Internet Area WG
Internet Draft
Intended status: Informational
Expires: January 2016

J. Touch
USC/ISI
M. Townsley
Cisco

July 20, 2015

IP Tunnels in the Internet Architecture
draft-ietf-intarea-tunnels-01.txt

Copyright Notice

Abstract

This document discusses the role of IP tunnels in the Internet architecture. It explains their relationship to existing protocol layers and the challenges in supporting IP tunneling based on the equivalence of tunnels to links.

Table of Contents

1. Introduction

The Internet is loosely based on the ISO seven layer stack, in which data units traverse the stack by being wrapped inside data units one layer down. A tunnel is a mechanism for transmitting data units between endpoints by wrapping them as data units of the same or higher layers, e.g., IP in IP (Figure 1) or IP in UDP (Figure 2).

```
+----+----+--------------+
| IP'| IP |     Data     |
+----+----+--------------+
```

Figure 1 IP inside IP

```
+----+-----+----+-------------+
| IP'| UDP | IP |    Data     |
+----+-----+----+-------------+
```

Figure 2 IP in UDP in IP in Ethernet

This document focuses on tunnels that transit IP packets, i.e., in which an IP packet is the payload of another protocol. Tunnels provide a virtual link that can help decouple the network topology seen by transiting packets from the underlying physical network [To98][RFC2473]. For example, tunnels were critical in the development of multicast because not all routers were capable of processing multicast packets [Er94]. Tunnels allowed multicast packets to transit between multicast-capable routers over paths that did not support multicast. Similar techniques have been used to support other protocols, such as IPv6 [RFC2460].

Use of tunnels is common in the Internet. The word "tunnel" occurs in over 100 RFCs, and is supported within numerous protocols, including:

- o Generic UDP Encapsulation (GUE) - IP in UDP (in IP)[He15a][He15b]

- o Generic IPv6 tunneling [RFC2473]

- o Generic Router Encapsulation (GRE) - an encapsulation framework allowing different messages to tunnel over a variety of tunnels, e.g., IP in GRE in IP [RFC2473][RFC2784][RFC7588][Pi15]

- o IP in IP / mobile IP [RFC2003][RFC2473][RFC5944]

- o IPsec - hides the original traffic destination [RFC4301]

- o L2TP - Tunnels PPP over IP, used largely in DSL/FTTH access networks to extend a subscriber's connection from an access line provider to an ISP [RFC3931]

- o L2VPNs - provides a link topology different from that provided by physical links [RFC4664]

- o L3VPNs - provides a network topology different from that provided by ISPs [RFC4176]

- o LISP - reduces routing table load within an enclave of routers [RFC6830]

- o MPLS - tunnels IP over a circuit-like path in which identifiers are rewritten on each hop, often used for traffic provisioning [RFC3031]

- o NVO3 - tunnels for data center network sharing (which includes use of GUE, above) [RFC7364]

- o PWE3 - tunnels to emulate wire-like services over packet-switched services [RFC3985]

- o SEAL/AERO - a generic mechanism for IP in IP tunneling designed to overcome the limitations of RFC2003 [RFC5320][Te15]

- o TRILL - enables L3 routing (typically IS-IS) in an enclave of Ethernet bridges [RFC5556][RFC6325]

The variety of tunnel mechanisms raises the question of the role of tunnels in the Internet architecture and the potential need for these mechanisms to have similar and predictable behavior. In particular, the ways in which packet sizes (i.e., Maximum Transmission Unit or MTU) mismatch and error signals (e.g., ICMP) are handled may benefit from a coordinated approach.

It is useful to note that, regardless of the layer in which encapsulation occurs, tunnels emulate a link. As links, they are subject to link issues, e.g., MTU discovery, signaling, and the potential utility of native support for broadcast and multicast [RFC2460][RFC3819]. They have advantages over native links, being potentially easier to reconfigure and control.

The remainder of this document describes the general principles of IP tunneling and discusses the key considerations in the design of a protocol that tunnels IP datagrams. It derives its conclusions from the equivalence of tunnels and links. Note that all considerations are in the context of existing standards and requirements.

2. Conventions used in this document

2.1. Key Words

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 RFC-2119 [RFC2119].

2.2. Terminology

This document uses the following terminology. These definitions are given in the most general terms, but will be used primarily to discuss IP tunnels in this document. They are presented in order from most fundamental to those derived on earlier definitions:

o Messages: variable length data labeled with globally-unique endpoint IDs [RFC791]

o Endpoint: a network device that sources or sinks messages labeled from/to its IDs, also known as a host [RFC1122].

o Forwarder: a network device that relays IP messages using longest-prefix match of destination IDs and local context, when possible, also known as a gateway or router [RFC1812].

o Network node (node): an endpoint or forwarder. For Internet messages (IP datagrams), these are hosts or gateways/routers, respectively.

o Source: the origin host of a message.

o Destination: the receiving host of a message.

o Link: a communication device that transfers messages between network devices, i.e., by which a message can traverse between devices without being processed by a forwarder. Note that the notion of forwarder is relative to the layer at which message processing is considered [RFC1122][RFC1812].

o Path: a communications path by which a message can traverse between network nodes, which may or may not involve being processed by a forwarding node.

o Tunnel: a protocol mechanism that transits messages using encapsulation to allow a path to appear as a link. Note that a protocol can be used to tunnel itself (IP over IP) and that this includes the conventional layering of the ISO stack (i.e., by this definition, Ethernet is a tunnel for IP).

o Ingress: a network node that receives messages, encapsulates them according to the tunnel protocol, and transmits them into the tunnel. Note that the ingress and source can be co-located.

o Egress: a network node that receives messages that have finished transiting a tunnel. The egress decapsulates datagrams for further transit to the destination. Note that the egress and destination can be co-located.

o Tunnel transit packet: the packet arriving at a node connected to a tunnel that enters the ingress and exits the egress, i.e., the packet carried over the tunnel. This is sometimes known as the "tunneled packet", i.e., the packet carried over the tunnel.

o Tunnel link packet: packets that traverse from ingress to egress, in which resides all or part of a tunnel transit packet. This is sometimes known as the "tunnel packet", i.e., the packet of the tunnel itself.

o Link MTU (LMTU): the largest message that can transit a link. Note that this need not be the native size of messages on the link.

o Reassembly MTU (RMTU): the largest message that can be reassembled by a receiver, and is not directly related to the link or path MTU. Sometimes also referred to as "receiver MTU".

o Path MTU (PMTU): the largest message that can transit a path. Typically, this is the minimum of the link MTUs of the links of the path.

o Tunnel MTU (TMTU): the largest message that can transit a tunnel. Typically, this is limited by the egress reassembly MTU.

3. The Tunnel Model

A network architecture is an abstract description of a distributed communications system, its components and their relationships, the requisite properties of those components and the emergent properties of the system that result [To03]. Such descriptions can help explain behavior, as when the OSI seven-layer model is used as a teaching example [Zi80]. Architectures describe capabilities - and, just as importantly, constraints.

A network can be defined as a system of endpoints and relays interconnected by communication paths, abstracting away issues of naming in order to focus on message forwarding. To the extent that the Internet has a single, coherent interpretation, its architecture is defined by its core protocols (IP [RFC791], TCP [RFC793], UDP [RFC768]) and messages, hosts, routers, and links [Cl88][To03], as shown in Figure 3:

```
               +------+    ------      ------    +------+
               |      |   /      \    /      \   |      |
               | HOST |--+ ROUTER +--+ ROUTER +--| HOST |
               |      |   \      /    \      /   |      |
               +------+    ------      ------    +------+
```

Figure 3 Basic Internet architecture

As a network architecture, the Internet is a system of hosts and routers interconnected by links that exchange messages when possible. "When possible" defines the Internet's "best effort" principle. The limited role of routers and links represents the End-to-End Principle [Sa84] and longest-prefix match enables hierarchical forwarding.

Although the definitions of host, router, and link seem absolute, they are often relative as viewed within the context of one OSI layer, each of which can be considered a distinct network architecture. An Internet gateway is a Layer 3 router when it transits IP datagrams but it acts as a Layer 2 host as it sources or sinks Layer 2 messages on attached links to accomplish this transit capability. In this way, a single device (Internet gateway) behaves as different components (router, host) at different layers.

Even though a single device may have multiple roles - even concurrently - at a given layer, each role is typically static and location-independent. An Internet gateway always acts as a Layer 2 host and that behavior does not depend on where the gateway is viewed from within Layer 2. In the context of a single layer, a device's behavior is modeled as a single component from all viewpoints in that layer.

3.1. What is a tunnel?

A tunnel can be modeled as a link in another network [To98][To01][To03]. In Figure 4, a source host (Hsrc) and destination host (Hdst) communicating over a network M in which two routers (Ra and Rd) are connected by a tunnel.

```
                 --_                                 --
     +------+   /  \                                /  \   +------+
     | Hsrc |--+ Ra +----      --      --      ----+ Rd +--| Hdst |
     +------+   \  /    /\    /  \    /  \    /\    \  /   +------+
                 --    /I \--+ Rb +--+ Rc +--/E \    --
                       \  /   \  /    \  /   \  /
                        \/     --      --     \/
                       <------ Network N ------->
     <----------------------- Network M ------------------------>
```

Figure 4 The big picture

The tunnel consists of two elements (ingress I, egress E), that lie along a path connected by a (possibly different) network N. Regardless of how the ingress and egress are connected, the tunnel serves as a link to the devices it connects (here, Ra and Rb).

IP packets arriving at the ingress are encapsulated to traverse network N. We call these packets "tunnel transit packets" because they will now transit the tunnel inside one or more "tunnel link packets". Tunnel link packets use the source address of the ingress and the destination address of the egress - using whatever address is appropriate to the Layer at which the ingress and egress operate (Layer 2, Layer 3, Layer 4, etc.). The egress decapsulates those messages, which then continue on network M as if emerging from a link. To tunnel transit packets, and to the routers the tunnel connects (Ra and Rb), the tunnel acts as a link.

The model of each component (ingress, egress) and the entire system (tunnel) depends on the layer from which you view the tunnel. From the perspective of the outermost hosts (Hsrc and Hdst), the tunnel appears as a link between two routers (Ra and Rd). For routers along the tunnel (e.g., Rb and Rc), the ingress and egress appear as the endpoint hosts and Hsrc and Hdst are invisible.

When the tunnel network (N) is implemented using the same protocol as the endpoint network (M), the picture looks flatter (Figure 5), as if it were running over a single network. However, note that this appearance is incorrect - nothing has changed. From the perspective of the endpoints, Rb and Rc and network N don't exist and aren't visible, and from the perspective of the tunnel, network M doesn't exist. The fact that network N and M use the same protocol, and may traverse the same links is irrelevant.

```
                 --_           --      --            --
     +------+   /  \    /\    /  \    /  \    /\    /  \   +------+
     | Hsrc |--+ Ra +--/I \--+ Rb +--+ Rc +--/E \--+ Rd +--| Hdst |
     +------+   \  /   \  /   \  /    \  /   \  /   \  /   +------+
                 --     \/     --      --     \/     --
                       <------ Network N ------->
     <------------------------ Network M ------------------------>
```

Figure 5 IP in IP network picture

3.2. View from the Outside

From outside the tunnel, to network M, the entire tunnel acts as a link (Figure 6). It may be numbered or unnumbered and the addresses associated with the ingress and egress are irrelevant from outside.

```
                 --_                                 --
     +------+   /  \                                /  \   +------+
     | Hsrc |--+ Ra +------------------------------+ Rd +--| Hdst |
     +------+   \  /                                \  /   +------+
                 --                                  --
     <------------------------ Network M ------------------------>
```

Figure 6 Tunnels as viewed from the outside

A tunnel is effectively invisible to the network in which it resides, except that it behaves exactly as a link. Consequently [RFC3819] requirements for links supporting IP also apply to tunnels.

E.g., the IP datagram hop count (IPv4 Time-to-Live [RFC791] and IPv6 Hop Limit [RFC2460]) are decremented when traversing a router, not by traversing a link - or thus a tunnel. Tunnels have a tunnel MTU - the largest datagram that can transit, just as links have a corresponding link MTU. A link MTU may not reflect the native link message sizes (ATM AAL5 48 byte messages support a 9KB MTU) and the same is true for a tunnel.

3.3. View from the Inside

Within network N, i.e., from inside the tunnel itself, the ingress is a source of tunnel link packets and the egress is a sink - both are hosts on network N (Figure 7). Consequently [RFC1122] Internet host requirements apply to ingress and egress nodes when Network N uses IP (and thus the ingress/egress use IP encapsulation).

```
                   _           --      --
                        /\    /  \    /  \    /\
                       /I \--+ Rb +--+ Rc +--/E \
                       \  /   \  /    \  /   \  /
                        \/     --      --     \/
                       <------ Network N ------->
```

Figure 7 Tunnels, as viewed from within the tunnel

Viewed from within the tunnel, the outer network (M) doesn't exist. Tunnel link packets can be fragmented by the source (ingress) and reassembled at the destination (egress), just as at any endpoint. The path between ingress and egress may have a path MTU but the endpoints can exchange messages as large as can be reassembled at the destination (egress), i.e., an egress MTU. Information about the network - i.e., regarding MTU sizes, network reachability, etc. - are relayed from the destination (egress) and intermediate routers back to the source (ingress), without regard for the external network (M).

3.4. Location of the Ingress and Egress

The ingress and egress are endpoints of the tunnel and the tunnel is a link. The ingress and egress are thus link endpoints at the network nodes the tunnel interconnects. Such link endpoints are typically described as "network interfaces".

Tunnel interfaces may be physical or virtual. The interface may be implemented inside the node where the tunnel attaches, e.g., inside a host or router. The interface may also be implemented as a "bump in the wire" (BITW), somewhere along a link between the two nodes the link interconnects. IP in IP tunnels are often implemented as interfaces, where IPsec tunnels are sometimes implemented as BITW. These implementation variations determine only whether information available at the link endpoints (ingress/egress) can be easily shared with the connected network nodes.

3.5. Implications of This Model

This approach highlights a few key features of a tunnel as a network architecture construct:

- o To the tunnel transit packets, tunnels turn a network (Layer 3) path into a (Layer 2) link
- o To devices the tunnel traverses, the tunnel ingress and egress act as hosts that source and sink tunnel link packets

The consequences of these features are as follow:

- Like a link, a tunnel has an MTU defined by the reassembly MTU of the receiving interface (egress).

- Path MTU discovery in the network layer (i.e., outer network M) has no direct relation to the MTU of the hops within the link layer of the links (or thus tunnels) that connect its components.

- Hops remain defined as the number of routers encountered on a path or the time spent at a router [RFC1812]. Hops are not decremented solely by the transit of a link, e.g., a packet with a hop count of zero should successfully transit a link (and thus a tunnel) that connects two hosts.

- The addresses of a tunnel ingress and egress correspond to link layer addresses to the tunnel transit packet and outer network M. Many point-to-point tunnels are unnumbered in the network in which they reside (even though they must have addresses in the network they transit).

- Like network interfaces, the ingress and egress are never a direct source of ICMP messages but may provide information to their attached host or router to generate those ICMP messages.

These observations make it much easier to determine what a tunnel must do to transit IP packets, notably it must satisfy all requirements expected of a link.

4. IP Tunnel Requirements

The requirements of an IP tunnel are defined by the requirements of an IP link because both transit IP packets. A tunnel must transit the IP MTU, i.e., 68B for IPv4 and 1280B for IPv6, and a tunnel must support address resolution when there is more than one egress.

The requirements of the tunnel ingress and egress are defined by the network over which they exchange messages (tunnel link packets). For IP-over-IP, this means that the ingress MUST NOT exceed the (fragment) Identification field uniqueness requirements [RFC6864].

These requirements remain even though tunnels have some unique issues, including the need for additional space for encapsulation headers and the potential for tunnel path MTU variation.

4.1. Fragmentation

As with any link layer, the MTU of a tunnel is defined as the receiving interface reassembly MTU, and must satisfy the requirements of the IP packets the tunnel transits.

Note that many of the issues with tunnel fragmentation and MTU handling were discussed in [RFC4459], but that document described a variety of alternatives as if they were independent. This document explains the combined approach that is necessary.

An IPv4 tunnel must transit 68 byte packets without further fragmentation [RFC791][RFC1122] and an IPv6 tunnel must transit 1280 byte packets without further fragmentation [RFC2460]. The tunnel MTU interacts with routers or hosts it connects the same way as would a link MTU. In the following pseudocode, TTPsize is the size of the tunnel transit packet, and egressRMTU is the receive MTU of the egress. As with any link, the link MTU is defined not by the native path of the link (the path MTU inside the tunnel) but by the egress reassembly MTU (egressRMTU). This is because the ICMP "packet too big" message indicates failure, not preference. There is no ICMP message for "larger than I'd like, but I can still transit it".

These rules apply at the host/router where the tunnel is attached:

```
if (TTP > linkMTU) then
   if (TTP can be fragmented, e.g., IPv4 DF=0) then
      split TTP into fragments of TunMTU size
      and send each fragment into the tunnel ingress
   else
      drop TTP and send ICMP "too big" to TTP source
   endif
else
   send TTP into the tunnel "interface" (the ingress)
endif
```

These rules apply at the tunnel ingress:

```
if (sizeof(TTP) <= TunnelPathMTU) then
   encapsulate TTP as received and emit
else
   if (TunnelPathMTU < sizeof(TTP) <= egressRMTU) then
      fragment TTP into TunMTU chunks
      encapsulate and emit each TTP
   else
      {never happens; host/router already dropped by now}
   endif
endif
```

For IPv4 or IPv6 over IPv6, the tunnel path MTU is a minimum of 1280 minus the encapsulation header (40 bytes) with its options (TOptSz) and the egress reassembly MTU is 1500 minus the same amount:

```
if (sizeof(TTP) <= (1240 - TOptSz)) then
   encapsulate TTP as received and emit
else
   if ((1240 - TOptSz) < sizeof(TTP) <= (1460 - TOptSz)) then
      fragment TTP into (1240 - TOptSz) chunks
      encapsulate and emit each TTP
   else
      {never happens; host/router already dropped by now}
   endif
endif
```

This tunnel supports IPv6 transit only if TOptSize is smaller than 180 bytes, and supports IPv4 transit if TOptSize is smaller than 884 bytes. IPv6 tunnel transit packets of 1280 bytes may be guaranteed transit the outer network (M) without needing fragmentation there but they may require ongoing fragmentation and reassembly if the tunnel MTU is not at least 1320 bytes.

When using IP directly over IP, the minimum egress reassembly MTU for IPv4 is 576 bytes and for IPv6 is 1500 bytes. This means that tunnels of IPv4-over-IPv4, IPv4-over-IPv6, and IPv6-over-IPv6 are possible without additional requirements, but this may involve ingress fragmentation and egress reassembly. IPv6 cannot be tunneled directly over IPv4 without additional requirements, notably that the egress reassembly MTU or the link path MTU are at least 1280 bytes. Fragmentation and reassembly cannot be avoided for IPv6-over-IPv6 without similar requirements.

When ongoing ingress fragmentation and egress reassembly would be prohibitive or costly, larger MTUs can be supported by design and confirmed either out-of-band (by design) or in-band (e.g., using PLMTUD [RFC4821], as done in SEAL [RFC5320] and AERO [Te15]). Alternately, an ingress can encapsulate packets that fit and shut down once fragmentation is needed, but it must not continue to forward smaller packets while dropping larger packets that are still within required limits.

4.2. MTU discovery

MTU discovery enables a network path to support a larger path MTU and egress MTU than it can assume from the protocol over which it operates. There are two ways in which MTU discovery interact with tunnels: the MTU of the path over the tunnel and the MTU of the tunnel itself.

A tunnel has two different MTU values: the largest payload that can traverse from ingress to egress without further fragmentation (the tunnel path MTU) and the largest payload that can traverse from ingress to egress. The latter is defined by the egress reassembly MTU, not the tunnel path MTU, and is the tunnel MTU.

The path MTU over the tunnel is limited by the tunnel MTU (the egress reassembly MTU) but not the tunnel path MTU. There is temptation to optimize tunnel traversal so that packets are not fragmented between ingress and egress, i.e., to tune the network path MTU to the tunnel link MTU. This is hazardous for many reasons:

- o The tunnel is capable of transiting packets as large as the egress reassembly MTU, which is always at least as large as the tunnel path MTU and typically is larger.

- o ICMP has only one type of error message regarding large packets - "too big", i.e., too large to transit. There is no optimization message of "bigger than I'd like, but I can deal with if needed".

- o IP tunnels often involve some level of recursion, i.e., encapsulation over itself [RFC4459].

Recursive tunneling occurs whenever a protocol ends up encapsulated in itself. This happens directly, as when IPv4 is encapsulated in IPv4, or indirectly, as when IP is encapsulated in UDP which then is a payload inside IP. It can involve many layers of encapsulation because a tunnel provider isn't always aware of whether the packets it transits are already tunneled.

Recursion is impossible when the tunnel transit packets are limited to that of the native size of the tunnel path MTU. Arriving tunnel transit packets have a minimum supported size (1280 for IPv6) and the tunnel path MTU has the same size; there would be no room for the additional encapsulation headers. The result would be an IPv6 tunnel that cannot satisfy IPv6 transit requirements.

It is more appropriate to require the tunnel to satisfy IP transit requirements and enforce that requirement at design time or during operation (the latter using PLMTUD [RFC4821]). Conventional path MTU discovery (PMTUD) relies existing endpoint ICMP processing of explicit negative feedback from routers along the path via "message to big" ICMP packets in the reverse direction of the tunnel [RFC1191]. This technique is susceptible to the "black hole" phenomenon, in which the ICMP messages never return to the source due to policy-based filtering [RFC2923]. PLMTUD requires a separate, direct control channel from the egress to the ingress that provides positive feedback; the direct channel is not blocked by policy filters and the positive feedback ensures fail-safe operation if feedback messages are lost [RFC4821].

4.3. IP ID exhaustion

In IPv4, the IP Identification (ID) field is a 16-bit value that is unique for every packet for a given source address, destination address, and protocol, such that it does not repeat within the Maximum Segment Lifetime (MSL) [RFC791][RFC1122]. Although the ID field was originally intended for fragmentation and reassembly, it can also be used to detect and discard duplicate packets, e.g., at congested routers (see Sec. 3.2.1.5 of [RFC1122]). For this reason, and because IPv4 packets can be fragmented anywhere along a path, all packets between a source and destination of a given protocol must have unique ID values over a period of an MSL, which is typically interpreted as two minutes (120 seconds). These requirements have recently been somewhat relaxed in recognition of the primary use of this field for reassembly and the need to handle only fragment misordering at the receiver [RFC6864].

The uniqueness of the IP ID is a known problem for high speed devices, because it limits the speed of a single protocol between two endpoints [RFC4963]. Although this suggests that the uniqueness of the IP ID is moot, tunnels exacerbate this condition. A tunnel often aggregates traffic from a number of different source and destination addresses, of different protocols, and encapsulates them in a header with the same ingress and egress addresses, all using a single encapsulation protocol. The result is one of the following:

1. The IP ID rules are enforced, and the tunnel throughput is severely limited.

2. The IP ID rules are enforced, and the tunnel consumes large numbers of ingress/egress IP addresses solely to ensure ID uniqueness.

3. The IP ID rules are ignored.

The last case is the most obvious solution, because it corresponds to how endpoints currently behave. Fortunately, fragmentation is somewhat rare in the current Internet at large, but it can be common along a tunnel. Fragments that repeat the IP ID risk being reassembled incorrectly, especially when fragments are reordered or lost. Reassembly errors are not always detected by other protocol layers (see Sec. 4.8), and even when detected they can result in excessive overall packet loss and can waste bandwidth between the egress and ultimate packet destination.

4.4. Hop Count

This section considers the selection of the value of the hop count of the tunnel link header, as well as the potential impact on the tunnel transit header. The former is affected by the number of hops within the tunnel. The latter determines whether the tunnel has visible effect on the transit packet.

In general, the Internet hop count field is used to detect and avoid forwarding loops that cannot be corrected without a synchronized reboot. The IPv4 Time-to-Live (TTL) and IPv6 Hop Limit field each serve this purpose [RFC791][RFC2460].

The IPv4 TTL field was originally intended to indicate packet expiration time, measured in seconds. A router is required to decrement the TTL by at least one or the number of seconds the packet is delayed, whichever is larger [RFC1812]. Packets are rarely held that long, and so the field has come to represent the count of the number of routers traversed. IPv6 makes this meaning more explicit.

These hop count fields represent the number of network forwarding elements traversed by an IP datagram. An IP datagram with a hop count of zero can traverse a link between two hosts because it never visits a router (where it would need to be decremented and would have been dropped).

An IP datagram traversing a tunnel thus need not have its hopcount modified, i.e., the tunnel transit header need not be affected. A

zero hop count datagram should be able to traverse a tunnel as easily as it traverses a link. A router MAY be configured to decrement packets traversing a particular link (and thus a tunnel), which may be useful in emulating a path as if it had traversed one or more routers, but this is strictly optional. The ability of the outer network and tunnel network to avoid indefinitely looping packets does not rely on the hop counts of the tunnel traversal packet and tunnel link packet being related in any way at all.

The hop count field is also used by several protocols to determine whether endpoints are "local", i.e., connected to the same subnet (link-local discovery and related protocols [RFC4861]). A tunnel is a way to make a remote address appear directly-connected, so it makes sense that the other ends of the tunnel appear local and that such link-local protocols operate over tunnels unless configured explicitly otherwise. When the interfaces of a tunnel are numbered, these can be interpreted the same way as if they were on the same link subnet.

4.5. Signaling

In the current Internet architecture, signaling goes upstream, either from routers along a path or from the destination, back toward the source. Such signals are typically contained in ICMP messages, but can involve other protocols such as RSVP, transport protocol signals (e.g., TCP RSTs), or multicast control or transport protocols.

A tunnel behaves like a link and acts like a link interface at the nodes where it is attached. As such, it can provide information that enhances IP signaling (e.g., ICMP), but itself does not directly generate ICMP messages.

For tunnels, this means that there are two separate signaling paths. The outer network M devices can each signal the source of the tunnel transit packets, Hsrc (Figure 8). Inside the tunnel, the inner network N devices can signal the source of the tunnel link packets, the ingress I (Figure 9).

```
        +--------+-----------------------------------+--------+
        |        |                                   |        |
        v        --_                                 --       v
     +------+   /  \                                /  \   +------+
     | Hsrc |--+ Ra +----      --      --      ----+ Rd +--| Hdst |
     +------+   \  /    /\    /  \    /  \    /\    \  /   +------+
                 --    /I \--+ Rb +--+ Rc +--/E \    --
                       \  /   \  /    \  /   \  /
                        \/     --      --     \/
                       <------ Network N ------->
     <------------------------ Network M ------------------------->
```

Figure 8 Signals outside the tunnel

```
                         +-----+-------+------+
                 --_     |     |       |      |      --
     +------+   /  \     v     |       |      |     /  \   +------+
     | Hsrc |--+ Ra +----      --      --      ----+ Rd +--| Hdst |
     +------+   \  /    /\    /  \    /  \    /\    \  /   +------+
                 --    /I \--+ Rb +--+ Rc +--/E \    --
                       \  /   \  /    \  /   \  /
                        \/     --      --     \/
                       <------ Network N ------->
     <------------------------ Network M ------------------------->
```

Figure 9 Signals inside the tunnel

These two signal paths are inherently distinct except where information is exchanged between the network interface of the tunnel (the ingress) and its attached device (Ra, in both figures).

It is always possible for a network interface to provide hints to its attached device (host or router), which can be used for optimization. In this case, when signals inside the tunnel indicate a change to the tunnel, the ingress (i.e., the tunnel network interface) can provide information to the router (Ra, in both figures), so that Ra can generate the appropriate signal in return to Hsrc. This relaying may be difficult, because signals inside the tunnel may not return enough information to the ingress to support direct relaying to Hsrc.

In all cases, the tunnel ingress needs to determine how to relay the signals from inside the tunnel into signals back to the source. For some protocols this is either simple or impossible (such as for ICMP), for others, it can even be undefined (e.g., multicast). In some cases, the individual signals relayed from inside the tunnel may result in corresponding signals in the outside network, and in other cases they may just change state of the tunnel interface. In the

latter case, the result may cause the router Ra to generate new ICMP errors when later messages arrive from Hsrc or other sources in the outer network.

The meaning of the relayed information must be carefully translated. In the case of soft or hard ICMP errors, the translation may be obvious. ICMP "packet too big" messages from inside the tunnel do not necessarily have a direct impact on Ra unless they arrive from the egress (where they would update egressRMTU). Inside the tunnel, these messages could be used to adjust the ingress fragmentation.

In addition to ICMP, messages typically considered for translation include Explicit Congestion Notification (ECN [RFC6040]) and multicast (IGMP, e.g.).

4.6. Relationship of Header Fields

Some tunnel specifications attempt to relate the fields of the tunnel transit packet and tunnel link packet, i.e., the packet arriving at the ingress and the encapsulation header. These two headers are effectively independent and there is no utility in requiring their contents to be related.

In specific, the encapsulation header source and destination addresses are network endpoints in the tunnel network N, but have no meaning in the outer network M, even when the tunneled packet traverses the same network. The addresses are effectively independent, and the tunnel endpoint addresses are link addresses to the tunnel transit packet.

Because the tunneled packet uses source and destination addresses with a separate meaning, it is inappropriate to copy or reuse the IPv4 Identification or IPv6 Fragment ID fields of the tunnel transit packet. These fields need to be generated based on the context of the encapsulation header, not the tunnel transit header.

Similarly, the DF field need not be copied from the tunnel transit packet to the encapsulation header of the tunnel link packet (presuming both are IPv4). Path MTU discovery inside the tunnel does not directly correspond to path MTU discovery outside the tunnel.

The same is true for most other fields. When a field value is generated in the encapsulation header, its meaning should be derived from what is desired in the context of the tunnel as a link. When feedback is received from these fields, they should be presented to the tunnel ingress and egress as if they were network interfaces. The

behavior of the node where these interfaces attach should be identical to that of a conventional link.

There are exceptions to this rule that are explicitly intended to relay signals from inside the tunnel to outside the tunnel. The primary example is ECN [RFC6040], which copies the ECN bits from the tunnel transit header to the tunnel link header during encapsulation at the ingress and modifies the tunnel transit header at egress based on a combination of the bits of the two headers. This is intended to allow congestion notification within the tunnel to be interpreted as if it were on the direct path. Other examples may involve the DSCP flags. In both cases, it is assumed that the intent of copying values on encapsulation and merging values on decapsulation has the effect of allowing the tunnel to act as if it participates in the same type of network as outside the tunnel (network M).

4.7. Congestion

In general, tunnels carrying IP traffic need not react directly to congestion any more than would any other link layer [RFC5405]. IP traffic is not generally expected to be congestion reactive.

[text from David Black on ECN relaying?]

4.8. Checksums

IP traffic transiting a tunnel needs to expect a similar level of error detection and correction as it would expect from any other link. In the case of IPv4, there are no such expectations, which is partly why it includes a header checksum [RFC791].

IPv6 omitted the header checksum because it already expects most link errors to be detected and dropped by the link layer and because it also assumes transport protection [RFC2460]. When transiting IPv6 over IPv6, the tunnel fails to provide the expected error detection. This is why IPv6 is often tunneled over layers that include separate protection, such as GRE [RFC2784].

The fragmentation created by the tunnel ingress can increase the need for stronger error detection and correction, especially at the tunnel egress to avoid reassembly errors. The Internet checksum is known to be susceptible to reassembly errors that could be common [RFC4963], and should not be relied upon for this purpose. This is why SEAL and AERO include a separate checksum [RFC5320][Te15]. This requirement can be undermined when using UDP as a tunnel with no UDP checksum (as per [RFC6935][RFC6936]) when fragmentation occurs because the egress has no checksum with which to validate reassembly. For this reason,

it is safe to use UDP with a zero checksum for atomic (non-fragmented, non-fragmentable) tunnel link packets only; when used on fragments, whether generated at the ingress or en-route inside the tunnel, omission of such a checksum can result in reassembly errors that can cause additional work (capacity, forwarding processing, receiver processing) downstream of the egress.

4.9. Numbering

Tunnel ingresses and egresses have addresses associated with the encapsulation protocol. These addresses are the source and destination (respectively) of the encapsulated packet while traversing the tunnel network.

Tunnels may or may not have addresses in the network whose traffic they transit (e.g., network M in Figure 4). In some cases, the tunnel is an unnumbered interface to a point-to-point virtual link. When the tunnel has multiple egresses, tunnel interfaces require separate addresses in network M.

To see the effect of tunnel interface addresses, consider traffic sourced at router Ra in Figure 4. Even before being encapsulated by the ingress, that traffic needs a source IP network address that belongs to the router. One option is to use an address associated with one of the other interfaces of the router [RFC1122]. Another option is to assign a number to the tunnel interface itself. Regardless of which address is used, the resulting IP packet is then encapsulated by the tunnel ingress using the ingress address as a separate operation.

4.10. Multicast

[To be addressed]

Note that PMTU for multicast is difficult. PIM carries an option that may help in the Population Count Extensions to PIM [RFC6807].

IMO, again, this is no different than any other multicast link.

4.11. NAT / Load Balancing

[To be addressed]

4.12. Recursive tunnels.

The rules described in this document already support tunnels over tunnels, sometimes known as "recursive" tunnels, in which IP is

transited over IP either directly or via intermediate encapsulation (IP-UDP-IP).

There are known hazards to recursive tunneling, notably that the independence of the tunnel transit header and tunnel link header hop counts can result in a tunneling loop. Such looping can be avoided when using direct encapsulation (IP in IP) by use of a header option to track the encapsulation count and to limit that count [RFC2473]. This looping cannot be avoided when other protocols are used for tunneling, e.g., IP in UDP in IP, because the encapsulation count may not be visible where the recursion occurs.

5. Observations (implications)

[Leave this as a shopping list for now]

5.1. Tunnel protocol designers

Account for egress MTU/path MTU differences.

Include a stronger checksum.

Ensure the egress MTU is always larger than the path MTU.

Ensure that the egress reassembly can keep up with line rate OR design PLMTUD into the tunneling protocol.

5.2. Tunnel implementers

Detect when the egress MTU is exceeded.

Detect when the egress MTU drops below the required minimum and shut down the tunnel if that happens - configuring the tunnel down and issuing a hard error may be the only way to detect this anomaly, and it's sufficiently important that the tunnel SHOULD be disabled.

Do NOT decrement the TTL as part of being a tunnel. It's always already OK for a router to decrement the TTL based on different next-hop routers, but TTL is a property of a router not a link.

5.3. Tunnel operators

Keep the difference between "enforced by operators" vs. "enforced by active protocol mechanism" in mind. It's fine to assume something the tunnel cannot or does not test, as long as you KNOW you can assume it. When the assumption is wrong, it will NOT be signaled by the tunnel. Do NOT decrement the TTL as part of being a tunnel. It's

always already OK for a router to decrement the TTL based on different next-hop routers, but TTL is a property of a router not a link.

Do NOT decrement the TTL as part of being a tunnel. It's always already OK for a router to decrement the TTL based on different next-hop routers, but TTL is a property of a router not a link.

5.4. For existing standards

5.4.1. Generic UDP Encapsulation (GUE - IP in UDP in IP)

[He15a][He15b]

5.4.2. Generic Packet Tunneling in IPv6

[RFC2473]

Consistent with this doc:

Considers the endpoints of the tunnel as virtual interfaces.

Considers the tunnel a virtual link.

Requires source fragmentation at the ingress and reassembly at the egress.

Includes a recursion limit to prevent unlimited re-encapsulation.

Sets tunnel transit header hop limit independently.

Sends ICMPs back at the ingress based on the arriving tunnel transit packet and its relation to the tunnel MTU (though it uses the incorrect value of the tunnel MTU; see below).

Allows for ingress relaying of internal tunnel errors (but see below; it does not discuss retaining state about these).

Inconsistent with this doc:

Decrements the tunnel transit header by 1, i.e., incorrectly assuming that tunnel endpoints occur at routers only and that the tunnel, rather than the router, is responsible for this decrement.

This doc goes to pains to describe the decapsulation process as if it were distinct from conventional protocol processing by the receiver (when it should not be).

Copies traffic class from tunnel link to tunnel transit header (as one variant).

Treats the tunnel MTU as the tunnel path MTU, rather than the tunnel egress MTU.

Incorrectly fragments IPv4 DF=0 tunnel transit packets that arrive larger than the tunnel MTU at the IPv6 layer; the relationship between IPv4 and the tunnel is more complex (as noted in this doc).

Fails to retain state from the tunnel based on ingress receiving ICMP messages from inside the tunnel, e.g., such as might cause future tunnel transit packets arriving at the ingress to be discarded with an ICMP error response rather than allowing them to proceed into the tunnel.

5.4.3. Geneve (NVO3)

[RFC7364][Gr15]

Consistent with this doc:

Generation of the link header fields is not discussed and presumed independent of transit packet.

Inconsistent with this doc:

Tries to match transit to tunnel path MTU rather than egress MTU.

5.4.4. GRE (IP in GRE in IP)

IPv4 [RFC2784][RFC7588][Pi15]:

Consistent with this doc:

Does not address link header generation.

Non-default behavior allows fragmentation of link packet to match tunnel path MTU up to the limit of the egress MTU.

Default behavior sets link DF independently.

Shuts the tunnel down if the tunnel path MTU isn't => 1280.

Inconsistent with this doc:

Based on tunnel path MTU, not egress MTU.

Claims that the tunnel (GRE) mechanism is responsible for generating ICMP error messages.

Default behavior fragments transit packet (where possible) based on tunnel path MTU (it should fragment based on egress MTU).

Default behavior does not support the minimum MTU of IPv6 when run over IPv6.

Non-default behavior allows copying DF for IPv4 in IPv4.

5.4.5. IP in IP / mobile IP

IPv4 [RFC2003][RFC5944]:

Consistent with this doc:

Generate link ID independently

Generate link DF independently when transit DF=0

Generate ECN/update ECN based on sharing info [RFC6040]

Set link TTL to transit to egress only (independently)

Do not decrement TTL on entry except when part of forwarding

Do not decrement TTL on exit except when part of forwarding

Options not copied, but used as a hint to desired services.

Generally treat tunnel as a link, e.g., for link-local.

Inconsistent with this doc

Set link DF when transit DF=1 (won't work unless I-E runs PLMTUD)

Drop at egress if transit TTL=0 (wrong TTL for host-host tunnels)

Drop when transit source is router's IP (prevents tun from router)

Drop when transit source matches egress (prevents tun to router)

Use tunnel ICMPs to generate upper ICMPs, copying context (ICMPs are now coming from inside a link!); these should be handled by setting errors as a "network interface" and letting the attached host/router figure out what to send.

Using tunnel MTU discovery to tune the transit packet to the tunnel path MTU rather than egress MTU.

IPv6 [RFC2473]:

Consistent with this doc:

Doesn't discuss lots of header fields, but implies they're set independently.

Sets link TTL independently.

Inconsistent with this doc:

Tunnel issues ICMP PTBs.

ICMP PTB issued if larger then 1280 - header, rather than egress reassembly MTU.

Fragments IPv6 over IPv6 fragments only if transit is <= 1280 (i.e., forces all tunnels to have a max MTU of 1280).

Fragments IPv4 over IPv6 fragments only if IPv4 DF=0 (misinterpreting the "can fragment the IPv4 packet" as permission to fragment at the IPv6 link header)

Considers encapsulation a forwarding operation and decrements the transit TTL.

5.4.6. IPsec tunnel mode (IP in IPsec in IP)

[RFC4301]

Consistent with this doc:

Most of the rules, except as noted below.

Inconsistent with this doc:

Writes its own header copying rules (Sec 5.1.2), rather than referring to existing standards.

Uses policy to set, clear, or copy DF (policy isn't the issue)

Intertwines tunneling with forwarding rather than presenting the tunnel as a network interface; this can be corrected by using IPsec transport mode with an IP-in-IP tunnel [RFC3884].

5.4.7. L2TP

[RFC3931]

Consistent with this doc:

Does not address most link headers, which are thus independent.

Inconsistent with this doc:

Manages tunnel access based on tunnel path MTU, instead of egress MTU.

Refers to RFC2473 (IPv6 in IPv6), which is inconsistent with this doc as noted above.

5.4.8. L2VPN

[RFC4664]

5.4.9. L3VPN

[RFC4176]

5.4.10. LISP

[RFC6830]

5.4.11. MPLS

[RFC3031]

5.4.12. PWE

[RFC3985]

5.4.13. SEAL/AERO

[RFC5320][Te15]

5.4.14. TRILL

[RFC5556][RFC6325]

Consistent with this doc:

Puts IP in Ethernet, so most of the issues don't come up.

Ethernet doesn't have TTL or fragment.

Rbridge (trill) TTL header is independent of transit packet.

5.5. For future standards

Larger IPv4 MTU (2K? or just 2x path MTU?) for reassembly

Always include frag support for at least two frags; do NOT try to deprecate fragmentation.

Limit encapsulation option use/space.

Augment ICMP to have two separate messages: PTB vs P-bigger-than-optimal

Include MTU as part of BGP as a hint - SB

Hazards of multi-MTU draft-van-beijnum-multi-mtu-04

6. Security Considerations

Tunnels may introduce vulnerabilities or add to the potential for receiver overload and thus DOS attacks. These issues are primarily related to the fact that a tunnel is a link that traverses a network path and to fragmentation and reassembly. ICMP signal translation introduces a new security issue and must be done with care. ICMP generation at the router or host attached to a tunnel is already covered by existing requirements (e.g., should be throttled).

Tunnels traverse multiple hops of a network path from ingress to egress. Traffic along such tunnels may be susceptible to on-path and off-path attacks, including fragment injection, reassembly buffer overload, and ICMP attacks. Some of these attacks may not be as visible to the endpoints of the architecture into which tunnels are deployed and these attacks may thus be more difficult to detect.

Fragmentation at routers or hosts attached to tunnels may place an undue burden on receivers where traffic is not sufficiently diffuse, because tunnels may induce source fragmentation at hosts and path fragmentation (for IPv4 DF=0) more for tunnels than for other links. Care should be taken to avoid this situation, notably by ensuring that tunnel MTUs are not significantly different from other link MTUs.

Tunnel ingresses emitting IP datagrams MUST obey all existing IP requirements, such as the uniqueness of the IP ID field. Failure to

either limit encapsulation traffic, or use additional ingress/egress IP addresses, can result in high speed traffic fragments being incorrectly reassembled.

[management?]

[Access control?]

describe relationship to [RFC6169] - JT (as per INTAREA meeting notes, don't cover Teredo-specific issues in RFC6169, but include generic issues here)

7. IANA Considerations

This document has no IANA considerations.

The RFC Editor should remove this section prior to publication.

8. References

8.1. Normative References

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

8.2. Informative References

[Cl88] Clark, D., "The design philosophy of the DARPA internet protocols," Proc. Sigcomm 1988, p.106-114, 1988.

[Er94] Eriksson, H., "MBone: The Multicast Backbone," Communications of the ACM, Aug. 1994, pp.54-60.

[Gr15] Gross, J., et al., "Geneve: Generic Network Virtualization Encapsulation," draft-ietf-nvo3-geneve-00, May 2015.

[He15a] Herbert, T., L. Yong, O. Zia, "Generic UDP Encapsulation," draft-ietf-nvo3-gue-01, June 2015.

[He15b] Herbert, T., F. Templin, "Fragmentation option for Generic UDP Encapsulation," draft-herbert-gue-fragmentation-00, Mar. 2015.

[Pi15] Pignataro, C., R. Bonica, S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)," draft-ietf-intarea-gre-ipv6-11, July 2015.

[RFC768] Postel, J, "User Datagram Protocol," RFC 768, Aug. 1980

[RFC791] Postel, J., "Internet Protocol," RFC 791 / STD 5, September 1981.

[RFC793] Postel, J, "Transmission Control Protocol," RFC 793, Sept. 1981.

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

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

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers," RFC 1812, June 1995.

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

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

[RFC2473] Conta, A., "Generic Packet Tunneling in IPv6 Specification," RFC 2473, Dec. 1998.

[RFC2784] Farinacci, D., T. Li, S. Hanks, D. Meyer, P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

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

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

[RFC3031] Rosen, E., A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised" RFC 5944, Nov. 2010.

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

[RFC3884] Touch, J., L. Eggert, Y. Wang, "Use of IPsec Transport Mode for Dynamic Routing," RFC 3884, September 2004.

[RFC3931] Lau, J., Ed., M. Townsley, Ed., I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)," RFC 3931, March 2005.

[RFC3985] Bryant, S., P. Pate (Eds.), "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4176] El Mghazli, Y., Ed., T. Nadeau, M. Boucadair, K. Chan, A. Gonguet, "Framework for Layer 3 Virtual Private Networks (L3VPN) Operations and Management," RFC 4176, October 2005.

[RFC4301] Kent, S., and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, December 2005.

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

[RFC4664] Andersson, L., Ed., E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)," RFC 4664, September 2006.

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

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

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

[RFC5320] Templin, F., Ed., "The Subnetwork Encapsulation and Adaptation Layer (SEAL)," RFC 5320, Feb. 2010.

[RFC5405] Eggert, L., G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers," RFC 5405, Nov. 2008.

[RFC5556] Touch, J., R. Perlman, "Transparently Interconnecting Lots of Links (TRILL): Problem and Applicability Statement," RFC 5556, May 2009.

[RFC6040] Briscoe, B., "Tunneling of Explicit Congestion Notification," RFC 6040, Nov. 2010.

[RFC6169] Krishnan, S., D. Thaler, J. Hoagland, "Security Concerns With IP Tunneling," RFC 6169, Apr. 2011.

[RFC6325] Perlman, R., D. Eastlake, D. Dutt, S. Gai, A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification," RFC 6325, July 2011.

[RFC6807] Farinacci, D., G. Shepherd, S. Venaas, Y. Cai, "Population Count Extensions to Protocol Independent Multicast (PIM)," RFC 6807, Dec. 2012.

[RFC6830] Farinacci, D., V. Fuller, D. Meyer, D. Lewis, "The Locator/ID Separation Protocol," RFC 6830, Jan. 2013.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field," Proposed Standard, RFC 6864, Feb. 2013.

[RFC6935] Eubanks, M., P. Chimento, M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets," RFC 6935, Apr. 2013.

[RFC6936] Fairhurst, G., M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums," RFC 6936, Apr. 2013.

[RFC7364] Narten, T., Gray, E., Black, D., Fang, L., Kreeger, L., M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, October 2014.

[RFC7588] Bonica, R., C. Pignataro, J. Touch, "A Widely-Deployed Solution to the Generic Routing Encapsulation Fragmentation Problem," RFC 7588, July 2015.

[Sa84] Saltzer, J., D. Reed, D. Clark, "End-to-end arguments in system design," ACM Trans. on Computing Systems, Nov. 1984.

[Te15] Templin, F., "Asymmetric Extended Route Optimization," draft-templin-aerolink-58, June 2015.

[To01] Touch, J., "Dynamic Internet Overlay Deployment and Management Using the X-Bone," Computer Networks, July 2001, pp. 117-135.

[To03] Touch, J., Y. Wang, L. Eggert, G. Finn, "Virtual Internet Architecture," USC/ISI Tech. Report 570, Aug. 2003.

[To98] Touch, J., S. Hotz, "The X-Bone," Proc. Globecom Third Global Internet Mini-Conference, Nov. 1998.

[Zi80] Zimmermann, H., "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," IEEE Trans. on Comm., Apr. 1980.

9. Acknowledgments

This document originated as the result of numerous discussions among the authors, Jari Arkko, Stuart Bryant, Lars Eggert, Ted Faber, Gorry Fairhurst, Dino Farinacci, Matt Mathis, and Fred Templin, as well as members participating in the Internet Area Working Group.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695
U.S.A.

Phone: +1 (310) 448-9151
Email: touch@isi.edu

W. Mark Townsley
Cisco
L'Atlantis, 11, Rue Camille Desmoulins
Issy Les Moulineaux, ILE DE FRANCE 92782

Email: townsley@cisco.com

Appendix A. Fragmentation

There are two places where fragmentation can occur in a tunnel, called Outer Fragmentation and Inner Fragmentation.

A.1. Outer Fragmentation

The simplest case is Outer Fragmentation, as shown in Figure 10. The bottom of the figure shows the network topology, where packets start at the source, enter the tunnel at the encapsulator, exit the tunnel at the decapsulator, and arrive finally at the destination. The packet traffic is shown above the topology, where the end-to-end packets are shown at the top. The packets are composed of an inner header (iH) and inner data (iD); the term "inner") is relative to the tunnel, as will become apparent. When the packet (iH,iD) arrives at the encapsulator, it is placed inside the tunnel packet structure, here shown as adding just an outer header, oH, in step (a).

When the encapsulated packet exceeds the MTU of the tunnel, the packet needs to be fragmented. In this case we fragment the packet at the outer header, with the fragments shown as (b1) and (b2). Note that the outer header indicates fragmentation (as ' and "),the inner header occurs only in the first fragment, and the inner data is broken across the two packets. These fragments are reassembled at the encapsulator in step (c), and the resulting packet is decapsulated and sent on to the destination.

```
    +----+----+                                              +----+----+
    | iH | iD |------+ -  -  -  -  -  -  -  -  -  -  +------>| iH | iD |
    +----+----+      |                               |       +----+----+
                     v                               |
              +----+----+----+               +----+----+----+
          (a) | oH | iH | iD |               | oH | iH | iD | (c)
              +----+----+----+               +----+----+----+
                     |                               ^
                     |       +----+----+-----+       |
                (b1) +----- >| oH'| iH | iD1 |-------+
                     |       +----+----+-----+       |
                     |                               |
                     |       +----+-----+            |
                (b2) +----- >| oH"| iD2 |------------+
                             +----+-----+
   +-----+         +---+                             +---+         +-----+
   |     |        /     \ ======================= /     \        |     |
   | Src |=======|  Enc  |=======================|  Dec  |=======| Dst |
   |     |        \     / ======================= \     /        |     |
   +-----+         +---+                             +---+         +-----+
```

Figure 10 Fragmentation of the outer packet

Outer fragmentation isolates Source and Destination from tunnel encapsulation duties. This can be considered a benefit in clean, layered network design, but also may result in complex decapsulator design, especially where tunnels aggregate large amounts of traffic, such as IP ID overload (see Sec. 4.3). Outer fragmentation is valid for any tunnel encapsulation protocol that supports fragmentation (e.g., IPv4 or IPv6), where the tunnel endpoints act as the host endpoints of that protocol.

Along the tunnel, the inner header is contained only in the first fragment, which can interfere with mechanisms that 'peek' into lower layer headers, e.g., as for ICMP, as discussed in Sec. 4.5.

A.2. Inner Fragmentation

Inner Fragmentation distributes the impact of tunneling across both the decapsulator and destination, and is shown in Figure 11. Again, the network topology is shown at the bottom of the figure, and the original packets show at the top. Packets arrive at the encapsulator, and are fragmented there based on the inner header into (a1) and (a2). The fragments arrive at the decapsulator, which removes the outer header and forwards the resulting fragments on to the destination. The destination is then responsible for reassembling the fragments into the original packet.

```
   +----+----+                                               +----+----+
   | iH | iD |-------+-  -  -  -  -  -  -  -  -  -  -  -  - >| iH | iD |
   +----+----+       |                                       +----+----+
                     v                                            ^
                +----+-----+                    +----+-----+      |
           (a1) | iH'| iD1 |                    | iH'| iD1 |------+
                +----+-----+                    +----+-----+      |
                                                                  |
                +----+---                       +----+-----+      |
           (a2) | iH"| iD2 |                    | iH"| iD2 |------+
                +----+-----+                    +----+-----+
                     |                               ^
                     |       +----+----+-----        |
                (b1) +----- >| oH | iH'| iD1 |-------+
                     |       +----+----+-----+       |
                     |                               |
                     |       +----+----+-----+       |
                (b2) +----- >| oH | iH"| iD2 |-------+
                             +----+----+-----+
   +-----+         +---+                           +---+         +-----+
   |     |        /     \ ======================= /     \        |     |
   | Src |=======|  Enc  |=======================|  Dec  |=======| Dst |
   |     |        \     / ======================= \     /        |     |
   +-----+         +---+                           +---+         +-----+
```

Figure 11 Fragmentation of the inner packet

As noted, inner fragmentation distributes the effort of tunneling across the decapsulator and destinations; this can be especially important when the tunnel aggregates large amounts of traffic. Note that this mechanism is thus valid only when the original source packets can be fragmented on-path, e.g., as in IPv4.

Along the tunnel, the inner headers are copied into each fragment, and so are available to mechanisms that 'peek' into headers (e.g., ICMP, as discussed in Sec. 4.5). Because fragmentation happens on the inner header, the impact of IP ID is reduced.

APPENDIX B: Fragmentation efficiency

B.1. Selecting fragment sizes

There are different ways to fragment a packet. Consider a network with an MTU as shown in Figure 12, where packets are encapsulated over the same network layer as they arrive on (e.g., IP in IP). If a packet as large as the MTU arrives, it must be fragmented to accommodate the additional header.

```
                 X===========================X (MTU)
                 +----+----------------------+
                 | iH | DDDDDDDDDDDDDDDDDDDD |
                 +----+----------------------+
                   |
                   |  X===========================X (MTU)
                   |  +---+----+------------------+
               (a) +->| H'| iH | DDDDDDDDDDDDDDDD |
                   |  +---+----+------------------+
                   |      |
                   |      |  X===========================X (MTU)
                   |      |  +----+---+----+-------------+
                   | (a1) +->| nH'| H | iH | DDDDDDDDDDD |
                   |      |  +----+---+----+-------------+
                   |      |
                   |      |  +----+-------+
                   | (a2) +->| nH"| DDDDD |
                   |         +----+-------+
                   |
                   |  +---+------+
               (b) +->| H"| DDDD |
                      +---+------+
                          |
                          |  +----+---+------+
                     (b1) +->| nH'| H"| DDDD |
                             +----+---+------+
```

Figure 12Fragmenting via maximum fit

Figure 12 shows this process, using Outer Fragmentation as an example (the situation is the same for Inner Fragmentation, but the headers that are affected differ). The arriving packet is first split into (a) and (b), where (a) is of the MTU of the network. However, this tunnel then traverses over another tunnel, whose impact the first tunnel ingress has not accommodated. The packet (a) arrives at the second tunnel ingress, and needs to be encapsulated again, but because it is already at the MTU, it needs to be fragmented as well,

into (a1) and (a2). In this case, packet (b) arrives at the second tunnel ingress and is encapsulated into (b1) without fragmentation, because it is already below the MTU size.

In Figure 13, the fragmentation is done evenly, i.e., by splitting the original packet into two roughly equal-sized components, (c) and (d). Note that (d) contains more packet data, because (c) includes the original packet header because this is an example of Outer Fragmentation. The packets (c) and (d) arrive at the second tunnel encapsulator, and are encapsulated again; this time, neither packet exceeds the MTU, and neither requires further fragmentation.

```
                 X===========================X (MTU)
                 +----+----------------------+
                 | iH | DDDDDDDDDDDDDDDDDDDD |
                 +----+----------------------+
                   |
                   |  X===========================X (MTU)
                   |  +---+----+----------+
               (c) +->| H'| iH | DDDDDDDD |
                   |  +---+----+----------+
                   |      |
                   |      |  X===========================X (MTU)
                   |      |  +----+---+----+----------+
                   | (c1) +->| nH | H'| iH | DDDDDDDD |
                   |         +----+---+----+----------+
                   |
                   |  +---+--------------+
               (d) +->| H"| DDDDDDDDDDDD |
                      +---+--------------+
                          |
                          |  +----+---+--------------+
                     (d1) +->| nH | H"| DDDDDDDDDDDD |
                             +----+---+--------------+
```

Figure 13 Fragmenting evenly

B.2. Packing

Encapsulating individual packets to traverse a tunnel can be inefficient, especially where headers are large relative to the packets being carried. In that case, it can be more efficient to encapsulate many small packets in a single, larger tunnel payload. This technique, similar to the effect of packet bursting in Gigabit Ethernet (regardless of whether they're encoded using L2 symbols as delineators), reduces the overhead of the encapsulation headers

(Figure 14). It reduces the work of header addition and removal at the tunnel endpoints, but increases other work involving the packing and unpacking of the component packets carried.

```
          +-----+-----+
          | iHa | iDa |
          +-----+-----+
                |
                |     +-----+-----+
                |     | iHb | iDb |
                |     +-----+-----+
                |           |
                |           |     +-----+-----+
                |           |     | iHc | iDc |
                |           |     +-----+-----+
                |           |           |
                v           v           v
     +----+-----+-----+-----+-----+-----+-----+
     | oH | iHa | iHa | iHb | iDb | iHc | iDc |
     +----+-----+-----+-----+-----+-----+-----+
```

Figure 14 Packing packets into a tunnel

[NOTE: PPP chopping and coalescing?]