Abstract

Subnetworks are connected network regions bounded by border routers that forward unicast and multicast packets over a virtual topology manifested by tunneling. This virtual topology resembles a "virtual ethernet" link, but may span multiple IP- and/or sub-IP layer forwarding hops that can introduce packet duplication and/or traverse 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.
1. Introduction

For the purpose of this document, subnetworks are defined as connected network regions bounded by border routers. Examples include the global Internet interdomain routing core, Mobile Ad Hoc Networks (MANETs) and enterprise networks. These subnetworks are manifested as a virtual topology that may span many underlying networks and traditional IP subnets, e.g., in the internal organization of an enterprise network.

Subnetwork border routers forward unicast and multicast packets over the virtual topology across multiple IP- and/or sub-IP layer forwarding hops which may introduce packet duplication and/or traverse links with diverse Maximum Transmission Units (MTUs). It is also expected that these subnetwork border routers will support operation of the Internet protocols [RFC0791][RFC2460].

As internet technology and communication has grown and matured, many techniques have developed that use virtual topologies (frequently tunnels of one form or another) over an actual IP network. Those virtual topologies have elements which 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 which are often not even visible to the end-points of the virtual hop. This introduces many failure modes that are not dealt with well in current approaches.

The use of IP encapsulation has long been considered as an alternative 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 shown to be harmful through operational experience and studies conducted over the course of many years [FRAG][FOLK][RFC2923][RFC4459][RFC4963].

This document proposes a Subnetwork Encapsulation and Adaptation Layer (SEAL) for the operation of IP over subnetworks that connect routers via Ingress- and Egress Tunnel Endpoints (ITEs/ETEs). SEAL supports simple and robust duplicate packet detection, and accommodates links with diverse MTUs by introducing a new encapsulation format. The SEAL encapsulation introduces an extended Identification field for packet identification and enables a mid-layer segmentation and reassembly capability that allows an in-the-network cutting and pasting of packets without invoking IP fragmentation. The SEAL protocol is specified in the following sections.
2. Terminology and Requirements

The term "subnetwork" in this document refers to a connected network region bounded by border routers that connect over a virtual topology manifested through tunneling that appears as a fully-connected shared link, i.e., a "Virtual Ethernet (VET)" [I-D.templin-autoconf-dhcp].

The terms "inner" and "outer" are used extensively throughout this document to respectively refer to the innermost IP {layer, protocol, header, packet, etc.} *before* any encapsulation, and the outermost IP {layer, protocol, header, packet etc.} *after* any encapsulation. Between these inner and outer layers, there may also be mid-layer encapsulations, including the SEAL encapsulation. These mid-layer encapsulations are denoted as '*' (where '*' may signify NULL, a single mid-layer encapsulation, or multiple mid-layer encapsulations.)

The notation IPvX/*/IPvY refers to an inner IPvX packet encapsulated in any '*' mid-layer headers followed by an outer IPvY header.

The notation "IP" means either IP protocol version (IPv4 or IPv6).

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

Subnetwork - a connected network region that is bounded by border routers

SEAL - Subnetwork Encapsulation and Adaptation Layer

VET - Virtual ETHERnet

MANET - Mobile Ad-hoc Network

ITE - Ingress Tunnel Endpoint

ETE - Egress Tunnel Endpoint

MTU - Maximum Transmission Unit

S-MSS - SEAL Maximum Segment Size

EMTU_R - Effective MTU to Receive

PTB - an ICMPv6 "Packet Too Big" or an ICMPv4 "fragmentation needed" message
DF - the IPv4 header Don’t Fragment flag

ENCAPS - the size of the outer encapsulating SEAL/*/IPv4 headers

FRAGREP - a Fragmentation Report message

SEAL packet - a segment of an inner IP packet encapsulated in outer SEAL/*/IPv4 headers

SEAL-ID - a 32-bit Identification value; randomly initialized and monotonically incremented for each SEAL packet

Unfragmentable - an IPv4 packet with DF=1, or an IPv6 packet

The keywords 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 inserts an additional mid-layer encapsulation when IP/*/IPv4 encapsulation is used, and appears as a subnetwork encapsulation as seen by inner layers. SEAL was motivated by the specific use case of subnetwork abstraction for MANETs, however the domain of applicability also extends to subnetwork abstractions of enterprise networks, the interdomain routing core, etc.

SEAL can be used as a mid-layer encapsulation above an outer UDP/IPv4 encapsulation, however the technique of concatenating the SEAL 16-bit ID Extension and the IPv4 ID (i.e., co-mingling the two identifier spaces) will not work when there are network address translators (NATs) in the path that may re-write the IPv4 ID, e.g., such as for the Teredo domain of applicability [RFC4380]. A variation of this proposal that maintains separate ID spaces for the SEAL-ID and IPv4 ID and that operates in the presence of NATs and firewalls will be specified in a future version of this document.

The current document version speaks exclusively to the use of IPv4 as the outer encapsulation layer, however the same principles apply when IPv6 is the outer layer. In-the-network fragmentation is not permitted for encapsulations over IPv6, however, so the "implicit" probing capabilities specified for IPv4 in this document are not available. Still, encapsulations over IPv6 can use "explicit" probing as well as the same architectural concepts as specified for IPv4 herein. A future version of this document will address the case of IPv6 as the outer encapsulation layer in more detail.
For further study, SEAL may also be useful for "transport-mode" applications, e.g., when the inner layer includes ordinary protocol data rather than an encapsulated IP packet.

4. SEAL Protocol Specification

4.1. Model of Operation

Ingres Tunnel Endpoints (ITEs) insert a SEAL header in the IP/*/IPv4-encapsulated packets they 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 defines a new IP protocol type and a new mid-layer encapsulation for both unicast and multicast inner IP packets. The ITE inserts a SEAL header during encapsulation as shown in Figure 1:

```
+-------------------------+        +-------------------------+
|                         |        |                         |
|   Outer */IPv4 headers   |        |   Any mid-layer * headers |
|                         |        |                         |
+-------------------------+        +-------------------------+
|                         |        |                         |
|       SEAL Header       |        |       Any mid-layer * headers |
|                         |        |                         |
+-------------------------+        +-------------------------+
|                         |        |                         |
| Any mid-layer * headers |        | Any mid-layer * headers |
|                         |        |                         |
+-------------------------+        +-------------------------+
|                         |        |                         |
|          Inner IP       |  --->  |          Inner IP       |
|                         |        |                         |
+-------------------------+        +-------------------------+
|                         |        |                         |
|                         |        |    Any mid-layer trailers |
|                         |        |                         |
+-------------------------+        +-------------------------+
|                         |        |    Any mid-layer trailers |
|                         |        |                         |
+-------------------------+        +-------------------------+
|                         |        |                         |
|                         |        |    Any outer trailers   |
|                         |        |                         |
+-------------------------+
```

Figure 1: SEAL Encapsulation

where the SEAL header is inserted as follows:

- For simple IP/IPv4 encapsulations (e.g., [RFC2003][RFC2004][RFC4213]), the SEAL header is inserted between the inner IP and outer IPv4 headers as: IP/SEAL/IPv4.
o For tunnel-mode IPsec encapsulations over IPv4, [RFC4301], the SEAL header is inserted between the {AH,ESP} header and outer IPv4 headers as: IP/*/{AH,ESP}/SEAL/IPv4.

o For IP encapulations over transports such as UDP (e.g., [I-D.farinacci-lisp]), the SEAL header is inserted immediately after the outer transport layer header, e.g., as IP/*/SEAL/UDP/IPv4.

Encapsulation and tunneling establishes an abstraction of the subnetwork that connects all ITEs and ETEs as single-hop neighbors as though they were attached to a virtual ethernet (VET). From a physical perspective, however, packets sent over the subnetwork may be forwarded across many IP and/or sub-IP layer hops.

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 ID value in the outer IPv4 header as the least-significant bits. Routers use the SEAL-ID for duplicate packet detection within the subnetwork as well as for SEAL segmentation and reassembly.

SEAL enables a multi-level segmentation and reassembly capability. First, the ITE can use IPv4 fragmentation for fragmentable inner IPv4 packets before encapsulation to avoid lower-level segmentation and reassembly. Secondly, the SEAL layer itself provides a simple mid-layer cutting-and-pasting of inner IP packets to avoid IPv4 fragmentation on the outer packet. Finally, ordinary IPv4 fragmentation for the outer IPv4 packet after SEAL encapsulation is permitted under certain limited and carefully managed circumstances.

4.2. Packetization

4.2.1. Packet Size Considerations

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 be delivered by the network without loss due to an MTU restriction on the path, or a suitable ICMP PTB message returned. However, PTB messages can be dropped in the network, and any PTBs received could be erroneous or maliciously fabricated. (Indeed, in the case of treating the global Internet interdomain routing core as a subnetwork, the PTB messages could come from anywhere in the Internet.) The ITE therefore requires a means for conveying 1500 byte (or smaller) original packets to the ETE without loss due to link MTU restrictions and/or triggering PTB messages from within the subnetwork.
In common deployments, there may be many forwarding hops between the source and the ITE. Within those hops, there may be additional encapsulations (IPSec, L2TP, etc.) such that a 1500 byte original packet might grow to a larger size by the time it reaches the ITE. Similarly, additional encapsulations on the path from the ITE to the ETE could cause the packet to become larger still and trigger in-the-network fragmentation. In order to preserve the end system expectation of delivery for 1500 byte and smaller packets, the ITE therefore requires a means for conveying this larger packet to the ETE even though there may be links within the subnetwork that configure a smaller MTU.

The ITE upholds the 1500-byte-and-smaller packet delivery expectation by instituting a SEAL Maximum Segment Size (S-MSS) variable, configurable within the range of [128 - 2KB]. The ITE also institutes a segmentation region for packet sizes [S-MSS - 2KB] such that all inner IP packets within this size range are segmented into multiple SEAL packets to avoid in-the-network IPv4 fragmentation.

The ITE must be configured to either drop unfragmentable inner IP packets larger than 2KB (and return a suitable ICMP PTB message), or admit them into the tunnel as single-segment SEAL packets. If the ITE is configured to admit such packets, it MUST maintain sufficient state for caching the MTU values reported in PTB messages received from within the tunnel. Configuration can be either on a per-interface or per-ETE basis.

4.2.2. Inner Fragmentation

The IPv4 layer of a subnetwork border router that configures an ITE fragments inner IPv4 packets larger than 2KB and with the IPv4 Don’t Fragment (DF) bit set to 0 into IPv4 fragments no larger than MIN(2KB, S-MSS). The IPv4 layer then submits each inner IPv4 fragment to the ITE as an independent IP packet for encapsulation. Note that inner fragmentation may not be available for certain ITE types, e.g., for tunnel-mode IPsec. Any inner IPv4 fragments created in this fashion will be reassembled by the final destination.

4.2.3. SEAL Segmentation and Encapsulation

After any inner fragmentation, the ITE encapsulates each inner IP packet/fragment according to its size.

When the ITE is configured to admit unfragmentable inner IP packets larger than 2KB into the tunnel, it MUST NOT break them into smaller segments but rather MUST encapsulate each inner packet as a single segment SEAL packet. When the ITE is configured to discard unfragmentable inner packets larger than 2KB, it drops the packet and
sends a suitable ICMP PTB message back to the original source.

For inner IP packets no larger than 2KB, the ITE encapsulates the packet in any mid-layer '*' headers, then performs SEAL segmentation on this inner packet based on a segment size (S-MSS) that will avoid IPv4 fragmentation within the subnetwork. The ITE maintains S-MSS for each ETR (including IPv4 multicast destinations) as per-ETR soft state, where S-MSS is configured to a value within the [128 - 2KB] range based on static configuration and/or dynamic segment size probing.

Note that this SEAL segmentation ignores the DF bit in the inner IPv4 header or (in the case of IPv6) ignores the fact that the network is not permitted to perform IPv6 fragmentation. This segmentation process is a mid-layer (not an IP layer) operation employed by the ITE to adapt the inner IP packet to the subnetwork path characteristics, and the ETE will restore the inner packet to its original form when it removed the packet from the subnetwork. Therefore, the fact that the packet may have been segmented within the subnetwork is not observable by the final destination.

The ITE breaks inner IP packets no larger than 2KB into N segments (N \(\leq 16\)) that are no larger than S-MSS bytes each, i.e., even if the inner packet is unfragmentable. Each segment except the final one MUST be of equal length, while the final segment MAY be of different length and 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 encapsulates each segment in a SEAL header formatted as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          ID Extension         |R|M|CTL|Segment|  Next Header  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: SEAL Header Format

where the header fields are defined as follows:

- ID Extension (16)
  a 16-bit extension of the 16-bit ID field in the outer IPv4 header; encodes the most-significant 16 bits of a 32 bit SEAL-ID value.
Internet-Draft                    SEAL                        April 2008

R (1)
   Reserved.

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

CTL (2)
    a 2-bit "Control" field that identifies the type of SEAL packet as follows:
    ’00’ - a Fragmentation Report (FRAGREP).
    ’01’ - a non-probe SEAL packet.
    ’10’ - an implicit probe.
    ’11’ - an explicit probe.

Segment (4)
    a 4-bit Segment number. Encodes a segment number between 0 - 15.

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

For single-segment inner IP packets, the ITE encapsulates the segment in a SEAL header with (M=0; Segment=0). For N-segment inner packets (N <= 16), the ITE encapsulates each segment in a header of the same format with (M=1; Segment=0) for the first segment, (M=1; Segment=1) for the second segment, etc., with the final segment setting (M=0; Segment=N-1).

The ITE next sets CTL in the SEAL header of each segment according to the SEAL packet type (see: Section 4.6), writes the IP protocol number corresponding to the inner payload in the ‘Next Header’ field, and encapsulates the segment in the requisite */IPv4 outer headers.

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 packet sent to the ETE. For each SEAL packet, the ITE writes the least-significant 16 bits of the SEAL-ID value in the ID field in the outer IPv4 header, and writes the most-significant 16 bits in the ID Extension field in the SEAL header.

The ITE finally sets other fields in the outer */IPv4 headers according to the specific encapsulation format (e.g., [RFC2003],...
4.2.4. Sending Packets

For unfragmentable inner IP packets larger than 2KB, if the ITE is configured to drop the packet it sends an ICMP PTB message back to the original source with an MTU value of 2KB. Otherwise, it determines whether the size of the packet plus the size of the SEAL/*/IPv4 encapsulation headers is larger than the IPv4 path MTU for the ETE. If the packet is too large, the ITE discards it and sends a PTB message back to the original source with an MTU value set to the IPv4 path MTU minus the size of the encapsulating headers. Otherwise, the ITE sets the Don’t Fragment (DF) bit in the outer IPv4 header to DF=1 and admits the packet into the tunnel.

For inner IP packets that were no larger than 2KB before segmentation, the ITE sets DF=0 in the outer IPv4 header of each segment and sends them into the tunnel in canonical order, i.e., Segment 0 first, then Segment 1, etc.

4.3. Reassembly

4.3.1. Reassembly Buffer Requirements

ETEs MUST be capable of using IPv4-layer reassembly to reassemble SEAL packets of at least (2KB+ENCAPS) bytes, i.e., ETEs MUST configure an IPv4 Effective MTU to Receive (EMTU_R) of at least (2KB+ENCAPS).

ETEs MUST also be capable of using SEAL-layer reassembly to reassemble inner IP packets of at least 2KB, i.e., ETEs MUST configure a SEAL EMTU_R of at least 2KB.

4.3.2. IPv4-Layer Reassembly

The ETE performs IPv4 reassembly as-normal, and maintains a conservative high- and low-water mark for the number of outstanding reassemblies pending for each ITE per common operational practices. 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 such as drop-eldest, Random Early Drop (RED), etc.) until the size falls below the low-water mark.

After reassembly, the ETE either accepts or discards the reassembled SEAL packet based on the current status of the IPv4 reassembly cache (congested vs uncongested). 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.).

The 32-bit 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.

4.3.3. SEAL-Layer Reassembly

After any IPv4-layer reassembly, the ETE performs SEAL-layer reassembly for N-segment inner IP packets through simple in-order concatenation of the encapsulated segments from N consecutive SEAL packets. These packets contain Segment numbers 0 through N-1, and with consecutive SEAL-ID values encoded in the 32-bit concatenation of the ID Extension field in the SEAL header and the ID field in the IPv4 header. That is, for an N-segment packet, reassembly of the inner packet entails the concatenation of the encapsulated segments of SEAL packets 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)). (The SEAL header and outer */IPv4 headers are discarded during this process.) This requires the ETE to maintain a cache of recently received SEAL packets for a hold time that would allow for reasonable inter-segment delays.

As for IPv6 reassembly [RFC2460], SEAL reassembly uses a maximum segment lifetime of 60 seconds, i.e., the time after which an incomplete reassembly is discarded. However, the ETE must also actively discard any pending reassemblies that appear to have no opportunity for completion, e.g., when a considerable number of SEAL packets have been received before a packet that completes the pending reassembly has arrived. This assumes that any packet reordering within the subnetwork will be on the order of a small number of positions and that any gross reordering will be short-lived in nature.

4.3.4. Reassembly Integrity Checks

TBD - a future version of this draft may specify an integrity check vector, inserted by the ITE during encapsulation and used by the ETE to detect packet splicing errors during IPv4 reassembly. Such an integrity check capability is specified in [I-D.templin-inetmtu].
4.4. Generating Fragmentation Reports

When the ETE receives the first fragment of a SEAL packet that was delivered as multiple IPv4 fragments and with CTL=’1X’ in the SEAL header, it generates a Fragmentation Report (FRAGREP) message to send back to the ITE. The ETE also generates a FRAGREP for any SEAL packet with CTL=’11’ even if the packet was not fragmented.

The ETE prepares the FRAGREP message by encapsulating the leading 128 bytes (or up to the end) of the first fragment in outer SEAL/*/IPv4 headers. The ETE next sets CTL=’00’ in the SEAL header and sets the fields of the outer */IPv4 headers according to the specific encapsulation type. In particular, the ETE sets the destination address of the FRAGREP to the source address that was included in the first fragment, and sets the source address of the FRAGREP to the destination address that was included in the first fragment. If the destination address in the first fragment was multicast, the ETE instead sets the source address of the FRAGREP to an address assigned to the underlying IPv4 interface.

The FRAGREP message has the following format:

```
+-------------------------+       |       +-------------------------+
|                         |       |       |                         |
|   Outer */IPv4 headers   |       |       |   First 128 bytes of IPv4 first fragment |
|                         |       |       |                         |
| +-------------------------+       |       +-------------------------+
|  SEAL Header            |       |       |                         |
|     (CTL='00', Segment=0)|       |       |                         |
| +-------------------------+       |       +-------------------------+
```

Figure 3: Fragmentation Report (FRAGREP) Message

4.5. Receiving Fragmentation Reports

When the ITE receives a potential FRAGREP message, it first verifies that the message was formatted correctly by the ETE (per Section 4.4) and confirms that the FRAGREP matches one of the implicit/explicit probes that it actually sent to the ETE, e.g., by examining the SEAL-ID embedded in the encapsulated IPv4 first fragment. If the FRAGREP matches one of its probes, the ITE advances its window of outstanding probes (see: Section 4.6).
For each FRAGREP that contains the leading portion of a whole IPv4 packet, if the length field in the whole packet contains a value larger than S-MSS the ITE sets S-MSS for this ETE to this length minus ENCAPS. For each FRAGREP that contains the leading portion of an IPv4 fragment, if the length field in the fragment contains a value larger than \( (128+\text{ENCAPS}) \), the ITE sets S-MSS for this ETE to this length minus ENCAPS; otherwise, it sets S-MSS = \( \min(S-MSS/2, 128) \).

The above "limited halving" procedure accounts for the possibility that the ETE receives IPv4 first fragments that were created as the smallest fragment (rather than the largest). In that case, convergence to an acceptable S-MSS size may require multiple iterations of sending SEAL packets and receiving FRAGREP messages in a manner that parallels classical path MTU discovery \([RFC1191]\), albeit with all feedback coming from the ETE and not a network middlebox. This limited halving procedure ensures that convergence will occur quickly even in extreme cases and without packet loss, while the correct MTU will normally be determined in a single iteration since routers typically produce the first fragment as the largest \([RFC1812]\).

### 4.6. S-MSS Probing

For inner IP packets no larger than 2KB, when S-MSS is larger than 128 the ITE uses each packet as an implicit probe to detect any in-the-network IPv4 fragmentation. The ITE sets CTL='10' in the SEAL header and DF=0 in the outer IPv4 header of each SEAL packet, and will receive FRAGREP messages from the ETE if fragmentation occurs. When S-MSS=128, the ITE instead sets CTL='01' in the SEAL header to avoid generating FRAGREPs for unavoidable in-the-network fragmentation.

The ITE should also send explicit probes periodically to manage a "window" of outstanding probes that allows the ITE to validate any FRAGREPs it receives (e.g., by examining the SEAL-ID). The ITE sends explicit probes by setting CTL='11' in the SEAL header and DF=0 in the IPv4 header. The ITE can also probe for larger S-MSS values by sending explicit probes with trailing padding added to create a probe packet of up to 2KB. When the ETE receives an explicit probe, it will return a FRAGREP message whether or not any in-the-network fragmentation occurred, which the ITE will process exactly as for any FRAGREP per Section 4.5.

For inner IP packets larger than 2KB, the ITE set DF=1 in the outer IPv4 header and may set CTL to any value other than '00', i.e., the packets may be sent as either non-probes or implicit/explicit probes but their use for probing may be of little value.
4.7. Processing ICMP PTBs

The ITE may receive ICMP PTB messages in response to any packets that were admitted into the tunnel with DF=1. The ITE SHOULD consult the SEAL 32-bit ID included in the packet-in-error to ensure that the PTB corresponds to a recently-sent packet. The ITE then records the MTU value from the PTB message in the IPv4 path MTU cache. If the PTB message includes enough information, the ITE then translates the message into a suitable PTB to send back to the original source; otherwise, it discards the message. During translation, the ITE sets the MTU value in the PTB message to MAX(2KB, the MTU reported in the non-translated PTB).

5. Link Requirements

Subnetwork designers are strongly encouraged to follow the recommendations in [RFC3819] when configuring link MTUs.

6. End System Requirements

End systems that send unfragmentable IP packets larger than 1500 bytes are strongly encouraged to use Packetization Layer Path MTU Discovery per [RFC4821], since the network may not always be able to return ICMP PTB messages in 1-to-1 correspondence with dropped packets.

7. Router Requirements

IPv4 routers observe the requirements in [RFC1812].

8. IANA Considerations

A new IP protocol number for the SEAL protocol is requested.

9. 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 spoofed IPv4 first fragments to an ETE, resulting in a stream of FRAGREP 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 FRAGREPs.

The SEAL header is sent in-the-clear (outside of any IPsec/ESP
encapsulations) the same as for the IPv4 header. As for IPv6
extension headers, the SEAL header is also protected only by L2
integrity checks, and is not covered under any L3 integrity checks.

10. Acknowledgments

Path MTU determination through the report of fragmentation
experienced by the final destination was first proposed by Charles
Lynn of BBN on the TCP-IP mailing list in May 1987. An historical
analysis of the evolution of path MTU discovery appears in
http://www.tools.ietf.org/html/draft-templin-v6v4-ndisc-01 and is
reproduced in Appendix A of this document.

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

11. References

11.1. Normative References

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


[ RFC2119 ] Bradner, S., "Key words for use in RFCs to Indicate


11.2. Informative References

[FOLK] C, C., D., D., and k. k, "Beyond Folklore: Observations on


Appendix A. Historic Evolution of PMTUD (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 proposal (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 MTUWG 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.

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

Fred L. Templin (editor)
Boeing Phantom Works
P.O. Box 3707
Seattle, WA  98124
USA

Email: fltemplin@acm.org