=0), if the packet is not a SEAL-encapsulated ICMPv4 message, the ETE generates a SEAL-encapsulated ICMPv4 Parameter Problem message with pointer set to the flags field in the SEAL header, sends the message back to the ITE in the same manner specified in Section 4.3.3, then drops the packet. For all other SEAL packets, the ETE adds the packet to a SEAL-Layer pending-reassembly queue if either the M bit or the SEG field in the SEAL header is non-zero.

The ETE performs SEAL-layer reassembly through simple in-order concatenation of the encapsulated segments from N consecutive SEAL protocol packets from the same mid-layer packet. SEAL-layer

reassembly requires the ETE to maintain a cache of recently received segments for a hold time that would allow for reasonable inter-segment delays. The ETE uses a SEAL maximum segment lifetime of 15 seconds for this purpose, i.e., the time after which it will discard an incomplete reassembly. However, the ETE should also 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.

The ETE reassembles the mid-layer packet segments in SEAL protocol packets that contain segment numbers 0 through N-1, with M=1/0 in non-final/final segments, respectively, and with consecutive SEAL_ID values. That is, for an N-segment mid-layer packet, reassembly entails the concatenation of the SEAL-encapsulated segments with (segment 0, SEAL_ID i), followed by (segment 1, SEAL_ID ((i + 1) mod 2^32)), etc. up to (segment N-1, SEAL_ID ((i + N-1) mod 2^32)). (For SEAL encapsulations specifically designed for traversal of IPv4 NATs, the ETE instead uses only a 16-bit SEAL_ID value, and uses mod 2^16 arithmetic to associate the segments of the same packet.)

4.3.5. Delivering Packets to Upper Layers

Following SEAL-layer reassembly, the ETE silently discards the reassembled packet if it was a NULL packet (see Section 4.2.4). In the same manner, the ETE silently discards any reassembled mid-layer packet larger than (2KB - OHLEN) that either experienced IPv4 fragmentation or did not arrive as a single SEAL segment.

Next, if the ETE determines that the inner packet would cause an ICMPv4 error message to be generated, it generates a SEAL-encapsulated ICMPv4 error message, sends the message back to the ITE in the same manner specified in Section 4.3.3, then either accepts or drops the packet according to the type of error. Otherwise, the ETE delivers the inner packet to the upper-layer protocol indicated in the Next Header field.

5. SEAL Protocol Specification - Transport Mode

Section 4 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.

6. Link Requirements

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

7. 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 use Packetization Layer Path MTU Discovery per [RFC4821].

8. 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.

9. IANA Considerations

SEAL_PROTO, SEAL_PORT, and SEAL_OPTION are taken from their respective range of experimental values documented in [RFC3692] and [RFC4727]. These values are for experimentation purposes only, and not to be used for any kind of deployments (i.e., they are not to be shipped in any products).

10. Security Considerations

Unlike IPv4 fragmentation, overlapping fragment attacks are not possible due to the requirement that SEAL segments be non-overlapping.

An amplification/reflection attack is possible when an attacker sends IPv4 first fragments with spoofed source addresses to an ETE, resulting in a stream of ICMPv4 Fragmentation Needed messages

returned to a victim ITE. The encapsulated segment of the spoofed IPv4 first fragment provides mitigation for the ITE to detect and discard spurious ICMPv4 Fragmentation Needed messages.

The SEAL header is sent in-the-clear (outside of any IPsec/ESP encapsulations) the same as for the outer */IPv4 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.

11. 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 tunnel-mode SEAL. SEAL could therefore be considered as a fully functioned manifestation of the method postulated by that informational reference; however, SEAL also supports other modes of operation including transport-mode and duplicate packet detection.

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 A of this document.

12. 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, IPv4 fragmentation provides a short-term interim mechanism for ensuring that packets are delivered while SEAL adjusts its packet sizing parameters.

2. Classical path MTU discovery requires that routers generate an ICMP PTB message for *all* packets lost due to an MTU restriction; this situation is exacerbated at high data rates and becomes severe for in-the-network tunnels that service many communicating end systems. Since SEAL ensures that packets no larger than S_MRU are delivered, however, it is sufficient for the ETE to return ICMPv4 Fragmentation Needed messages subject to rate limiting and not for every packet-in-error.

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

4. 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.

5. Using SEAL, ETEs encapsulate ICMP error messages in an outer SEAL header such that packet-filtering network middleboxes can distinguish them from "raw" ICMP messages that may be generated by an attacker.

6. Most importantly, all SEAL packets have a 32-bit Identification value that can be used for duplicate packet detection purposes and to match ICMP error messages with actual packets sent without requiring per-packet state. Moreover, the SEAL ITE can be configured to accept ICMP feedback only from the legitimate ETE; hence, the packet spoofing-related denial-of-service attack vectors open to the classical methods are eliminated.

In summary, the SEAL approach represents an architecturally superior method for ensuring that packets of various sizes are either delivered or deterministically dropped. When end systems use their own end-to-end MTU determination mechanisms [RFC4821], the SEAL advantages are further enhanced.

13. Acknowledgments

The following individuals are acknowledged for helpful comments and suggestions: Jari Arkko, Fred Baker, Iljitsch van Beijnum, Teco Boot, Bob Braden, Brian Carpenter, Steve Casner, Ian Chakeres, Remi Denis-Courmont, Aurnaud Ebalard, Gorry Fairhurst, Joel Halpern, John Heffner, Thomas Henderson, Bob Hinden, Christian Huitema, Joe Macker, Matt Mathis, Erik Nordmark, Dan Romascanu, Dave Thaler, Joe Touch, Magnus Westerlund, Robin Whittle, James Woodyatt, and members of the Boeing PhantomWorks 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.

14. References

14.1. 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.

14.2. Informative References

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

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

[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.

[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.

[RFC2004] Perkins, C., "Minimal Encapsulation within IP", RFC 2004, October 1996.

[RFC2764] Gleeson, B., Lin, A., Heinanen, J., 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.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, January 2004.

[RFC3819] Karn, P., Ed., 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.

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

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 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.

[RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, November 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.

[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.

Appendix A. Historic Evolution of PMTUD

(Taken from "Neighbor Affiliation Protocol for IPv6-over-(foo)-over-IPv4"; written 10/30/2002):

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 leaving only the final to consider. But, [FOLK] has shown that IP_ID wraparound simply does not occur within several orders of magnitude the reassembly timeout window on high-bandwidth networks.

(Author's 2/11/08 note: this final point was based on a loose interpretation of [FOLK], and is more accurately addressed in [RFC4963].)

Appendix B. Reliability Extensions

The SEAL header includes a Reserved (RSV) field that is set to zero for the purpose of this specification. This field may be used by future updates to this specification for the purpose of improved reliability in the face of loss due to congestion, signal intermittence, etc. Automatic Repeat-ReQuest (ARQ) mechanisms are used to ensure reliable delivery between the endpoints of physical links (e.g., on-link neighbors in an IEEE 802.11 network) as well as between the endpoints of an end-to-end transport (e.g., the endpoints of a TCP connection). However, ARQ mechanisms may be poorly suited to in-the-network elements such as the SEAL ITE and ETE, since retransmission of lost segments would require unacceptable state maintenance at the ITE and would result in packet reordering within the subnetwork.

Instead, alternate reliability mechanisms such as Forward Error Correction (FEC) may be specified in future updates to this specification for the purpose of improved reliability. Such mechanisms may entail the ITE performing proactive transmissions of redundant data, e.g., by sending multiple copies of the same data. Signaling from the ETE (e.g., by sending SEAL-encapsulated ICMPv4 Source Quench messages) may be specified in a future document as a means for the ETE to dynamically inform the ITE of changing FEC conditions.

Author's Address

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

EMail: fltemplin@acm.org