Internet Area WG   J. Touch
Internet Draft
Intended status: Best Current Practice   M. Townsley
Updates: 4459   Cisco
Expires: July 2018   January 19, 2018

IP Tunnels in the Internet Architecture
draft-ietf-intarea-tunnels-08.txt

Status of this Memo

Abstract

This document discusses the role of IP tunnels in the Internet architecture. An IP tunnel transits IP datagrams as payloads in non-link layer protocols. This document explains the relationship of IP tunnels to existing protocol layers and the challenges in supporting IP tunneling, based on the equivalence of tunnels to links. The implications of this document are used to derive recommendations that update MTU and fragment issues in RFC 4459.

Table of Contents

1. Introduction

The Internet layering architecture is loosely based on the ISO seven layer stack, in which data units traverse the stack by being wrapped inside data units of the next layer down [Cl88][Zi80]. 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).

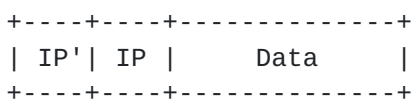


Figure 1 IP inside IP

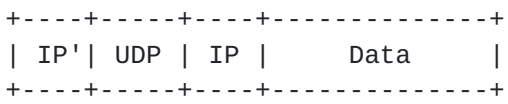


Figure 2 IP in UDP in IP in Ethernet

This document focuses on tunnels that transit IP packets, i.e., in which an IP packet is the payload of another protocol, other than a typical link layer. A tunnel is a virtual link that can help decouple the network topology seen by transiting packets from the underlying physical network [To98][RFC2473]. 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 efficiently between multicast-capable routers over paths that did not support native link-layer multicast. Similar techniques have been used to support incremental deployment of other protocols over legacy substrates, such as IPv6 [RFC2546].

Use of tunnels is common in the Internet. The word "tunnel" occurs in nearly 1,500 RFCs (of nearly 8,000 current RFCs, close to 20%), and is supported within numerous protocols, including:

- IP in IP / mobile IP - IPv4 in IPv4 tunnels using protocol 4 [RFC2003][RFC2473][RFC5944] and its precursor called "IPIP" using protocol 94 [RFC1853]

- IP in IPv6 - IPv6 or IPv4 in IPv6 [RFC2473]

- IPsec - includes a tunnel mode to enable encryption or authentication of the an entire IP datagram inside another IP datagram [RFC4301]

- Generic Router Encapsulation (GRE) - a shim layer for tunneling any network layer in any other network layer, as in IP in GRE in IP [RFC2784][RFC7588][RFC7676], or inside UDP in IP [RFC8086]

- MPLS - a shim layer for tunneling IP over a circuit-like path over a link layer [RFC3031] or inside UDP in IP [RFC7510], in which identifiers are rewritten on each hop, often used for traffic provisioning

- LISP - a mechanism that uses multipoint IP tunnels to reduce routing table load within an enclave of routers at the expense of more complex tunnel ingress encapsulation tables [RFC6830]

- TRILL - a mechanism that uses multipoint L2 tunnels to enable use of L3 routing (typically IS-IS) in an enclave of Ethernet bridges [RFC5556][RFC6325]

- Generic UDP Encapsulation (GUE) - IP in UDP in IP [He17]

- Automatic Multicast Tunneling (AMT) - IP in UDP in IP for multicast [RFC7450]

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

- L2VPNs - provides a link topology different from that provided by physical links [RFC4664]; many of these are not classical tunnels, using only tags (Ethernet VLAN tags) rather than encapsulation

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

- NVO3 - data center network sharing (to be determined, which may include use of GUE or other tunnels) [RFC7364]

- PWE3 - emulates wire-like services over packet-switched services [RFC3985]

- SEAL/AERO -IP in IP tunneling with an additional shim header designed to overcome the limitations of RFC2003 [RFC5320][Te18]

- A number of legacy variants, including swIPe (an IPsec precursor), a GRE precursor, and the Internet Encapsulation Protocol, all of which included a shim layer [RFC1853]

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 size (i.e., Maximum Transmission Unit or MTU) mismatches and error signals (e.g., ICMP) are handled may benefit from a coordinated approach.

Regardless of the layer in which encapsulation occurs, tunnels emulate a link. The only difference is that a link operates over a physical communication channel, whereas a tunnel operates over other software protocol layers. Because tunnels are links, they are subject to the same issues as any link, e.g., MTU discovery, signaling, and the potential utility of native support for broadcast and multicast [RFC3819]. Tunnels have some advantages over native links, being potentially easier to reconfigure and control because they can

generally rely on existing out-of-band communication between its endpoints.

The first attempt to use large-scale tunnels was to transit multicast traffic across the Internet in 1988, and this resulted in 'tunnel collapse'. At the time, tunnels were not implemented as encapsulation-based virtual links, but rather as loose source routes on un-encapsulated IP datagrams [RFC1075]. Then, as now, routers did not support use of the loose source route IP option at line rate, and the multicast traffic caused overload of the so-called "slow path" processing of IP datagrams in software. Using encapsulation tunnels avoided that collapse by allowing the forwarding of encapsulated packets to use the "fast path" hardware processing [Er94].

The remainder of this document describes the general principles of IP tunneling and discusses the key considerations in the design of any protocol that tunnels IP datagrams. It derives its conclusions from the equivalence of tunnels and links and from requirements of existing standards for supporting IPv4 and IPv6 as payloads.

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].

In this document, these key words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

2.2. Terminology

This document uses the following terminology. Optional words in the term are indicated in parentheses, e.g., "(link or network) interface" or "egress (interface)".

Terms from existing RFCs:

- Messages: variable length data labeled with globally-unique endpoint IDs, also known as a datagram for IP messages [RFC791].

- Node: a physical or logical network device that participates as either a host [RFC1122][RFC6434] or router [RFC1812]. This term originally referred to gateways since some very early RFCs [RFC5], but is currently the common way to describe a point in a network at which messages are processed.

- Host or endpoint: a node that sources or sinks messages labeled from/to its IDs, typically known as a host for both IP and higher-layer protocol messages [RFC1122].

- Source or sender: the node that generates a message [RFC1122].

- Destination or receiver: the node that consumes a message [RFC1122].

- Router or gateway: a node that relays IP messages using destination IDs and local context [RFC1812]. Routers also act as hosts when they source or sink messages. Also known as a forwarder for IP messages. Note that the notion of router is relative to the layer at which message processing is considered [To16].

- Link: a communications medium (or emulation thereof) that transfers IP messages between nodes without traversing a router (as would require decrementing the hop count) [RFC1122][RFC1812].

- Link packet: a link layer message, which can carry an IP datagram as a payload

- (Link or network) Interface: a location on a link co-located with a node where messages depart onto that link or arrive from that link. On physical links, this interface formats the message for transmission and interprets the received signals.

- Path: a sequence of one or more links over which an IP message traverses between source and destination nodes (hosts or routers).

- (Link) MTU: the largest message that can transit a link [RFC791], also often referred to simply as "MTU". It does not include the size of link-layer information, e.g., link layer headers or trailers, i.e., it refers to the message that the link can carry as a payload rather than the message as it appears on the link. This is thus the largest network layer packet (including network layer headers, e.g., IP datagram) that can transit a link. Note that this need not be the native size of messages on the link, i.e., the link may internally fragment and reassemble messages. For IPv4, the smallest MTU must be at least 68 bytes [RFC791], and for IPv6 the smallest MTU must be at least 1280 bytes [RFC8200].

- EMTU_S (effective MTU for sending): the largest message that can transit a link, possibly also accounting for fragmentation that happens before the fragments are emitted onto the link [RFC1122]. When source fragmentation is possible, EMTU_S = EMTU_R. When source fragmentation is not possible, EMTU_S = (link) MTU. For IPv4, this is MUST be at least 68 bytes [RFC791] and for IPv6 this MUST be at least 1280 bytes [RFC8200].

- EMTU_R (effective MTU to receive): the largest payload message that a receiver must be able to accept. This thus also represents the largest message that can traverse a link, taking into account reassembly at the receiver that happens after the fragments are received [RFC1122]. For IPv4, this is MUST be at least 576 bytes [RFC791] and for IPv6 this MUST be at least 1500 bytes [RFC8200].

- Path MTU (PMTU): the largest message that can transit a path of links [RFC1191][RFC8201]. Typically, this is the minimum of the link MTUs of the links of the path, and represents the largest network layer message (including network layer headers) that can transit a path without requiring fragmentation while in transit. Note that this is not the largest network packet that can be sent between a source and destination, because that network packet might have been fragmented at the network layer of the source and reassembled at the network layer of the destination.

- Tunnel: a protocol mechanism that transits messages between an ingress interface and egress interface using encapsulation to allow an existing network path to appear as a single link [RFC1853]. Note that a protocol can be used to tunnel itself (IP over IP). There is essentially no difference between a tunnel and the conventional layering of the ISO stack (i.e., by this definition, Ethernet is can be considered tunnel for IP). A tunnel is also known as a virtual link.

- Ingress (interface): the virtual link interface of a tunnel that receives messages within a node, encapsulates them according to the tunnel protocol, and transmits them into the tunnel [RFC2983]. An ingress is the tunnel equivalent of the outgoing (departing) network interface of a link, and its encapsulation processing is the tunnel equivalent of encoding a message for transmission over a physical link. The ingress virtual link interface can be co-located with the traffic source.

  The term 'ingress' in other RFCs also refers to 'network ingress', which is the entry point of traffic to a transit network. Because this document focuses on tunnels, the term "ingress" used in the remainder of this document implies "tunnel ingress".

- Egress (interface): a virtual link interface of a tunnel that receives messages that have finished transiting a tunnel and presents them to a node [RFC2983]. For reasons similar to ingress, the term 'egress' will refer to 'tunnel egress' throughout the remainder of this document. An egress is the tunnel equivalent of the incoming (arriving) network interface of a link and its decapsulation processing is the tunnel equivalent of interpreting a signal received from a physical link. The egress decapsulates messages for further transit to the destination. The egress virtual link interface can be co-located with the traffic destination.

- Ingress node: network device on which an ingress is attached as a virtual link interface [RFC2983]. Note that a node can act as both an ingress node and an egress node at the same time, but typically only for different tunnels.

- Egress node: device where an egress is attached as a virtual link interface [RFC2983]. Note that a device can act as both a ingress node and an egress node at the same time, but typically only for different tunnels.

- Inner header: the header of the message as it arrives to the ingress [RFC2003].

- Outer header(s): one or more headers added to the message by the ingress, as part of the encapsulation for tunnel transit [RFC2003].

- Mid-tunnel fragmentation: Fragmentation of the message during the tunnel transit, as could occur for IPv4 datagrams with DF=0 [RFC2983].

- Atomic packet, datagram, or fragment: an IP packet that has not been fragmented and which cannot be fragmented further [RFC6864] [RFC6946].

The following terms are introduced by this document:

- (Tunnel) transit packet: the packet arriving at a node connected to a tunnel that enters the ingress interface and exits the egress interface, i.e., the packet carried over the tunnel. This is sometimes known as the 'tunneled packet', i.e., the packet carried over the tunnel. This is the tunnel equivalent of a network layer packet as it would traverse a link. This document focuses on IPv4 and IPv6 transit packets.

- (Tunnel) link packet (TLP): packets that traverse between two interfaces, e.g., from ingress interface to egress interface, in which resides all or part of a transit packet. A tunnel link packet is the tunnel equivalent of a link (layer) packet as it would traverse a link, which is why we use the same terminology.

- Tunnel MTU: the largest transit packet that can traverse a tunnel, i.e., the tunnel equivalent of a link MTU, which is why we use the same terminology. This is the largest transit packet which can be reassembled at the egress interface.

- Tunnel maximum atomic packet (MAP): the largest transit packet that can traverse a tunnel as an atomic packet, i.e., without requiring tunnel link packet fragmentation either at the ingress or on-path between the ingress and egress.

- Inner fragmentation: fragmentation of the transit packet that arrives at the ingress interface before any additional headers are added. This can only correctly occur for IPv4 DF=0 datagrams.

- Outer fragmentation: source fragmentation of the tunnel link packet after encapsulation; this can involve fragmenting the outermost header or any of the other (if any) protocol layers involved in encapsulation.

- Maximum frame size (MFS): the link-layer equivalent of the MTU, using the OSI term 'frame'. For Ethernet, the MTU (network packet size) is 1500 bytes but the MFS (link frame size) is 1518 bytes originally, and 1522 bytes assuming VLAN (802.1Q) tagging support.

- EMFS_S: the link layer equivalent of EMTU_S.

- EMFS_R: the link layer equivalent of EMTU_R.

- Path MFS: the link layer equivalent of PMTU.

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]) whose messages are handled by hosts, routers, and links [Cl88][To03], as shown in Figure 3:

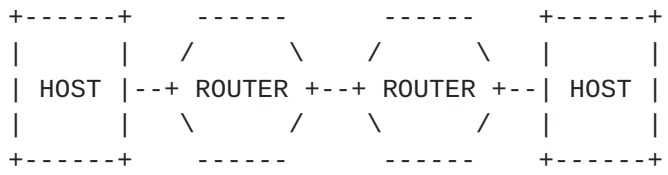


Figure 3 Basic Internet architecture

As a network architecture, the Internet is a system of hosts (endpoints) and routers (relays) 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 using compact tables.

Although the definitions of host, router, and link seem absolute, they are often relative as viewed within the context of one protocol layer, each of which can be considered a distinct network architecture. An Internet gateway is an OSI Layer 3 router when it transits IP datagrams but it acts as an OSI Layer 2 host as it sources or sinks Layer 2 messages on attached links to accomplish this transit capability. In this way, one 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 determined by context. 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 typically modeled as a single component from all viewpoints in that layer (with some notable exceptions, e.g., Network Address Translators, which appear as hosts and routers, depending on the direction of the viewpoint [To16]).

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. Keep in mind that it is possible that both network N and network M can both be components of the Internet, i.e., there may be regular traffic as well as tunneled traffic over any of the routers shown.

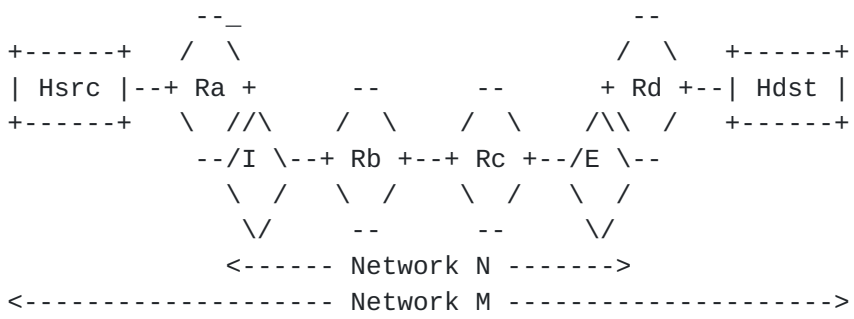


Figure 4 The big picture

The tunnel consists of two interfaces - an ingress (I) and an egress (E) that lie along a path connected by network N. Regardless of how the ingress and egress interfaces are connected, the tunnel serves as a link between the nodes it connects (here, Ra and Rd).

IP packets arriving at the ingress interface are encapsulated to traverse network N. We call these packets 'tunnel transit packets' (or just 'transit packets') because they will transit the tunnel inside one or more of what we call 'tunnel link packets'. Transit packets correspond to network (IP) packets traversing a conventional link and tunnel link packets correspond to the packets of a conventional link layer (which can be called just 'link packets').

Link packets use the source address of the ingress interface and the destination address of the egress interface - using whatever address is appropriate to the Layer at which the ingress and egress interfaces operate (Layer 2, Layer 3, Layer 4, etc.). The egress interface decapsulates those messages, which then continue on network M as if emerging from a link. To transit packets and to the routers the tunnel connects (Ra and Rd), the tunnel acts as a link and the ingress and egress interfaces act as network interfaces to that link.

The model of each component (ingress and egress interfaces) and the entire system (tunnel) depends on the layer from which they are viewed. 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 interfaces appear as the endpoint hosts on network N.

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, this appearance is incorrect - nothing has changed from the previous case. 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

As already observed, from outside the tunnel, to network M, the entire tunnel acts as a link (Figure 6). Consequently all requirements for links supporting IP also apply to tunnels [RFC3819].

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

Figure 6 Tunnels as viewed from the outside

For example, the IP datagram hop counts (IPv4 Time-to-Live [RFC791] and IPv6 Hop Limit [RFC8200]) are decremented when traversing a router, but not when traversing a link - or thus a tunnel. Similarly, because the ingress and egress are interfaces on this outer network, they should never issue ICMP messages. A router or host would issue the appropriate ICMP, e.g., "packet too big" (IPv4 fragmentation needed and DF set [RFC792] or IPv6 packet too big [RFC4443]), when trying to send a packet to the egress, as it would for any interface.

Tunnels have a tunnel MTU - the largest message that can transit that tunnel, just as links have a link MTU. This MTU may not reflect the native message size of hops within a multihop link (or tunnel) and

the same is true for a tunnel. In both cases, the MTU is defined by the link's (or tunnel's) effective MTU to receive (EMTU_R).

3.3. View from the Inside

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

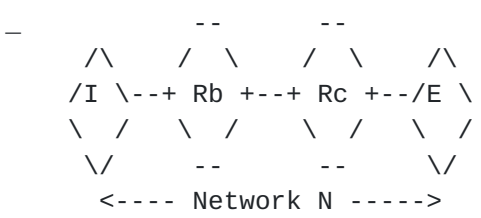


Figure 7 Tunnels, as viewed from within the tunnel

Viewed from within the tunnel, the outer network (M) doesn't exist. Tunnel link packets can be fragmented by the source (ingress interface) and reassembled at the destination (egress interface), just as at conventional hosts. The path between ingress and egress interfaces has a path MTU, but the endpoints can exchange messages as large as can be reassembled at the destination (egress interface), i.e., the EMTU_R of the egress interface. However, in both cases, these MTUs refer to the size of the message that can transit the links and between the hosts of network N, which represents a link layer to network M. I.e., the MTUs of network N represent the maximum frame sizes (MFSs) of the tunnel as a link in network M.

Information about the network - i.e., regarding network N MTU sizes, network reachability, etc. - are relayed from the destination (egress interface) and intermediate routers back to the source (ingress interface), without regard for the external network (M). When such messages arrive at the ingress interface, they may affect the properties of that interface (e.g., its reported MTU to network M), but they should never directly cause new ICMPs in the outer network M. Again, events at interfaces don't generate ICMP messages; it would be the host or router at which that interface is attached that would generate ICMPs, e.g., upon attempting to use that interface.

3.4. Location of the Ingress and Egress

The ingress and egress interfaces are endpoints of the tunnel. 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 on nodes, whereas IPsec tunnels are sometimes implemented as BITW. These implementation variations determine only whether information available at the link endpoints (ingress/egress interfaces) can be easily shared with the connected network nodes.

An ingress or egress can be implemented as an integrated component, appearing equivalent to any other network interface, or can be more complex. In the simple variant, each is tightly coupled to another network interface, e.g., where the ingress emits encapsulated packets directly into another network interface, or where the egress receives packets to decapsulate directly from another network interface.

The other implementation variant is more modular, but more complex to explain. The ingress acts like a network interface by receiving IP packets to transmit from an upper layer protocol (or relay mechanism of a router), but then acts like an upper layer protocol (or relay mechanism of a router) when it emits encapsulated packets back into the same node. The egress acts like an upper layer interface (or relay mechanism of a router) by receiving packets from a network interface, but then acts like a network interface when it emits decapsulated packets back in to the same node. To the existing network interfaces, the ingress/egress act like upper layer interfaces (i.e., sending or receiving application stacks), while to the interior of the node, the ingress/egress act like network interfaces. This dual nature inside the node reflects the duality of the tunnel as transit link and host-host channel.

3.5. Implications of This Model

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

- To the transit packets, tunnels turn a network (Layer 3) path into a (Layer 2) link

- To nodes the tunnel traverses, the tunnel ingress and egress interfaces act as hosts that source and sink tunnel link packets

The consequences of these features are as follow:

- Like a link MTU, a tunnel MTU is defined by the effective MTU of the receiver (i.e., EMTU_R of the egress).

- The messages inside the tunnel are treated like any other link layer, i.e., the MTU is determined by the largest (transit) payload that traverses the link.

- The tunnel path MFS is not relevant to the transited traffic. There is no mechanism or protocol by which it can be determined.

- Because routers, not links, alter hop counts [RFC1812], hopcounts are not decremented solely by the transit of a tunnel. 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 interface correspond to link layer addresses to the transit packet. Like links, some tunnels may not have their own addresses. Like network interfaces, ingress and egress interfaces typically require network layer addresses.

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

- Like network interfaces and links, two nodes may be connected by any combination of tunnels and links, including multiple tunnels. As with multiple links, existing network layer forwarding determines which IP traffic uses each link or tunnel.

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 [RFC1122][RFC3819]. The remainder of this document explores these implications in greater detail.

3.6. Fragmentation

There are two places where fragmentation can occur in a tunnel, called 'outer fragmentation' and 'inner fragmentation'. This document assumes that only outer fragmentation is viable because it is the only approach that works for both IPv4 datagrams with DF=1 and IPv6.

3.6.1. Outer Fragmentation

Outer fragmentation is shown in Figure 8. The bottom of the figure shows the network topology, where transit packets originate at the

source, enter the tunnel at the ingress interface for encapsulation, exit the tunnel at the egress interface where they are decapsulated, and arrive at the destination. The packet traffic is shown above the topology, where the transit packets are shown at the top. In this diagram, the ingress interface is located on router 'Ra' and the egress interface is located on router 'Rd'.

When the link packet - which is the encapsulated transit packet - would exceed the tunnel MTU, the packet needs to be fragmented. In this case the packet is fragmented at the outer (link) header, with the fragments shown as (b1) and (b2). The outer header indicates fragmentation (as ' and "), the inner (transit) header occurs only in the first fragment, and the inner (transit) data is broken across the two packets. These fragments are reassembled at the egress interface during decapsulation in step (c), where the resulting link packet is reassembled and decapsulated so that the transit packet can continue on its way to the destination.

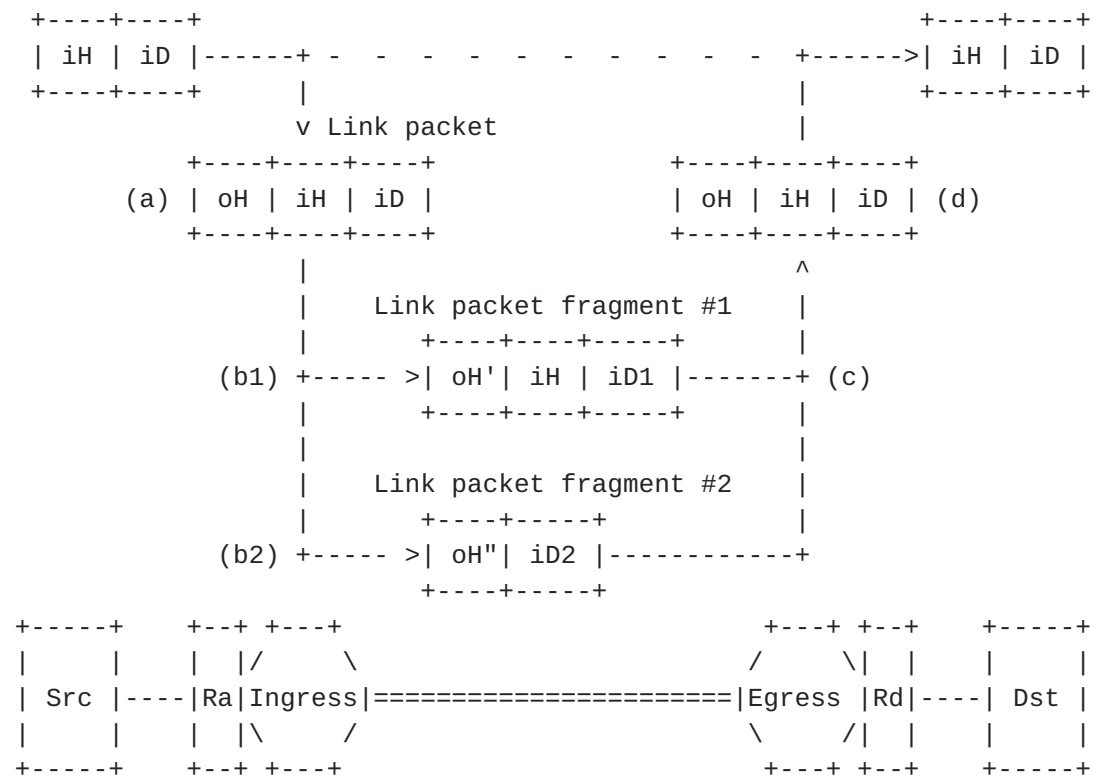


Figure 8 Fragmentation of the (outer) link packet

Outer fragmentation isolates the tunnel encapsulation duties to the ingress and egress interfaces. This can be considered a benefit in clean, layered network design, but also may require complex egress

interface decapsulation, especially where tunnels aggregate large amounts of traffic, such as may result in IP ID overload (see Sec. 4.1.4). Outer fragmentation is valid for any tunnel link protocol that supports fragmentation (e.g., IPv4 or IPv6), in which the tunnel endpoints act as the host endpoints of that protocol.

Along the tunnel, the inner (transit) header is contained only in the first fragment, which can interfere with mechanisms that 'peek' into lower layer headers, e.g., as for relayed ICMP (see Sec. 4.3).

3.6.2. Inner Fragmentation

Inner fragmentation distributes the impact of tunnel fragmentation across both egress interface decapsulation and transit packet destination, as shown in Figure 9; this can be especially important when the tunnel would otherwise need to source (outer) fragment large amounts of traffic. However, this mechanism is valid only when the transit packets can be fragmented on-path, e.g., as when the transit packets are IPv4 datagrams with DF=0.

Again, the network topology is shown at the bottom of the figure, and the original packets show at the top. Packets arrive at the ingress node (router Ra) and are fragmented there based into transit packet fragments #1 (a1) and #2 (a2). These fragments are encapsulated at the ingress interface in steps (b1) and (b2) and each resulting link packet traverses the tunnel. When these link packets arrive at the egress interface they are decapsulated in steps (c1) and (c2) and the egress node (router) forwards the transit packet fragments to their destination. This destination is then responsible for reassembling the transit packet fragments into the original transit packet (d).

Along the tunnel, the inner headers are copied into each fragment, and so can be 'peeked at' inside the tunnel (see Sec. 4.3). Fragmentation shifts from the ingress interface to the ingress router and reassembly shifts from the egress interface to the destination.

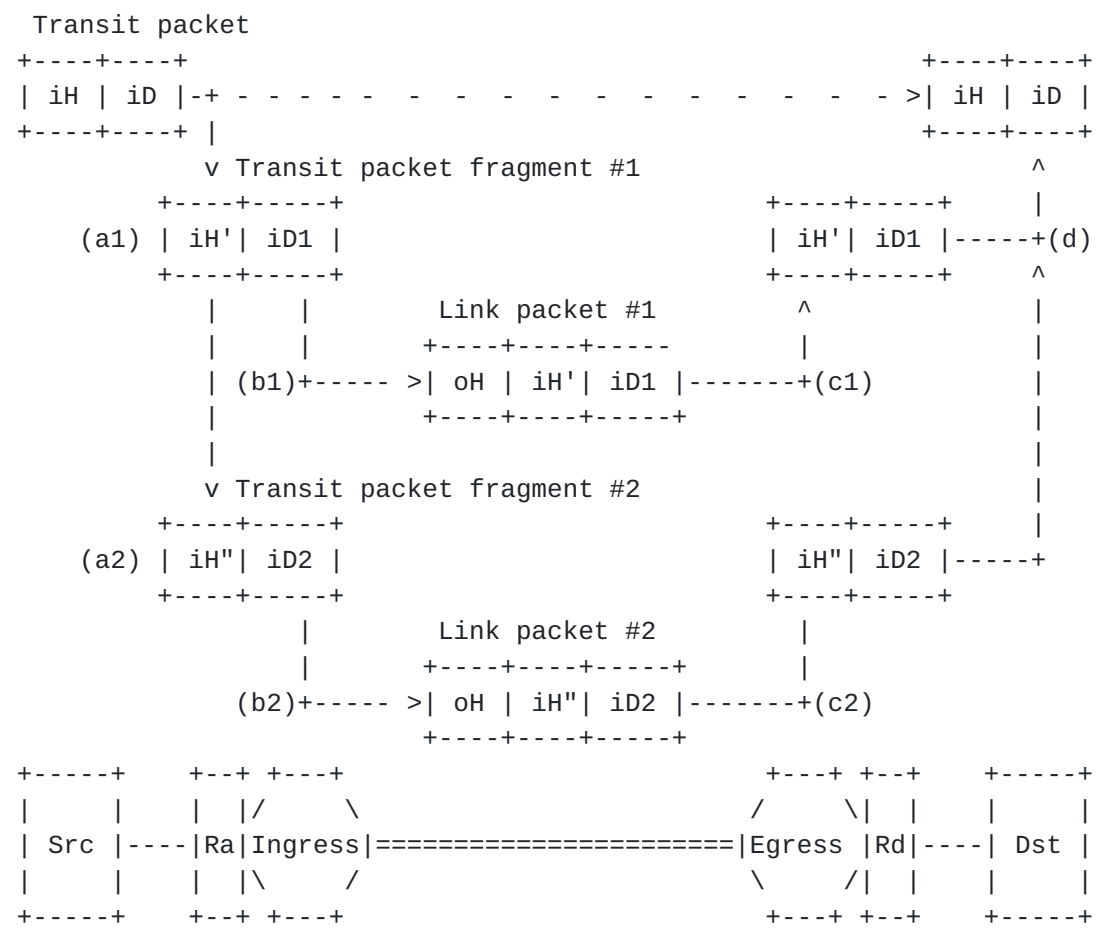


Figure 9 Fragmentation of the inner (transit) packet

3.6.3. The Necessity of Outer Fragmentation

Fragmentation is critical for tunnels that support transit packets for protocols with minimum MTU requirements, while operating over tunnel paths using protocols that have their own MTU requirements. Depending on the amount of space used by encapsulation, these two minimums will ultimately interfere (especially when a protocol transits itself either directly, as with IP-in-IP, or indirectly, as in IP-in-GRE-in-IP), and the transit packet will need to be fragmented to both support a tunnel MTU while traversing tunnels with their own tunnel path MTUs.

Outer fragmentation is the only solution that supports all IPv4 and IPv6 traffic, because inner fragmentation is allowed only for IPv4 datagrams with DF=0.

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 thus must transit the IP minimum MTU, i.e., 68 bytes for IPv4 [RFC793] and 1280 bytes for IPv6 [RFC8200] and a tunnel must support address resolution when there is more than one egress interface for that tunnel.

The requirements of the tunnel ingress and egress interfaces are defined by the network over which they exchange messages (link packets). For IP-over-IP, this means that the ingress interface MUST NOT exceed the IP fragment identification field uniqueness requirements [RFC6864]. Uniqueness is more difficult to maintain at high packet rates for IPv4, whose fragment ID field is only 16 bits.

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

4.1. Encapsulation Header Issues

Tunnel encapsulation uses a non-link protocol as a link layer. The encapsulation layer thus has the same requirements and expectations as any other IP link layer when used to transit IP packets. These relationships are addressed in the following subsections.

4.1.1. General Principles of Header Fields Relationships

Some tunnel specifications attempt to relate the header fields of the transit packet and tunnel link packet. In some cases, this relationship is warranted, whereas in other cases the two protocol layers need to be isolated from each other. For example, the tunnel link header source and destination addresses are network endpoints in the tunnel network N, but have no meaning in the outer network M. The two sets of addresses are effectively independent, just as are other network and link addresses.

Because the tunneled packet uses source and destination addresses with a separate meaning, it is inappropriate to copy or reuse the IPv4 Identification (ID) or IPv6 Fragment ID fields of the tunnel transit packet (see Section 4.1.4). Similarly, the DF field of the transit packet is not related to that field in the tunnel link packet header (presuming both are IPv4) (see Section 4.2). Most other fields are similarly independent between the transit packet and tunnel link packet. 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 the network outside the tunnel, typically relevant only when the tunnel network N and the outer network M use the same network. These apply only when that coordination is defined, as with explicit congestion notification (ECN) [RFC6040] (see Section 4.3.2), and differentiated services code points (DSCPs) [RFC2983]. Equal-cost multipath routing may also affect how some encapsulation fields are set, including IPv6 flow labels [RFC6438] and source ports for transport protocols when used for tunnel encapsulation [RFC8085] (see Section 4.3.4).

4.1.2. Addressing Fields

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, 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.

4.1.3. Hop Count Fields

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][RFC8200]. 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 (routers) 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 hop count 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 tunnel path as if it were a network path that traversed one or more routers, but this is strictly optional. The ability of the outer network M and tunnel network N to avoid indefinitely looping packets does not rely on the hop counts of the transit packet and tunnel link packet being related.

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 network 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.

4.1.4. IP Fragment Identification Fields

Both IPv4 and IPv6 include an IP Identification (ID) field to support IP datagram fragmentation and reassembly [RFC791][RFC1122][RFC8200]. When used, the ID field is intended to be 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).

For IPv4, this field is in the default header and is meaningful only when either source fragmented or DF=0 ("non-atomic packets") [RFC6864]. For IPv6, this field is contained in the optional Fragment Header [RFC8200]. Although IPv6 supports only source fragmentation, the field may occur in atomic fragments [RFC6946].

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 non-atomic IPv4 packets and all IPv6 packets between a source and destination of a given protocol must have unique ID values over the potential fragment reordering period [RFC6864][RFC8200].

The uniqueness of the IP ID is a known problem for high speed nodes, because it limits the speed of a single protocol between two endpoints [RFC4963]. Although this RFC 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. If the ingress enforces IP ID uniqueness, this can either severely limit tunnel throughput or can require substantial resources; the alternative is to ignore IP ID uniqueness and risk reassembly errors. Although fragmentation is somewhat rare in the current Internet at large, it can be common along a tunnel. Reassembly errors are not always detected by other protocol layers (see Sec. 4.3.3) , and even when detected they can result in excessive overall packet loss and can waste bandwidth between the egress and ultimate packet destination.

The 32-bit IPv6 ID field in the Fragment Header is typically used only during source fragmentation. The size of the ID field is typically sufficient that a single counter can be used at the tunnel ingress, regardless of the endpoint addresses or next-header protocol, allowing efficient support for very high throughput tunnels.

The smaller 16-bit IPv4 ID is more difficult to correctly support. A recent update to IPv4 allows the ID to be repeated for atomic packets [RFC6864]. When either source fragmentation or on-path fragmentation is supported, the tunnel ingress may need to keep independent ID counters for each tunnel source/destination/protocol tuple.

4.1.5. 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 [RFC8200]. 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].

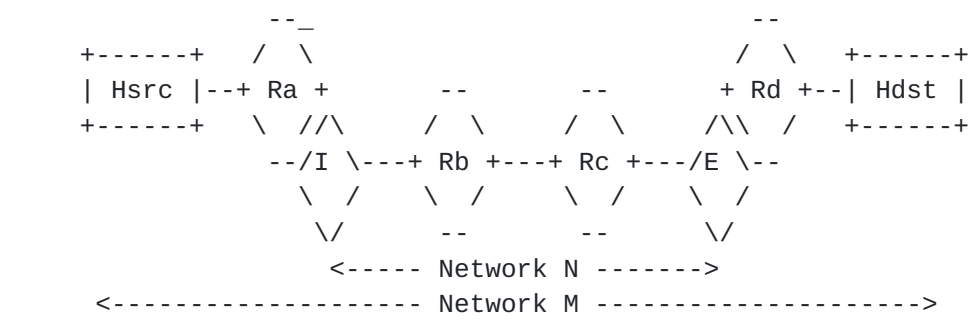
The fragmentation created by the tunnel ingress can increase the need for stronger error detection and correction, especially at the tunnel egress to avoid reassembly errors. The Internet checksum is known to be susceptible to reassembly errors that could be common [RFC4963], and should not be relied upon for this purpose. This is why some tunnel protocols, e.g., SEAL and AERO [RFC5320][Te18] and GRE [RFC2784] as well as legacy protocols swIPe and the Internet Encapsulation Protocol [RFC1853], include a separate checksum. 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 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.

4.2. MTU Issues

Link MTUs, IP datagram limits, and transport protocol segment sizes are already related by several requirements [RFC768][RFC791][RFC1122][RFC1812][RFC8200] and by a variety of protocol mechanisms that attempt to establish relationships between them, including path MTU discovery (PMTUD) [RFC1191][RFC8201], packetization layer path MTU discovery (PLMTUD) [RFC4821], as well as mechanisms inside transport protocols [RFC793][RFC4340][RFC4960]. The following subsections summarize the interactions between tunnels and MTU issues, including minimum tunnel MTUs, tunnel fragmentation and reassembly, and MTU discovery.

4.2.1. Minimum MTU Considerations

There are a variety of values of minimum MTU values to consider, both in a conventional network and in a tunnel as a link in that network. These are indicated in Figure 10, an annotated variant of Figure 4. Note that a (link) MTU (a) corresponds to a tunnel MTU (d) and that a path MTU (b) corresponds to a tunnel path MTU (e). The tunnel MTU is the EMTU_R of the egress interface, because that defines the largest transit packet message that can traverse the tunnel as a link in network M. The ability to traverse the hops of the tunnel - in network N - is not related, and only the ingress need be concerned with that value.



Communication in network M viewed at that layer:
(a) <-> Link MTU
(b) <---- Tunnel MTU --------->
(c) <----------- Path MTU ----------------->
(d) <------------------- EMTU_R --------------------------->

Communication in network N viewed at that layer:
(e) <--> Link MTU
(f) <--- Path MTU ------>
(g) <----- EMTU_R --------->

Communication in network N viewed from network M:
(h) <--> MFS
(i) <--- Path MFS ------>
(j) <----- EMFS_R --------->

Figure 10 The variety of MTU values

Consider the following example values. For IPv6 transit packets, the minimum (link) MTU (a) is 1280 bytes, which similarly applies to tunnels as the tunnel MTU (b). The path MTU (c) is the minimum of the links (including tunnels as links) along a path, and indicates the smallest IP message (packet or fragment) that can traverse a path between a source and destination without on-path fragmentation (e.g., supported in IPv4 with DF=0). Path MTU discovery, either at the network layer (PMTUD [RFC1191][RFC8201]) or packetization layer (PLPMTUD [RFC4821]) attempts to tune the source IP packets and fragments (i.e., EMTU_S) to fit within this path MTU size to avoid fragmentation and reassembly [Ke95]. The minimum EMTU_R (d) is 1500 bytes, i.e., the minimum MTU for endpoint-to-endpoint communication.

The tunnel is a source-destination communication in network N. Messages between the tunnel source (the ingress interface) and tunnel destination (egress interface) similarly experience a variety of network N MTU values, including a link MTU (e), a path MTU (f), and an EMTU_R (g). The network N message maximum is limited by the path MTU, and the source-destination message maximum (EMTU_S) is limited

by the path MTU when source fragmentation is disabled and by EMTU_R otherwise, just as it was in for those types of MTUs in network M. For an IPv6 network N, its link and path MTUs must be at least 1280 and its EMTU_R must be at least 1500.

However, viewed from the context of network M, these network N MTUs are link layer properties, i.e., maximum frame sizes (MFS (h)). The network N EMTU_R determines the largest message that can transit between the source (ingress) and destination (egress), but viewed from network M this is a link layer, i.e., EMFS_R (j). The tunnel EMTU_R is EMFS_R minus the link (encapsulation) headers and includes the encapsulation headers of the link layer. Just as the path MTU has no bearing on EMTU_R, the path MFS (i) in network N has no bearing on the MTU of the tunnel.

For IPv6 networks M and N, these relationships are summarized as follows:

- Network M MTU = 1280, the largest transit packet (i.e., payload) over a single IPv6 link in the base network without source fragmentation

- Network M path MTU = 1280, the transit packet (i.e., payload) that can traverse a path of links in the base network without source fragmentation

- Network M EMTU_R = 1500, the largest transit packet (i.e., payload) that can traverse a path in the base network with source fragmentation

- Network N MTU = 1280 (for the same reasons as for network M)

- Network N path MTU = 1280 (for the same reasons as for network M)

- Network N EMTU_R = 1500 (for the same reasons as for network M)

- Tunnel MTU = 1500-encapsulation (typically 1460), the network N EMTU_R payload

- Tunnel MAP (maximum atomic packet) = largest network M message that transits a tunnel as an atomic packet using network N as a link layer: 1280-encapsulation, i.e., the network N path MTU payload (which is itself limited by the tunnel path MFS)

The difference between the network N MTU and its treatment as a link layer in network M is the reason why the tunnel ingress interfaces need to support fragmentation and tunnel egress interfaces need to

support reassembly in the encapsulation layer(s). The high cost of fragmentation and reassembly is why it is useful for applications to avoid sending messages too close to the size of the tunnel path MTU [Ke95], although there is no signaling mechanism that can achieve this (see Section 4.2.3).

4.2.2. Fragmentation

A tunnel interacts with fragmentation in two different ways. As a link in network M, transit packets might be fragmented before they reach the tunnel - i.e., in network M either during source fragmentation (if generated at the same node as the ingress interface) or forwarding fragmentation (for IPv4 DF=0 datagrams). In addition, link packets traversing inside the tunnel may require fragmentation by the ingress interface - i.e., source fragmentation by the ingress as a host in network N. These two fragmentation operations are no more related than are conventional IP fragmentation and ATM segmentation and reassembly; one occurs at the (transit) network layer, the other at the (virtual) link layer.

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

Like any other link, an IPv4 tunnel must transit 68 byte packets without requiring source fragmentation [RFC791][RFC1122] and an IPv6 tunnel must transit 1280 byte packets without requiring source fragmentation [RFC8200]. The tunnel MTU interacts with routers or hosts it connects the same way as would any other link MTU. The pseudocode examples in this section use the following values:

- TP: transit packet
- TLP: tunnel link packet
- TPsize: size of the transit packet (including its headers)
- encaps: ingress encapsulation overhead (tunnel link headers)
- tunMTU: tunnel MTU, i.e., network N egress EMTU_R - encaps
- tunMAP: tunnel maximum atomic packet as limited by the tunnel path MFS

These rules apply at the host/router where the tunnel is attached, i.e., at the network layer of the transit packet (we assume that all tunnels, including multipoint tunnels, have a single, uniform MTU). These are basic source fragmentation rules (or transit refragmentation for IPv4 DF=0 datagrams), and have no relation to the tunnel itself other than to consider the tunnel MTU as the effective link MTU of the next hop.

Inside the source during transit packet generation or a router during transit packet forwarding, the tunnel is treated as if it were any other link (i.e., this is not tunnel processing, but rather typical source or router processing), as indicated in the pseudocode in Figure 11.

```
if (TPsize > tunMTU) then
   if (TP can be on-path fragmented, e.g., IPv4 DF=0) then
      split TP into TP fragments of tunMTU size
      and send each TP fragment to the tunnel ingress interface
   else
      drop the TP and send ICMP "too big" to the TP source
   endif
else
   send TP to the tunnel ingress (i.e., as an outbound interface)
endif
```

Figure 11 Router / host packet size processing algorithm

The tunnel ingress acts as host on the tunnel path, i.e., as source fragmentation of tunnel link packets (we assume that all tunnels, even multipoint tunnels, have a single, uniform tunnel MTU), using the pseudocode shown in Figure 12. Note that ingress source fragmentation occurs in the encapsulation process, which may involve more than one protocol layer. In those cases, fragmentation can occur at any of the layers of encapsulation in which it is supported, based on the configuration of the ingress.

```
if (TPsize <= tunMAP) then
   encapsulate the TP and emit
else
   if (tunMAP < TPsize) then
      encapsulate the TP, creating the TLP
      fragment the TLP into tunMAP chunks
      emit the TLP fragments
   endif
endif
```

Figure 12 Ingress processing algorithm

Note that these Figure 11 and Figure 12 indicate that a node might both "fragment then encapsulate" and "encapsulate then fragment", i.e., the effect is "on-path fragment, then encapsulate, then source fragment". The first (on-path) fragmentation occurs only for IPv4 DF=0 packets, based on the tunnel MTU. The second (source) fragmentation occurs for all packets, based on the tunnel maximum atomic packet (MAP) size. The first fragmentation is a convenience for a subset of IPv4 packets; it is the second (source) fragmentation that ensures that messages traverse the tunnel.

Just as a network interface should never receive a message larger than its MTU, a tunnel should never receive a message larger than its tunnel MTU limit (see the host/router processing above). A router attempting to process such a message would already have generated an ICMP "packet too big" and the transit packet would have been dropped before entering into this algorithm. Similarly, a host would have generated an error internally and aborted the attempted transmission.

As an example, consider IPv4 over IPv6 or IPv6 over IPv6 tunneling, where IPv6 encapsulation adds a 40 byte fixed header plus IPv6 options (i.e., IPv6 header extensions) of total size 'EHsize'. The tunnel MTU will be at least 1500 - (40 + EHsize) bytes. The tunnel path MTU will be at least 1280 - (40 + EHsize) bytes, which then also represents the tunnel maximum atomic packet size (MAP). Transit packets larger than the tunnel MTU will be dropped by a node before ingress processing, and so do not need to be addressed as part of ingress processing. Considering these minimum values, the previous algorithm uses actual values shown in the pseudocode in Figure 13.

```
if (TPsize <= (1240 - EHsize)) then
   encapsulate TP and emit
else
   if ((1240 - EHsize) < TPsize) then
      encapsulate the TP, creating the TLP
      fragment the TLP into (1240 - EHsize) chunks
      emit the TLP fragments
   endif
endif
```

Figure 13 Ingress processing for an tunnel over IPv6

IPv6 cannot necessarily support all tunnel encapsulations. When the egress EMTU_R is the default of 1500 bytes, an IPv6 tunnel supports IPv6 transit only if EHsize is 180 bytes or less; otherwise the incoming transit packet would have been dropped as being too large by the host/router. Under the same EMTU_R assumption, an IPv6 tunnel supports IPv4 transit only if EHsize is 884 bytes or less. In this

example, transit packets of up to (1240 - Ehsize) can traverse the tunnel without ingress source fragmentation and egress reassembly.

When using IP directly over IP, the minimum transit packet EMTU_R 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 EMTU_R is at least 1280 bytes.

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 PLPMTUD [RFC4821], as done in SEAL [RFC5320] and AERO [Te18]). In particular, many tunnel specifications are often able to avoid persistent fragmentation because they operationally assume larger EMTU_R and tunnel MAP sizes than are guaranteed for IPv4 [RFC1122] or IPv6 [RFC8200].

4.2.3. Path MTU Discovery

Path MTU discovery (PMTUD) enables a network path to support a larger PMTU than it can assume from the minimum requirements of protocol over which it operates. Note, however, that PMTUD never discovers EMTU_R that is larger than the required minimum; that information is available to some upper layer protocols, such as TCP [RFC1122], but cannot be determined at the IP layer.

There is temptation to optimize tunnel traversal so that packets are not fragmented between ingress and egress, i.e., to attempt tune the network M PMTU to the tunnel MAP size rather than to the tunnel MTU, to avoid ingress fragmentation. This is often impossible because the ICMP "packet too big" message (IPv4 fragmentation needed [RFC792] or IPv6 packet too big [RFC4443]) indicates the complete failure of a link to transit a packet, not a preference for a size that matches that internal the mechanism of the link. ICMP messages are intended to indicate whether a tunnel MTU is insufficient; there is no ICMP message that can indicate when a transit packet is "too big for the tunnel path MTU, but not larger than the tunnel MTU". If there were, endpoints might receive that message for IP packets larger than 40 bytes (the payload of a single ATM cell, allowing for the 8-byte AAL5 trailer), but smaller than 9K (the ATM EMTU_R payload).

In addition, attempting to try to tune the network transit size to natively match that of the link internal transit can be hazardous for many reasons:

- The tunnel is capable of transiting packets as large as the network N EMTU_R - encapsulation, which is always at least as large as the tunnel MTU and typically is larger.

- 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".

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

Tunnels that use IPv4 as the encapsulation layer SHOULD set DF=0, but this requires generating unique fragmentation ID values, which may limit throughput [RFC6864]. These tunnels might have difficulty assuming ingress EMTU_S values over 64 bytes, so it may not be feasible to assume that larger packets with DF=1 are safe.

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 ingress payload. Arriving tunnel transit packets have a minimum supported size (1280 for IPv6) and the tunnel PMFS has the same requirement; there would be no room for the tunnel's "link layer" headers, i.e., the encapsulation layer. 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 PLPMTUD [RFC4821]). Conventional path MTU discovery (PMTUD) relies on existing endpoint ICMP processing of explicit negative feedback from routers along the path via "packet to big" ICMP packets in the reverse direction of the tunnel [RFC1191][RFC8201]. 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]. PLPMTUD 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].

PLPMTUD might require that the ingress consider the potential impact of multipath forwarding (see Section 4.3.4). In such cases, probes generated by the ingress might need to track different flows, e.g., that might traverse different tunnel paths. Additionally, encapsulation might need to consider mechanisms to ensure that probes traverse the same path as their corresponding traffic, even when labeled as the same flow (e.g., using the IPv6 flow ID). In such cases, the transit packet and probe may need to be encrypted or encapsulated in an additional flow-based transport header, to avoid differential path traversal based on deep-packet inspection within the tunnel.

4.3. Coordination Issues

IP tunnels interact with link layer signals and capabilities in a variety of ways. The following subsections address some key issues of these interactions. In general, they are again informed by treating a tunnel as any other link layer and considering the interactions between the IP layer and link layers [RFC3819].

4.3.1. 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 nodes can each signal the source of the tunnel transit packets, Hsrc (Figure 14). Inside the tunnel, the inner network N nodes can signal the source of the tunnel link packets, the ingress I (Figure 15).

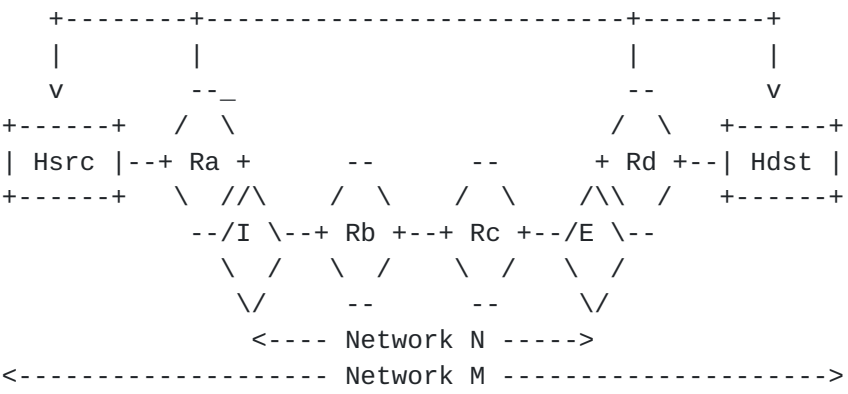


Figure 14 Signals outside the tunnel

Figure 15 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 node (Ra, in both figures).

It is always possible for a network interface to provide hints to its attached node (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. An ICMP error within a tunnel indicates a failure of the path inside the tunnel to support an egress atomic packet or packet fragment size. It can be very difficult to convert that ICMP error into a corresponding ICMP message from the ingress node back to the transit packet source. The ICMP message may not contain enough of a packet prefix to extract the transit packet header sufficient to generate the appropriate ICMP message. The relationship between the egress EMTU_R and the transit packet may be indirect, e.g., the ingress node may be performing source fragmentation that should be adjusted instead of propagating the ICMP upstream.

Some messages have detailed specifications for relaying between the tunnel link packet and transit packet, including Explicit Congestion Notification (ECN [RFC6040]) and multicast (IGMP, e.g.).

4.3.2. Congestion

Tunnels carrying IP traffic (i.e., the focus of this document) need not react directly to congestion any more than would any other link layer [RFC8085]. IP transit packet traffic is already expected to be congestion controlled.

It is useful to relay network congestion notification between the tunnel link and the tunnel transit packets. Explicit congestion notification requires that ECN bits are copied from the tunnel transit packet to the tunnel link packet on encapsulation, as well as copied back at the egress based on a combination of the bits of the two headers [RFC6040]. This allows congestion notification within the tunnel to be interpreted as if it were on the direct path.

4.3.3. Multipoint Tunnels and Multicast

Multipoint tunnels are tunnels with more than two ingress/egress endpoints [RFC2529][RFC5214][Te18]. Just as tunnels emulate links, multipoint tunnels emulate multipoint links, and can support multicast as a tunnel capability. Multipoint tunnels can be useful on their own, or may be used as part of more complex systems, e.g., LISP and TRILL configurations [RFC6830][RFC6325].

Multipoint tunnels require a support for egress determination, just as multipoint links do. This function is typically supported by ARP [RFC826] or ARP emulation (e.g., LAN Emulation, known as LANE

[RFC2225]) for multipoint links. For multipoint tunnels, a similar mechanism is required for the same purpose - to determine the egress address for proper ingress encapsulation (e.g., LISP Map-Service [RFC6833]).

All multipoint systems - tunnels and links - might support different MTUs between each ingress/egress (or link entrance/exit) pair. In most cases, it is simpler to assume a uniform MTU throughout the multipoint system, e.g., the minimum MTU supported across all ingress/egress pairs. This applies to both the ingress EMTU_S and egress EMTU_R (the latter determining the tunnel MTU). Values valid across all receivers need to be confirmed in advance (e.g., via IPv6 ND announcements or out-of-band configuration information) before a multipoint tunnel or link can use values other than the default, otherwise packets may reach some receivers but be "black-holed" to others (e.g., if PMTUD fails [RFC2923]).

A multipoint tunnel MUST have support for broadcast and multicast (or their equivalent), in exactly the same way as this is already required for multipoint links [RFC3819]. Both modes can be supported either by a native mechanism inside the tunnel or by emulation using serial replication at the tunnel ingress (e.g., AMT [RFC7450]), in the same way that links may provide the same support either natively (e.g., via promiscuous or automatic replication in the link itself) or network interface emulation (e.g., as for non-broadcast multiaccess networks, i.e., NBMAs).

IGMP snooping enables IP multicast to be coupled with native link layer multicast support [RFC4541]. A similar technique may be relevant to couple transit packet multicast to tunnel link packet multicast, but the coupling of the protocols may be more complex because many tunnel link protocols rely on their own network N multicast control protocol, e.g., via PIM-SM [RFC6807][RFC7761].

4.3.4. Load Balancing

Load balancing can impact the way in which a tunnel operates. In particular, multipath routing inside the tunnel can impact some of the tunnel parameters to vary, both over time and for different transit packets. The use of multiple paths can be the result of MPLS link aggregation groups (LAGs), equal-cost multipath routing (ECMP [RFC2991]), or other load balancing mechanisms. In some cases, the tunnel exists as the mechanism to support ECMP, as for GRE in UDP [RFC8086].

A tunnel may have multiple paths between the ingress and egress with different tunnel path MTU or tunnel MAP values, causing the ingress

EMTU_S to vary [RFC7690]. When individual values cannot be correlated to transit traffic, the EMTU_S can be set to the minimum of these different path MTU and MAP values.

In some cases, these values can be correlated to paths, e.g., IPv6 packets include a flow label to enable multipath routing to keep packets of a single flow following the same path, as well as to help differentiate path properties (e.g., for path MTU discovery [RFC4821]). It is important to preserve the semantics of that flow label as an aggregate identifier of the encapsulated link packets of a tunnel. This is achieved by hashing the transit IP addresses and flow label to generate a new flow label for use between the ingress and egress addresses [RFC6438]. It is not appropriate to simply copy the flow label from the transit packet into the link packet because of collisions that might arise if a label is used for flows between different transit packet addresses that traverse the same tunnel.

When the transit packet is visible to forwarding nodes inside the tunnel (e.g., when it is not encrypted), those nodes use deep packet inspection (DPI) context to send a single flow over different paths. This sort of "DPI override" of the IP flow information can interfere with both PMTUD and PLPMTUD mechanisms. The only way to ensure that intermediate nodes do not interfere with PLPMTUD is to encrypt the transit packet when it is encapsulated for tunnel traversal, or to provide some other signals (e.g., an additional layer of encapsulation header including transport ports) that preserves the flow semantics.

4.3.5. 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, as in GUE [He17]).

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

The following subsections summarize the observations of this document and a summary of issues with existing tunnel protocol specifications. It also includes advice for tunnel protocol designers, implementers, and operators. It also includes

5.1. Summary of Recommendations

- Tunnel endpoints are network interfaces, tunnel are virtual links
  - ICMP messages MUST NOT be generated by the tunnel (as a link)
  - ICMP messages received by the ingress inside link change the link properties (they do not generate transit-layer ICMP messages)
  - Link headers (hop, ID, options) are largely independent of arriving ID (with few exceptions based on translation, not direct copying, e.g., ECN and IPv6 flow IDs)
- MTU values should treat the tunnel as any other link
  - Require source ingress source fragmentation and egress reassembly at the tunnel link packet layer
  - The tunnel MTU is the tunnel egress EMTU_R less headers, and not related at all to the ingress-egress MFS
- Tunnels must obey core IP requirements
  - Obey IPv4 DF=1 on arrival at a node (nodes MUST NOT fragment IPv4 packets where DF=1 and routers MUST NOT clear the DF bit)
  - Shut down an IP tunnel if the tunnel MTU falls below the required minimum

5.2. Impact on Existing Encapsulation Protocols

Many existing and proposed encapsulation protocols are inconsistent with the guidelines of this document. The following list summarizes only those inconsistencies, but omits places where a protocol is inconsistent solely by reference to another protocol.

[should this be inverted as a table of issues and a list of which RFCs have problems?]

- IP in IP / mobile IP [RFC2003][RFC4459] - IPv4 in IPv4
  - Sets link DF when transit DF=1 (fails without PLPMTUD)
  - Drops at egress if hopcount = 0 (host-host tunnels fail)
  - Drops based on transit source (same as router IP, matches egress), i.e., performs routing functions it should not
  - Ingress generates ICMP messages (based on relayed context), rather than using inner ICMP messages to set interface properties only
  - Treats tunnel MTU as tunnel path MTU, not tunnel egress MTU
- IPv6 tunnels [RFC2473] -- IPv6 or IPv4 in IPv6
  - Treats tunnel MTU as tunnel path MTU, not tunnel egress MTU
  - Decrements transiting packet hopcount (by 1)
  - Copies traffic class from tunnel link to tunnel transit header
  - Ignores IPv4 DF=0 and fragments at that layer upon arrival
  - Fails to retain soft ingress state based on inner ICMP messages affecting tunnel MTU
  - Tunnel ingress issues ICMPs
  - 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)
- IPsec tunnel mode (IP in IPsec in IP) [RFC4301] -- IP in IPsec
  - Uses security policy to set, clear, or copy DF (rather than generating it independently, which would also be more secure)
  - Intertwines tunnel selection with security selection, rather than presenting tunnel as an interface and using existing forwarding (as with transport mode over IP-in-IP [RFC3884])
- GRE (IP in GRE in IP or IP in GRE in UDP in IP) [RFC2784][RFC7588][RFC7676][RFC8086]
  - Treats tunnel MTU as tunnel path MTU, not tunnel egress MTU

- o Requires ingress to generate ICMP errors
- o Copies IPv4 DF to outer IPv4 DF
- o Violates IPv6 MTU requirements when using IPv6 encapsulation

- o LISP [RFC6830]
  - o Treats tunnel MTU as tunnel path MTU, not tunnel egress MTU
  - o Requires ingress to generate ICMP errors
  - o Copies inner hop limit to outer
- o L2TP [RFC3931]
  - o Treats tunnel MTU as tunnel path MTU, not tunnel egress MTU
  - o Requires ingress to generate ICMP errors
- o PWE [RFC3985]
  - o Treats tunnel MTU as tunnel path MTU, not tunnel egress MTU
  - o Requires ingress to generate ICMP errors
- o GUE (Generic UDP encapsulation) [He17] - IP (et. al) in UDP in IP
  - o Allows inner encapsulation fragmentation
- o Geneve [RFC7364][Gr18] - IP (et al.) in Geneve in UDP in IP
  - o Treats tunnel MTU as tunnel path MTU, not tunnel egress MTU
- o SEAL/AERO [RFC5320][Te18] - IP in SEAL/AERO in IP
  - o Some issues with SEAL (MTU, ICMP), corrected in AERO
- o RTG DT encapsulations [No16]
  - o Assumes fragmentation can be avoided completely
  - o Allows encapsulation protocols that lack fragmentation
  - o Relies on ICMP PTB to correct for tunnel path MTU
- o No known issues

- o L2VPN (framework for L2 virtualization) [RFC4664]
- o L3VPN (framework for L3 virtualization) [RFC4176]
- o MPLS (IP in MPLS) [RFC3031]
- o TRILL (Ethernet in Ethernet) [RFC5556][RFC6325]

5.3. Tunnel Protocol Designers

[To be completed]

Recursive tunneling + minimum MTU = frag/reassembly is inevitable, at least to be able to split/join two fragments

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 PLPMTUD into the tunneling protocol.

5.3.1. For Future Standards

[To be completed]

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

5.3.2. Diagnostics

[To be completed]

Some current implementations include diagnostics to support monitoring the impact of tunneling, especially the impact on fragmentation and reassembly resources, the status of path MTU discovery, etc.

>> Because a tunnel ingress/egress is a network interface, it SHOULD have similar resources as any other network interface. This includes resources for packet processing as well as monitoring.

5.4. Tunnel Implementers

[To be completed]

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. This is always better than blindly assuming the tunnel has been deployed correctly, i.e., that the solution has been engineered.

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.

5.5. Tunnel Operators

[To be completed]

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.

Consider the circuit breakers doc to provide diagnostics and last-resort control to avoid overload for non-reactive traffic (see Gorry's RFC-to-be)

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.

>>>> PLPMTUD can give multiple conflicting PMTU values during ECMP or LAG if PMTU is cached per endpoint pair rather than per flow -- but so can PMTUD! This is another reason why ICMP should never drive up the effective MTU (if aggregate, treat as the minimum of received messages over an interval).

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.

Tunnels are susceptible to attacks at both the inner and outer network layers. The tunnel ingress/egress endpoints appear as network interfaces in the outer network, and are as susceptible as any other network interface. This includes vulnerability to fragmentation reassembly overload, traffic overload, and spoofed ICMP messages that misreport the state of those interfaces. Similarly, the ingress/egress appear as hosts to the path traversed by the tunnel, and thus are as susceptible as any other host to attacks as well.

[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.

[are there others? 3819? ECN? Flow label issues?]

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.

[Gr18] Gross, J. (Ed.), I. Ganga (Ed.), T. Sridhar (Ed.), "Geneve: Generic Network Virtualization Encapsulation," draft-ietf-nvo3-geneve-05, Sep. 2017.

[He17] Herbert, T., L. Yong, O. Zia, "Generic UDP Encapsulation," draft-ietf-intarea-gue-05, Dec. 2017.

[Ke95] Kent, S., J. Mogul, "Fragmentation considered harmful," ACM Sigcomm Computer Communication Review (CCR), V25 N1, Jan. 1995, pp. 75-87.

[No16] Nordmark, E. (Ed.), A. Tian, J. Gross, J. Hudson, L. Kreeger, P. Garg, P. Thaler, T. Herbert, "Encapsulation Considerations," draft-ietf-rtgwg-dt-encap-02, Oct. 2016.

[RFC5] Rulifson, J, "Decode Encode Language (DEL)," RFC 5, June 1969.

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

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

[RFC792] Postel, J., "Internet Control Message Protocol," RFC 792, Sep. 981.

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

[RFC826] Plummer, D., "An Ethernet Address Resolution Protocol -- or -- Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware," RFC 826, Nov. 1982.

[RFC1075] Waitzman, D., C. Partridge, S. Deering, "Distance Vector Multicast Routing Protocol," RFC 1075, Nov. 1988.

[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.

[RFC1853] Simpson, W., "IP in IP Tunneling," RFC 1853, Oct. 1995.

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

[RFC2225] Laubach, M., J. Halpern, "Classical IP and ARP over ATM," RFC 2225, Apr. 1998.

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

[RFC2529] Carpenter, B., C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," RFC 2529, Mar. 1999.

[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.

[RFC2983] Black, D., "Differentiated Services and Tunnels," RFC 2983, Oct. 2000.

[RFC2991] Thaler, D., C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection," RFC 2991, Nov. 2000.

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

[RFC2546] Durand, A., B. Buclin, "6bone Routing Practice," RFC 2540, Mar. 1999.

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

[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.

[RFC4340] Kohler, E., M. Handley, S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, Mar. 2006.

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

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

[RFC4541] Christensen, M., K. Kimball, F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches," RFC 4541, May 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.

[RFC4960] Stewart, R. (Ed.), "Stream Control Transmission Protocol," RFC 4960, Sep. 2007.

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

[RFC5214] Templin, F., T. Gleeson, D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)," RFC 5214, Mar. 2008.

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

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

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

[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.

[RFC6434] Jankiewicz, E., J. Loughney, T. Narten, "IPv6 Node Requirements," RFC 6434, Dec. 2011.

[RFC6438] Carpenter, B., S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels," RFC 6438, Nov. 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.

[RFC6833] Fuller, V., D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface," RFC 6833, 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.

[RFC6946] Gont, F., "Processing of IPv6 "Atomic" Fragments," RFC 6946, May 2013.

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

[RFC7450] Bumgardner, G., "Automatic Multicast Tunneling," RFC 7450, Feb. 2015.

[RFC7510] Xu, X., N. Sheth, L. Yong, R. Callon, D. Black, "Encapsulating MPLS in UDP," RFC 7510, April 2015.

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

[RFC7676] Pignataro, C., R. Bonica, S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)," RFC 7676, Oct 2015.

[RFC7690] Byerly, M., M. Hite, J. Jaeggli, "Close Encounters of the ICMP Type 2 Kind (Near Misses with ICMPv6 Packet Too Big (PTB))," RFC 7690, Jan. 2016.

[RFC7761] Fenner, B., M. Handley, H. Holbrook, I. Kouvelas, R. Parekh, Z. Zhang, L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 7761, Mar. 2016.

[RFC8085] Eggert, L., G. Fairhurst, G. Shepherd, "Unicast UDP Usage Guidelines," RFC 8085, Oct. 2015.

[RFC8086] Yong, L. (Ed.), E. Crabbe, X. Xu, T. Herbert, "GRE-in-UDP Encapsulation," RFC 8086, Feb. 2017.

[RFC8200] Deering, S., R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 8200, Jul. 2017.

[RFC8201] McCann, J., S. Deering, J. Mogul, R. Hinden (Ed.), "Path MTU Discovery for IP version 6," RFC 8201, Jul. 2017.

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

[Te18] Templin, F., "Asymmetric Extended Route Optimization," draft-templin-aerolink-78, Jan. 2018.

[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 ISI-TR-570, Aug. 2003.

[To16] Touch, J., "Middleboxes Models Compatible with the Internet," USC/ISI Tech. Report ISI-TR-711, Oct. 2016.

[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. It benefitted substantially from detailed feedback from Toerless Eckert, Vincent Roca, and Lucy Yong, as well as other members of the Internet Area Working Group.

This work is partly supported by USC/ISI's Postel Center.

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

Authors' Addresses

Joe Touch
Manhattan Beach, CA 90266
U.S.A.

Phone: +1 (310) 560-0334
Email: touch@strayalpha.com

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

Email: townsley@cisco.com

APPENDIX A: Fragmentation efficiency

A.1. Selecting fragment sizes

There are different ways to fragment a packet. Consider a network with a PMTU as shown in Figure 16, 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 PMTU arrives, it must be fragmented to accommodate the additional header.

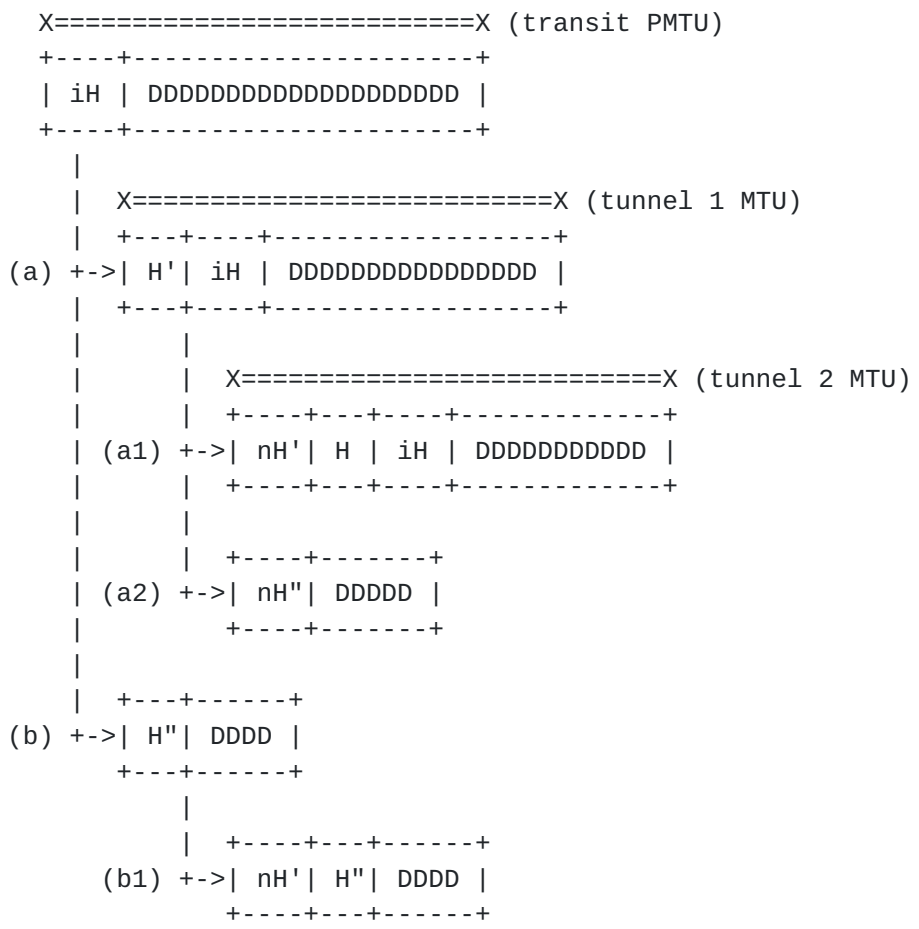


Figure 16 Fragmenting via maximum fit

Figure 16 shows this process using "maximum fit", assuming outer fragmentation as an example (the situation is the same for inner fragmentation, but the headers that are affected differ). In maximum fit, the arriving packet is split into (a) and (b), where (a) is the size of the first tunnel, i.e., the tunnel 1 MTU (the maximum that fits over the first tunnel). However, this tunnel then traverses over another tunnel (number 2), 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 it needs to be fragmented as well to fit into the tunnel 2 MTU, 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 tunnel 2 MTU size.

In Figure 17, the fragmentation is done using "even split", 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 tunnel 2 MTU, and neither requires further fragmentation.

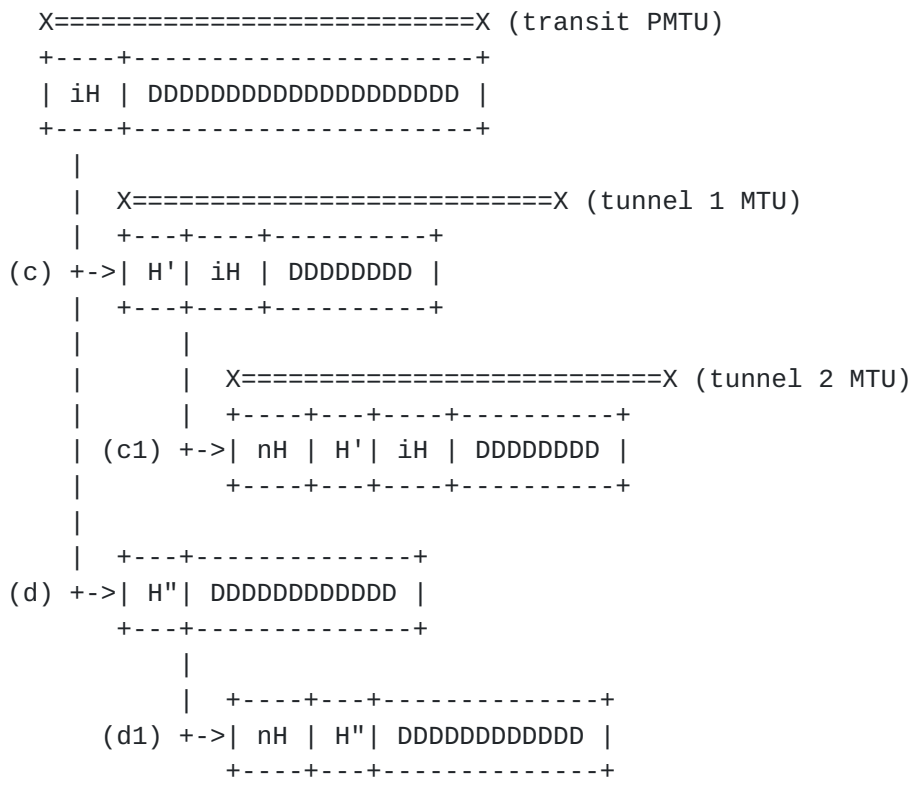


Figure 17 Fragmenting via "even split"

A.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 18). 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.

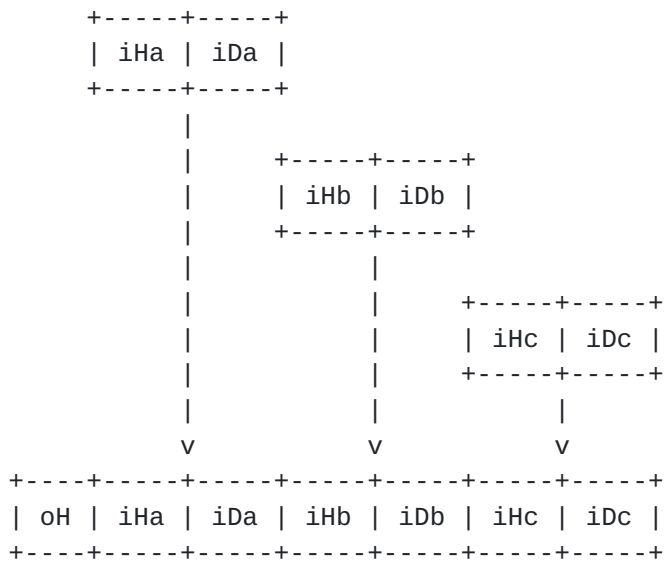


Figure 18 Packing packets into a tunnel