Internet Area WG
Internet Draft
Intended status: Informational
Expires: September 2010

J. Touch
USC/ISI
M. Townsley
Cisco
March 5, 2010

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

Status of this Memo

Abstract

This document discusses the role of tunnels in the Internet architecture. It explains their relationship to existing protocol layers, and the challenges in supporting tunneling.

Table of Contents

1. Introduction

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

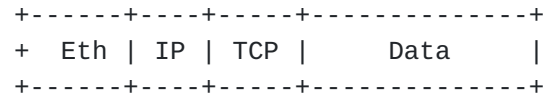

Figure 1 TCP inside IP inside Ethernet

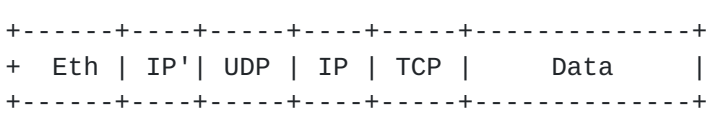

Figure 2 IP in UDP in IP in Ethernet

Tunnels help decouple topology from that provided by the physical network components. For example, they were critical in the development of multicast, where not all routers were capable of processing multicast packets. Multicast routers were interconnected by tunnels where not directly connected. Similar techniques have been used to support other protocols, such as IPv6.

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

- IPsec - hides the original traffic destination [RFC4301]
- L2TP - Tunnels PPP over IP, used largely in DSL/FTTH access networks to extend a subscriber's connection from an access line provider to an ISP [RFC3931]
- Mobile IP - forwards traffic to the home agent [RFC2003]

- L2VPNs - provides a link topology different from that provided by physical links [RFC4664]
- L3VPNs - provides a network topology different from that provided by ISPs [RFC4176]
- SEAL - a generic mechanism for IP in IP tunneling designed to overcome the limitations of RFC2003 [RFC5320]
- LISP - reduces routing table load within an enclave of routers [Fa10]
- TRILL - enables L3 routing in an enclave of bridges [Pe10][RFC5556]
- MPLS - ? {need description/ref}
- PWE3 - ? {need description/ref}

The variety of tunnel mechanisms begs the question of the roles of tunnels in the Internet architecture, and the potential need for coordination of these mechanisms. In particular, the ways in which MTU mismatch, error signals (e.g., ICMP), and is handled may benefit from a coordinated approach.

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

2. Conventions used in this document

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].

3. Known Issues

Most of the known issues with tunnels arise from the complications of encapsulation, or from the introduction of artificial endpoints along a data path. Encapsulation exacerbates MTU issues, often because a data unit will traverse at least one layer of a protocol stack more than once (e.g., as in Figure 2), which requires space for additional headers. This space complicates MTU discovery, and often results in fragmentation.

Tunnel encapsulation and decapsulation nodes act as network endpoints. They may source and sink much higher bandwidth streams from single IP addresses, and thus can be affected by many of the issues of other high bandwidth edge devices, such as fragmentation efficiency and IP ID exhaustion (in IPv4). These endpoints also introduce complexity in end-to-end and path signaling, in the translation between signals inside a tunnel and signals outside on the end-to-end path.

3.1. MTU discovery

MTU discovery is a known challenge in the current Internet, and tunnels can complicate its proper operation. Encapsulation increases the size of a packet during tunnel transit that can exceed the MTU of the links of the tunnel path. This is especially true for recursive tunnels, i.e., tunnels that reuse layers of the protocol stack (e.g., IPv4 over IPv4). These issues are discussed in detail in [RFC4459]; the following provides a brief overview of the issues. Note that the impact of tunnels on MTU discovery may be mitigated somewhat by the ubiquity of workarounds already needed in the Internet, e.g., the deduction of a 'tunnel tax' for all MTUs (i.e., maxing out the MTU at 1200-1400 bytes, rather than 1500).

Conventional path MTU discovery (PMTUD) relies on explicit negative feedback from routers along the path (ICMP "message to big" signals) [RFC1191]. This technique is susceptible to the "black hole" phenomenon, in which the ICMP messages never return to the source [RFC2923]. In the typical Internet case, lost ICMPs are often the result of filtering, e.g., for policy reasons.

A more recent alternative is packetization-layer path MTU discovery (PLPMTUD) [RFC4821]. This variant relies on feedback from the endpoint, indicating either the success or failure of probe packets. It is not susceptible to "black holing", but requires explicit participation by the receiver.

Either of these techniques (PMTUD, PLPMTUD) can be applied to tunnels. The encapsulator must react to "message to big" signals in either case, by either adjusting its fragmentation, relaying a corresponding signal to the packet origin outside the tunnel, or both. Fragmentation adjustment is easy to incorporate, but can result in inefficient transmission of packets over the tunnel (e.g., where every source packet is fragmented). Relaying the signal to the source can be much more efficient, but it can be difficult to determine what signal to forward. E.g., in PMTUD, routers along the tunnel may not return a sufficiently long prefix to determine the decapsulated packet origin.

Tunnels thus may need to participate in MTU discovery, either forwarding or recomputing ICMPs received inside the tunnel path. The tunnel may incorporate its own MTU discovery between ingress and egress, e.g., as proposed in SEAL [RFC5320].

3.2. Fragmentation

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

3.2.1. Outer Fragmentation

The simplest case is Outer Fragmentation, as shown in Figure 3. The bottom of the figure shows the network toplogy, where packets start at the source, enter the tunnel at the encapsulator, exit the tunnel at the decapsulator, and arrive finally at the destination. The packet traffic is shown above the topology, where the end-to-end packets are shown at the top. The packets are composed of an inner header (iH) and inner data (iD); the term "inner") is relative to the tunnel, as will become apparent. When the packet (iH,iD) arrives at the encapsulator, it is placed inside the tunnel packet structure, here shown as adding just an outer header, oH, in step (a).

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

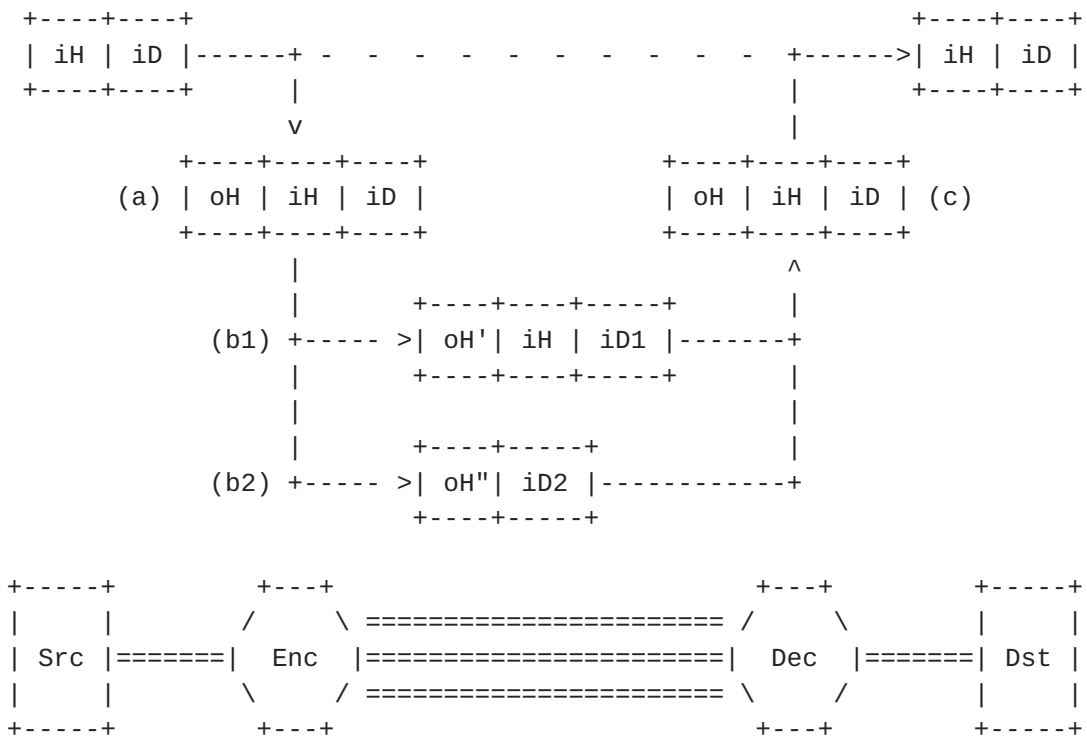

Figure 3 Fragmentation of the outer packet

Outer fragmentation isolates Source and Destination from tunnel encapsulation duties. This can be considered a benefit in clean, layered network design, but also may result in complex decapsulator design, especially where tunnels aggregate large amounts of traffic, such as IP ID overload (see Sec. 3.2.5). Outer fragmentation is valid for any tunnel encapsulation protocol that supports fragmentation (e.g., IPv4 or IPv6), where the tunnel endpoints act as the host endpoints of that protocol.

Along the tunnel, the inner header is contained only in the first fragment, which can interfere with mechanisms that 'peek' into lower layer headers, e.g., as for ICMP, as discussed in Sec. 3.3.

3.2.2. Inner Fragmentation

Inner Fragmentation distributes the impact of tunneling across both the decapsulator and destination, and is shown in Figure 4. Again, the network topology is shown at the bottom of the figure, and the original packets show at the top. Packets arrive at the encapsulator, and are fragmented there based on the inner header into (a1) and (a2). The fragments arrive at the decapsulator, which removes the outer header and forwards the resulting fragments on to the

destination. The destination is then responsible for reassembling the fragments into the original packet.

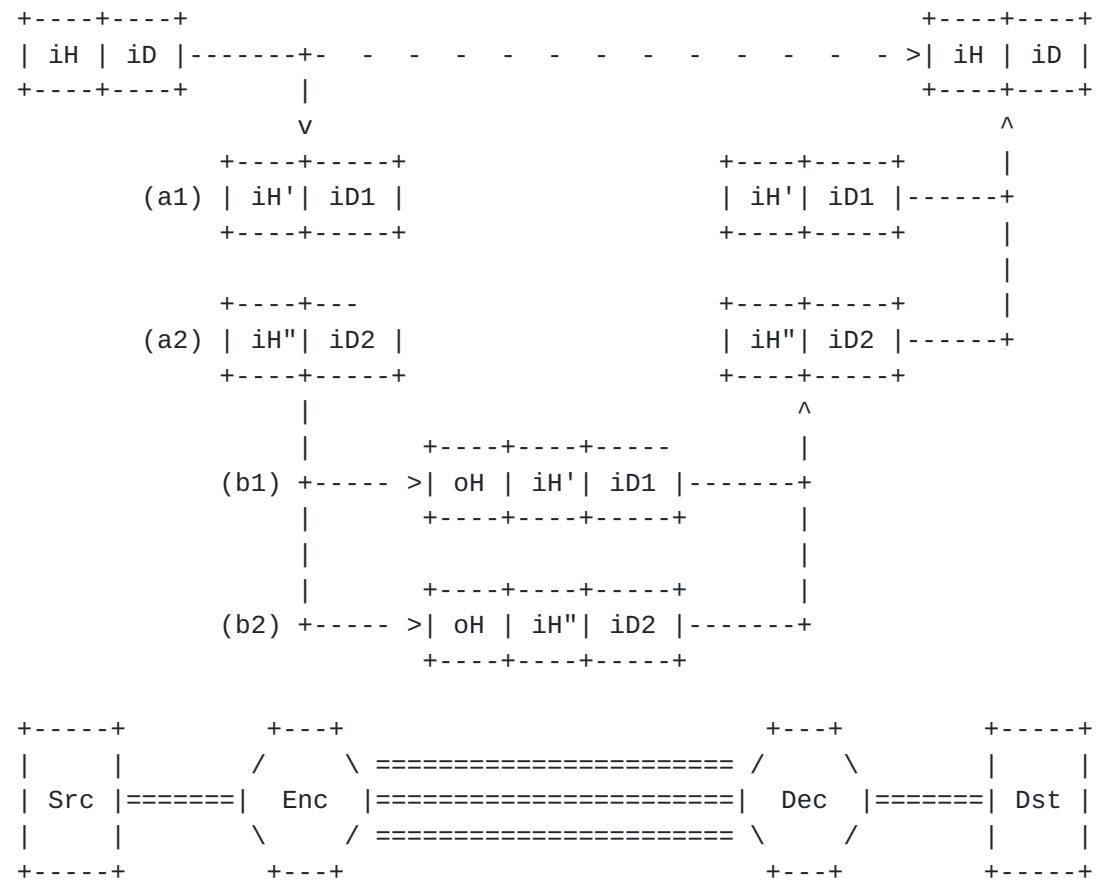

Figure 4 Fragmentation of the inner packet

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

Along the tunnel, the inner headers are copied into each fragment, and so are available to mechanisms that 'peek' into headers (e.g., ICMP, as discussed in Sec. 3.3). Because fragmentation happens on the inner header, the impact of IP ID is reduced.

3.2.3. Fragmentation efficiency

There are different ways to fragment a packet. Consider a network with an MTU as shown in Figure 5, where packets are encapsulated over the same network layer as they arrive on (e.g., IP in IP). If a

packet as large as the MTU arrives, it must be fragmented to accommodate the additional header.

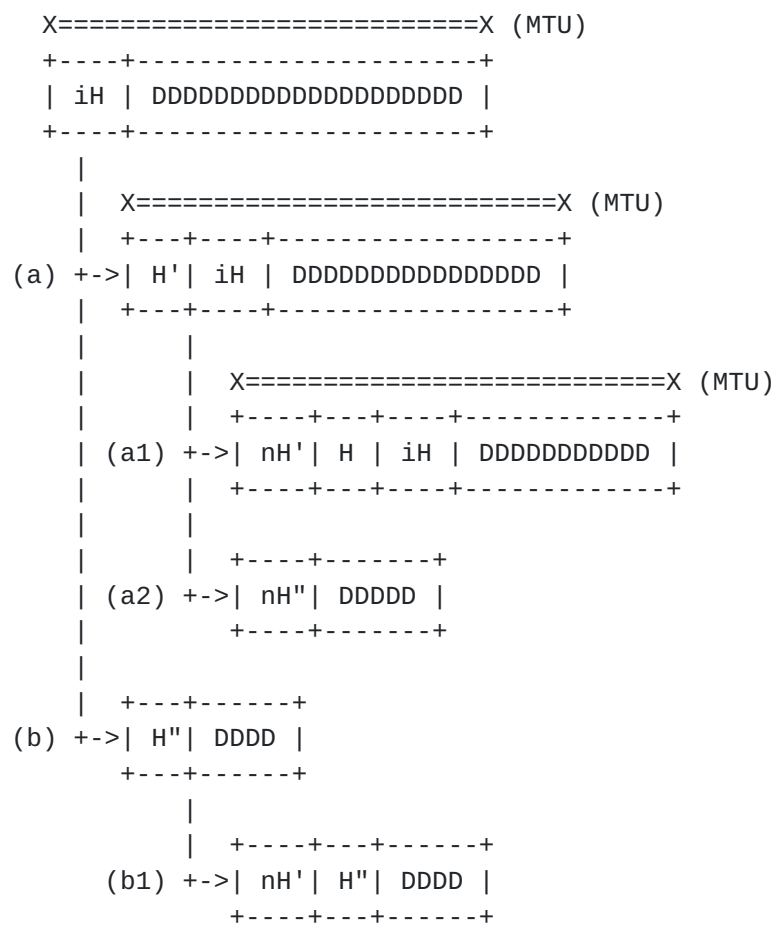

Figure 5 Fragmenting via maximum fit

Figure 5 shows this process, using Outer Fragmentation as an example (the situation is the same for Inner Fragmentation, but the headers that are affected differ). The arriving packet is first split into (a) and (b), where (a) is of the MTU of the network. However, this tunnel then traverses over another tunnel, whose impact the first tunnel ingress has not accommodated. The packet (a) arrives at the second tunnel ingress, and needs to be encapsulated again, but because it is already at the MTU, it needs to be fragmented as well, into (a1) and (a2). In this case, packet (b) arrives at the second tunnel ingress and is encapsulated into (b1) without fragmentation, because it is already below the MTU size.

In Figure 6, the fragmentation is done evenly, i.e., by splitting the original packet into two roughly equal-sized components, (c) and (d). Note that (d) contains more packet data, because (c) includes the

original packet header because this is an example of Outer Fragmentation. The packets (c) and (d) arrive at the second tunnel encapsulator, and are encapsulated again; this time, neither packet exceeds the MTU, and neither requires further fragmentation.

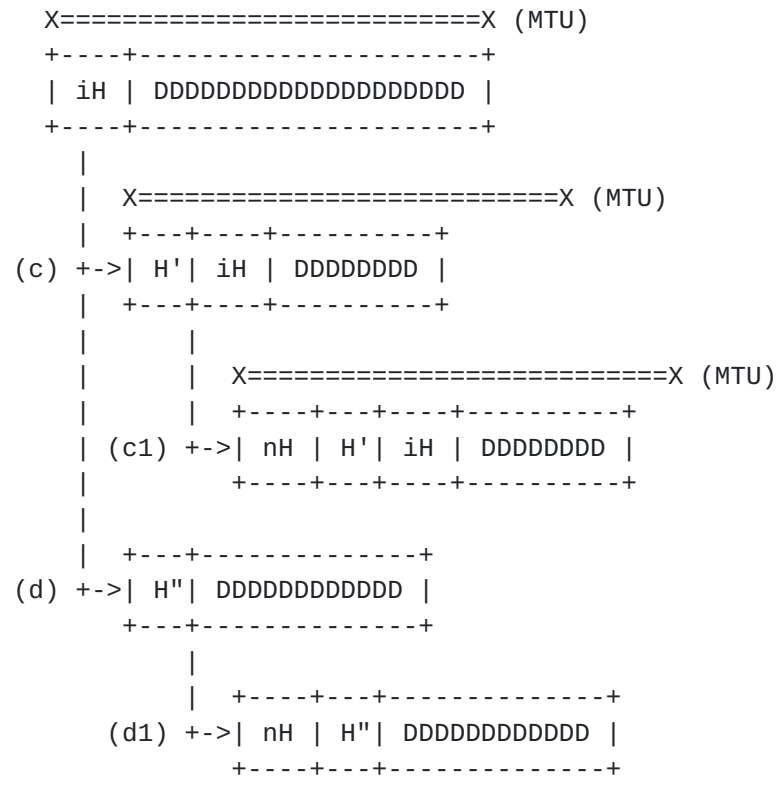

Figure 6 Fragmenting evenly

3.2.4. Packing (ala GigE bursting)

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, reduces the overhead of the encapsulation headers (Figure 7). 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.

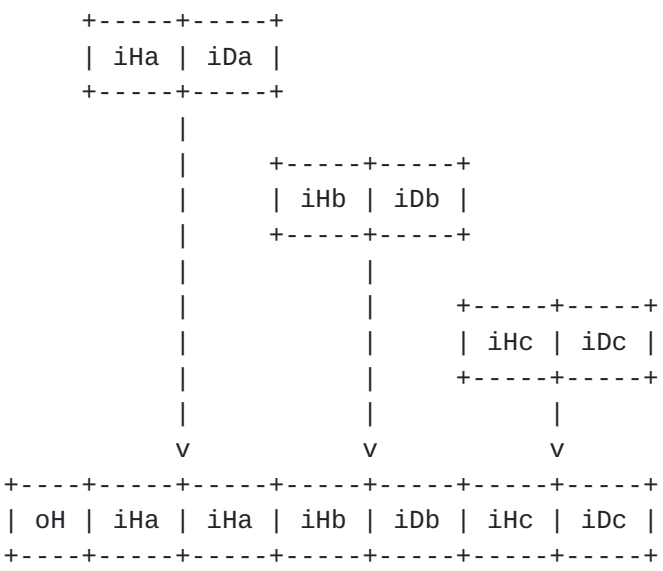

Figure 7 Packing packets into a tunnel

3.2.5. IP ID exhaustion

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

The uniqueness of the IP ID is a known problem for high speed devices, because it limits the speed of a single protocol between two endpoints [RFC4963]. With the maximum IP packet size of 64KB, a 16-bit ID field that does not repeat within 120 seconds means that the sum of all TCP connections between two endpoints is limited to roughly 286 Mbps; for more typical MTUs of 1500 bytes, this drops to 6.4 Mbps.

Although this strongly suggests that the uniqueness of the IP ID is moot, tunnels exacerbate this condition. A tunnel often aggregates traffic from a number of different source and destination addresses, of different protocols, and encapsulates them in a header with the same ingress and egress addresses, all using a single encapsulation protocol. The result is one of the following:

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

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

3. The IP ID rules are ignored.

The last case is the most obvious solution, because it corresponds to how endpoints currently behave. Fortunately, fragmentation is somewhat rare in the current Internet at large, but it can be common along a tunnel. Fragments that repeat the IP ID risk being reassembled incorrectly, especially when fragments are reordered or lost. Although such errors may be detected at the transport layer, this results in excessive overall packet loss, as well as wasting bandwidth between the egress and ultimate packet destination.

3.3. Signaling

In the current Internet architecture, signals tend to go upstream, either from routers along a path or from the destination, back toward the source (Figure 8). 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.

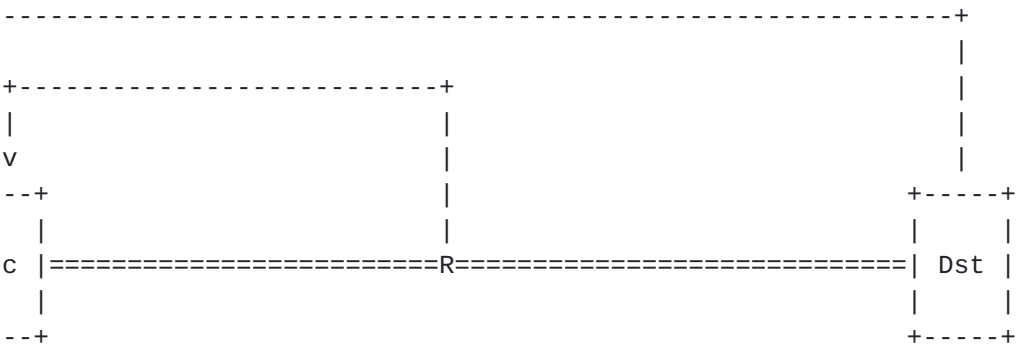

Figure 8 Signaling paths in an Internet

Tunnels interfere with these known signaling paths. As shown in Figure 9, signals from routers along the tunnel path (R2), as well as those from the tunnel egress, need to be relayed by the ingress. This relaying may be difficult, because R2 may not return enough information to the ingress to support relaying (e.g., when ICMP returns only the outermost headers in a "message to big", and the source transport port information is lost). Signals from routers

downstream of the egress (R3 in Figure 9) need to traverse the tunnel in reverse.

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).

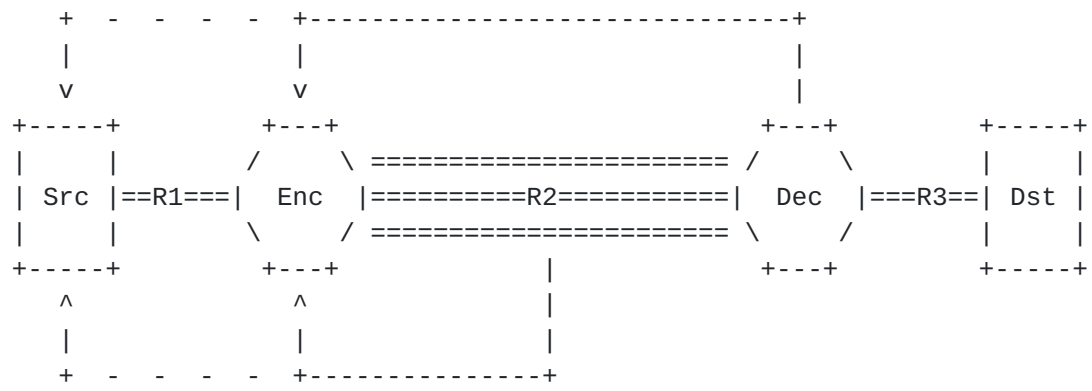

Figure 9 Signaling paths introduced by a tunnel

4. Current Tunnel Standards

This section reviews two common Internet tunnel standards. They are notable because they both ultimately rely on IP in IP encapsulation, although they each handle MTU discovery, fragmentation, and signaling differently.

[There are other tunnel mechanisms, such as IPv4 in IPv6, which may be added to this discussion later.]

4.1. IP in IP

The simplest tunnel encapsulation mechanism is IP in IP, explained here for IPv4 [RFC2003]. This protocol was standardized for use in mobile IP, so that packets sent from a source to a Home Agent could be forwarded unmodified to the different address of the Mobile Node [RFC3344]. It has come to be used much more generally, e.g., to support multicast, as well as in overlay network systems [Er94][To01].

4.1.1. MTU discovery

When an IPv4 packet arrives at an IP-in-IP ingress, the DF flag from the inner packet is copied to the outer header. This enforces DF of the packet within the tunnel when requested by the packet source.

Packets which are too large are dropped at the ingress, and a corresponding ICMP "message to big" is returned to the source. Internally, IP-in-IP tunneling requires that the tunnel MUST support ICMP-based path MTU discovery (i.e., PMTUD). Note that due to common filtering of ICMP messages, this requirement is impossible to determine and thus to enforce.

4.1.2. Fragmentation

IP-in-IP tunneling supports Inner Fragmentation. The inner packet MAY be fragmented if DF=0, otherwise the packet would have been dropped if too big, as noted earlier. The tunnel MUST NOT fragment at the outer header if DF=1 is set, i.e., this tunnel protocol assumes the network honors the DF bit (note that some tunnels, as well as some network devices, do not honor the DF bit). Further, if the DF bit is set in the inner header, it MUST be set in the outer; if not, it MAY be set in the outer.

4.1.3. Signaling

IP-in-IP tunnels MAY relay ICMPs from inside the tunnel to the source, i.e., at the ingress. They SHOULD relay network and host unreachable messages, and MUST relay "message too big" messages; these reflect network conditions that the source should be informed about. They MUST NOT relay port unreachable messages, because these are meaningless for encapsulated packets, and thus reflect internal link conditions that the source should not care about at all. They MUST NOT relay and SHOULD handle locally messages that affect the ingress as if it were a host, e.g., source quench and router errors.

Most notably, IP-in-IP notes that the tunnel SHOULD keep sufficient soft state to assist with relaying. Such state may involve keeping copies of recently sent packets, to have sufficient context to relay when lacking in the received ICMP message.

4.2. IPsec

The Internet network security standard, IPsec, incorporates IP-in-IP encapsulation as part of its tunnel mode of operation [RFC4301]. Although IP-in-IP packets can be secured via IPsec transport mode, resulting in identical packets [RFC3884], the rules affecting IPsec tunnel mode MTU discovery, fragmentation, and signaling mode are specified by IPsec, rather than IP-in-IP.

4.2.1. MTU discovery

Tunnel mode IPsec MTU discovery supports ICMP-based path MTU discovery (PMTUD), but only as a SHOULD. If an IPv4 packet arrives with DF=1, or an IPv6 packet arrives, and either is too large for the tunnel, the ingress SHOULD discard and send an ICMP to the source. If IPv4 and DF=0, the ingress SHOULD perform Outer Fragmentation, and SHOULD NOT send an ICMP to the source.

4.2.2. Fragmentation

IPsec performs only Outer Fragmentation; this distinguishes it from IP-in-IP, which performs only Inner Fragmentation.

It requires that implementations of tunnel mode allow the security policy to decide how the IPv4 DF bit should propagate from the inner to the outer header. It may be copied, cleared, or set, again, differing from IP-in-IP which allows only copy or set.

4.2.3. Signaling

IPsec, like IP-in-IP, relays ICMP "message to big" signals from the ingress back to the source. The size indicated is adjusted to take into account for the space for both encapsulation and security information. Further, it allows that any ICMP message may be blocked, on a per-security association basis; this filtering is for security reasons, but also can directly result in "black holing".

5. Issues

As has been shown in only two examples, even similar mechanisms for encapsulation can result in very different approaches to tunneling. Although these approaches result in different MTU discovery, fragmentation, and signaling mechanisms, they result from different architectural perspectives on the role of tunnels in the Internet. This section discusses these more fundamental perspectives, and their impact on the mechanisms.

5.1. Tunnel model

The Internet architecture is composed of hosts, gateways (i.e., routers), and links [Cl88]. A host is a source or sink of network packet traffic, a router redirects packets from one set of links to another, and links interconnect hosts and routers. Although originally described for the Internet's network layer, this architecture, with a bit of renaming (e.g., routers become bridges), applies equally well for link layers.

Tunnels could, in principle, be related to this basic model in one of three ways:

- Tunnel as a link
- Tunnel as a router/bridge
- Tunnel as invisible

Tunnels require distinct ingress and egress addresses, to use during encapsulation, and to direct encapsulated traffic from the ingress to the egress. As a result, a tunnel is most usefully considered a link in the architecture in which they are deployed. As a result, tunnel designers should consider and apply link design issues [RFC3819]. This also implies that operating systems designers should represent tunnels as links; this may be conveniently represented as virtual interfaces.

[this includes tunnel as point-point vs. tunnel as multipoint]

5.2. Parties participating

The description of a tunnel focuses on the functions of the ingress and egress, but not all functions need be located at one of these two points. Recall inner fragmentation, in which fragment reassembly occurs at the destination, not the egress - this imposes load on the destination as a result of behavior of the ingress.

Containing all tunnel functions solely inside the tunnel endpoints, as with outer fragmentation, is architecturally clean. It also obeys the 'clean up your own mess' principle; the impact of encapsulation and fragmentation caused by the ingress is then handled by the egress, without imposing load on the destination.

Distributing tunnel functions across both egress and destination, as with inner fragmentation, can be more efficient. The impact of the limited IPv4 IP ID space is more prominent in the outer header, due to aggregation of traffic at the ingress. Using the inner header for fragmentation allows use of a larger effective IP ID space because of the additional IP source/destination addresses present there. Reassembly can be distributed among a large number of destinations (where present), and the impact of reassembly can be isolated to only affected destinations. Further, fragmenting once at the ingress can avoid repeated fragmentation/reassembly steps when packets traverse multiple tunnels in succession.

The primary case in favor of distributed tunnel functions, and thus inner encapsulation is that high speed ingress devices can be implemented, but that corresponding high speed egresses are difficult or costly. Unfortunately, network operators cannot always know in advance that high-speed ingresses are being deployed where the destination traffic is sufficiently diffuse; deploying such a device where the traffic focuses on a single destination puts an undue burden on that destination.

6. Potential Ways Forward

There are a number of issues which may benefit from a coordinated review. These include unification of various tunneling standards, and revision of tunnel standards to address:

- Relation of inner/outer headers (i.e., which fields are copied, derived, etc.)
- MTU discovery
- Fragmentation
- Signaling

This revision may suggest the utility of a single, configurable tunnel mechanism that includes various solutions as alternatives, rather than developing custom tunnel solutions on-demand. It may also suggest the development of new solutions, such as:

- The use of PLPMTUD for tunnels
- Addressing the IP ID issue and fragmentation
- New ICMP signals
- Optimization solutions, such as packing

SEAL addresses a few of these issues, notably the first two [RFC5320]. It adds an active signal exchange between ingress and egress for intra-tunnel MTU discovery, and an extension to the IP ID space to detect collisions.

Tunnels are further evidence that the current requirements for IPv4 ID uniqueness may need revision. In particular, it is clear that even moderate speed transport connections already violate these requirements. We recommend revisiting the requirements as suggested in [To10].

Note that this document does not argue for a single, generic tunneling protocol or mechanism. Such a mechanism is no more likely to be useful than would a 'one size fits all' transport protocol. It does argue, however, for consistency in tunnel design, and abstraction and reuse of mechanism where possible.

7. Notes for future updates

[This area includes notes for future updates which have been reported but not yet fully included - it represents a holding area for comments, and should not appear in the final document.]

tunnel as virtualization - Stewart Bryant (SB)

tunnel as endpoint only, not on-path (not MPLS, e.g.) - JT/coauthor

gigE packing like PWE3 ATM packing - SB

PPP chopping and coalescing - MT/coauthor

end sec 2 "we need large seq num and to frag at the tunnel" / maybe, but do we want recommendations? - SB

security should add addr management and ACLs (?) - SB

MTU as part of BGP? - SB (Will this even work - JT)

section 2 it says: "The IPv6 fragment header is present only when a packet has been fragmented", but I know of at least one effort in MANET that is proposing to include the fragment header even for unfragmented IPv6 packets. That would seem to bend the rules set forth in RFC2460, but I just thought it might be worth pointing out that some people are considering bending them. - Fred Templin

NATs - i.e., One other thought; where the IP ID problem becomes truly pathological is for tunnels that traverse IPv4 NATs. First, the NATs could rewrite the ID to something the ingress tunnel endpoint never intended. Secondly, multiple ingress tunnel endpoints that traverse the same NAT could have IP ID "collisions" from the perspective of the outside world. This may deserve a section unto itself? - FT

NAT as half-tunnel - JT

tunnel endpoint as following host rules - JT (as with ECN in CAPWAP, per Magnus' email of 10/10/08)

the need for larger min MTU - FT (see SEAL)

describe relationship to [Ho08] - JT (as per INTAREA meeting notes, don't cover Teredo-specific issues in Ho08, but include generic issues here)

8. 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. Regarding ICMP signals, tunnels have similar security issues to routers, in that they SHOULD throttle ICMPs sent to a given source, and SHOULD send ICMPs that correspond to events inside the tunnel. Such ICMPs MUST have the tunnel ingress IP address as the source IP, because IP addresses inside a tunnel path may have no meaning outside the tunnel.

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 may result in these attacks being more difficult to detect.

Inner fragmentation can present an undue burden on destinations where traffic is not sufficiently diffuse; tunnels SHOULD NOT employ inner fragmentation except where such diffusion is confirmed either by the tunnel mechanism or network designer. All tunnel fragmentation - inner and outer - MUST obey all existing fragmentation requirements, i.e., IPv6 tunnels MUST NOT employ inner fragmentation, and IPv4 tunnels MUST NOT use inner fragmentation where the inner header DF=1.

Tunnels MUST obey all existing IP requirements, such as the uniqueness of the IP ID field, until otherwise exceptioned or revoked. Failure to either limit encapsulation traffic, or use additional ingress/egress IP addresses, can result in high speed traffic fragments being incorrectly reassembled.

9. IANA Considerations

This document has no IANA considerations.

The RFC Editor should remove this section prior to publication.

10. References

10.1. Normative References

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

10.2. 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.

[Fa10] Farinacci, D., V. Fuller, D. Meyer, D. Lewis, "Locator/ID Separation Protocol (LISP)," (work in progress), draft-ietf-lisp-06, Jan. 2010.

[Ho08] Hoagland, J., S. Krishnan, D. Thaler, "Security Concerns With IP Tunneling," (work in progress), draft-ietf-v6ops-tunnel-security-concerns-01, Oct. 2008.

[Pe10] Perlman, R., D. Eastlake, D. Dutt, S. Gai, A. Ghanwani, "RBridges: Base Protocol Specification," (work in progress), trill draft-ietf-trill-rbridge-protocol-15, Jan. 2010.

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

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

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

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

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

[RFC3344] Perkins, C., Ed., "IP Mobility Support for IPv4," RFC 3344, August 2002.

[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.

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

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

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

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

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

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

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

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

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

[To10] Touch, J., "Updated Specification of the IPv4 ID Field," (work in progress), draft-touch-intarea-ipv4-id-update, Feb. 2010.

11. Acknowledgments

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

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

Authors' Addresses

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

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

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

Email: townsley@cisco.com