Network Working Group
Request for Comments: 1933
Category: Standards Track

R. Gilligan
E. Nordmark
Sun Microsystems, Inc.

April 1996

Transition Mechanisms for IPv6 Hosts and Routers

Status of this Memo

Abstract

This document specifies IPv4 compatibility mechanisms that can be implemented by IPv6 hosts and routers. These mechanisms include providing complete implementations of both versions of the Internet Protocol (IPv4 and IPv6), and tunneling IPv6 packets over IPv4 routing infrastructures. They are designed to allow IPv6 nodes to maintain complete compatibility with IPv4, which should greatly simplify the deployment of IPv6 in the Internet, and facilitate the eventual transition of the entire Internet to IPv6.

1. Introduction

The key to a successful IPv6 transition is compatibility with the large installed base of IPv4 hosts and routers. Maintaining compatibility with IPv4 while deploying IPv6 will streamline the task of transitioning the Internet to IPv6. This specification defines a set of mechanisms that IPv6 hosts and routers may implement in order to be compatible with IPv4 hosts and routers.

The mechanisms in this document are designed to be employed by IPv6 hosts and routers that need to interoperate with IPv4 hosts and utilize IPv4 routing infrastructures. We expect that most nodes in the Internet will need such compatibility for a long time to come, and perhaps even indefinitely.

However, IPv6 may be used in some environments where interoperability with IPv4 is not required. IPv6 nodes that are designed to be used in such environments need not use or even implement these mechanisms.

The mechanisms specified here include:

- Dual IP layer. Providing complete support for both IPv4 and IPv6 in hosts and routers.

- IPv6 over IPv4 tunneling. Encapsulating IPv6 packets within IPv4 headers to carry them over IPv4 routing infrastructures. Two types of tunneling are employed: configured and automatic.

Additional transition and compatibility mechanisms may be developed in the future. These will be specified in other documents.

1.2. Terminology

The following terms are used in this document:

Types of Nodes

IPv4-only node:

A host or router that implements only IPv4. An IPv4-only node does not understand IPv6. The installed base of IPv4 hosts and routers existing before the transition begins are IPv4-only nodes.

IPv6/IPv4 node:

A host or router that implements both IPv4 and IPv6.

IPv6-only node:

A host or router that implements IPv6, and does not implement IPv4. The operation of IPv6-only nodes is not addressed here.

IPv6 node:

Any host or router that implements IPv6. IPv6/IPv4 and IPv6-only nodes are both IPv6 nodes.

IPv4 node:

Any host or router that implements IPv4. IPv6/IPv4 and IPv4-only nodes are both IPv4 nodes.

Types of IPv6 Addresses

IPv4-compatible IPv6 address:

An IPv6 address, assigned to an IPv6/IPv4 node, which bears the high-order 96-bit prefix 0:0:0:0:0:0, and an IPv4 address in the low-order 32-bits. IPv4-compatible addresses are used by the automatic tunneling mechanism.

IPv6-only address:

The remainder of the IPv6 address space. An IPv6 address that bears a prefix other than 0:0:0:0:0:0.

Techniques Used in the Transition

IPv6-over-IPv4 tunneling:

The technique of encapsulating IPv6 packets within IPv4 so that they can be carried across IPv4 routing infrastructures.

IPv6-in-IPv4 encapsulation:

IPv6-over-IPv4 tunneling.

Configured tunneling:

IPv6-over-IPv4 tunneling where the IPv4 tunnel endpoint address is determined by configuration information on the encapsulating node.

Automatic tunneling:

IPv6-over-IPv4 tunneling where the IPv4 tunnel endpoint address is determined from the IPv4 address embedded in the IPv4-compatible destination address of the IPv6 packet.

1.3. Structure of this Document

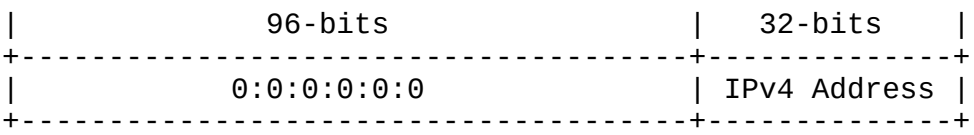
The remainder of this document is organized into three sections:

- Section 2 discusses the IPv4-compatible address format.
- Section 3 discusses the operation of nodes with a dual IP layer, IPv6/IPv4 nodes.

- Section 4 discusses IPv6-over-IPv4 tunneling.

2. Addressing

The automatic tunneling mechanism uses a special type of IPv6 address, termed an "IPv4-compatible" address. An IPv4-compatible address is identified by an all-zeros 96-bit prefix, and holds an IPv4 address in the low-order 32-bits. IPv4-compatible addresses are structured as follows:



IPv4-Compatible IPv6 Address Format

IPv4-compatible addresses are assigned to IPv6/IPv4 nodes that support automatic tunneling. Nodes that are configured with IPv4-compatible addresses may use the complete address as their IPv6 address, and use the embedded IPv4 address as their IPv4 address.

The remainder of the IPv6 address space (that is, all addresses with 96-bit prefixes other than 0:0:0:0:0:0) are termed "IPv6-only Addresses."

3. Dual IP Layer

The most straightforward way for IPv6 nodes to remain compatible with IPv4-only nodes is by providing a complete IPv4 implementation. IPv6 nodes that provide a complete IPv4 implementation in addition to their IPv6 implementation are called "IPv6/IPv4 nodes." IPv6/IPv4 nodes have the ability to send and receive both IPv4 and IPv6 packets. They can directly interoperate with IPv4 nodes using IPv4 packets, and also directly interoperate with IPv6 nodes using IPv6 packets.

The dual IP layer technique may or may not be used in conjunction with the IPv6-over-IPv4 tunneling techniques, which are described in section 4. An IPv6/IPv4 node that supports tunneling may support only configured tunneling, or both configured and automatic tunneling. Thus three configurations are possible:

- IPv6/IPv4 node that does not perform tunneling.

- IPv6/IPv4 node that performs configured tunneling only.

- IPv6/IPv4 node that performs configured tunneling and

automatic tunneling.

3.1. Address Configuration

Because they support both protocols, IPv6/IPv4 nodes may be configured with both IPv4 and IPv6 addresses. Although the two addresses may be related to each other, this is not required. IPv6/IPv4 nodes may be configured with IPv6 and IPv4 addresses that are unrelated to each other.

Nodes that perform automatic tunneling are configured with IPv4-compatible IPv6 addresses. These may be viewed as single addresses that can serve both as IPv6 and IPv4 addresses. The entire 128-bit IPv4-compatible IPv6 address is used as the node's IPv6 address, while the IPv4 address embedded in low-order 32-bits serves as the node's IPv4 address.

IPv6/IPv4 nodes may use the stateless IPv6 address configuration mechanism [5] or DHCP for IPv6 [3] to acquire their IPv6 address. These mechanisms may provide either IPv4-compatible or IPv6-only IPv6 addresses.

IPv6/IPv4 nodes may use IPv4 mechanisms to acquire their IPv4 addresses.

IPv6/IPv4 nodes that perform automatic tunneling may also acquire their IPv4-compatible IPv6 addresses from another source: IPv4 address configuration protocols. A node may use any IPv4 address configuration mechanism to acquire its IPv4 address, then "map" that address into an IPv4-compatible IPv6 address by pre-pending it with the 96-bit prefix 0:0:0:0:0:0. This mode of configuration allows IPv6/IPv4 nodes to "leverage" the installed base of IPv4 address configuration servers. It can be particularly useful in environments where IPv6 routers and address configuration servers have not yet been deployed.

The specific algorithm for acquiring an IPv4-compatible address using IPv4-based address configuration protocols is as follows:

1) The IPv6/IPv4 node uses standard IPv4 mechanisms or protocols to acquire its own IPv4 address. These include:

   - The Dynamic Host Configuration Protocol (DHCP) [2]
   - The Bootstrap Protocol (BOOTP) [1]
   - The Reverse Address Resolution Protocol (RARP) [9]
   - Manual configuration
   - Any other mechanism which accurately yields the node's own IPv4 address

2) The node uses this address as its IPv4 address.

3) The node prepends the 96-bit prefix 0:0:0:0:0:0 to the 32-bit IPv4 address that it acquired in step (1). The result is an IPv4-compatible IPv6 address with the node's own IPv4-address embedded in the low-order 32-bits. The node uses this address as its own IPv6 address.

3.1.1. IPv4 Loopback Address

Many IPv4 implementations treat the address 127.0.0.1 as a "loopback address" -- an address to reach services located on the local machine. Per the host requirements specification [10], section 3.2.1.3, IPv4 packets addressed from or to the loopback address are not to be sent onto the network; they must remain entirely within the node. IPv6/IPv4 implementations may treat the IPv4-compatible IPv6 address ::127.0.0.1 as an IPv6 loopback address. Packets with this address should also remain entirely within the node, and not be transmitted onto the network.

3.2. DNS

The Domain Naming System (DNS) is used in both IPv4 and IPv6 to map hostnames into addresses. A new resource record type named "AAAA" has been defined for IPv6 addresses [6]. Since IPv6/IPv4 nodes must be able to interoperate directly with both IPv4 and IPv6 nodes, they must provide resolver libraries capable of dealing with IPv4 "A" records as well as IPv6 "AAAA" records.

3.2.1. Handling Records for IPv4-Compatible Addresses

When an IPv4-compatible IPv6 addresses is assigned to an IPv6/IPv4 host that supports automatic tunneling, both A and AAAA records are listed in the DNS. The AAAA record holds the full IPv4-compatible IPv6 address, while the A record holds the low-order 32-bits of that address. The AAAA record is needed so that queries by IPv6 hosts can be satisfied. The A record is needed so that queries by IPv4-only hosts, whose resolver libraries only support the A record type, will locate the host.

DNS resolver libraries on IPv6/IPv4 nodes must be capable of handling both AAAA and A records. However, when a query locates an AAAA record holding an IPv4-compatible IPv6 address, and an A record holding the corresponding IPv4 address, the resolver library need not necessarily return both addresses. It has three options:

- Return only the IPv6 address to the application.

- Return only the IPv4 address to the application.

- Return both addresses to the application.

The selection of which address type to return in this case, or, if both addresses are returned, in which order they are listed, can affect what type of IP traffic is generated. If the IPv6 address is returned, the node will communicate with that destination using IPv6 packets (in most cases encapsulated in IPv4); If the IPv4 address is returned, the communication will use IPv4 packets.

The way that DNS resolver implementations handle redundant records for IPv4-compatible addresses may depend on whether that implementation supports automatic tunneling, or whether it is enabled. For example, an implementation that does not support automatic tunneling would not return IPv4-compatible IPv6 addresses to applications because those destinations are generally only reachable via tunneling. On the other hand, those implementations in which automatic tunneling is supported and enabled may elect to return only the IPv4-compatible IPv6 address and not the IPv4 address.

4. IPv6-over-IPv4 Tunneling

In most deployment scenarios, the IPv6 routing infrastructure will be built up over time. While the IPv6 infrastructure is being deployed, the existing IPv4 routing infrastructure can remain functional, and can be used to carry IPv6 traffic. Tunneling provides a way to utilize an existing IPv4 routing infrastructure to carry IPv6 traffic.

IPv6/IPv4 hosts and routers can tunnel IPv6 datagrams over regions of IPv4 routing topology by encapsulating them within IPv4 packets. Tunneling can be used in a variety of ways:

- Router-to-Router. IPv6/IPv4 routers interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans one segment of the end-to-end path that the IPv6 packet takes.

- Host-to-Router. IPv6/IPv4 hosts can tunnel IPv6 packets to an intermediary IPv6/IPv4 router that is reachable via an IPv4 infrastructure. This type of tunnel spans the first segment of the packet’s end-to-end path.

- Host-to-Host. IPv6/IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans the entire end-to-end path that the packet takes.

- Router-to-Host. IPv6/IPv4 routers can tunnel IPv6 packets to their final destination IPv6/IPv4 host. This tunnel spans only the last segment of the end-to-end path.

Tunneling techniques are usually classified according to the mechanism by which the encapsulating node determines the address of the node at the end of the tunnel. In the first two tunneling methods listed above -- router-to-router and host-to-router -- the IPv6 packet is being tunneled to a router. The endpoint of this type of tunnel is an intermediary router which must decapsulate the IPv6 packet and forward it on to its final destination. When tunneling to a router, the endpoint of the tunnel is different from the destination of the packet being tunneled. So the addresses in the IPv6 packet being tunneled do not provide the IPv4 address of the tunnel endpoint. Instead, the tunnel endpoint address must be determined from configuration information on the node performing the tunneling. We use the term "configured tunneling" to describe the type of tunneling where the endpoint is explicitly configured.

In the last two tunneling methods -- host-to-host and router-to-host -- the IPv6 packet is tunneled all the way to its final destination.

The tunnel endpoint is the node to which the IPv6 packet is addressed. Since the endpoint of the tunnel is the destination of the IPv6 packet, the tunnel endpoint can be determined from the destination IPv6 address of that packet: If that address is an IPv4-compatible address, then the low-order 32-bits hold the IPv4 address of the destination node, and that can be used as the tunnel endpoint address. This technique avoids the need to explicitly configure the tunnel endpoint address. Deriving the tunnel endpoint address from the embedded IPv4 address of the packet’s IPv6 address is termed "automatic tunneling".

The two tunneling techniques -- automatic and configured -- differ primarily in how they determine the tunnel endpoint address. Most of the underlying mechanisms are the same:

- The entry node of the tunnel (the encapsulating node) creates an encapsulating IPv4 header and transmits the encapsulated packet.

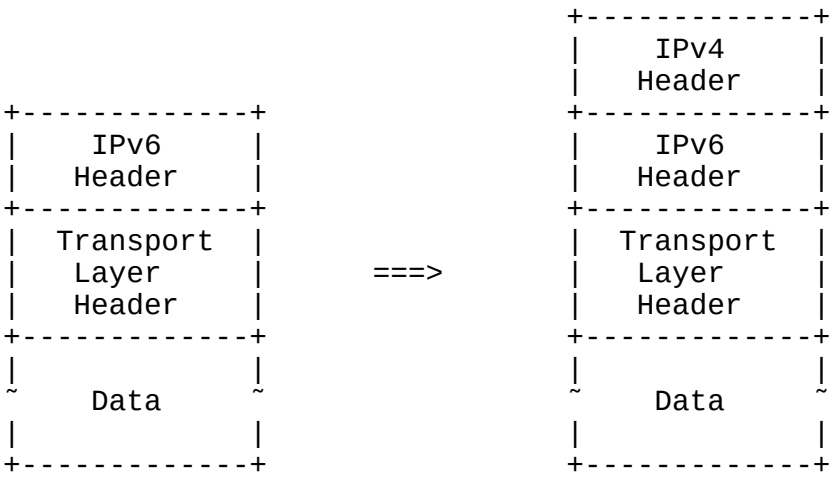
- The exit node of the tunnel (the decapsulating node) receives the encapsulated packet, removes the IPv4 header, updates the IPv6 header, and processes the received IPv6 packet.

- The encapsulating node may need to maintain soft state information for each tunnel recording such parameters as the MTU of the tunnel in order to process IPv6 packets forwarded into the tunnel. Since the number of tunnels that any one host or router may be using may grow to be quite large, this state information can be cached and discarded when not in use.

The next section discusses the common mechanisms that apply to both types of tunneling. Subsequent sections discuss how the tunnel endpoint address is determined for automatic and configured tunneling.

4.1. Common Tunneling Mechanisms

The encapsulation of an IPv6 datagram in IPv4 is shown below:



Encapsulating IPv6 in IPv4

In addition to adding an IPv4 header, the encapsulating node also has to handle some more complex issues:

- Determine when to fragment and when to report an ICMP "packet too big" error back to the source.

- How to reflect IPv4 ICMP errors from routers along the tunnel path back to the source as IPv6 ICMP errors.

Those issues are discussed in the following sections.

4.1.1. Tunnel MTU and Fragmentation

The encapsulating node could view encapsulation as IPv6 using IPv4 as a link layer with a very large MTU (65535-20 bytes to be exact; 20 bytes "extra" are needed for the encapsulating IPv4 header). The encapsulating node would need only to report IPv6 ICMP "packet too big" errors back to the source for packets that exceed this MTU. However, such a scheme would be inefficient for two reasons:

1) It would result in more fragmentation than needed. IPv4 layer fragmentation should be avoided due to the performance problems caused by the loss unit being smaller than the retransmission unit [11].

2) Any IPv4 fragmentation occurring inside the tunnel would have to be reassembled at the tunnel endpoint. For tunnels that terminate at a router, this would require additional memory to reassemble the IPv4 fragments into a complete IPv6 packet before that packet could be forwarded onward.

The fragmentation inside the tunnel can be reduced to a minimum by having the encapsulating node track the IPv4 Path MTU across the tunnel, using the IPv4 Path MTU Discovery Protocol [8] and recording the resulting path MTU. The IPv6 layer in the encapsulating node can then view a tunnel as a link layer with an MTU equal to the IPv4 path MTU, minus the size of the encapsulating IPv4 header.

Note that this does not completely eliminate IPv4 fragmentation in the case when the IPv4 path MTU would result in an IPv6 MTU less than 576 bytes. (Any link layer used by IPv6 has to have an MTU of at least 576 bytes [4].) In this case the IPv6 layer has to "see" a link layer with an MTU of 576 bytes and the encapsulating node has to use IPv4 fragmentation in order to forward the 576 byte IPv6 packets.

The encapsulating node can employ the following algorithm to determine when to forward an IPv6 packet that is larger than the tunnel's path MTU using IPv4 fragmentation, and when to return an IPv6 ICMP "packet too big" message:

```
if (IPv4 path MTU - 20) is less than or equal to 576
        if packet is larger than 576 bytes
                Send IPv6 ICMP "packet too big" with MTU = 576.
                Drop packet.
        else
                Encapsulate but do not set the Don't Fragment
                flag in the IPv4 header. The resulting IPv4
                packet might be fragmented by the IPv4 layer on
                the encapsulating node or by some router along
```

```
                        the IPv4 path.
                endif
        else
                if packet is larger than (IPv4 path MTU - 20)
                        Send IPv6 ICMP "packet too big" with
                        MTU = (IPv4 path MTU - 20).
                        Drop packet.
                else
                        Encapsulate and set the Don't Fragment flag
                        in the IPv4 header.
                endif
        endif
```

Encapsulating nodes that have a large number of tunnels might not be able to store the IPv4 Path MTU for all tunnels. Such nodes can, at the expense of additional fragmentation in the network, avoid using the IPv4 Path MTU algorithm across the tunnel and instead use the MTU of the link layer (under IPv4) in the above algorithm instead of the IPv4 path MTU.

In this case the Don't Fragment bit must not be set in the encapsulating IPv4 header.

4.1.2. Hop Limit

IPv6-over-IPv4 tunnels are modeled as "single-hop". That is, the IPv6 hop limit is decremented by 1 when an IPv6 packet traverses the tunnel. The single-hop model serves to hide the existence of a tunnel. The tunnel is opaque to users of the network, and is not detectable by network diagnostic tools such as traceroute.

The single-hop model is implemented by having the encapsulating and decapsulating nodes process the IPv6 hop limit field as they would if they were forwarding a packet on to any other datalink. That is, they decrement the hop limit by 1 when forwarding an IPv6 packet. (The originating node and final destination do not decrement the hop limit.)

The TTL of the encapsulating IPv4 header is selected in an implementation dependent manner. The current suggested value is published in the "Assigned Numbers RFC. Implementations may provide a mechanism to allow the administrator to configure the IPv4 TTL.

4.1.3. Handling IPv4 ICMP errors

In response to encapsulated packets it has sent into the tunnel, the encapsulating node may receive IPv4 ICMP error messages from IPv4 routers inside the tunnel. These packets are addressed to the

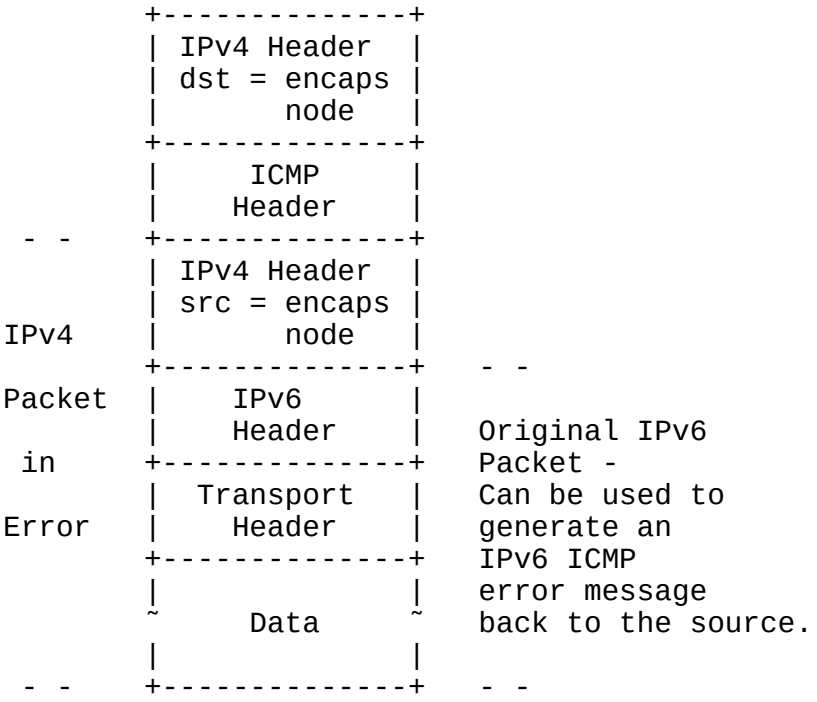
encapsulating node because it is the IPv4 source of the encapsulated packet.

The ICMP "packet too big" error messages are handled according to IPv4 Path MTU Discovery [8] and the resulting path MTU is recorded in the IPv4 layer. The recorded path MTU is used by IPv6 to determine if an IPv6 ICMP "packet too big" error has to be generated as described in section 4.1.1.

The handling of other types of ICMP error messages depends on how much information is included in the "packet in error" field, which holds the encapsulated packet that caused the error.

Many older IPv4 routers return only 8 bytes of data beyond the IPv4 header of the packet in error, which is not enough to include the address fields of the IPv6 header. More modern IPv4 routers may return enough data beyond the IPv4 header to include the entire IPv6 header and possibly even the data beyond that.

If the offending packet includes enough data, the encapsulating node may extract the encapsulated IPv6 packet and use it to generating an IPv6 ICMP message directed back to the originating IPv6 node, as shown below:



IPv4 ICMP Error Message Returned to Encapsulating Node

4.1.4. IPv4 Header Construction

When encapsulating an IPv6 packet in an IPv4 datagram, the IPv4 header fields are set as follows:

Version:

4

IP Header Length in 32-bit words:

5 (There are no IPv4 options in the encapsulating header.)

Type of Service:

0

Total Length:

Payload length from IPv6 header plus length of IPv6 and IPv4 headers (i.e. a constant 60 bytes).

Identification:

Generated uniquely as for any IPv4 packet transmitted by the system.

Flags:

Set the Don't Fragment (DF) flag as specified in section 4.1.1. Set the More Fragments (MF) bit as necessary if fragmenting.

Fragment offset:

Set as necessary if fragmenting.

Time to Live:

Set in implementation-specific manner.

Protocol:

41 (Assigned payload type number for IPv6)

Header Checksum:

Calculate the checksum of the IPv4 header.

Source Address:

IPv4 address of outgoing interface of the encapsulating node.

Destination Address:

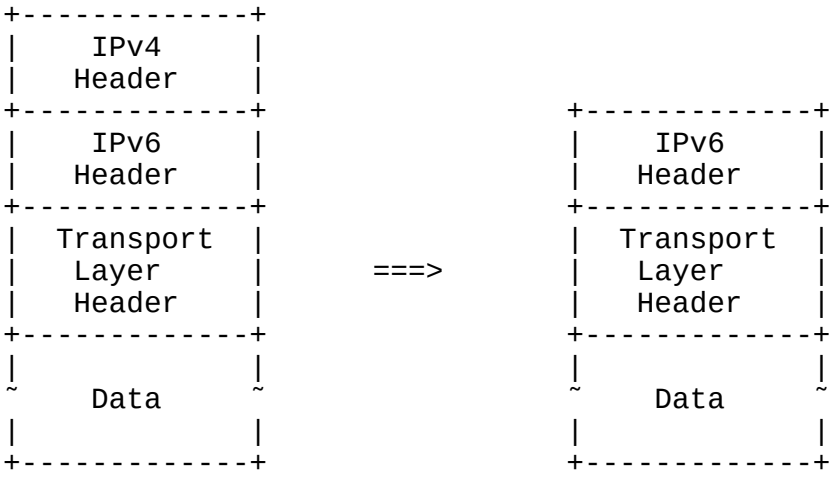
IPv4 address of tunnel endpoint.

Any IPv6 options are preserved in the packet (after the IPv6 header).

4.1.5. Decapsulating IPv6-in-IPv4 Packets

When an IPv6/IPv4 host or a router receives an IPv4 datagram that is addressed to one of its own IPv4 address, and the value of the protocol field is 41, it removes the IPv4 header and submits the IPv6 datagram to its IPv6 layer code.

The decapsulation is shown below:



Decapsulating IPv6 from IPv4

When decapsulating the IPv6-in-IPv4 packet, the IPv6 header is not modified. If the packet is subsequently forwarded, its hop limit is decremented by one.

The encapsulating IPv4 header is discarded.

The decapsulating node performs IPv4 reassembly before decapsulating the IPv6 packet. All IPv6 options are preserved even if the encapsulating IPv4 packet is fragmented.

After the IPv6 packet is decapsulated, it is processed the same as any received IPv6 packet.

4.2. Configured Tunneling

In configured tunneling, the tunnel endpoint address is determined from configuration information in the encapsulating node. For each tunnel, the encapsulating node must store the tunnel endpoint address. When an IPv6 packet is transmitted over a tunnel, the tunnel endpoint address configured for that tunnel is used as the destination address for the encapsulating IPv4 header.

The determination of which packets to tunnel is usually made by routing information on the encapsulating node. This is usually done via a routing table, which directs packets based on their destination address using the prefix mask and match technique.

4.2.1. Default Configured Tunnel

Nodes that are connected to IPv4 routing infrastructures may use a configured tunnel to reach an IPv6 "backbone". If the IPv4 address of an IPv6/IPv4 router bordering the backbone is known, a tunnel can be configured to that router. This tunnel can be configured into the routing table as a "default route". That is, all IPv6 destination addresses will match the route and could potentially traverse the tunnel. Since the "mask length" of such default route is zero, it will be used only if there are no other routes with a longer mask that match the destination.

The tunnel endpoint address of such a default tunnel could be the IPv4 address of one IPv6/IPv4 router at the border of the IPv6 backbone. Alternatively, the tunnel endpoint could be an IPv4 "anycast address". With this approach, multiple IPv6/IPv4 routers at the border advertise IPv4 reachability to the same IPv4 address. All of these routers accept packets to this address as their own, and will decapsulate IPv6 packets tunneled to this address. When an IPv6/IPv4 node sends an encapsulated packet to this address, it will be delivered to only one of the border routers, but the sending node will not know which one. The IPv4 routing system will generally carry the traffic to the closest router.

Using a default tunnel to an IPv4 "anycast address" provides a high degree of robustness since multiple border router can be provided, and, using the normal fallback mechanisms of IPv4 routing, traffic

will automatically switch to another router when one goes down.

4.3. Automatic Tunneling

In automatic tunneling, the tunnel endpoint address is determined from the packet being tunneled. The destination IPv6 address in the packet must be an IPv4-compatible address. If it is, the IPv4 address component of that address -- the low-order 32-bits -- are extracted and used as the tunnel endpoint address. IPv6 packets that are not addressed to an IPv4-compatible address can not be tunneled using automatic tunneling.

IPv6/IPv4 nodes need to determine which IPv6 packets can be sent via automatic tunneling. One technique is to use the IPv6 routing table to direct automatic tunneling. An implementation can have a special static routing table entry for the prefix 0:0:0:0:0:0/96. (That is, a route to the all-zeros prefix with a 96-bit mask.) Packets that match this prefix are sent to a pseudo-interface driver which performs automatic tunneling. Since all IPv4-compatible IPv6 addresses will match this prefix, all packets to those destinations will be auto-tunneled.

4.4. Default Sending Algorithm

This section presents a combined IPv4 and IPv6 sending algorithm that IPv6/IPv4 nodes can use. The algorithm can be used to determine when to send IPv4 packets, when to send IPv6 packets, and when to perform automatic and configured tunneling. It illustrates how the techniques of dual IP layer, configured tunneling, and automatic tunneling can be used together. Note that is just an example to show how the techniques can be combined; IPv6/IPv6 implementations may provide different algorithms. This algorithm has the following properties:

- Sends IPv4 packets to all IPv4 destinations.

- Sends IPv6 packets to all IPv6 destinations on the same link.

- Using automatic tunneling, sends IPv6 packets encapsulated in IPv4 to IPv6 destinations with IPv4-compatible addresses that are located off-link.

- Sends IPv6 packets to IPv6 destinations located off-link when IPv6 routers are present.

- Using the default IPv6 tunnel, sends IPv6 packets encapsulated in IPv4 to IPv6 destinations with IPv6-only addresses when no IPv6 routers are present.

The algorithm is as follows:

1) If the address of the end node is an IPv4 address then:

   1.1) If the destination is located on an attached link, then send an IPv4 packet addressed to the end node.

   1.2) If the destination is located off-link, then;

      1.2.1) If there is an IPv4 router on link, then send an IPv4 format packet. The IPv4 destination address is the IPv4 address of the end node. The datalink address is the datalink address of the IPv4 router.

      1.2.2) Else, the destination is treated as "unreachable" because it is located off link and there are no on-link routers.

2) If the address of the end node is an IPv4-compatible IPv6 address (i.e. bears the prefix 0:0:0:0:0:0), then:

   2.1) If the destination is located on an attached link, then send an IPv6 format packet (not encapsulated). The IPv6 destination address is the IPv6 address of the end node. The datalink address is the datalink address of the end node.

   2.2) If the destination is located off-link, then:

      2.2.1) If there is an IPv4 router on an attached link, then send an IPv6 packet encapsulated in IPv4. The IPv6 destination address is the address of the end node. The IPv4 destination address is the low-order 32-bits of the end node's address. The datalink address is the datalink address of the IPv4 router.

      2.2.2) Else, if there is an IPv6 router on an attached link, then send an IPv6 format packet. The IPv6 destination address is the IPv6 address of the end node. The datalink address is the datalink address of the IPv6 router.

      2.2.3) Else, the destination is treated as "unreachable" because it is located off-link and there are no on-link routers.

3) If the address of the end node is an IPv6-only address, then:

3.1) If the destination is located on an attached link, then send an IPv6 format packet. The IPv6 destination address is the IPv6 address of the end node. The datalink address is the datalink address of the end node.

3.2) If the destination is located off-link, then:

3.2.1) If there is an IPv6 router on an attached link, then send an IPv6 format packet. The IPv6 destination address is the IPv6 address of the end node. The datalink address is the datalink address of the IPv6 router.

3.2.2) Else, if the destination is reachable via a configured tunnel, and there is an IPv4 router on an attached link, then send an IPv6 packet encapsulated in IPv4. The IPv6 destination address is the address of the end node. The IPv4 destination address is the configured IPv4 address of the tunnel endpoint. The datalink address is the datalink address of the IPv4 router.

3.2.3) Else, the destination is treated as "unreachable" because it is located off-link and there are no on-link IPv6 routers.

A summary of these sending rules are given in the table below:

End Node Address Type	End Node On Link?	IPv4 Router On Link?	IPv6 Router On Link?	Packet Format To Send	IPv6 Dest Addr	IPv4 Dest Addr	DLink Dest Addr
IPv4	Yes	N/A	N/A	IPv4	N/A	E4	EL
IPv4	No	Yes	N/A	IPv4	N/A	E4	RL
IPv4	No	No	N/A	UNRCH	N/A	N/A	N/A
IPv4-compat	Yes	N/A	N/A	IPv6	E6	N/A	EL
IPv4-compat	No	Yes	N/A	IPv6/4	E6	E4	RL
IPv4-compat	No	No	Yes	IPv6	E6	N/A	RL
IPv4-compat	No	No	No	UNRCH	N/A	N/A	N/A
IPv6-only	Yes	N/A	N/A	IPv6	E6	N/A	EL
IPv6-only	No	N/A	Yes	IPv6	E6	N/A	RL
IPv6-only	No	Yes	No	IPv6/4	E6	T4	RL
IPv6-only	No	No	No	UNRCH	N/A	N/A	N/A

Key to Abbreviations

N/A: Not applicable or does not matter.
E6: IPv6 address of end node.
E4: IPv4 address of end node (low-order 32-bits of IPv4-compatible address).
EL: Datalink address of end node.
T4: IPv4 address of the tunnel endpoint.
R6: IPv6 address of router.
R4: IPv4 address of router.
RL: Datalink address of router.
IPv4: IPv4 packet format.
IPv6: IPv6 packet format.
IPv6/4: IPv6 encapsulated in IPv4 packet format.
UNRCH: Destination is unreachable. Don't send a packet.

4.4.1 On/Off Link Determination

Part of the process of determining what packet format to use includes determining whether a destination is located on an attached link or not. IPv4 and IPv6 employ different mechanisms. IPv4 uses an algorithm in which the destination address and the interface address are both logically ANDed with the netmask of the interface and then compared. If the resulting two values match, then the destination is located on-link. This algorithm is discussed in more detail in Section 3.3.1.1 of the host requirements specification [10]. IPv6 uses the neighbor discovery algorithm described in "Neighbor Discovery for IP Version 6" [7].

IPv6/IPv4 nodes need to use both methods:

- If a destination is an IPv4 address, then the on/off link determination is made by comparison with the netmask, as described in RFC 1122 section 3.3.1.1.

- If a destination is represented by an IPv4-compatible IPv6 address (prefix 0:0:0:0:0:0), the decision is made using the IPv4 netmask comparison algorithm using the low-order 32-bits (IPv4 address part) of the destination address.

- If the destination is represented by an IPv6-only address (prefix other than 0:0:0:0:0:0), the on/off link determination is made using the IPv6 neighbor discovery mechanism.

5. Acknowledgements

We would like to thank the members of the IPng working group and the IPng transition working group for their many contributions and extensive review of this document. Special thanks to Jim Bound, Ross Callon, and Bob Hinden for many helpful suggestions and to John Moy for suggesting the IPv4 "anycast address" default tunnel technique.

6. Security Considerations

Security issues are not discussed in this memo.

7. Authors' Addresses

Robert E. Gilligan
Sun Microsystems, Inc.
2550 Garcia Ave.
Mailstop UMTV 05-44
Mountain View, California 94043

Phone: 415-336-1012
Fax: 415-336-6015
EMail: Bob.Gilligan@Eng.Sun.COM

Erik Nordmark
Sun Microsystems, Inc.
2550 Garcia Ave.
Mailstop UMTV 05-44
Mountain View, California 94043

Phone: 415-336-2788
Fax: 415-336-6015
EMail: Erik.Nordmark@Eng.Sun.COM

7. References

[1] Croft, W., and J. Gilmore, "Bootstrap Protocol", RFC 951, September 1985.

[2] Droms, R., "Dynamic Host Configuration Protocol", RFC 1541. October 1993.

[3] Bound, J., "Dynamic Host Configuration Protocol for IPv6 for IPv6 (DHCPv6)", Work in Progress, November 1995.

[4] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, December 1995.

[5] Thomson, S., and T. Nartan, "IPv6 Stateless Address Autoconfiguration, Work in Progress, December 1995.

[6] Thomson, S., and C. Huitema. "DNS Extensions to support IP version 6", RFC 1886, December 1995.

[7] Nartan, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", Work in Progress, November 1995.

[8] Mogul, J., and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[9] Finlayson, R., Mann, T., Mogul, J., and M. Theimer, "Reverse Address Resolution Protocol", RFC 903, June 1984.

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

[11] Kent, C., and J. Mogul, "Fragmentation Considered Harmful". In Proc. SIGCOMM '87 Workshop on Frontiers in Computer Communications Technology. August 1987.