Network Working Group
Request for Comments: 2473
Category: Standards Track

A. Conta
Lucent Technologies Inc.
S. Deering
Cisco Systems

December 1998

Generic Packet Tunneling in IPv6 Specification

Status of this Memo

Copyright Notice

Abstract

This document defines the model and generic mechanisms for IPv6 encapsulation of Internet packets, such as IPv6 and IPv4. The model and mechanisms can be applied to other protocol packets as well, such as AppleTalk, IPX, CLNP, or others.

Table of Contents

1. Introduction

This document specifies a method and generic mechanisms by which a packet is encapsulated and carried as payload within an IPv6 packet. The resulting packet is called an IPv6 tunnel packet. The forwarding path between the source and destination of the tunnel packet is called an IPv6 tunnel. The technique is called IPv6 tunneling.

A typical scenario for IPv6 tunneling is the case in which an intermediate node exerts explicit routing control by specifying particular forwarding paths for selected packets. This control is achieved by prepending IPv6 headers to each of the selected original packets. These prepended headers identify the forwarding paths.

In addition to the description of generic IPv6 tunneling mechanisms, which is the focus of this document, specific mechanisms for tunneling IPv6 and IPv4 packets are also described herein.

The keywords MUST, MUST NOT, MAY, OPTIONAL, REQUIRED, RECOMMENDED, SHALL, SHALL NOT, SHOULD, SHOULD NOT are to be interpreted as defined in RFC 2119.

2. Terminology

original packet

a packet that undergoes encapsulation.

original header

the header of an original packet.

tunnel

a forwarding path between two nodes on which the payloads of packets are original packets.

tunnel end-node

a node where a tunnel begins or ends.

tunnel header

the header prepended to the original packet during encapsulation. It specifies the tunnel end-points as source and destination.

tunnel packet

a packet that encapsulates an original packet.

tunnel entry-point

the tunnel end-node where an original packet is encapsulated.

tunnel exit-point

the tunnel end-node where a tunnel packet is decapsulated.

IPv6 tunnel

a tunnel configured as a virtual link between two IPv6 nodes, on which the encapsulating protocol is IPv6.

tunnel MTU

the maximum size of a tunnel packet payload without requiring fragmentation, that is, the Path MTU between the tunnel entry-point and the tunnel exit-point nodes minus the size of the tunnel header.

tunnel hop limit

the maximum number of hops that a tunnel packet can travel from the tunnel entry-point to the tunnel exit-point.

inner tunnel

a tunnel that is a hop (virtual link) of another tunnel.

outer tunnel

a tunnel containing one or more inner tunnels.

nested tunnel packet

a tunnel packet that has as payload a tunnel packet.

nested tunnel header

the tunnel header of a nested tunnel packet.

nested encapsulation

encapsulation of an encapsulated packet.

recursive encapsulation

encapsulation of a packet that reenters a tunnel before exiting it.

tunnel encapsulation limit

the maximum number of nested encapsulations of a packet.

3. IPv6 Tunneling

IPv6 tunneling is a technique for establishing a "virtual link" between two IPv6 nodes for transmitting data packets as payloads of IPv6 packets (see Fig.1). From the point of view of the two nodes, this "virtual link", called an IPv6 tunnel, appears as a point to point link on which IPv6 acts like a link-layer protocol. The two IPv6 nodes play specific roles. One node encapsulates original packets received from other nodes or from itself and forwards the resulting tunnel packets through the tunnel. The other node decapsulates the received tunnel packets and forwards the resulting original packets towards their destinations, possibly itself. The encapsulator node is called the tunnel entry-point node, and it is the source of the tunnel packets. The decapsulator node is called the tunnel exit-point, and it is the destination of the tunnel packets.

Note:
This document refers in particular to tunnels between two nodes identified by unicast addresses - such tunnels look like "virtual point to point links". The mechanisms described herein apply also to tunnels in which the exit-point nodes are identified by other types of addresses, such as anycast or multicast. These tunnels may look like "virtual point to multipoint links". At the time of writing this document, IPv6 anycast addresses are a subject of ongoing specification and experimental work.

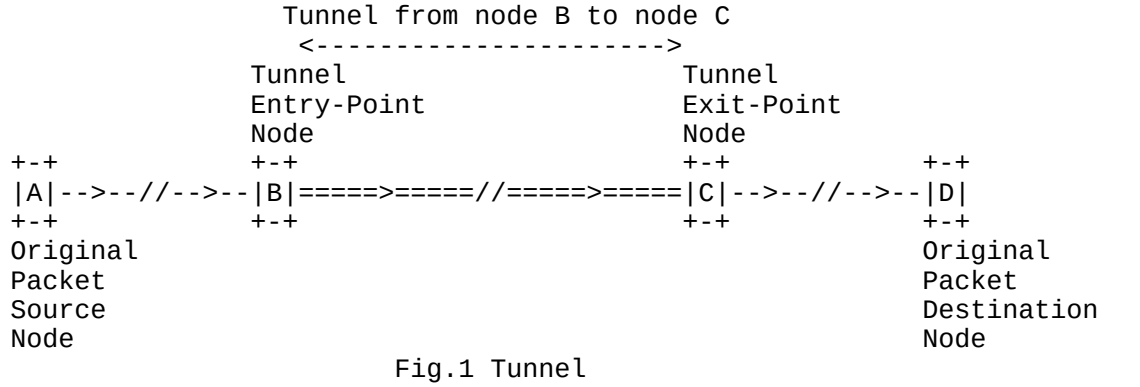


Fig.1 Tunnel

An IPv6 tunnel is a unidirectional mechanism - tunnel packet flow takes place in one direction between the IPv6 tunnel entry-point and exit-point nodes (see Fig.1).

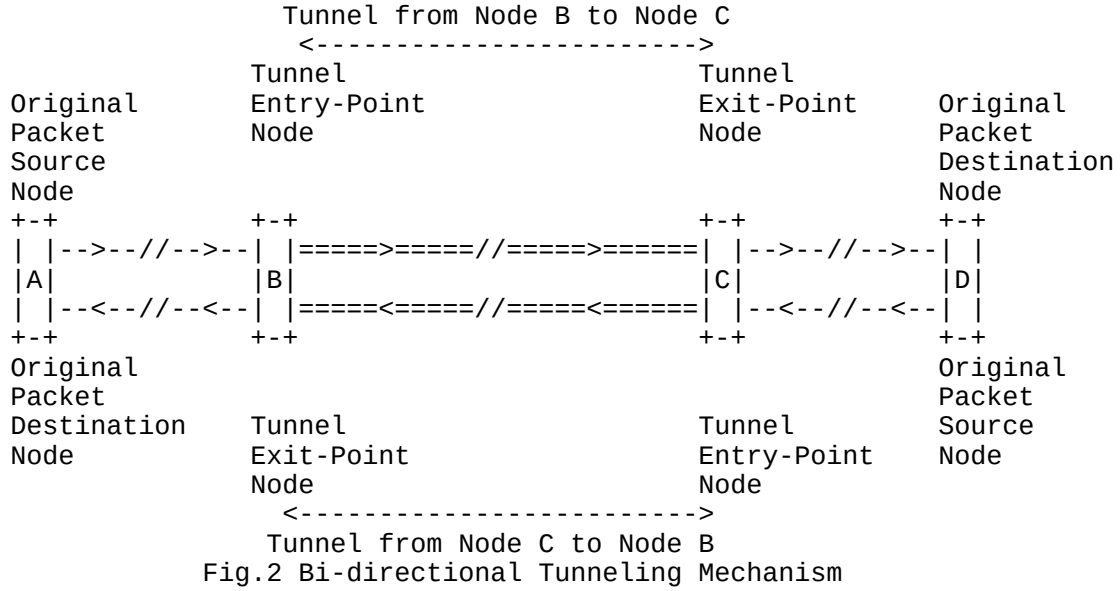


Fig.2 Bi-directional Tunneling Mechanism

Bi-directional tunneling is achieved by merging two unidirectional mechanisms, that is, configuring two tunnels, each in opposite direction to the other - the entry-point node of one tunnel is the exit-point node of the other tunnel (see Fig.2).

3.1 IPv6 Encapsulation

IPv6 encapsulation consists of prepending to the original packet an IPv6 header and, optionally, a set of IPv6 extension headers (see Fig.3), which are collectively called tunnel IPv6 headers. The encapsulation takes place in an IPv6 tunnel entry-point node, as the result of an original packet being forwarded onto the virtual link represented by the tunnel. The original packet is processed during forwarding according to the forwarding rules of the protocol of that packet. For instance if the original packet is an:

(a) IPv6 packet, the IPv6 original header hop limit is decremented by one.

(b) IPv4 packet, the IPv4 original header time to live field (TTL) is decremented by one.

At encapsulation, the source field of the tunnel IPv6 header is filled with an IPv6 address of the tunnel entry-point node, and the destination field with an IPv6 address of the tunnel exit-point. Subsequently, the tunnel packet resulting from encapsulation is sent towards the tunnel exit-point node.

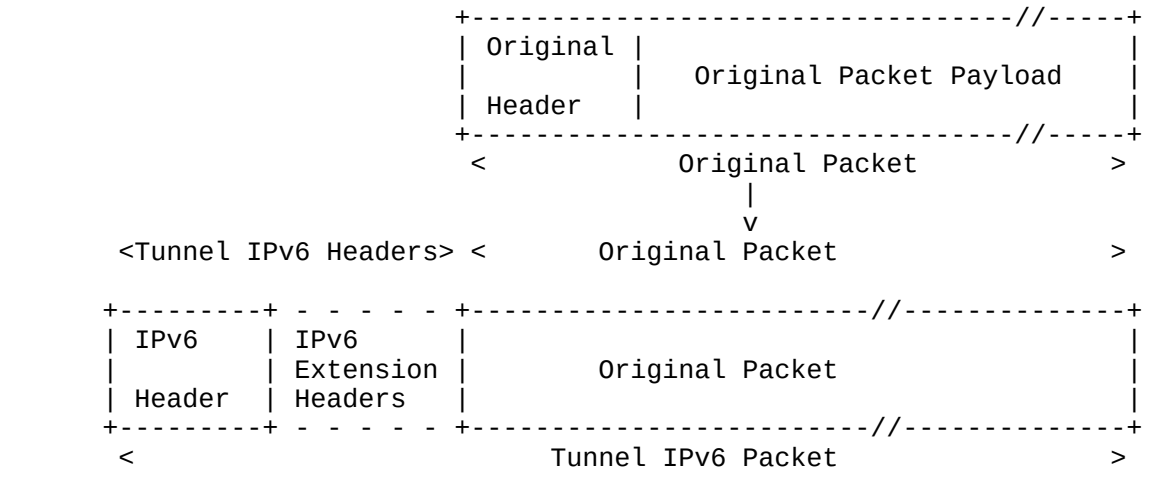


Fig.3 Encapsulating a Packet

Tunnel extension headers should appear in the order recommended by the specifications that define the extension headers, such as [IPv6-Spec].

A source of original packets and a tunnel entry-point that encapsulates those packets can be the same node.

3.2 Packet Processing in Tunnels

The intermediate nodes in the tunnel process the IPv6 tunnel packets according to the IPv6 protocol. For example, a tunnel Hop by Hop Options extension header is processed by each receiving node in the tunnel; a tunnel Routing extension header identifies the intermediate processing nodes, and controls at a finer granularity the forwarding path of the tunnel packet through the tunnel; a tunnel Destination Options extension header is processed at the tunnel exit-point node.

3.3 IPv6 Decapsulation

Decapsulation is graphically shown in Fig.4:

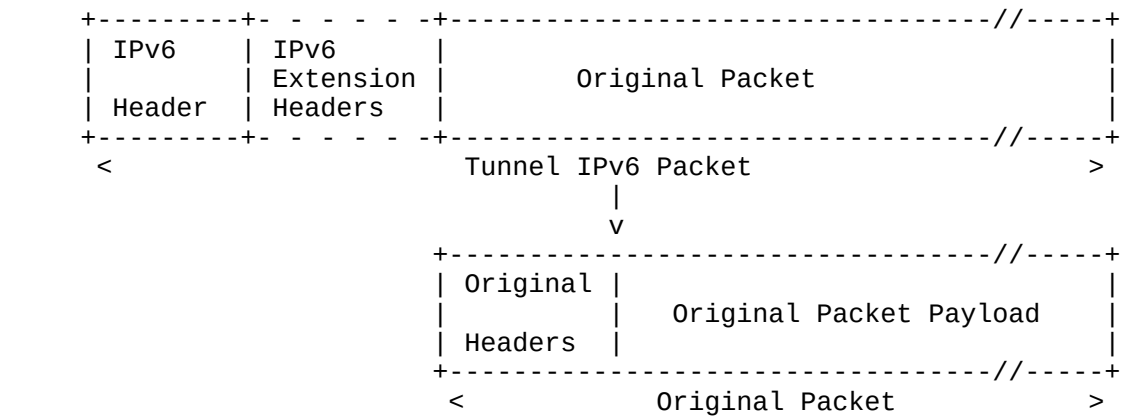


Fig.4 Decapsulating a Packet

Upon receiving an IPv6 packet destined to an IPv6 address of a tunnel exit-point node, its IPv6 protocol layer processes the tunnel headers. The strict left-to-right processing rules for extension headers is applied. When processing is complete, control is handed to the next protocol engine, which is identified by the Next Header field value in the last header processed. If this is set to a tunnel protocol value, the tunnel protocol engine discards the tunnel headers and passes the resulting original packet to the Internet or lower layer protocol identified by that value for further processing.

For example, in the case the Next Header field has the IPv6 Tunnel Protocol value, the resulting original packet is passed to the IPv6 protocol layer.

The tunnel exit-point node, which decapsulates the tunnel packets, and the destination node, which receives the resulting original packets can be the same node.

3.4 IPv6 Tunnel Protocol Engine

Packet flow (paths #1-7) through the IPv6 Tunnel Protocol Engine on a node is graphically shown in Fig.5:

Note:

In Fig.5, the Upper-Layer Protocols box represents transport protocols such as TCP, UDP, control protocols such as ICMP, routing protocols such as OSPF, and internet or lower-layer protocol being "tunneled" over IPv6, such as IPv4, IPX, etc. The Link-Layer Protocols box represents Ethernet, Token Ring, FDDI, PPP, X.25, Frame Relay, ATM, etc..., as well as internet layer "tunnels" such as IPv4 tunnels.

The IPv6 tunnel protocol engine acts as both an "upper-layer" and a "link-layer", each with a specific input and output as follows:

(u.i) "tunnel upper-layer input" - consists of tunnel IPv6 packets that are going to be decapsulated. The tunnel packets are incoming through the IPv6 layer from:

(u.i.1) a link-layer - (path #1, Fig.5)

These are tunnel packets destined to this node and will undergo decapsulation.

(u.i.2) a tunnel link-layer - (path #7, Fig.5)

These are tunnel packets that underwent one or more decapsulations on this node, that is, the packets had one or more nested tunnel headers and one nested tunnel header was just discarded. This node is the exit-point of both an outer tunnel and one or more of its inner tunnels.

For both above cases the resulting original packets are passed back to the IPv6 layer as "tunnel link-layer" output for further processing (see b.2).

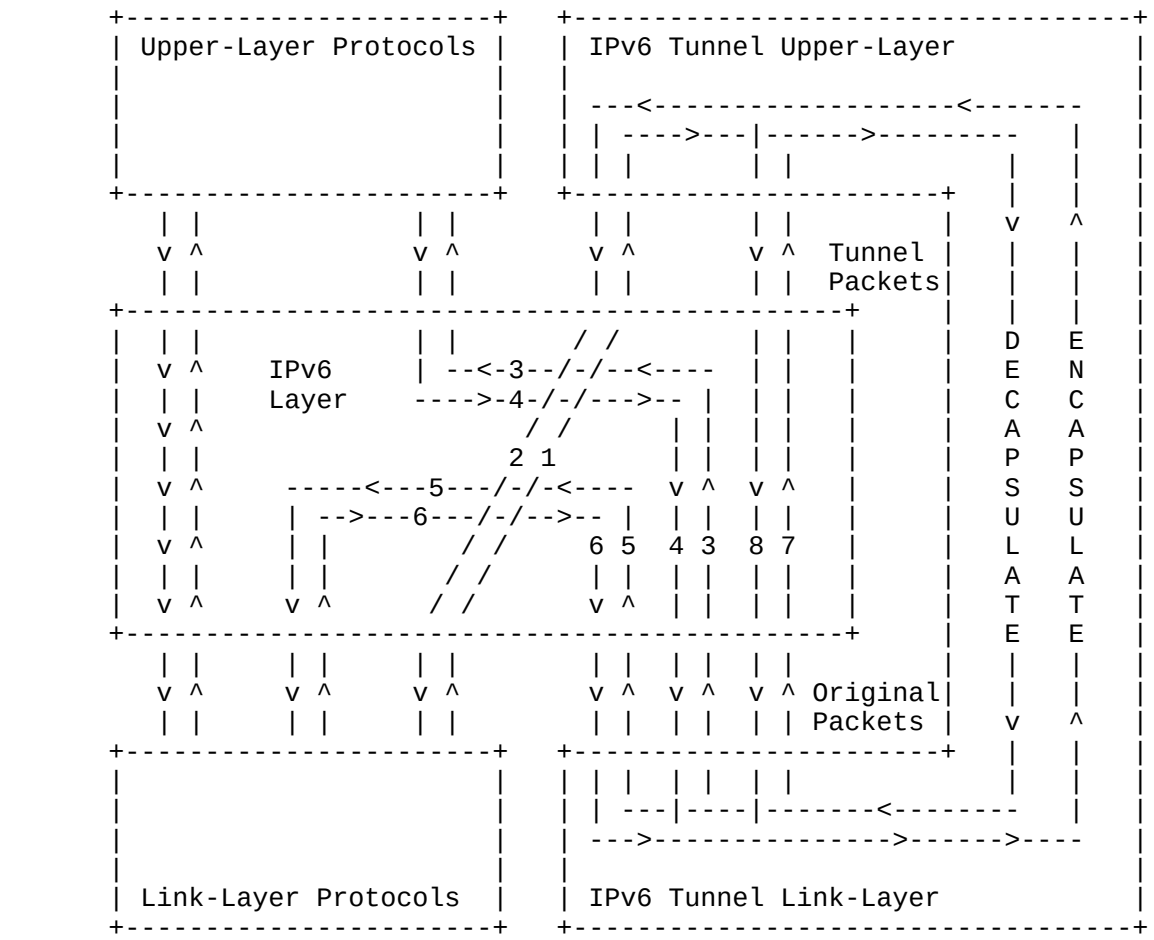


Fig.5 Packet Flow in the IPv6 Tunneling Protocol Engine on a Node

(u.o) "tunnel upper-layer output" - consists of tunnel IPv6 packets that are passed through the IPv6 layer down to:

(u.o.1) a link-layer - (path #2, Fig.5)

These packets underwent encapsulation and are sent towards the tunnel exit-point

(u.o.2) a tunnel link-layer - (path #8, Fig.5)

These tunnel packets undergo nested encapsulation. This node is the entry-point node of both an outer tunnel and one or more of its inner tunnel.

Implementation Note:

The tunnel upper-layer input and output can be implemented similar to the input and output of the other upper-layer protocols.

The tunnel link-layer input and output are as follows:

(l.i) "tunnel link-layer input" - consists of original IPv6 packets that are going to be encapsulated.

The original packets are incoming through the IPv6 layer from:

(l.i.1) an upper-layer - (path #4, Fig.5)

These are original packets originating on this node that undergo encapsulation. The original packet source and tunnel entry-point are the same node.

(l.i.2) a link-layer - (path #6, Fig.5)

These are original packets incoming from a different node that undergo encapsulation on this tunnel entry-point node.

(l.i.3) a tunnel upper-layer - (path #8, Fig.5)

These packets are tunnel packets that undergo nested encapsulation. This node is the entry-point node of both an outer tunnel and one or more of its inner tunnels.

The resulting tunnel packets are passed as tunnel upper-layer output packets through the IPv6 layer (see u.o) down to:

(l.o) "tunnel link-layer output" - consists of original IPv6 packets resulting from decapsulation. These packets are passed through the IPv6 layer to:

(l.o.1) an upper-layer - (path #3, Fig.5)

These original packets are destined to this node.

(l.o.2) a link-layer - (path #5, Fig.5)

These original packets are destined to another node; they are transmitted on a link towards their destination.

(l.o.3) a tunnel upper-layer - (path #7, Fig.5)

These packets undergo another decapsulation; they were nested tunnel packets. This node is both the exit-point node of an outer tunnel and one or more inner tunnels.

Implementation Note:

The tunnel link-layer input and output can be implemented similar to the input and output of other link-layer protocols, for instance, associating an interface or pseudo-interface with the IPv6 tunnel.

The selection of the "IPv6 tunnel link" over other links results from the packet forwarding decision taken based on the content of the node's routing table.

4. Nested Encapsulation

Nested IPv6 encapsulation is the encapsulation of a tunnel packet. It takes place when a hop of an IPv6 tunnel is a tunnel. The tunnel containing a tunnel is called an outer tunnel. The tunnel contained in the outer tunnel is called an inner tunnel - see Fig.6. Inner tunnels and their outer tunnels are nested tunnels.

The entry-point node of an "inner IPv6 tunnel" receives tunnel IPv6 packets encapsulated by the "outer IPv6 tunnel" entry-point node. The "inner tunnel entry-point node" treats the receiving tunnel packets as original packets and performs encapsulation. The resulting packets are "tunnel packets" for the "inner IPv6 tunnel", and "nested tunnel packets" for the "outer IPv6 tunnel".

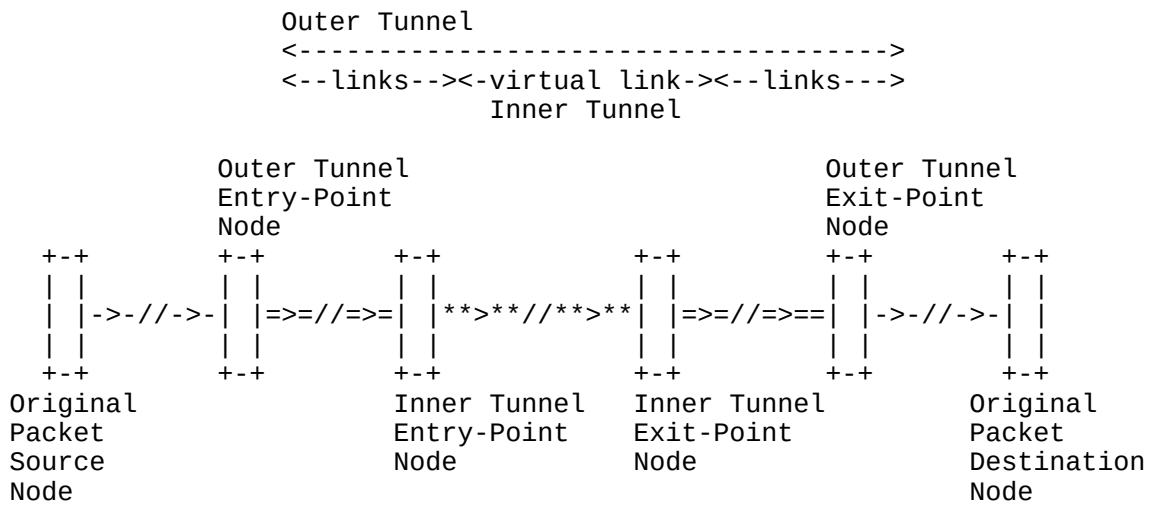


Fig.6. Nested Encapsulation

4.1 Limiting Nested Encapsulation

A tunnel IPv6 packet is limited to the maximum IPv6 packet size [IPv6-Spec]. Each encapsulation adds to the size of an encapsulated packet the size of the tunnel IPv6 headers. Consequently, the number of tunnel headers, and therefore, the number of nested encapsulations is limited by the maximum packet size. However this limit is so large (more than 1600 encapsulations for an original packet of minimum size) that it is not an effective limit in most cases.

The increase in the size of a tunnel IPv6 packet due to nested encapsulations may require fragmentation [IPv6-Spec] at a tunnel entry point - see section 7. Furthermore, each fragmentation, due to nested encapsulation, of an already fragmented tunnel packet results in a doubling of the number of fragments. Moreover, it is probable that once this fragmentation begins, each new nested encapsulation results in yet additional fragmentation. Therefore limiting nested encapsulation is recommended.

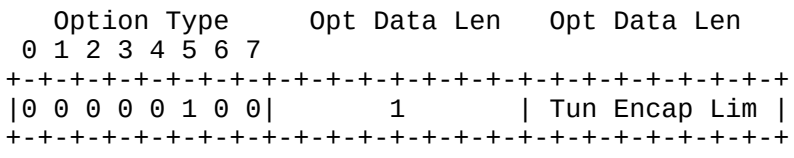
The proposed mechanism for limiting excessive nested encapsulation is a "Tunnel Encapsulation Limit" option, which is carried in an IPv6 Destination Options extension header accompanying an encapsulating IPv6 header.

4.1.1 Tunnel Encapsulation Limit Option

A tunnel entry-point node may be configured to include a Tunnel Encapsulation Limit option as part of the information prepended to all packets entering a tunnel at that node. The Tunnel Encapsulaton Limit option is carried in a Destination Options extension header [IPv6-Spec] placed between the encapsulating IPv6 header and the IPv6 header of the original packet. (Other IPv6 extension headers may also be present preceding or following the Destination Options extension header, depending on configuration information at the tunnel entry-point node.)

The Tunnel Encapsulation Limit option specifies how many additional levels of encapsulation are permitted to be prepended to the packet -- or, in other words, how many further levels of nesting the packet is permitted to undergo -- not counting the encapsulation in which the option itself is contained. For example, a Tunnel Encapsulation Limit option containing a limit value of zero means that a packet carrying that option may not enter another tunnel before exiting the current tunnel.

The Tunnel Encapsulation Limit option has the following format:



Option Type decimal value 4

- the highest-order two bits - set to 00 - indicate "skip over this option if the option is not recognized".

- the third-highest-order bit - set to 0 - indicates that the option data in this option does not change en route to the packet's destination [IPv6-Spec].

Opt Data Len value 1 - the data portion of the Option is one octet long.

Opt Data Value the Tunnel Encapsulation Limit value - 8-bit unsigned integer specifying how many further levels of encapsulation are permitted for the

Tunnel Encapsulation Limit options are of interest only to tunnel entry points. A tunnel entry-point node is required to execute the following procedure for every packet entering a tunnel at that node:

(a) Examine the packet to see if a Tunnel Encapsulation Limit option is present following its IPv6 header. The headers following the IPv6 header must be examined in strict "left-to-right" order, with the examination stopping as soon as any one of the following headers is encountered: (i) a Destination Options extension header containing a Tunnel Encapsulation Limit, (ii) another IPv6 header, (iii) a non-extension header, such as TCP, UDP, or ICMP, or (iv) a header that cannot be parsed because it is encrypted or its type is unknown. (Note that this requirment is an exception to the general IPv6 rule that a Destination Options extension header need only be examined by a packet's destination node. An alternative and "cleaner" approach would have been to use a Hop-by-Hop extension header for this purpose, but that would have imposed an undesirable extra processing burden, and possible consequent extra delay, at every IPv6 node along the path of a tunnel.)

(b) If a Tunnel Encapsulation Limit option is found in the packet entering the tunnel and its limit value is zero, the packet is discarded and an ICMP Parameter Problem message [ICMP-Spec] is sent to the source of the packet, which is the previous tunnel entry-point node. The Code field of the Parameter Problem message is set to zero ("erroneous header field encountered") and the Pointer field is set to point to the third octet of the Tunnel Encapsulation Limit option (i.e., the octet containing the limit value of zero).

(c) If a Tunnel Encapsulation Limit option is found in the packet entering the tunnel and its limit value is non-zero, an additional Tunnel Encapsulation Limit option must be included as part of the encapsulating headers being added at this entry point. The limit value in the encapsulating option is set to one less than the limit value found in the packet being encapsulated.

(d) If a Tunnel Encapsulation Limit option is not found in the packet entering the tunnel and if an encapsulation limit has been configured for this tunnel, a Tunnel Encapsulation Limit option must be included as part of the encapsulating headers being added at this entry point. The limit value in the option is set to the configured limit.

(e) If a Tunnel Encapsulation Limit option is not found in the packet entering the tunnel and if no encapsulation limit has been configured for this tunnel, then no Tunnel Encapsulation Limit option is included as part of the encapsulating headers being added at this entry point.

A Tunnel Encapsulation Limit option added at a tunnel entry-point node is removed as part of the decapsulation process at that tunnel's exit-point node.

Two cases of encapsulation that should be avoided are described below:

4.1.2 Loopback Encapsulation

A particular case of encapsulation which must be avoided is the loopback encapsulation. Loopback encapsulation takes place when a tunnel IPv6 entry-point node encapsulates tunnel IPv6 packets originated from itself, and destined to itself. This can generate an infinite processing loop in the entry-point node.

To avoid such a case, it is recommended that an implementation have a mechanism that checks and rejects the configuration of a tunnel in which both the entry-point and exit-point node addresses belong to the same node. It is also recommended that the encapsulating engine check for and reject the encapsulation of a packet that has the pair of tunnel entry-point and exit-point addresses identical with the pair of original packet source and final destination addresses.

4.1.3 Routing-Loop Nested Encapsulation

In the case of a forwarding path with multiple-level nested tunnels, a routing-loop from an inner tunnel to an outer tunnel is particularly dangerous when packets from the inner tunnels reenter an outer tunnel from which they have not yet exited. In such a case, the nested encapsulation becomes a recursive encapsulation with the negative effects described in 4.1. Because each nested encapsulation adds a tunnel header with a new hop limit value, the IPv6 hop limit mechanism cannot control the number of times the packet reaches the outer tunnel entry-point node, and thus cannot control the number of recursive encapsulations.

When the path of a packet from source to final destination includes tunnels, the maximum number of hops that the packet can traverse should be controlled by two mechanisms used together to avoid the negative effects of recursive encapsulation in routing loops:

(a) the original packet hop limit.

It is decremented at each forwarding operation performed on an original packet. This includes each encapsulation of the original packet. It does not include nested encapsulations of the original packet

(b) the tunnel IPv6 packet encapsulation limit.

It is decremented at each nested encapsulation of the packet.

For a discussion of the excessive encapsulation risk factors in nested encapsulation see Appendix A.

5. Tunnel IPv6 Header

The tunnel entry-point node fills out a tunnel IPv6 main header [IPv6-Spec] as follows:

Version:

value 6

Traffic Class:

Depending on the entry-point node tunnel configuration, the traffic class can be set to that of either the original packet or a pre-configured value - see section 6.4.

Flow Label:

Depending on the entry-point node tunnel configuration, the flow label can be set to a pre-configured value. The typical value is zero - see section 6.5.

Payload Length:

The original packet length, plus the length of the encapsulating (prepended) IPv6 extension headers, if any.

Next Header:

The next header value according to [IPv6-Spec] from the Assigned Numbers RFC [RFC-1700 or its successors].

For example, if the original packet is an IPv6 packet, this is set to:

- decimal value 41 (Assigned Next Header number for IPv6) - if there are no tunnel extension headers.

- value 0 (Assigned Next Header number for IPv6 Hop by Hop Options extension header) - if a hop by hop options extension header immediately follows the tunnel IPv6 header.

- decimal value 60 (Assigned Next Header number for IPv6 Destination Options extension header) - if a destination options extension header immediately follows the tunnel IPv6 header.

Hop Limit:

The tunnel IPv6 header hop limit is set to a pre-configured value - see section 6.3.

The default value for hosts is the Neighbor Discovery advertised hop limit [ND-Spec]. The default value for routers is the default IPv6 Hop Limit value from the Assigned Numbers RFC (64 at the time of writing this document).

Source Address:

An IPv6 address of the outgoing interface of the tunnel entry-point node. This address is configured as the tunnel entry-point node address - see section 6.1.

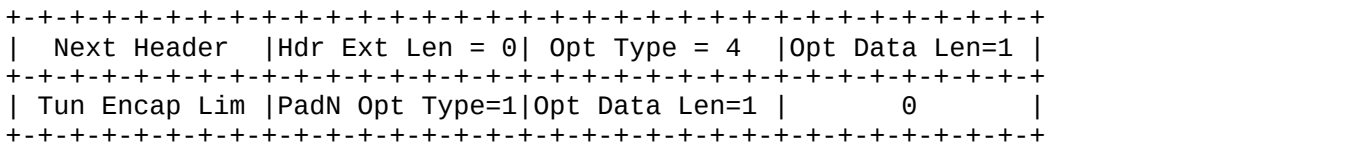
Destination Address:

An IPv6 address of the tunnel exit-point node. This address is configured as the tunnel exit-point node address - see section 6.2.

5.1 Tunnel IPv6 Extension Headers

Depending on IPv6 node configuration parameters, a tunnel entry-point node may append to the tunnel IPv6 main header one or more IPv6 extension headers, such as a Hop-by-Hop Options header, a Routing header, or others.

To limit the number of nested encapsulations of a packet, if it was configured to do so - see section 6.6 - a tunnel entry-point includes a Destination Options extension header containing a Tunnel Encapsulation Limit option. If that option is the only option present in the Destination Options header, the header has the following format:



Next Header:

Identifies the type of the original packet header. For example, if the original packet is an IPv6 packet, the next header protocol value is set to decimal value 41 (Assigned payload type number for IPv6).

Hdr Ext Len:

Length of the Destination Options extension header in 8-octet units, not including the first 8 octets. Set to value 0, if no other options are present in this destination options header.

Option Type:

value 4 - see section 4.1.1.

Opt Data Len:

value 1 - see section 4.1.1.

Tun Encap Lim:

8 bit unsigned integer - see section 4.1.1.

Option Type:

value 1 - PadN option, to align the header following this header.

Opt Data Len:

value 1 - one octet of option data.

Option Data:

value 0 - one zero-valued octet.

6. IPv6 Tunnel State Variables

The IPv6 tunnel state variables, some of which are or may be configured on the tunnel entry-point node, are:

6.1 IPv6 Tunnel Entry-Point Node Address

The tunnel entry-point node address is one of the valid IPv6 unicast addresses of the entry-point node - the validation of the address at tunnel configuration time is recommended.

The tunnel entry-point node address is copied to the source address field in the tunnel IPv6 header during packet encapsulation.

6.2 IPv6 Tunnel Exit-Point Node Address

The tunnel exit-point node address is used as IPv6 destination address for the tunnel IPv6 header. A tunnel acts like a virtual point to point link between the entry-point node and exit-point node.

The tunnel exit-point node address is copied to the destination address field in the tunnel IPv6 header during packet encapsulation.

The configuration of the tunnel entry-point and exit-point addresses is not subject to IPv6 Autoconfiguration or IPv6 Neighbor Discovery.

6.3 IPv6 Tunnel Hop Limit

An IPv6 tunnel is modeled as a "single-hop virtual link" tunnel, in which the passing of the original packet through the tunnel is like the passing of the original packet over a one hop link, regardless of the number of hops in the IPv6 tunnel.

The "single-hop" mechanism should be implemented by having the tunnel entry point node set a tunnel IPv6 header hop limit independently of the hop limit of the original header.

The "single-hop" mechanism hides from the original IPv6 packets the number of IPv6 hops of the tunnel.

It is recommended that the tunnel hop limit be configured with a value that ensures:

(a) that tunnel IPv6 packets can reach the tunnel exit-point node

(b) a quick expiration of the tunnel packet if a routing loop occurs within the IPv6 tunnel.

The tunnel hop limit default value for hosts is the IPv6 Neighbor Discovery advertised hop limit [ND-Spec]. The tunnel hop limit default value for routers is the default IPv6 Hop Limit value from the Assigned Numbers RFC (64 at the time of writing this document).

The tunnel hop limit is copied into the hop limit field of the tunnel IPv6 header of each packet encapsulated by the tunnel entry-point node.

6.4 IPv6 Tunnel Packet Traffic Class

The IPv6 Tunnel Packet Traffic Class indicates the value that a tunnel entry-point node sets in the Traffic Class field of a tunnel header. The default value is zero. The configured Packet Traffic Class can also indicate whether the value of the Traffic Class field in the tunnel header is copied from the original header, or it is set to the pre-configured value.

6.5 IPv6 Tunnel Flow Label

The IPv6 Tunnel Flow Label indicates the value that a tunnel entry-point node sets in the flow label of a tunnel header. The default value is zero.

6.6 IPv6 Tunnel Encapsulation Limit

The Tunnel Encapsulation Limit value can indicate whether the entry-point node is configured to limit the number of encapsulations of tunnel packets originating on that node. The IPv6 Tunnel Encapsulation Limit is the maximum number of additional encapsulations permitted for packets undergoing encapsulation at that entry-point node. Recommended default value is 4. An entry-point node configured to limit the number of nested encapsulations prepends a Destination Options extension header containing a Tunnel Encapsulation Limit option to an original packet undergoing encapsulation - see sections 4.1 and 4.1.1.

6.7 IPv6 Tunnel MTU

The tunnel MTU is set dynamically to the Path MTU between the tunnel entry-point and the tunnel exit-point nodes, minus the size of the tunnel headers: the maximum size of a tunnel packet payload that can

be sent through the tunnel without fragmentation [IPv6-Spec]. The tunnel entry-point node performs Path MTU discovery on the path between the tunnel entry-point and exit-point nodes [PMTU-Spec], [ICMP-Spec]. The tunnel MTU of a nested tunnel is the tunnel MTU of the outer tunnel minus the size of the nested tunnel headers.

7. IPv6 Tunnel Packet Size Issues

Prepending a tunnel header increases the size of a packet, therefore a tunnel packet resulting from the encapsulation of an IPv6 original packet may require fragmentation.

A tunnel IPv6 packet resulting from the encapsulation of an original packet is considered an IPv6 packet originating from the tunnel entry-point node. Therefore, like any source of an IPv6 packet, a tunnel entry-point node must support fragmentation of tunnel IPv6 packets.

A tunnel intermediate node that forwards a tunnel packet to another node in the tunnel follows the general IPv6 rule that it must not fragment a packet undergoing forwarding.

A tunnel exit-point node receiving tunnel packets at the end of the tunnel for decapsulation applies the strict left-to-right processing rules for extension headers. In the case of a fragmented tunnel packet, the fragments are reassembled into a complete tunnel packet before determining that an embedded packet is present.

Note:

A particular problem arises when the destination of a fragmented tunnel packet is an exit-point node identified by an anycast address. The problem, which is similar to that of original fragmented IPv6 packets destined to nodes identified by an anycast address, is that all the fragments of a packet must arrive at the same destination node for that node to be able to perform a successful reassembly, a requirement that is not necessarily satisfied by packets sent to an anycast address.

7.1 IPv6 Tunnel Packet Fragmentation

When an IPv6 original packet enters a tunnel, if the original packet size exceeds the tunnel MTU (i.e., the Path MTU between the tunnel entry-point and the tunnel exit-point, minus the size of the tunnel header(s)), it is handled as follows:

(a) if the original IPv6 packet size is larger than the IPv6 minimum link MTU [IPv6-Spec], the entry-point node discards the packet and sends an ICMPv6 "Packet Too Big" message to the source address of the original packet with the recommended MTU size field set to the tunnel MTU or the IPv6 minimum link MTU, whichever is larger, i.e. max (tunnel MTU, IPv6 minimum link MTU). Also see sections 6.7 and 8.2.

(b) if the original IPv6 packet is equal or smaller than the IPv6 minimum link MTU, the tunnel entry-point node encapsulates the original packet, and subsequently fragments the resulting IPv6 tunnel packet into IPv6 fragments that do not exceed the Path MTU to the tunnel exit-point.

7.2 IPv4 Tunnel Packet Fragmentation

When an IPv4 original packet enters a tunnel, if the original packet size exceeds the tunnel MTU (i.e., the Path MTU between the tunnel entry-point and the tunnel exit-point, minus the size of the tunnel header(s)), it is handled as follows:

(a) if in the original IPv4 packet header the Don’t Fragment - DF - bit flag is SET, the entry-point node discards the packet and returns an ICMP message. The ICMP message has the type = "unreachable", the code = "packet too big", and the recommended MTU size field set to the size of the tunnel MTU - see sections 6.7 and 8.3.

(b) if in the original packet header the Don’t Fragment - DF - bit flag is CLEAR, the tunnel entry-point node encapsulates the original packet, and subsequently fragments the resulting IPv6 tunnel packet into IPv6 fragments that do not exceed the Path MTU to the tunnel exit-point.

8. IPv6 Tunnel Error Processing and Reporting

IPv6 Tunneling follows the general rule that an error detected during the processing of an IPv6 packet is reported through an ICMP message to the source of the packet.

On a forwarding path that includes IPv6 tunnels, an error detected by a node that is not in any tunnel is directly reported to the source of the original IPv6 packet.

An error detected by a node inside a tunnel is reported to the source of the tunnel packet, that is, the tunnel entry-point node. The ICMP message sent to the tunnel entry-point node has as ICMP payload the tunnel IPv6 packet that has the original packet as its payload.

The cause of a packet error encountered inside a tunnel can be a problem with:

(a) the tunnel header, or

(b) the tunnel packet.

Both tunnel header and tunnel packet problems are reported to the tunnel entry-point node.

If a tunnel packet problem is a consequence of a problem with the original packet, which is the payload of the tunnel packet, then the problem is also reported to the source of the original packet.

To report a problem detected inside the tunnel to the source of an original packet, the tunnel entry point node must relay the ICMP message received from inside the tunnel to the source of that original IPv6 packet.

An example of the processing that can take place in the error reporting mechanism of a node is illustrated in Fig.7, and Fig.8:

Fig.7 path #0 and Fig.8 (a) - The IPv6 tunnel entry-point receives an ICMP packet from inside the tunnel, marked Tunnel ICMPv6 Message in Fig.7. The tunnel entry-point node IPv6 layer passes the received ICMP message to the ICMPv6 Input. The ICMPv6 Input, based on the ICMP type and code [ICMP-Spec] generates an internal "error code".

Fig.7 path #1 - The internal error code, is passed with the "ICMPv6 message payload" to the upper-layer protocol - in this case the IPv6 tunnel upper-layer error input.

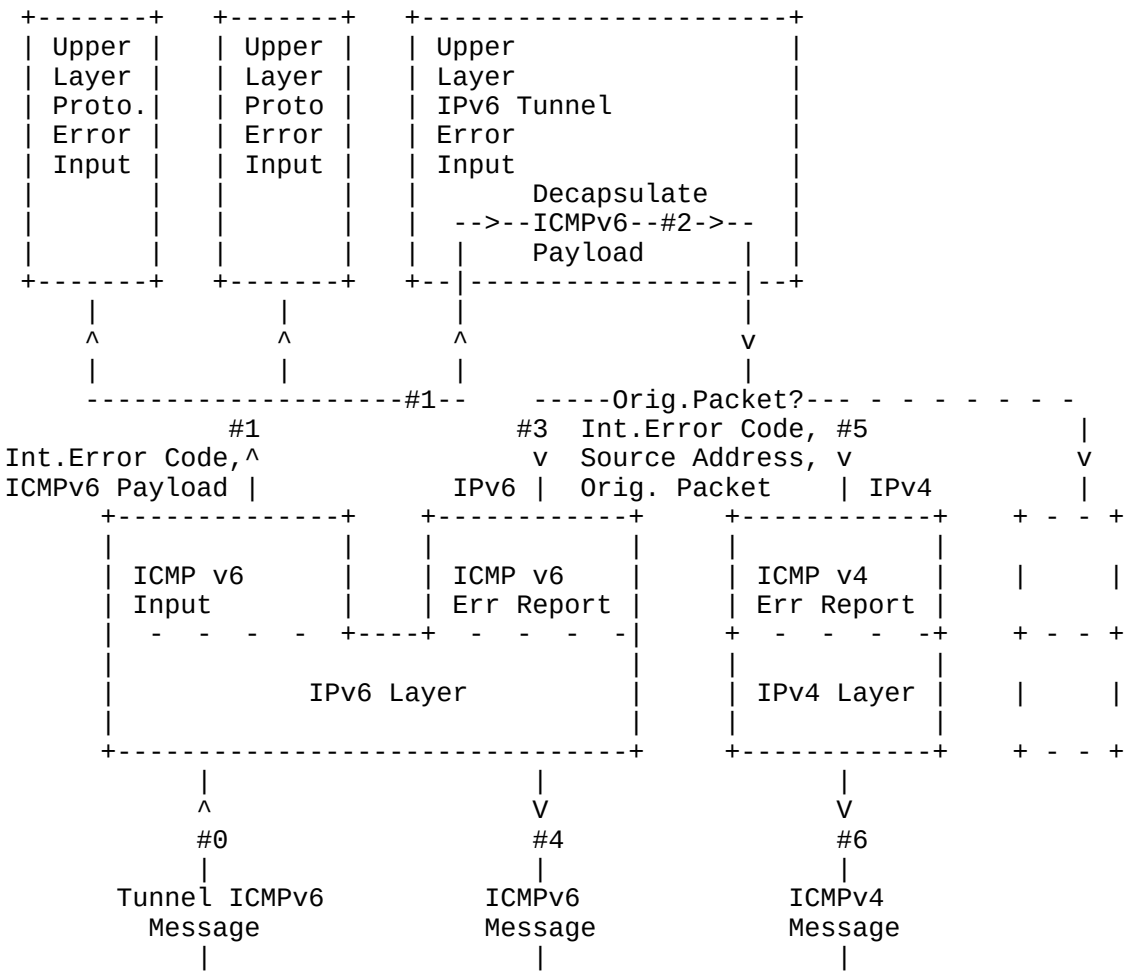


Fig.7 Error Reporting Flow in a Node (IPv6 Tunneling Protocol Engine)

Fig.7 path #2 and Fig.8 (b) - The IPv6 tunnel error input decapsulates the tunnel IPv6 packet, which is the ICMPv6 message payload, obtaining the original packet, and thus the original headers and dispatches the "internal error code", the source address from the original packet header, and the original packet, down to the error report block of the protocol identified by the Next Header field in the tunnel header immediately preceding the original packet in the ICMP message payload.

From here the processing depends on the protocol of the original packet:

(a) - for an IPv6 original packet

Fig.7 path #3 and Fig.8 (c.1)- for an IPv6 original packet, the ICMPv6 error report builds an ICMP message of a type and code according to the "internal error code", containing the "original packet" as ICMP payload.

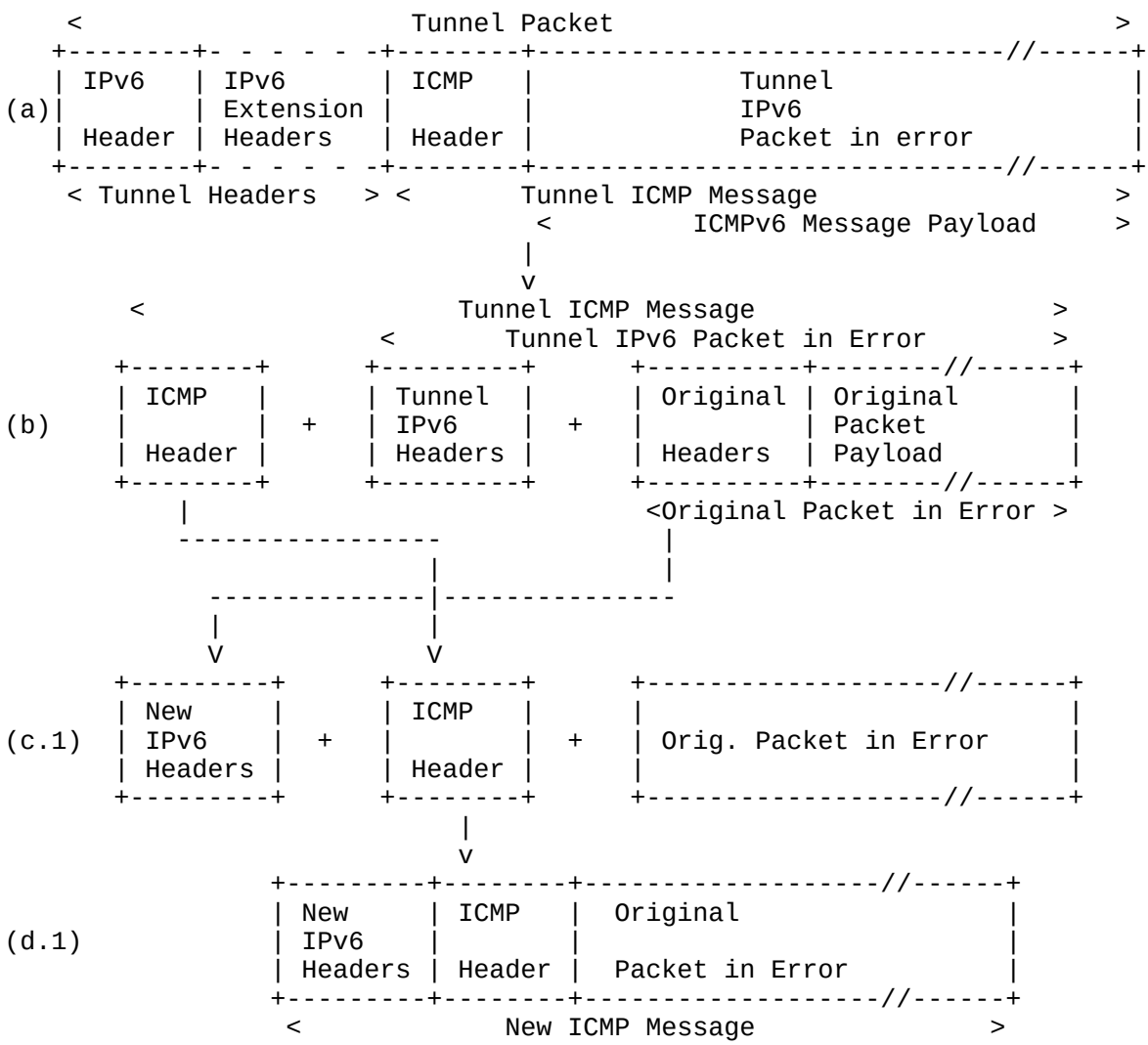
Fig.7 path #4 and Fig.8 (d.1)- The ICMP message has the tunnel entry-point node address as source address, and the original packet source node address as destination address. The tunnel entry-point node sends the ICMP message to the source node of the original packet.

(b) - for an IPv4 original packet

Fig.7 path #5 and Fig.8 (c.2) - for an IPv4 original packet, the ICMPv4 error report builds an ICMP message of a type and code derived from the the "internal error code", containing the "original packet" as ICMP payload.

Fig.7 path #6 and Fig.8 (d.2) - The ICMP message has the tunnel entry-point node IPv4 address as source address, and the original packet IPv4 source node address as destination address. The tunnel entry-point node sends the ICMP message to the source node of the original packet.

A graphical description of the header processing taking place is the following:


Tunnel Packet
IPv6 Header
IPv6 Extension Headers
ICMP Header
Tunnel IPv6 Packet in error
(a)
< Tunnel Headers >
Tunnel ICMP Message
ICMPv6 Message Payload
Tunnel ICMP Message
Tunnel IPv6 Packet in Error
ICMP Header
Tunnel IPv6 Headers
Original Packet Headers
Original Packet Payload
(b)
<Original Packet in Error >
New IPv6 Headers
ICMP Header
Orig. Packet in Error
(c.1)
New IPv6 Headers
ICMP Header
Original Packet in Error
(d.1)
New ICMP Message

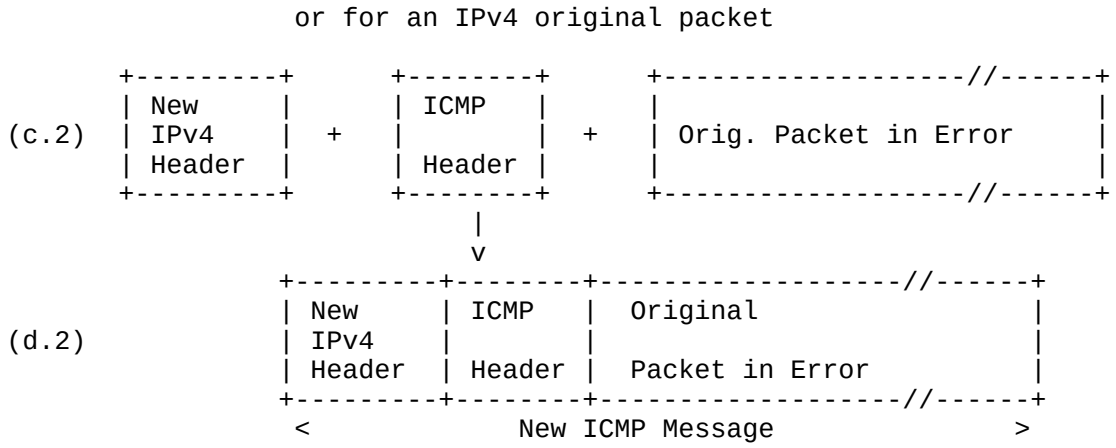


Fig.8 ICMP Error Reporting and Processing

8.1 Tunnel ICMP Messages

The tunnel ICMP messages that are reported to the source of the original packet are:

hop limit exceeded

The tunnel has a misconfigured hop limit, or contains a routing loop, and packets do not reach the tunnel exit-point node. This problem is reported to the tunnel entry-point node, where the tunnel hop limit can be reconfigured to a higher value. The problem is further reported to the source of the original packet as described in section 8.2, or 8.3.

unreachable node

One of the nodes in the tunnel is not or is no longer reachable. This problem is reported to the tunnel entry-point node, which should be reconfigured with a valid and active path between the entry and exit-point of the tunnel.

The problem is further reported to the source of the original packet as described in section 8.2, or 8.3.

parameter problem

A Parameter Problem ICMP message pointing to a valid Tunnel Encapsulation Limit Destination header with a Tun Encap Lim field value set to one is an indication that the tunnel

packet exceeded the maximum number of encapsulations allowed. The problem is further reported to the source of the original packet as described in section 8.2, or 8.3.

The above three problems detected inside the tunnel, which are a tunnel configuration and a tunnel topology problem, are reported to the source of the original IPv6 packet, as a tunnel generic "unreachable" problem caused by a "link problem" - see section 8.2 and 8.3.

packet too big

The tunnel packet exceeds the tunnel Path MTU.

The information carried by this type of ICMP message is used as follows:

- by a receiving tunnel entry-point node to set or adjust the tunnel MTU

- by a sending tunnel entry-point node to indicate to the source of an original packet the MTU size that should be used in sending IPv6 packets towards the tunnel entry-point node.

8.2 ICMP Messages for IPv6 Original Packets

The tunnel entry-point node builds the ICMP and IPv6 headers of the ICMP message that is sent to the source of the original packet as follows:

IPv6 Fields:

Source Address

A valid unicast IPv6 address of the outgoing interface.

Destination Address

Copied from the Source Address field of the Original IPv6 header.

ICMP Fields:

For any of the following tunnel ICMP error messages:

"hop limit exceeded"

"unreachable node"

"parameter problem" - pointing to a valid Tunnel Encapsulation Limit destination header with the Tun Encap Lim field set to a value zero:

Type 1 - unreachable node

Code 3 - address unreachable

For tunnel ICMP error message "packet too big":

Type 2 - packet too big

Code 0

MTU The MTU field from the tunnel ICMP message minus the length of the tunnel headers.

According to the general rules described in 7.1, an ICMP "packet too big" message is sent to the source of the original packet only if the original packet size is larger than the minimum link MTU size required for IPv6 [IPv6-Spec].

8.3 ICMP Messages for IPv4 Original Packets

The tunnel entry-point node builds the ICMP and IPv4 header of the ICMP message that is sent to the source of the original packet as follows:

IPv4 Fields:

Source Address

A valid unicast IPv4 address of the outgoing interface.

Destination Address

Copied from the Source Address field of the Original IPv4 header.

ICMP Fields:

For any of the following tunnel ICMP error messages:

"hop limit exceeded"

"unreachable node"

"parameter problem" - pointing to a valid Tunnel Enacpsulation Limit destination header with the Tun Encap Lim field set to a value zero:

Type 3 - destination unreachable

Code 1 - host unreachable

For a tunnel ICMP error message "packet too big":

Type 3 - destination unreachable

Code 4 - packet too big

MTU The MTU field from the tunnel ICMP message minus the length of the tunnel headers.

According to the general rules described in section 7.2, an ICMP "packet too big" message is sent to the original IPv4 packet source node if the the original IPv4 header has the DF - don't fragment - bit flag SET.

8.4 ICMP Messages for Nested Tunnel Packets

In case of an error uncovered with a nested tunnel packet, the inner tunnel entry-point, which receives the ICMP error message from the inner tunnel reporting node, relays the ICMP message to the outer tunnel entry-point following the mechanisms described in sections 8.,8.1, 8.2, and 8.3. Further, the outer tunnel entry-point relays the ICMP message to the source of the original packet, following the same mechanisms.

9. Security Considerations

An IPv6 tunnel can be secured by securing the IPv6 path between the tunnel entry-point and exit-point node. The security architecture, mechanisms, and services are described in [RFC2401], [RFC2402], and [RFC2406]. A secure IPv6 tunnel may act as a gateway-to-gateway secure path as described in [RFC2401].

For a secure IPv6 tunnel, in addition to the mechanisms described earlier in this document, the entry-point node of the tunnel performs security algorithms on the packet and prepends as part of the tunnel headers one or more security headers in conformance with [IPv6-Spec], [RFC2401], and [RFC2402], or [RFC2406].

The exit-point node of a secure IPv6 tunnel performs security algorithms and processes the tunnel security header[s] as part of the tunnel headers processing described earlier, and in conformance with [RFC2401], and [RFC2402], or [RFC2406]. The exit-point node discards the tunnel security header[s] with the rest of the tunnel headers after tunnel headers processing completion.

The degree of integrity, authentication, and confidentiality and the security processing performed on a tunnel packet at the entry-point and exit-point node of a secure IPv6 tunnel depend on the type of security header - authentication (AH) or encryption (ESP) - and parameters configured in the Security Association for the tunnel. There is no dependency or interaction between the security level and mechanisms applied to the tunnel packets and the security applied to the original packets which are the payloads of the tunnel packets. In case of nested tunnels, each inner tunnel may have its own set of security services, independently from those of the outer tunnels, or of those between the source and destination of the original packet.

10. Acknowledgments

This document is partially derived from several discussions about IPv6 tunneling on the IPng Working Group Mailing List and from feedback from the IPng Working Group to an IPv6 presentation that focused on IPv6 tunneling at the 33rd IETF, in Stockholm, in July 1995.

Additionally, the following documents that focused on tunneling or encapsulation were helpful references: RFC 1933 (R. Gilligan, E. Nordmark), RFC 1241 (R. Woodburn, D. Mills), RFC 1326 (P. Tsuchiya), RFC 1701, RFC 1702 (S. Hanks, D. Farinacci, P. Traina), RFC 1853 (W. Simpson), as well as RFC 2003 (C. Perkins).

Brian Carpenter, Richard Draves, Bob Hinden, Thomas Narten, Erik Nordmark (in alphabetical order) gave valuable reviewing comments and suggestions for the improvement of this document. Scott Bradner, Ross Callon, Dimitry Haskin, Paul Traina, and James Watt (in alphabetical order) shared their view or experience on matters of concern in this document. Judith Grossman provided a sample of her many years of editorial and writing experience as well as a good amount of probing technical questions.

11. References

[IPv6-Spec] Deering, S. and R. Hinden, "Internet Protocol Version 6 (IPv6) Specification", RFC 2460, December 1998.

[ICMP-Spec] Conta, A. and S. Deering "Internet Control Message Protocol for the Internet Protocol Version 6 (IPv6)", RFC 2463, December 1998.

[ND-Spec] Narten, T., Nordmark, E., and W. Simpson "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[PMTU-Spec] McCann, J., Deering, S. and J. Mogul, "Path MTU Discovery for IP Version 6 (IPv6)", RFC 1981, August 1996.

[RFC2401] Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2402] Atkinson, R., "IP Authentication Header", RFC 2402, November 1998.

[RFC2406] Atkinson, R., "IP Encapsulation Security Payload (ESP)", RFC 2406, November 1998.

[RFC-1853] Simpson, W., "IP in IP Tunneling", RFC 1853, October 1995.

[Assign-Nr] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. See also: http://www.iana.org/numbers.html

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

Authors' Addresses

Alex Conta
Lucent Technologies Inc.
300 Baker Ave
Concord, MA 01742-2168
+1-978-287-2842

EMail: aconta@lucent.com

Stephen Deering
Cisco Systems
170 West Tasman Dr
San Jose, CA 95132-1706

Phone: +1-408-527-8213
EMail: deering@cisco.com

Appendix A

A.1 Risk Factors in Nested Encapsulation

Nested encapsulations of a packet become a recursive encapsulation if the packet reenters an outer tunnel before exiting it. The cases which present a high risk of recursive encapsulation are those in which a tunnel entry-point node cannot determine whether a packet that undergoes encapsulation reenters the tunnel before exiting it. Routing loops that cause tunnel packets to reenter a tunnel before exiting it are certainly the major cause of the problem. But since routing loops exist, and happen, it is important to understand and describe, the cases in which the risk for recursive encapsulation is higher.

There are two significant elements that determine the risk factor of routing loop recursive encapsulation:

(a) the type of tunnel,

(b) the type of route to the tunnel exit-point, which determines the packet forwarding through the tunnel, that is, over the tunnel virtual-link.

A.1.1 Risk Factor in Nested Encapsulation - type of tunnel.

The type of tunnels which were identified as a high risk factor for recursive encapsulation in routing loops are:

"inner tunnels with identical exit-points".

Since the source and destination of an original packet is the main information used to decide whether to forward a packet through a tunnel or not, a recursive encapsulation can be avoided in case of a single tunnel (non-inner), by checking that the packet to be encapsulated is not originated on the entry-point node. This mechanism is suggested in [RFC-1853].

However, this type of protection does not seem to work well in case of inner tunnels with different entry-points, and identical exit-points.

Inner tunnels with different entry-points and identical exit-points introduce ambiguity in deciding whether to encapsulate a packet, when a packet encapsulated in an inner tunnel reaches the entry-point node of an outer tunnel by means of a routing loop. Because the source of the tunnel packet is the inner tunnel entry-point node which is different than the entry-point node of the outer tunnel, the source

address checking (mentioned above) fails to detect an invalid encapsulation, and as a consequence the tunnel packet gets encapsulated at the outer tunnel each time it reaches it through the routing loop.

A.1.2 Risk Factor in Nested Encapsulation - type of route.

The type of route to a tunnel exit-point node has been also identified as a high risk factor of recursive encapsulation in routing loops.

One type of route to a tunnel exit-point node is a route to a specified destination node, that is, the destination is a valid specified IPv6 address (route to node). Such a route can be selected based on the longest match of an original packet destination address with the destination address stored in the tunnel entry-point node routing table entry for that route. The packet forwarded on such a route is first encapsulated and then forwarded towards the tunnel exit-point node.

Another type of route to a tunnel exit-point node is a route to a specified prefix-net, that is, the destination is a valid specified IPv6 prefix (route to net). Such a route can be selected based on the longest path match of an original packet destination address with the prefix destination stored in the tunnel entry-point node routing table entry for that route. The packet forwarded on such a route is first encapsulated and then forwarded towards the tunnel exit-point node.

And finally another type of route to a tunnel exit-point is a default route, or a route to an unspecified destination. This route is selected when no-other match for the destination of the original packet has been found in the routing table. A tunnel that is the first hop of a default route is a "default tunnel".

If the route to a tunnel exit-point is a route to node, the risk factor for recursive encapsulation is minimum.

If the route to a tunnel exit-point is a route to net, the risk factor for recursive encapsulation is medium. There is a range of destination addresses that will match the prefix the route is associated with. If one or more inner tunnels with different tunnel entry-points have exit-point node addresses that match the route to net of an outer tunnel exit-point, then a recursive encapsulation may occur if a tunnel packet gets diverted from inside such an inner tunnel to the entry-point of the outer tunnel that has a route to its exit-point that matches the exit-point of an inner tunnel.

If the route to a tunnel exit-point is a default route, the risk factor for recursive encapsulation is maximum. Packets are forwarded through a default tunnel for lack of a better route. In many situations, forwarding through a default tunnel can happen for a wide range of destination addresses which at the maximum extent is the entire Internet minus the node's link. As consequence, it is likely that in a routing loop case, if a tunnel packet gets diverted from an inner tunnel to an outer tunnel entry-point in which the tunnel is a default tunnel, the packet will be once more encapsulated, because the default routing mechanism will not be able to discern differently, based on the destination.

Full Copyright Statement