IPv4-to-IPv6 Transition Strategies

By Tim Rooney
Director, Product Management
BT Diamond IP

Table of Contents

IPv4-to-IPv6 Transition Strategies

By Tim Rooney, Director, Product Management

Introduction

While deployments of IPv6 networks have increased over recent years, especially in Asia and Europe, interest in IPv6 in North America has been lackluster. Focus has intensified recently, however, especially among large service providers and government organizations. IPv6 provides a number of advanced features, though many of these have been retrofitted to IPv4. Nonetheless, the massive increase in address space is indisputably unique to IPv6 and represents the crowning objective for IP-address-hungry organizations. Unfortunately, this increase in address space comes at the cost of different address formats and notations, which affect not only network layer routing, but also applications that display IP addresses[1].

Organizations with existing IPv4 networks seeking to implement IPv6 face challenges in identifying impacts, planning the transition and executing the migration to IPv6. Given the common organizational reliance on external communications for partner links, home-based employees and Internet access for email, web browsing, etc., an overall plan should be compiled addressing the current environment, end users and the controlled steps to IPv6 deployment.

When we discuss migration, we're referring to an initial state of an IPv4-only network to which IPv6 nodes and networks are added or overlaid over time, resulting in an IPv6-only network, or more likely, a predominantly IPv6 network with continued IPv4 support.

This white paper will present an overview of the main migration technologies that can be used to transition from an IPv4 network to an IPv6 network, and will suggest several strategies to implement that transition.

---

[1] See the companion IPv6 Addressing and Management Challenges white paper, which provides an introduction to IPv6 address formats, notation, allocation and assignment, including DHCPv6.

Migration Technologies Overview

A variety of technologies are available to facilitate the migration to IPv6. These technologies will be discussed according to the following basic categories:

- Dual stack – support of both IPv4 and IPv6 on network devices
- Tunneling – encapsulation of an IPv6 packet within an IPv4 packet for transmission over an IPv4 network
- Translation – address or port translation of addresses such as via a gateway device or translation code in the TCP/IP code of the host or router

This white paper will also touch on application migration and provide some example migration scenarios and corresponding impact considerations. Implementation of the selected migration strategy(ies) will require effective coordination of the following:

- IPv4 and IPv6 network and subnet allocations, existing and planned
- Address assignment strategies for IPv4 and IPv6: static, autoconfiguration, DHCP for IPv4 and IPv6
- DNS resource record configuration corresponding to appropriate name resolution to address(es) for desired tunneling or translation
- Compatible client/host and router support of selected migration technologies, including translation and/or tunneling and application considerations
- Deployment of translation gateway(s) as appropriate

Dual-Stack Approach

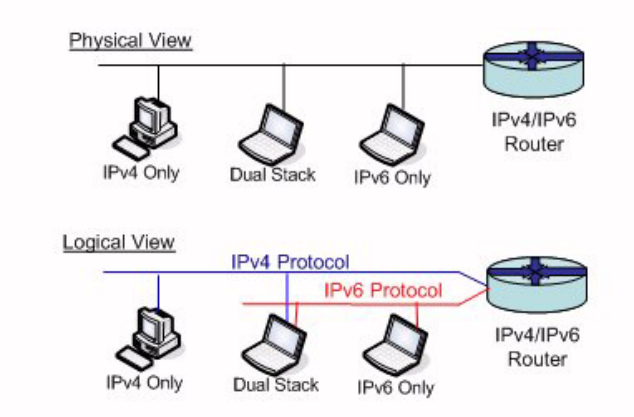
The dual-stack approach consists of implementing both IPv4 and IPv6 protocol stacks on devices requiring access to both network-layer technologies, including routers, other infrastructure devices and end-user devices. Such devices would be configured with both IPv4 and IPv6 addresses, and they may obtain these addresses via methods defined for the respective protocols as enabled by administrators. For example, an IPv4 address may be obtained via DHCPv4, while the IPv6 address may be autoconfigured.

Implementations may vary with dual-stack approaches with respect to the scope of the stack that is shared versus what is unique to each IP version. Ideally, only the network layer would be dualized, using a common application, transport and data link layer. This is the approach being implemented in Microsoft Vista, the newest Microsoft desktop operating system. This contrasts with the Microsoft XP implementation, which utilized dual transport and network layers, requiring, in some cases, redundant configuration by administrators of each stack. Other approaches may span the entire stack down to the physical layer, requiring a separate network interface for IPv6 vs. IPv4. This approach, while contrary to the benefits of a layered protocol model, may be intentional and even desirable, especially in the case of network servers with multiple applications or services, some of which support only one version or the other.

Dual-stack deployment

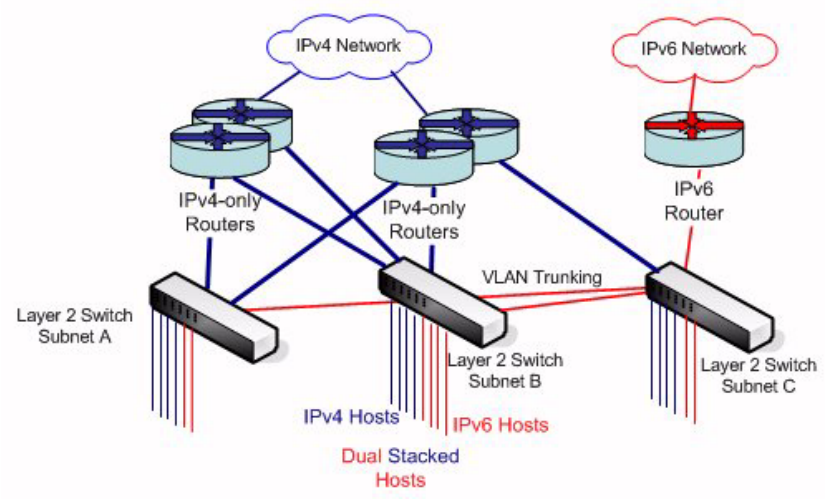
Deployment of dual-stacked devices sharing a common network interface implies the operation of both IPv4 and IPv6 over the same physical link. After all, Ethernet and other layer 2 technologies support either IPv4 or IPv6 payloads. Dual-stacked devices require routers supporting such links to be dual-stacked as well. This overlay approach is expected to be very common during the transition and is depicted in Figure 1. This diagram can be extended beyond a physical LAN to a multi-hop network where routers support IPv4 and IPv6 and route IPv4 packets among native IPv4 hosts and IPv6 packets among IPv6-capable hosts.

Figure 1: Dual-stacked Network Perspectives



While routers would generally be among the first IP elements to be upgraded to support both protocols, RFC 4554 is an informational RFC describing an innovative approach using VLANs to support an overlay configuration without requiring immediate router upgrades. This approach relies on VLAN tagging to enable Layer 2 switches to broadcast or trunk the Ethernet frames containing IPv6 payload to one or more IPv6 capable routers. By upgrading one router to support IPv6, the switch ports to which its interfaces are connected can be configured as the "IPv6 VLAN." Other IPv6 or dual-stacked devices could then be configured as members of the VLAN, and multiple VLANs could be configured likewise. An example of this deployment is displayed in Figure 2.

Figure 2: Dual-stacked VLAN Network



DNS considerations

DNS plays a crucial role in proper operation of each transition technology; after all, it provides the vital linkage between end-user naming and the destination IP address. End users attempting to access a dual-stack device will query DNS, which can be configured by administrators with an A resource record corresponding to the node's IPv4 address and a AAAA resource record corresponding to its IPv6 address. The owner field of the resource record may have a common host name corresponding to the device per the following example:

```
dual-stack-host.example.com.  86400   IN  A     10.200.0.16

dual-stack-host.example.com.  86400   IN  AAAA    4FFE:320:200::A
```

Resolution of the IP-address-to-host name may also be configured in DNS within the appropriate .arpa domain:

```
16.0.200.10.in-addr.arpa.      86400 IN PTR dual-stack-host.example.com.

A.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.0.2.3.0.E.F.F.4.ip6.arpa. 86400

 IN PTR dual-stack-host.example.com.
```

The dual stack node itself must be able to support receipt of A and AAAA records during its own DNS resolution processing and communicate with the intended destination using the address and protocol corresponding to the returned record. Resolver configuration may enable definition of the preferred network protocol when both an A and a AAAA record are returned from the query, not to mention the protocol to use when issuing DNS queries themselves. In addition, as we shall see in the next section on tunneling approaches, some automatic tunneling technologies utilize specific IPv6 address formats, so addresses corresponding to one or more tunneled address formats may also be returned and may be used to the extent that the resolving host supports the corresponding tunneling technology.

DHCP considerations

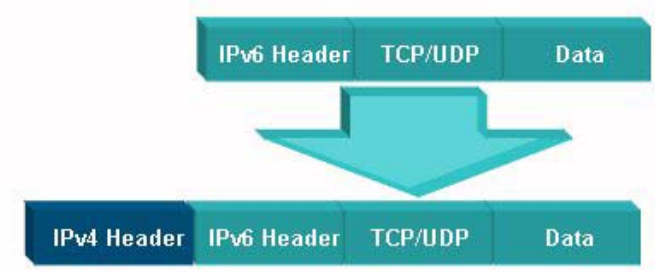
The mechanism for using DHCP under a dual-stack implementation is simply that each stack uses its version of DHCP. That is, in order to obtain an IPv4 address, use DHCP, and to obtain an IPv6 address or prefix, use DHCPv6. However, additional configuration information is provided by both forms of DHCP, such as which DNS or NTP server to use. The information obtained may lead to incorrect behavior on the client depending on how the information from both servers is merged together. This remains an ongoing area of concern, as documented in RFC 4477, but the current standard is to use a DHCP server for IPv4 and a DHCPv6 server for IPv6, possibly implemented on a common physical server.

Tunneling Approaches

A variety of tunneling technologies has been developed to support IPv4 over IPv6 as well as IPv6 over IPv4 tunneling. These technologies are generally categorized as *configured* or *automatic*. Configured tunnels are predefined, whereas automatic tunnels are created and torn down "on the fly." We'll discuss these two tunnel types after reviewing some tunneling basics.

In general, tunneling of IPv6 packets through an IPv4 network entails prefixing each IPv6 packet with an IPv4 header (Figure 3). This enables the tunneled packet to be routed over an IPv4 routing infrastructure. The entry node of the tunnel, whether a router or host, performs the encapsulation.[2] The source IPv4 address in the IPv4 header is populated with that node's IPv4 address and the destination address is that of the tunnel endpoint. The protocol field of the IPv4 header is set to 41 (decimal) indicating an encapsulated IPv6 packet. The exit node or tunnel endpoint performs decapsulation to strip off the IPv4 header and route the packet as appropriate to the ultimate destination via IPv6.

Figure 3: IPv6 over IPv4 Tunneling

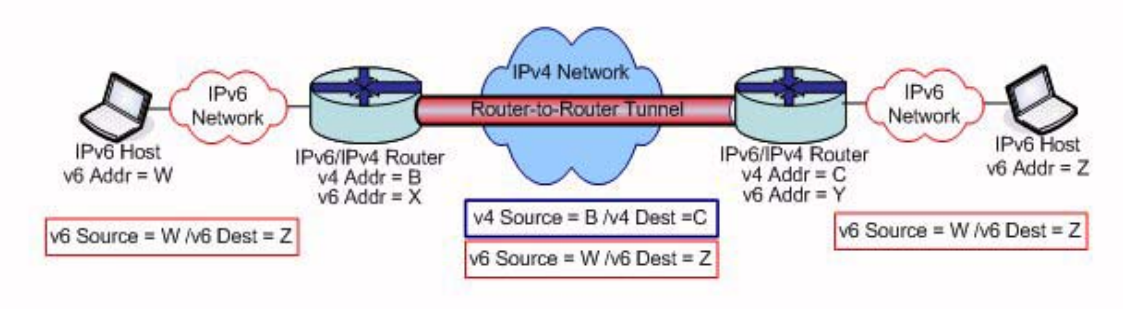


[2] The tunnel encapsulator endpoint must also manage the resulting tunneled packet size with respect to the tunnel's maximum transmission unit (MTU) or packet size, and inform the source if the packet is too large to tunnel.

Tunnel types

While the process of tunneling is the same for all types of tunnels, there is a variety of scenarios based on defined tunnel endpoints. Probably the most common configuration is a router-to-router tunnel, depicted in Figure 4, which is the typical approach for configured tunnels.

Figure 4: Router-to-Router Tunnel

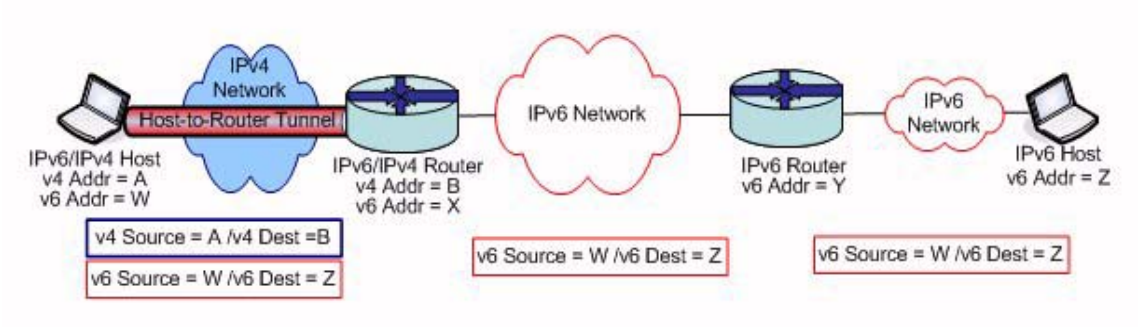


In this figure, the originating IPv6 host on the left has an IPv6 address of *W* (for simplicity and brevity for now). A packet[3] destined for the host on the far end of the diagram with an IPv6 address of *Z* is sent to a router serving the subnet. This router (with an IPv4 address of *B* and an IPv6 address of *X*) receives the IPv6 packet. Configured to tunnel packets destined for the network on which host *Z* resides, the router encapsulates the IPv6 packet with an IPv4 header. The router uses its IPv4 address (*B*) as the source IPv4 address and the tunnel endpoint router (with an IPv4 address of *C*) as the destination address, which is depicted beneath the IPv4 network in the center of the figure. The endpoint router decapsulates the packet, stripping off the IPv4 header and routes the original IPv6 packet to its intended destination (*Z*).

Another tunneling scenario features an IPv6/IPv4 host capable of supporting both IPv4 and IPv6 protocols, tunneling a packet to a router, which in turn decapsulates the packet and routes it natively via IPv6. This flow and packet header addresses are shown in Figure 5. The tunneling mechanism is the same as in the router-to-router case, but the tunnel endpoints are different.

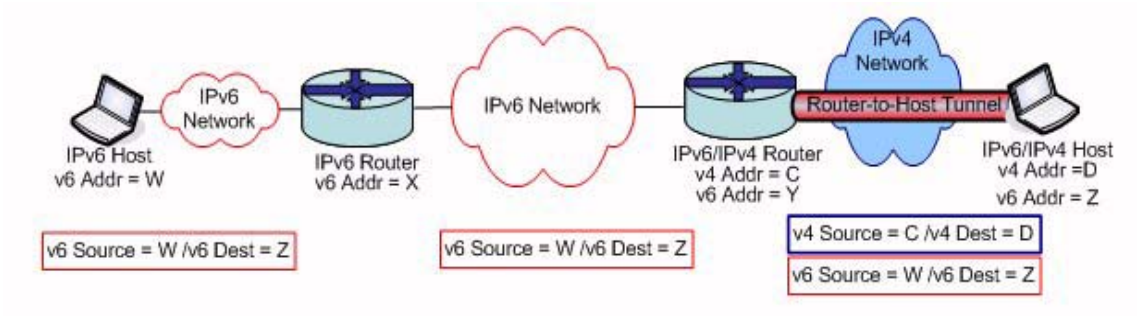
[3] This packet is identified in the figure as the red rectangle beneath the originating host displaying the packet's IPv6 source address of *W* and destination address of *Z*.

Figure 5: Host-to-Router Tunneling Configuration



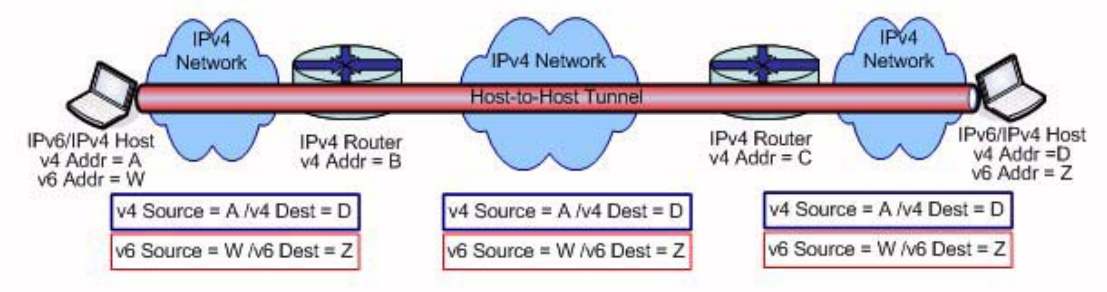
The router-to-host configuration, as shown in Figure 6, is also very similar to router-to-router tunneling. The originating IPv6 host on the left of the diagram sends the IPv6 packet to its local router, which routes it to a router closest to the destination. The serving router is configured to tunnel IPv6 packets over IPv4 to the host, as shown in the figure.

Figure 6: Router-to-Host Tunnel Configuration



The final tunneling configuration is one that spans end-to-end, from host-to-host. If the routing infrastructure has not yet been upgraded to support IPv6, this tunneling configuration enables two IPv6/IPv4 hosts to communicate via a tunnel over an IPv4 network as shown in Figure 7. In this example, the communication is IPv4 from end–to-end.

Figure 7: Host-to-Host Tunnel Configuration



Automatic tunneling of IPv6 packets over IPv4 networks

As mentioned previously, tunnels are either configured or automatic. Configured tunnels are pre-defined by administrators in advance of communications, much as static routes would be pre-configured. In the scenarios described in Figures 4-7, configuration of the respective tunnel endpoints is required to configure the device for when to tunnel IPv6 packets, i.e., based on destination, along with other tunnel configuration parameters that may be required by the tunnel implementation, such as maximum packet size (sometimes called MTU or maximum transmission unit).

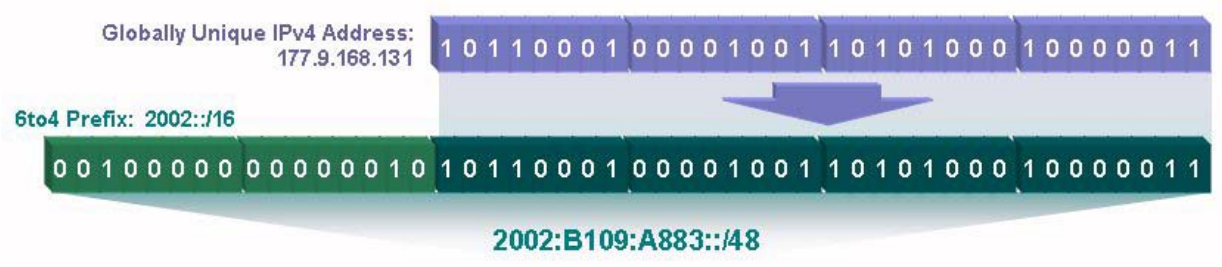
An automatic tunnel does not require pre-configuration. Tunnels are created based on information contained in the IPv6 packet, such as the source or destination IP address. The following automatic tunneling techniques are described in this section:

- 6to4 – automatic router-to-router tunneling based on a particular global address prefix and embedded IPv4 address
- ISATAP – automatic host-to-router, router-to-host or host-to-host tunneling based on a particular IPv6 address format with inclusion of an embedded IPv4 address
- 6over4 – automatic host-to-host tunneling using IPv4 multicasting
- Tunnel Brokers – automatic tunnel setup by a server acting as a tunnel broker in assigning tunnel gateway resources on behalf of hosts requiring tunneling
- Teredo – automatic tunneling through NAT firewalls over IPv4 networks
- Dual-Stack Transition Mechanism – enables automatic tunneling of IPv4 packets over IPv6 networks

6to4

6to4 is an IPv6-over-IPv4 tunneling technique that relies on a particular IPv6 address format to identify 6to4 packets and to tunnel them accordingly. The address format consists of a 6to4 prefix, 2002::/16, followed by a globally unique IPv4 address for the intended destination site. This concatenation forms a /48 prefix as shown in the example in Figure 8

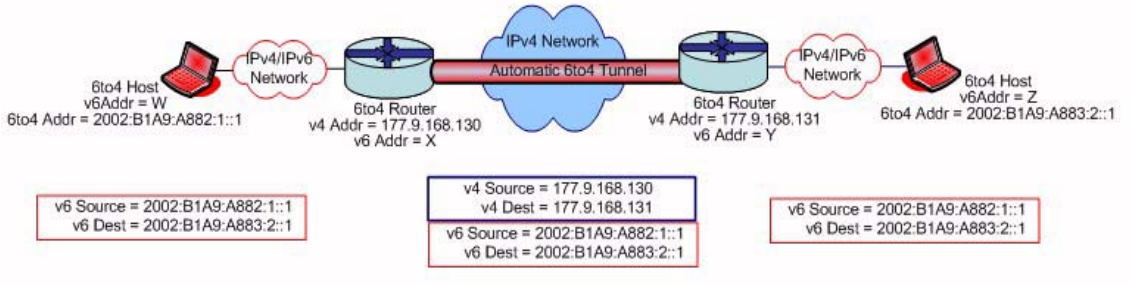
Figure 8: 6to4 Address Prefix Derivation



The unique IPv4 address represents the IPv4 address of the 6to4 router terminating the 6to4 tunnel. The 48-bit 6to4 prefix serves as the global routing prefix, and a Subnet ID can be appended as the next 16 bits, followed by an Interface ID to fully define the IPv6 address. Routers with 6to4 tunneling support (6to4 routers) must be employed, and IPv6 hosts that are to send/receive via 6to4 tunnels must be configured with a 6to4 address and are considered 6to4 hosts.

Let's consider an example where two sites containing 6to4 hosts want to communicate and are interconnected via 6to4 routers connected to a common IPv4 network; this could be the Internet or an internal IPv4 network. Per Figure 9, the IPv4 addresses of the routers' IPv4 interfaces are 177.9.168.130 and 177.9.168.131, respectively. Transforming these IPv4 addresses into 6to4 addresses, we arrive at 2002:B109:A882::/48 and 2002:B109:A883::/48, respectively. These prefixes now identify each site in terms of 6to4 reachability. The 6to4 host on the left is on Subnet ID = 1 and, for simplicity, has Interface ID = 1. Thus, this host's 6to4 address is 2002:B109:A882:1::1. This address would be configured on the device manually or automatically (autoconfiguration based on the devices' interface ID and the router's advertisement of the 2002:B109:A882:1/64 prefix). Similarly, the 6to4 host at the other site resides on subnet ID = 2 and interface ID = 1, resulting in a 6to4 address of 2002:B109:A883:2::1.

Figure 9: 6to4 Tunneling Example



The AAAA and PTR resource records corresponding to these 6to4 addresses should also be added to DNS within the appropriate domains. When our host on the left wants to communicate with the other host, a DNS lookup would return its 6to4 address and potentially other IPv6 address(es). The sending host will use its 6to4 address as the source and the destination 6to4 address as the destination. When this packet is received by the 6to4 router, the router will encapsulate the packet with an IPv4 header using its (source) and the other 6to4 router's (destination) IP addresses. The 6to4 router receiving the packet will decapsulate it and transmit the packet on its 2002:B109:A883:2/64 network to the destination 6to4 host.

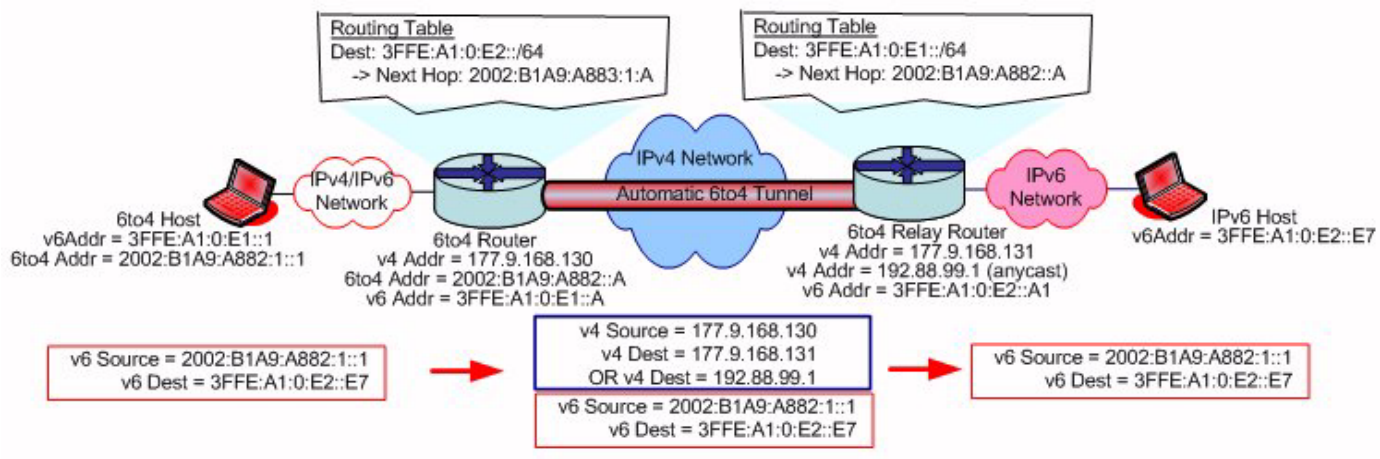
During the IPv4-to-IPv6 migration process, 6to4 can provide an efficient mechanism for IPv6 hosts to communicate over IPv4 networks. As networks are incrementally migrated to IPv6, *6to4 relay routers* (which are IPv6 routers that also support 6to4) can be used to relay packets from hosts on "pure" IPv6 networks to IPv6 hosts via IPv4 networks.

The same addressing and tunneling scheme applies, however, the 6to4 router requires knowledge of the 6to4 relay routers to map global unicast (native) IPv6 addresses to a 6to4 address for tunneling. There are three ways this knowledge may be imparted:

- Configure routes to destination native IPv6 networks with the 6to4 relay router as the next hop. This is illustrated in Figure 10.
- Utilize normal routing protocols, enabling the 6to4 relay router to advertise routes to IPv6 networks. This scenario would apply when advertising routes to migrated or internal IPv6 networks. If the pure IPv6 network in Figure 10 is the IPv6 Internet, the following default route option is likely a better alternative.

- Configure a default route to the 6to4 relay router for IPv6 networks. This scenario may apply where an IPv6 Internet connection is reachable only through an IPv4 network internal to the organization and few or no pure IPv6 networks exist within the organization. A variant on this scenario is to define the default route next hop as the 6to4 relay router anycast address for IPv6 networks. This variation supports the scenario with multiple 6to4 relay routers. RFC 3068 defines an anycast address for 6to4 relay routers: 2002:C058:6301::/48. This address corresponds to the IPv4 address of 192.88.99.1.

Figure 10: 6to4 Host Communicating with a Native IPv6 Host



Reviewing Figure 10, we see a 6to4 host on an in-transition IPv4/IPv6 network on the left with a native IPv6 address and a 6to4 address. This host sets out to communicate with a native IPv6 host with IP address 3FFE:A1:0:E2::E7 on the right side of the figure. This IPv6 address is returned within a AAAA resource record response from a DNS server when queried for the IP address of the destination host. Thus our 6to4 host on the left formulates an IPv6 packet using either its IPv6 or 6to4 (shown) address as the source IP address based on host address selection policies and the destination host's IPv6 address as the destination.

This packet then arrives at the 6to4 router. The router would need to have a routing table entry in order to route to the destination 3FFE:A1:0::E2/64 network, pointing to the 6to4 relay router's 6to4 address as shown in the figure. The 6to4 router then creates an automatic tunnel to the corresponding IPv4 address contained in bits 17-48 of the 6to4 address found in the routing table. Note that as just discussed, this routing table entry could alternatively be a default route for IPv6 packets to the 6to4 relay router's 6to4 unicast address or the 6to4 anycast address. While not shown in the router's routing table in the figure, the tunneled packet headers depicted below the IPv4 network cloud indicates that the destination IPv4 address encapsulating the IPv6 packet could be the IPv4 unicast address or the IPv4 address corresponding to the 6to4 unicast address. The 6to4 relay router, upon receiving the IPv4 packet, would then decapsulate it and transmit the native IPv6 packet to the intended recipient.

In the reverse direction, use of the recipient's 6to4 address as the destination IPv6 address would inform the 6to4 relay that this packet requires 6to4 tunneling to the corresponding 6to4 router. However, if the destination address is a native IPv6 address, the routing table within the 6to4

relay router must contain a mapping of the 6to4 address of the corresponding 6to4 router as the next hop towards the IPv6 host.

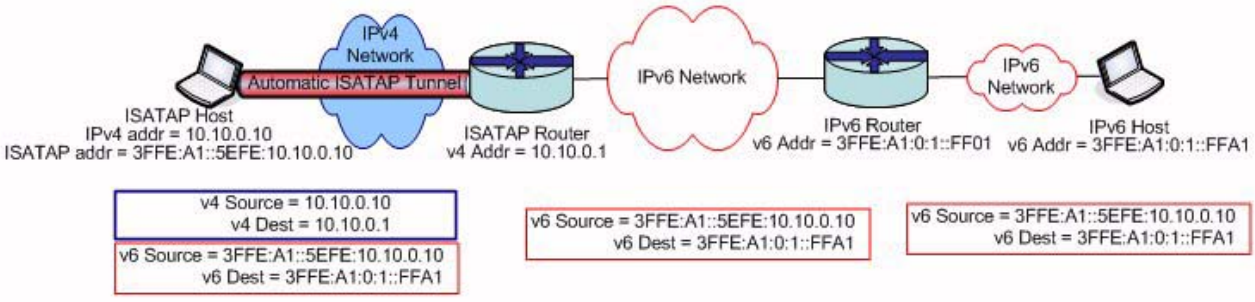
Intra-Site Automatic Tunneling Addressing Protocol (ISATAP)

ISATAP is an experimental protocol providing automatic IPv6-over-IPv4 tunneling for host-to-router, router-to-host and host-to-host configurations. ISATAP IPv6 addresses are formed using an IPv4 address to define its interface ID. The Interface ID is comprised of ::5EFE:a.b.c.d, where a.b.c.d is the dotted decimal IPv4 notation. So an ISATAP interface ID corresponding to 177.9.168.131 is denoted as ::5EFE:177.9.168.131. The IPv4 notation provides a clear indication that the ISATAP address contains an IPv4 address without having to translate the IPv4 address into hexadecimal. This ISATAP interface ID can be used as a normal interface ID in appending it to supported network prefixes to define IPv6 addresses. For example, the link local IPv6 address using the ISATAP interface ID above is FE80::5EFE:177.9.168.131.

Hosts supporting ISATAP are required to maintain a *potential router list* (PRL) containing the IPv4 address and associated address lifetime timer for each router advertising an ISATAP interface. ISATAP hosts solicit ISATAP support information from local routers via router solicitation over IPv4. The solicitation destination needs to be identified by the host by prior manual configuration by looking up the router in DNS with a hostname of "`isatap`" in the resolver's domain, or using a DHCP option indicating the IPv4 address(es) of the ISATAP router(s). The DNS technique requires administrators to create resource records for ISATAP routers using the `isatap` hostname.

An ISATAP host would encapsulate the IPv6 data packet with an IPv4 header as shown in Figure 11 using the IPv4 address corresponding to the chosen router from the PRL.

Figure 11: ISATAP Host-to-Router Example



ISATAP hosts can auto-configure their ISATAP interface IDs using configured IPv4 addresses whether the IPv4 address is defined statically or is obtained via DHCP. Microsoft XP and Windows 2003 servers perform such autoconfiguration if configured with IPv6. Microsoft Vista clients and Longhorn servers will support ISATAP autoconfiguration by default. The ISATAP interface ID can be appended to a 64-bit global network prefix and subnet ID provided by solicited ISATAP routers in their advertisement.

Reviewing Figure 11, the host on the left of the diagram identifies the destination host's IP address, in this case an IPv6 address, using DNS. An IPv6 packet would be formed by the host using its ISATAP IPv6 address as its source address and the destination IPv6 host address as the destination address. This packet is encapsulated in an IPv4 header, thereby forming an automatic tunnel. The source address is set to the ISATAP host's IPv4 address, the destination address is set to the ISATAP router's IPv4 address and the protocol field is set to decimal 41, indicating an encapsulated IPv6 packet. The ISATAP router need not be on the same physical network as the host, and the tunnel can span a generic IPv4 network (zero or more hops) between the host and the ISATAP router. The ISATAP router strips off the IPv4 header and routes the remaining IPv6 packet to the destination host using normal IPv6 routing.

The destination host can respond to the originating host using the originating host's ISATAP address. Since the ISATAP address contains a globally unique network prefix/Subnet ID, the destination packet is routed to the serving ISATAP router. Upon processing the interface ID, the ISATAP router can extract the IPv4 address of the destination host and encapsulate the IPv6 packet with an IPv4 header to the original host. In a like manner, the native IPv6 host to the right of Figure 11 could have initiated the communication to the ISATAP host. Going from right to left, the ISATAP router in this case would create the ISATAP tunnel to the host.

Similar to the host-to-host tunnel shown in Figure 7, host-to-host ISATAP tunnels can be initiated by ISATAP hosts residing on an IPv4 network, where a link-local (same subnet) or global network prefix can be prefixed to each host's ISATAP interface ID. In Figure 7, IPv6 addresses W and Z would represent ISATAP addresses formed from IPv4 addresses A and D respectively.

6over4

6over4 is an automatic tunneling technique that leverages IPv4 multicast. IPv4 multicast is required and is considered a *virtual link layer* or *virtual Ethernet* by the 6over4 scheme. Because of the virtual link layer perspective, IPv6 addresses are formed using a link local scope (FE80::/10 prefix). A host's IPv4 address comprises the 6over4 interface ID portion of its IPv6 address. For example, a 6over4 host with IPv4 address of 192.223.16.85 would formulate an IPv6 interface ID of ::C0DF:1055, and thus a 6over4 address of FE80::C0DF:1055. 6over4 tunnels can be of the form host-to-host, host-to-router and router-to-host, where respective hosts and routers must be configured to support 6over4. IPv6 packets are tunneled in IPv4 headers using corresponding IPv4 multicast addresses. All members of the multicast group receive the tunneled packets, thus the analogy of virtual Ethernet, and the intended recipient strips off the IPv4 header and processes the IPv6 packet. As long as at least one IPv6 router also running 6over4 is reachable via the IPv4 multicast mechanism, the router can serve as a tunnel endpoint and route the packet via IPv6.

6over4 supports IPv6 multicast as well as unicast, so hosts can perform IPv6 router and neighbor discovery to locate IPv6 routers. When tunneling IPv6 multicast messages, e.g., for neighbor discovery, the IPv4 destination address is formatted as 239.192.Y.Z, where Y and Z are the last two bytes of the IPv6 multicast address. Thus an IPv6 message to the all-routers, link-scoped, multicast address FF02::2 would be tunneled to IPv4 destination 239.192.0.2. The Internet Group Membership Protocol (IGMP) is used by 6over4 hosts to inform IPv4 routers of multicast group membership.

Tunnel brokers

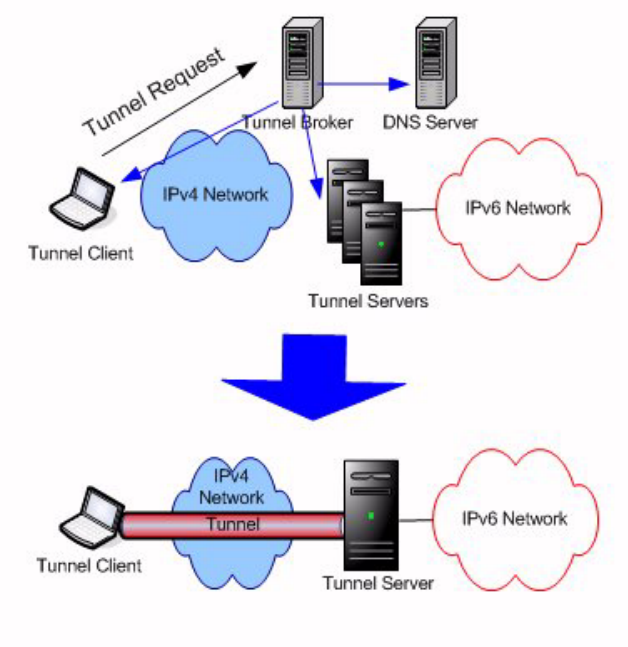
Tunnel brokers provide another technique for automatic tunneling over IPv4 networks. The tunnel broker manages tunnel requests from dual-stack clients and tunnel-broker servers, which connect to the intended IPv6 network. Dual-stack clients attempting to access an IPv6 network can optionally be directed via DNS name resolution to a tunnel broker web server for entry of authentication credentials to authorize use of the broker service. The tunnel broker may also manage certificates for authorization services. The client also provides the IPv4 address for its end of the tunnel, along with the desired FQDN of the client, the number of IPv6 addresses requested and whether the client is a host or a router.

Once authorized, the tunnel broker performs the following tasks to broker creation of the tunnel:

- Assigns and configures a tunnel server and informs the selected tunnel server of the new client
- Assigns an IPv6 address or prefix to the client based on the requested number of addresses and client type (router or host)
- Registers the client FQDN in DNS
- Informs the client of its assigned tunnel server and associated tunnel and IPv6 parameters including address/prefix and DNS name.

From an end-user perspective, setting up the tunnel connection to the IPv6 network appears similar to setting up a standard VPN connection. Figure 12 illustrates the client-tunnel broker interaction at the top of the figure, and the resulting tunnel between the client and the assigned tunnel server below.

Figure 12: Tunnel Broker Interaction



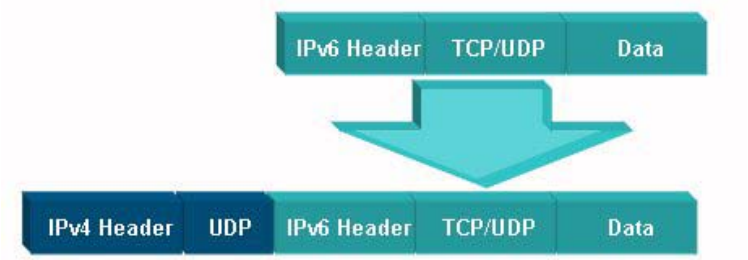
Teredo

Tunneling through firewalls that perform Network Address Translation (NAT) can be challenging, if not impossible. This is so because many NAT/firewall devices will not allow traversal of IPv4 packets with the protocol field set to 41, which is the setting for tunneling of IPv6 packets as described previously. Teredo enables NAT traversal of IPv6 packets tunneled over UDP over IPv4 for host-to-host automatic tunnels. Teredo incorporates the additional UDP header in order to facilitate NAT/firewall traversal. The additional UDP header further "buries" the tunnel to enable its traversal through NAT/firewall devices, most of which support UDP port translation.

Teredo is defined in RFC 4380 to provide "IPv6 access of last resort" due to its overhead, and will be used less and less as 6to4-enabled or IPv6-aware firewall routers are deployed. Teredo requires the following elements as shown in Figure 13:

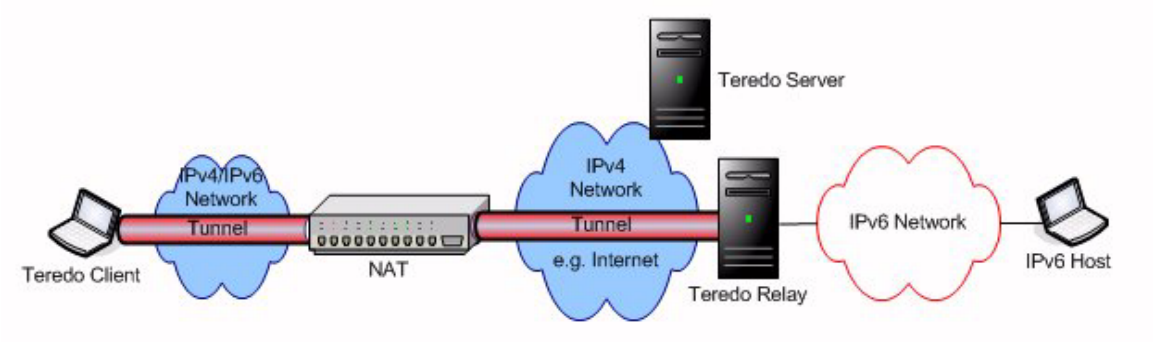
- Teredo client
- Teredo server
- Teredo relay

Figure 13: Teredo Tunnels add UDP then IPv4 Headers



The Teredo tunneling process starts with a Teredo client performing a qualification procedure to discover a Teredo relay closest to the intended destination IPv6 host and identify the type of intervening NAT firewall that is in place. Teredo hosts must be pre-configured with Teredo server IPv4 addresses to use. Determining the closest Teredo relay entails sending a ping (ICMPv6 echo request) to the destination host. The ping is encapsulated with a UDP and IPv4 header and sent to the Teredo server, which decapsulates and sends the native ICMPv6 packet to the destination. The destination host's response will be routed via native IPv6 to the nearest Teredo relay by virtue of routing metrics, then back to the originating host. In this manner, the client determines the appropriate Teredo relay's IPv4 address and port. Figures 14 and 15 illustrate the case with a Teredo client communicating to a native IPv6 host.

Figure 14: Teredo Client-to-IPv6 Host Connection



The type of intervening NAT may drive the need to perform an additional step to initialize NAT table mappings. The following NAT types have generally been defined:

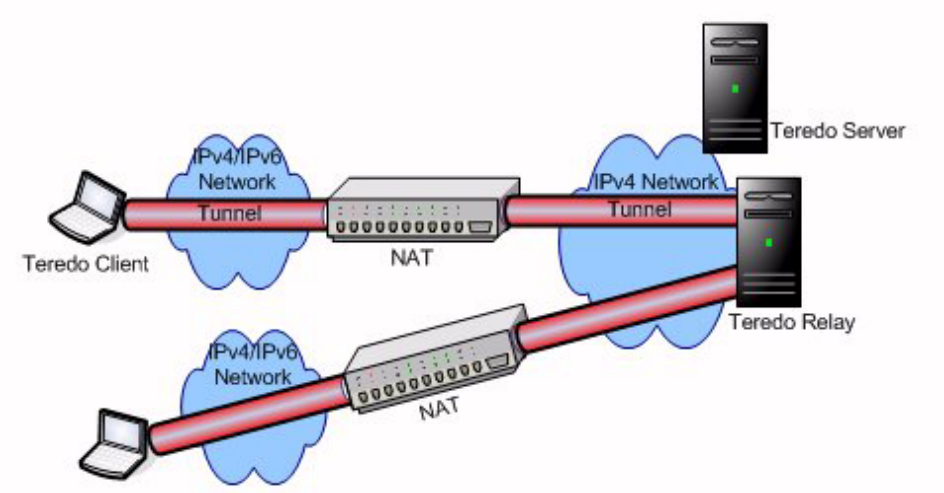
- *Full cone* – all IP packets from the same internal IP address and port are mapped by the NAT to a corresponding external address and port. External hosts can communicate with the host by simply transmitting to the mapped external address and port.
- *Restricted cone* – all IP packets from the same internal IP address and port are mapped by the NAT to a corresponding external address and port. An external host can communicate with the internal host only if the internal host had previously sent a packet to the external host. This form restricts unsolicited incoming packets.
- *Port restricted cone* – all IP packets from the same internal IP address and port are mapped by the NAT to a corresponding external address and port. External hosts can communicate with the internal host only if the internal host had previously sent a packet to the external host using the external host's address and the same port number.
- *Symmetric* – all IP packets from a given internal IP address and port to a specific external IP address and port are mapped to a particular IP address and port. Packets from the same host IP address and port to a different destination IP address or port results in a different external IP address and port mapping. External hosts can communicate with the internal host only if the internal host had previously sent a packet to the external host using the external host's address and the same port number.

Identification of the NAT as full cone requires no further qualification, but both of the restricted cone scenarios do. Teredo does not support traversal of symmetric NAT devices. To complete the mapping within the NAT of the internal host communiqué with the destination host, a *bubble packet*, which is an IPv6 header with no payload, is sent to the host. A bubble packet enables the NAT to complete the mapping of internal and external IP addresses and internal and external port numbers for the port-restricted cone scenario.

Generally, the bubble packet is sent directly from the source Teredo client to the destination host. But if the destination host is also behind a firewall, the bubble packet may be discarded. Thus, the Teredo client times-out, then sends the bubble packet to the destination via the Teredo server. Assuming the destination host is also a common Teredo client, a firewall mapping should already exist on the destination side between the destination host and the Teredo server. The destination host should receive the bubble packet, then respond to the originating host directly, completing

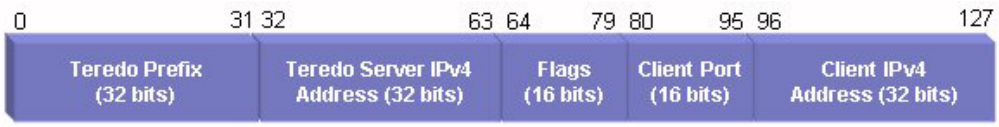
the firewall mapping from the destination and the originator on both NAT devices. Figure 15 illustrates this scenario with two Teredo clients communicating via a common Teredo relay.

Figure 15: Two Teredo Clients Communicating via the IPv4 Internet



The Teredo IPv6 address has the format shown in Figure 16.

Figure 16: Teredo IPv6 Address Format



The Teredo prefix is a pre-defined IPv6 prefix: 2001::/32. The Teredo server IPv4 address comprises the next 32 bits. Flags indicate the type of NAT as either full cone (value = 0x8000) or restricted or port-restricted (value = 0x0000). The client port and client IPv4 address fields represent obfuscated values of these respective values by reversing each bit value.

Automatic tunneling of IPv4 packets over IPv6 networks

During the IPv6 transition, some IPv6 clients on IPv6 networks may still need to communicate with IPv4 applications or hosts on IPv4 networks, such as the Internet. Tunneling of IPv4 packets over the IPv6 network provides a means to preserve this communications path.

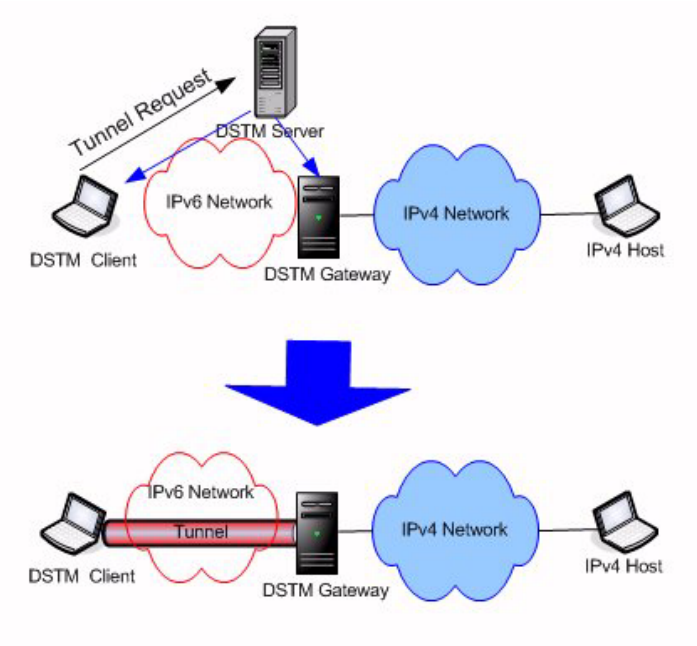
Dual-Stack Transition Mechanism (DSTM)

DSTM provides a means to tunnel IPv4 packets over IPv6 networks, ultimately to the destination IPv4 network and host. The host on the IPv6 network intending to communicate to the IPv4 host would require a DSTM client. Upon resolving the hostname of the intended destination host to

only an IPv4 address, the client would initiate the DSTM process, which is very similar to the tunnel-broker approach, and is illustrated in Figure 17. The process begins with the DSTM client contacting a DSTM server to obtain an IPv4 address, preferably via the DHCPv6 protocol,[4] as well as the IPV6 address of the DSTM gateway. The IPv4 address is used as the source address in the data packet to be transmitted. This packet is encapsulated with an IPv6 header using the DSTM client's source IPv6 address and the DSTM gateway's IPv6 address as the destination. The next field in the IPv6 header indicates an encapsulated IPv4 packet with this 4over6 tunneling approach.

A variant of DSTM supports VPN-based access from a DSTM client outside of the native network, e.g., a home-based worker. In this scenario, assuming the DSTM client obtains an IPv6 address but no IPv4 address, it can connect to the DSTM server to obtain an IPv4 address. This access should require authentication to establish a VPN between the DSTM client and DSTM gateway.

Figure 17: DSTM Tunnel Setup



[4] While DSTM RFC drafts denote DHCPv6 as the preferable method to obtain an IPv4 address, DHCPv6 does not currently define assignment of IPv4 addresses natively or via an option setting.

Tunneling summary

Table 1 summarizes the applicability of tunneling based on the source host capabilities/network type and the destination address resolution and network type. The green shaded cells in the table indicate use of a native IP version from end to end. Any intervening networks of the opposite protocol must be either tunneled through via a router-to-router protocol or translated at each boundary using a translation technology, discussed in the next section. The yellow shaded cells indicate a tunneling scenario, including configured tunnels. The "→" symbol represents a transition point or tunneling endpoint within the network that converts the corresponding native protocol to a tunneled protocol or vice versa. The red shaded cells indicate an invalid connection option via tunneling. However, translation technologies could be employed to bridge these gaps.

Table 1: Tunneling Scenarios

To→ *↓From*	IPv4 destination on IPv4 network	Dual stack destination on IPv4 network resolved to IPv4 address	Dual stack destination on IPv4 network resolved to IPv6 address	Dual stack destination on IPv6 network resolved as IPv4 address	Dual stack destination on IPv6 network resolved as IPv6 address	IPv6 destination on IPv6 network
IPv4 client on IPv4 network	Native IPv4	Native IPv4	N/A	Native IPv4 → IPv4-compatible	N/A	N/A
Dual stack client on IPv4 network	Native IPv4	Native IPv4	Host-to-host IPv6 over IPv4*	Native IPv4 → IPv4-compatible	Host-to-router IPv6 over IPv4*	Host-to-router IPv6 over IPv4*
Dual stack client on IPv6 network	DSTM → Native IPv4	DSTM → Native IPv4	Native IPv6→ Router-to-host IPv6 over IPv4*	DSTM	Native IPv6	Native IPv6
IPv6 client on IPv6 network	N/A	N/A	Native IPv6 → IPv6 over IPv4*	N/A	Native IPv6	Native IPv6

* Resolution to an IPv6 address could be a native IPv6 address, or a 6to4, ISATAP, Teredo, 6over4 or IPv4-compatible address. The host must select the destination address based on its corresponding feature support.

Translation Approaches

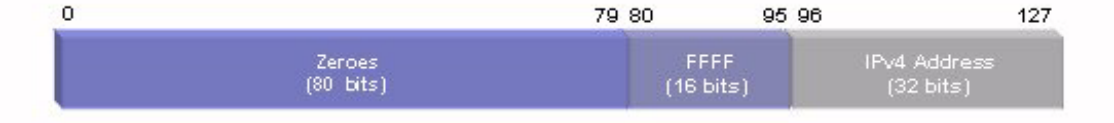
Translation techniques perform IPv4-to-IPv6 translation (and vice versa) at a particular layer of the protocol stack, typically the network, transport or application layer. Unlike tunneling, which does not alter the tunneled data packet, translation mechanisms do modify or translate IP packets commutatively between IPv4 and IPv6. Translation approaches are generally recommended in an environment with IPv6-only nodes communicating with IPv4-only nodes, i.e., for the red-shaded cell scenarios in Table 1. In dual-stack environments, native or tunneling mechanisms are preferable[5].

[5] Per RFC 2766.

Stateless IP/ICMP Translation (SIIT) algorithm

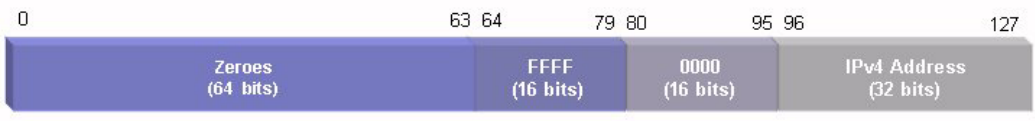
SIIT provides translation of IP packet headers between IPv4 and IPv6. SIIT resides on an IPv6 host and converts outgoing IPv6 packet headers into IPv4 headers, and incoming IPv4 headers into IPv6. To perform this task, the IPv6 host must be assigned an IPv4 address either configured on the host or obtained via a network service left unspecified in RFC 2765. When the IPv6 host desires to communicate with an IPv4 host, based on DNS resolution to an IPv4 address, the SIIT algorithm would convert the IPv6 packet header into IPv4 format. The SIIT algorithm recognizes such a case when the IPv6 address is an IPv4-mapped address, formatted as shown in Figure 18. The mechanism to convert the resolved IPv4 address into an IPv4-mapped address is provided by bump-in-the-stack (BIS) or bump-in-the-API (BIA) techniques described later in this paper.

Figure 18: IPv4 Mapped Address Format



Based on the presence of the IPv4-mapped address format as the destination IP address, the SIIT algorithm performs header translation as described in Table 2 to yield an IPv4 packet for transmission via the data link and physical layers. The source IP address uses a different format, that of the IPv4-translated format, shown in Figure 19. The IPv4-mapped address format is invalid as a source address for tunneling (per RFC 4213). Therefore, its use as the source address would disqualify communications through any intervening tunnels. The use of the IPv4-translated format bypasses this potential restriction.

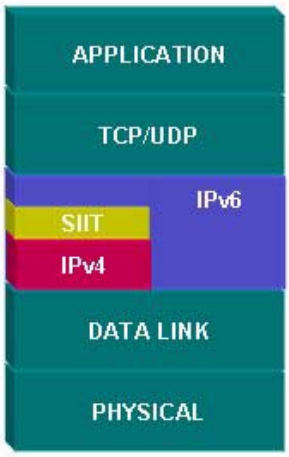
Figure 19: IPv4-Translated Address Format used within SIIT



.

An example protocol stack view of the SIIT algorithm is shown in Figure 20, though it is typically wrapped within a bump-in-the-stack or bump-in-the-API solution, discussed in the next section.

Figure 20: SIIT Stack Example



The basic header translation process for both directions is summarized in Table 2. The left-hand column highlights the resulting IPv6 header field values based on translating an IPv4 header, while the right-hand column details the resulting IPv4 header fields based on translating an IPv6 header.

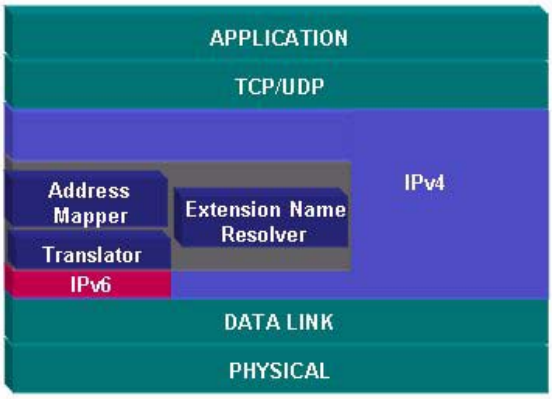
Table 2: Header Translation Process

IPv4 -> IPv6 Header Translation	IPv6 -> IPv4 Header Translation
Version = 6 **Traffic Class** = IPv4 header TOS bits **Flow Label** = 0 **Payload Length** = IPv4 header Total Length value – (IPv4 header length + IPv4 options length) **Next Header** = IPv4 header Protocol field value **Hop Limit** = IPv4 TTL field value - 1 **Source IP Address** = 0:0:0:0:FFFF::/80 concatenated with IPv4 header source IP address **Destination IP Address** = 0:0:0:0:0:FFFF::/96 concatenated with IPv4 header destination IP address	**Version** = 4 **Header Length** = 5 (no IPv4 options) **Type of Service** = IPv6 header Traffic Class field **Total Length** = IPv6 header payload length field + IPv4 header length **Identification** = 0 **Flags** = Don't Fragment = 1, More Fragments = 0 **Fragment Offset** = 0 **TTL** = IPv6 Hop Limit field value - 1 **Protocol** = IPv6 Next Header field value **Header Checksum** = Computed over the IPv4 header **Source IP Address** = low order 32 bits of IPv6 Source IP Address field (IPv4-translated address) **Destination IP Address** = low order 32 bits of IPv6 Destination IP Address field (IPv4-mapped address) **Options** = None

Bump-in-the-Stack (BIS)

BIS enables hosts using IPv4 applications to communicate over IPv6 networks. BIS snoops data flowing between the TCP/IPv4 module and link layer devices (e.g., network interface cards) and translates the IPv4 packet into IPv6. The components of BIS are shown in Figure 21.

Figure 21: Bump-in-the-Stack Components



The Translator component translates the IPv4 header into an IPv6 header according to the SIIT algorithm discussed previously. The Extension Name Resolver snoops DNS queries for A record types; upon detecting such a query, the Extension Name Resolver creates an additional query for both A and AAAA record types for the same host name. If no affirmative answer is received from the AAAA query, the communication ensues using IPv4; if the AAAA query is resolved, the Extension Name Resolver instructs the Address Mapper component to associate the returned IPv4 address (A record) with the returned IPv6 address (AAAA record). If only a AAAA response is received, the Address Mapper assigns an IPv4 address from a configured pool of addresses.

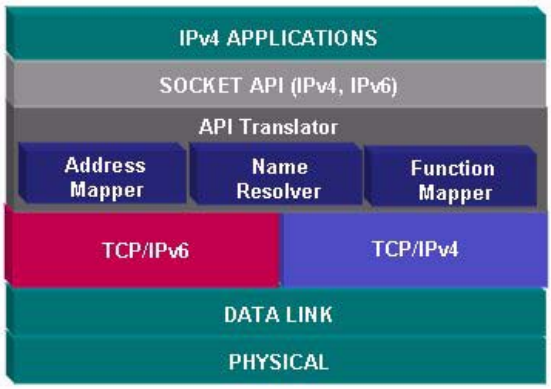
The IPv4 address is needed in order to provide a response up the stack to the application requesting resolution to the A query. Thus, the Address Mapper maintains the association of the real or self-assigned IPv4 address with the IPv6 address of the destination. Any data packets destined to the IPv4 address are then translated into IPv6 packets for transmission via IPv6 networks.

In the case of the BIS host receiving an IPv6 packet initiated from an external host that is not already mapped, the Address Mapper will assign an IPv4 address from its pool and translate the IPv6 header into IPv4 for communication up the stack.

Bump-in-the-API (BIA)

The BIA strategy enables the use of IPv4 applications while communicating over an IPv6 network. Unlike IP header modification provided by BIS, the BIA approach translates between IPv4 and IPv6 APIs. BIA is implemented between the application and TCP/UDP layer of the stack on the host and consists of an API Translator, an Address Mapper, Name Resolver and Function Mapper, as depicted in Figure 22.

Figure 22: Bump-in-the-API



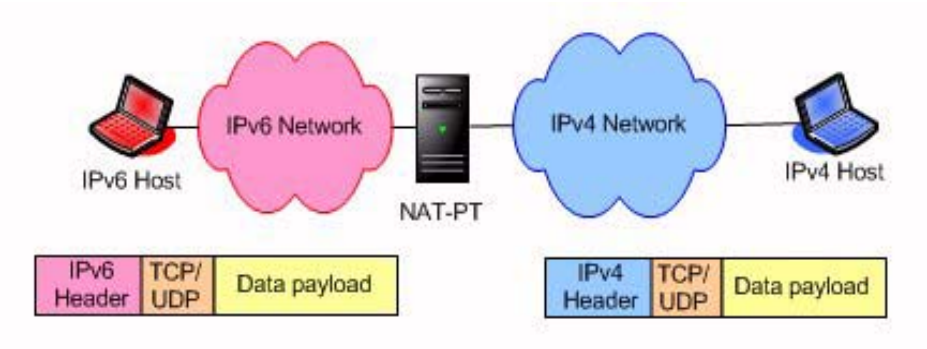
When the IPv4 application sends a DNS query to determine the IP address of a destination host, the Name Resolver intercepts the query and creates a new query requesting both A and AAAA records. A DNS reply with an A record will provide the answer with the given IPv4 address. A reply with only a AAAA record stimulates the name resolver to request an IPv4 address from the Address Mapper to map to the returned IPv6 address. The Name Resolver utilizes the mapped IPv4 address to return an A record response to the application. The Address Mapper maintains this mapping of IPv6 addresses to those assigned from an internal address pool consisting of the unassigned IPv4 address space (0.0.0.0/24). The Function Mapper intercepts API function calls and maps IPv4 API calls to IPv6 socket calls.

Network Address Translation with Protocol Translation (NAT-PT)

As the name implies, the NAT-PT process, which is still in the experimental stage, entails translating IPv4 addresses into IPv6 addresses, like a familiar IPv4 NAT, but also performs protocol header translation as described previously in the SIIT section.[6] A NAT-PT device serves as a gateway between an IPv6 network and an IPv4 network and enables native IPv6 devices to communicate with hosts on the IPv4 Internet, for example. The NAT-PT device maintains an IPv4 address pool and associates a given IPv4 address with an IPv6 address while the communicatios ensues. Figure 23 illustrates the architecture of a NAT-PT deployment.

[6] This excepts the source and destination IP address fields, which are governed by associations within the NAT-PT gateway.

Figure 23: NAT-PT Deployment Example



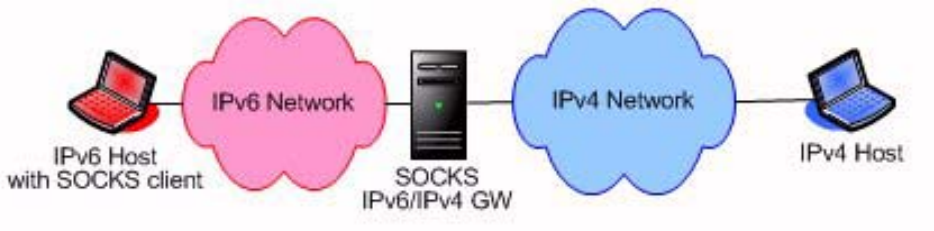
Network Address Port Translation with Protocol Translation (NAPT-PT)

NAPT-PT enables IPv6 nodes to communicate with IPv4 nodes using a single IPv4 address. Thus, in Figure 23, instead of maintaining a one-to-one association of an IPv6 address and a unique IPv4 address as in NAT-PT, NAP-PT maps each IPv6 address to a common IPv4 address with a unique TCP or UDP port value set in the corresponding IPv4 packet. The use of a single, shared IPv4 address eliminates the possibility of IPv4 address pool depletion under the NAT-PT scenario.

SOCKS IPv6/IPv4 Gateway

SOCKS, defined in RFC 1928, provides transport relay for applications traversing firewalls, effectively providing application proxy services. RFC 3089 applies the SOCKS protocol for translating IPv4 and IPv6 communications. And like the other translation technologies already discussed, this approach includes special DNS treatment, termed *DNS name resolving delegation*, which delegates name resolution from the resolver client to the SOCKS IPv6/IPv4 gateway. An IPv4 or IPv6 application can be "socksified" to communicate with the SOCKS gateway proxy for ultimate connection to a host supporting the opposite protocol. Figure 24 illustrates the case of an IPV6 host with a SOCKS client connecting to an IPv4 host, from left to right. A socksified IPv4 host could just as well communicate via the SOCKS gateway to an IPv6 host, from right to left.

Figure 24: Basic SOCKS Gateway Configuration

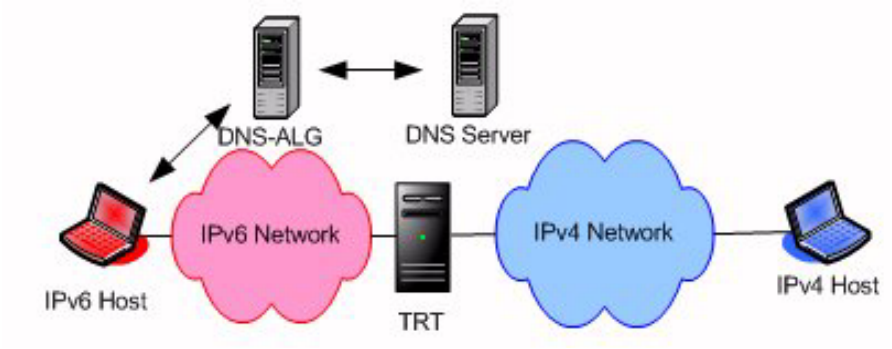


Transport Relay Translator (TRT)

Much like the SOCKS configuration, TRT features a stateful gateway device that interlinks two "independent" connections over different networks. The TCP/UDP connection from a host terminates on the TRT, and the TRT creates a separate connection to the destination host and relays between the two connections. TRT requires a DNS-Application Layer Gateway (DNS-ALG[7]), which acts as a DNS proxy. TRT is specified to enable IPv6 hosts to communicate with IPv4 destinations, e.g., web servers.

As such, the primary function of the DNS-ALG is to perform a AAAA resource record query as requested by IPv6 resolvers; if a AAAA record is returned, the reply is passed on to the resolver and the data connection ensues as an IPv6 connection. If no AAAA records are returned, the DNS-ALG performs an A record query, and if an answer is received, the DNS-ALG formulates an IPv6 address using the IPv4 address contained in the returned A record. RFC 3142, which defines TRT as an informational RFC, specifies use of the prefix C6::/64 followed by 32 zeroes plus the 32-bit IPv4 address. However, IANA has not allocated the C6::/64 prefix. Thus a locally configured prefix would be required.

Figure 25: TRT Configuration with DNS-ALG



Application Layer Gateway (ALG)

ALGs perform protocol translation at the application layer and perform application proxy functions similar to HTTP proxies. An application would typically need to be configured with the IP address of the proxy server, to which a connection would be made upon opening the application, e.g., web browser for the HTTP proxy case. An ALG may be useful for web or other application-specific access to the IPv4 Internet by hosts on an IPv6-only network.

Application Migration

The de facto application program interface (API) for TCP/IP applications is the *sockets* interface originally implemented on BSD UNIX (on which BIND was also originally implemented). The sockets interface defines program calls to enable applications to interface with TCP/IP layers to

[7] Sometimes referred to as "trick-or-treat DNS-ALG" or totd.

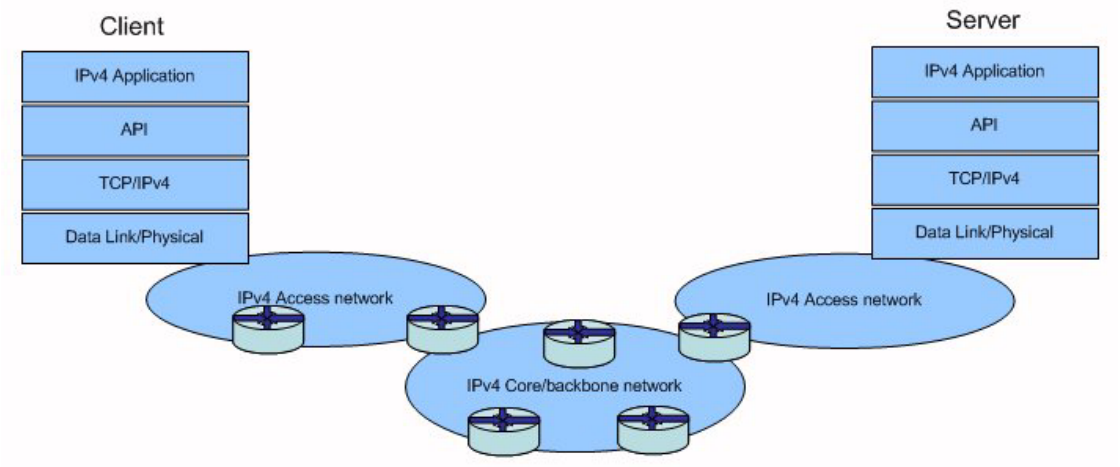
communicate over IP networks. Microsoft's Winsock API is also based on the sockets interface. Both sockets interfaces have been modified to support IPv6's longer address size and additional features. In fact, most major operating system providers have implemented support for sockets or Winsock including Microsoft (XP SP1, Server 2003), Solaris (8+), Linux (kernel 2.4+), Mac OS (X.10.2), AIX (4.3+) and HP-UX (11i with upgrade). The updated sockets interface supports both IPv4 and IPv6 and provides the ability for IPv6 applications to interoperate with IPv4 applications by use of IPv4-mapped IPv6 addresses. Microsoft's Vista and Longhorn releases will provide improved IPv6 integration as well as Windows applications support for IPv6. Check with the application vendors for IPv6 compatibility and requirements.

Migration Scenarios

There is certainly no shortage of technology options when considering a migration from IPv4 to IPv6. Having many options is good, but can be confusing. Selecting the right path will depend on your current environment in terms of end-user devices and operating systems, router models and versions, as well as key applications, budget and resources, not to mention schedule constraints. In this section, we'll review some basic migration scenarios to provide a flavor for macro-level approaches. In reviewing these, we'll use the basic diagram in Figure 26 as a baseline. In this figure, we have a client with IPv4 applications, IPv4 sockets API and TCP/IPv4 stack, and a server with a comparable configuration. We've split the interconnection network into access networks for the client and server, respectively, and a core or backbone network.

Note that this basic diagram illustrates a pair-wise, client-server connection. For a given use case, this could represent an internal client accessing an internal server via an all-internal access and core network. But it could also represent an internal client and internal server communicating over the Internet, an internal client communicating to an Internet-based server or an external client communicating via the Internet to an internal server. This convention simplifies the sheer number of unique use cases. Nuances among these variations will be pointed out as appropriate when describing the migration scenarios.

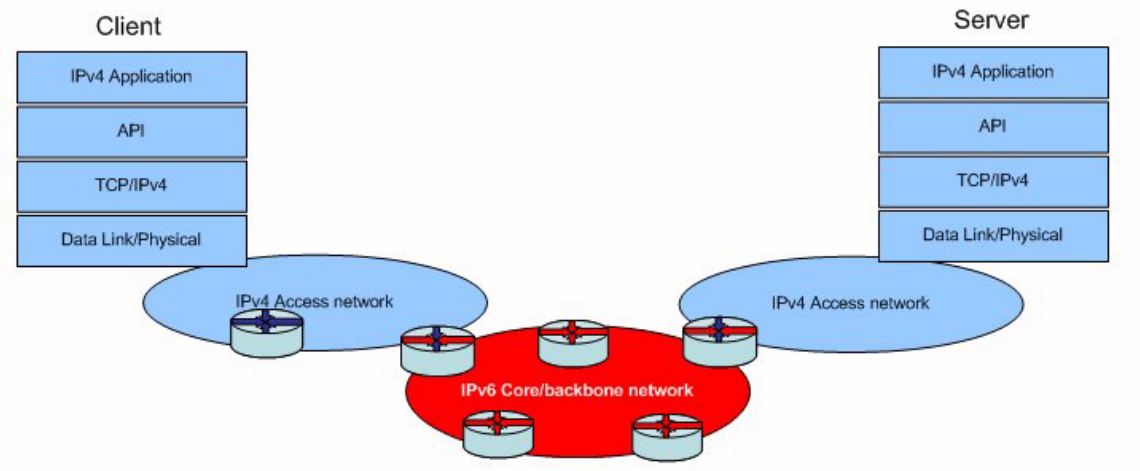
Figure 26: Base Case for IPv4 Network – Initial State Prior to Migration



Core migration

The core migration scenario involves first migrating the backbone or core network to IPv6. This requires upgrading all core routers to support IPv6 routing and routing protocols and for access-to-core boundary routers to support dual stacks. The core network could be an internal backbone or an IPv6 ISP network. A common implementation approach for the access-core boundary routers is to employ configured tunnels between them. This enables these boundary routers to advertise IPv4 routes and tunnel IPv4 packets across the IPv6 backbone. Alternatively, translation gateways could be used at these boundary points. Either way, this approach should have little to no effect on the client or server devices or software, and could provide a starting point for IPv6 experimentation without affecting end users.

Figure 27: Core Migration Scenario



Localized server-side migration

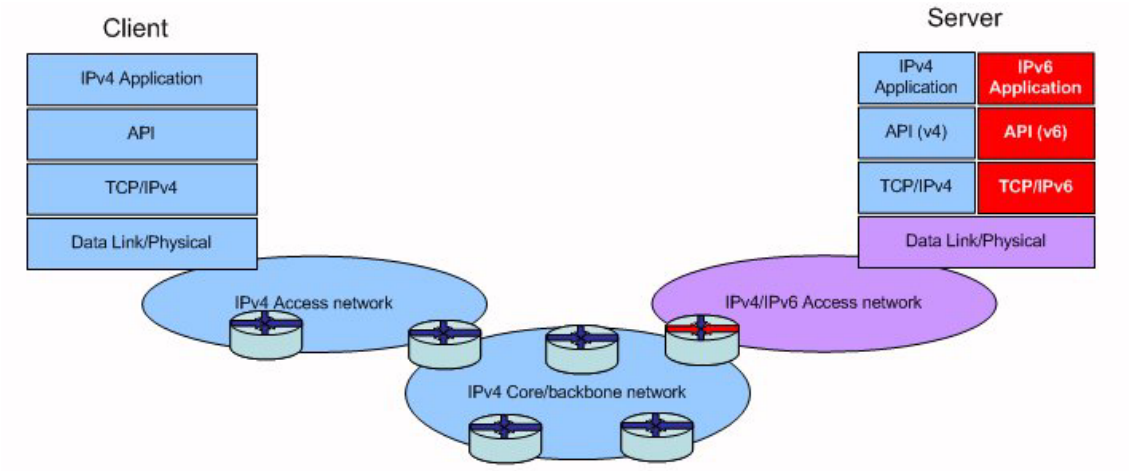
The localized server-side scenario involves upgrading servers and application hosts to dual-stack implementations. With the server still able to support IPv4 communications and applications, end clients should communicate as before via IPv4. However, the server would be able to serve IPv6 clients as well. This scenario may reflect the following use cases:

- Where the client and server in Figure 28 are within a given organization, the organization could upgrade just its servers to provide a means of overall readiness testing for IPv6 prior to upgrading and affecting end clients. End clients would not have access to any IPv6 applications in this case.
- An inter-organizational connection via an IPv4 network, e.g., the IPv4 Internet.
  - An organization migrating or having completed migration to IPv6 would likely implement a dual-stack server for Internet-facing IP applications, such as its web servers. In this case, an IPv4 browser client can access the web server via the Internet. The web server could serve both IPv4 clients and IPv6 browser clients. Depending on the ISP capabilities, the ISP could provide a translation gateway from an IPv6 access network, or tunneling could be used.

- An organization migrating or having completed a migration to IPv6 that requires network connections to partners running IPv4-only would also map to this scenario. Implementation of configured tunnels would be a good approach for such an inter-organizational link.

- Ignoring the IPv4 portion of the dual stack on the server and considering the server IP6-only, this scenario could represent an IPv4-only client attempting to access an IPv6-only server. Such a scenario would require the use of one of the following techniques:
  - An IPv4-IPv6 translation gateway bordering the IPv4-IPv6 networks, assuming the server access network is also IPv6-only.
  - If the server access network is IPv4-only, the server must employ SIIT or another host-based translation mechanism to translate the IPv4 header into IPv6. However, the Bump-in-the-API approach may be required instead to provide application level translation.

Figure 28: Server-Side Migration Scenario



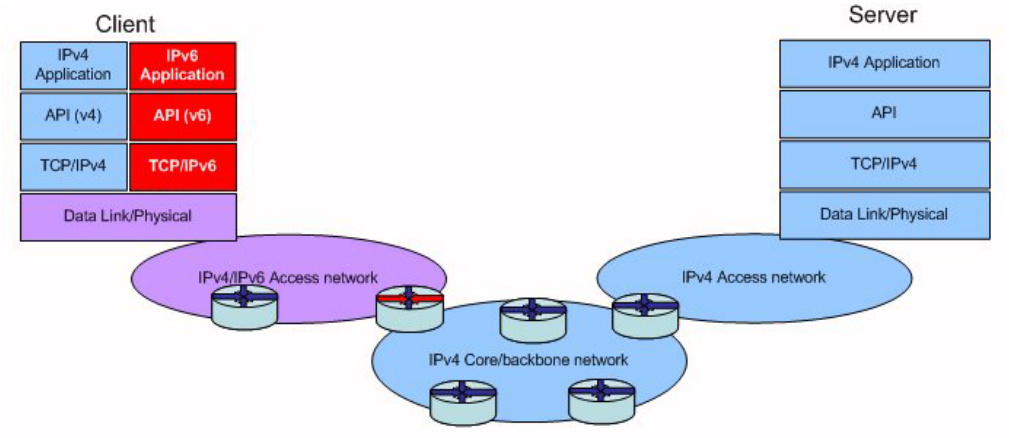
Client-side migration

Figure 29 represents a scenario of migrating clients to dual-stack implementations as well as access network routers. Existing IPv4 client devices would be supplemented with IPv6 applications, API and TCP/IP stack. Much of this is already provided when migrating to Windows XP SP1, Vista (with better integrated dual TCP/IP stack) or Mac OS X. This scenario represents the following example cases:

- After an organization's complete migration to IPv6, continued use of dual stack would support access to IPv4 web sites. Hence, this may be a post-migration scenario in effect for a number of years until Internet-accessible applications complete the transition to IPv6. However, a more likely configuration for such a scenario is the complete migration to IPv6 within the organization and a translation gateway to the IPv4 Internet (ISP) link.
- Such a configuration within a given organization is deemed unlikely outside of those working on IPv6 projects having a need to access external IPv6 applications. Typically, client deployments on a wide scale would require requisite server-side support.

- Ignoring the IPv4 portion of the client stack and considering the client IPv6-only, this scenario could depict a fully transitioned IPv6 user accessing an IPv4 application such as a web server. Such a scenario would typically feature a translation gateway on the IPv4 ISP connection, though 6to4, ISATAP or Teredo tunneling could be employed as well.

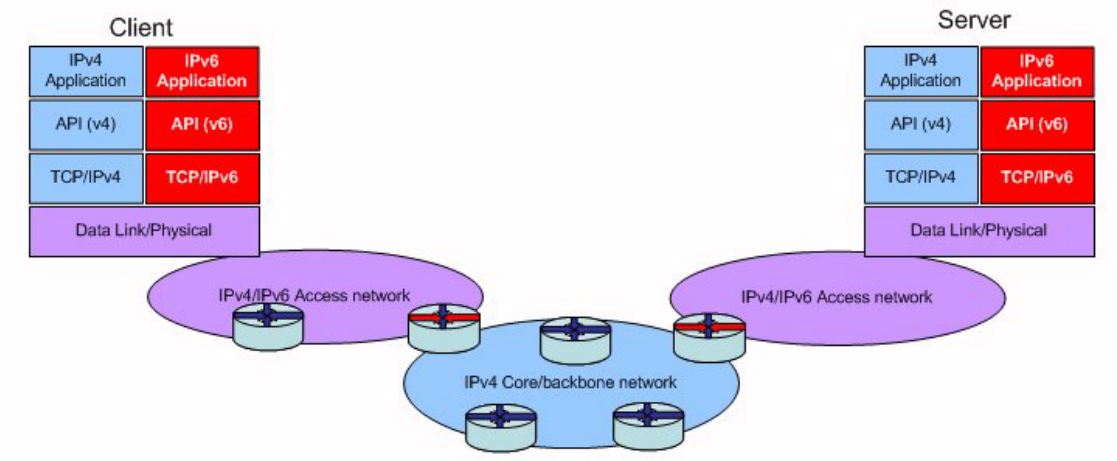
Figure 29: Client-Side Migration Scenario



Client-server migration

A common approach to IPv6 migration within an organization features upgrading of clients and servers at about the same time, including applications as appropriate. The use of dual-stack deployments, shown in Figure 30, facilitates the deployment to clients and servers over time. Special consideration must be given to application migration, especially for integrated enterprise applications accessed by a large user population. Support of mixed IPv4/IPv6 clients in transition is ideal but may not be practical.

Figure 30: Client-Server Deployment Scenario



This scenario reflects the intra-enterprise example described previously, as well as a broader set of examples per Table 3, considering also single-stack cases, i.e., both client and server as IPv6-only.

Table 3: Translation Technologies Summary

	Server resolved to IPv4 address	Server resolved to IPv6 address
IPv4 Client	Native IPv4 communications	Translation gateway or router-to-host tunnel
IPv6 Client	BIS, BIA, TRT, ALG	Native IPv6 communications

Migration Planning Support

The IPv4-to-IPv6 migration process presents a number of challenges and BT Diamond IP can help in a number of ways. As stated earlier, selecting the right migration path will depend on your current environment in terms of end-user devices and operating systems, router models and versions, as well as key applications, budget and resources, not to mention schedule constraints. Along the entire process, from initial assessment to planning and execution, BT Diamond IP professionals can provide expert assistance and support. The IPControl™ system from BT Diamond IP provides software and appliance solutions for modeling, tracking and implementing IPv4 and IPv6 address blocks, subnets, individual IP addresses and associated DHCP/DNS server configurations.

While there are numerous detailed steps involved, the following four high-level steps are key to planning and performing such a transition. Centralized management of a mixed IPv4/IPv6 address space using IPControl is key to documenting and tracking migration progress along the way.

1) Baseline the current environment
   a) Inventory and baseline the current IPv4 network environment to determine what IPv4 address spaces are in use, how they have been allocated, what individual IPv4 addresses have been assigned and are in use and other IP-related information per device.
   b) As a corollary to the prior step, inventory and baseline the associated Dynamic Host Configuration Protocol (DHCP) server configurations with respect to address pools and associated policies and options.
   c) Identify and record associated Domain Name Server (DNS) configurations and resource records associated with IPv4 devices.
   d) Inventory network devices (infrastructure and end user) to assess current and expected future IPv4/IPv6 level of support.
   e) Analyze applications for potential migration impacts and, if found, plan for addressing these impacts.
2) Plan the IPv6 deployment
   a) Map out strategy for IPv6 migration including consideration of the following:

i) Application migration or upgrades

ii) Networking/router migrations or upgrades

iii) Selection of transition technologies from tunneling, translation and/or dual-stack approaches and planning corresponding DNS impacts

iv) Selection of IPv6 device configuration strategy, e.g., stateless autoconfiguration, stateful configuration or hybrid

b) Consider timeframe requirements, if any, analyze dependencies across affected elements and determine budgeting requirements to derive a migration plan.

3) Execute the transition

a) Begin execution of the migration plan based on the prior step and project manage the process with respect to schedules, budget, dependencies and contingency plans.

b) Track the IP address inventory of baselined IPv4 space and associated IPv6 overlay space. Of course, growth and changes in the IPv4 space will most likely continue throughout the transition, so dovetail these updates into the plan as appropriate.

c) Update DNS and DHCP services to accommodate the transitioning environment, e.g. to support both IPv4 and IPv6 hosts, as well as IPv4 and IPv6 transport.

4) Complete the transition

a) As planned, complete the transition with decommissioning of IPv4 space or planned co-existence of IPv4-IPv6 address space. This can be done as portions of your network fully migrate or as a final decommissioning step across the network.

b) You may want to leave IPv4 protocol operational throughout, or on portions of your network, depending on requirements to service external hosts that may be IPv4-only based on internal policies.

Throughout this process and beyond, the IPControl solution provides a centralized repository of the IPv4 network baseline, IPv6 network planning capability and associated DHCP and DNS server configuration. IPControl supports both IPv4 and IPv6, and provides integrated network allocation, discovery and multi-vendor DHCP/DNS server configuration and management features. BT Diamond IP provides professional services to assist with planning, baselining, transitioning and operating the IPv4/IPv6 network or any combination of the above.

Conclusion

Numerous IPv4-to-IPv6 transition mechanisms have been devised to readily enable the migration. Leading router and operating system vendors already support IPv6, as well as various transition implementations. However, bringing it all together into a comprehensive migration plan for your network can be a daunting task. BT Diamond IP offers a number of services to assist with IPv6 readiness assessments, transition strategy analysis, migration plan development and execution and ongoing operations support. BT Diamond IP also offers its market-leading IP address management solution IPControl, the world's first integrated IPv4-to-IPv6 address management solution.

RFC Reference

RFC Number	RFC Category	Title
2529	Proposed Standard	Transmission of IPv6 Packets over IPv4 (aka 6over4)
2765	Proposed Standard	Stateless IP/ICMP Translation Algorithm (SIIT)
2766	Proposed Standard	Network Address Translation – Protocol Translation (NAT-PT)
2767	Informational	Dual-Stack Hosts using the "Bump-in-the-Stack" Technique (BIS)
3053	Informational	IPv6 Tunnel Broker
3056	Proposed Standard	Connection of IPv6 Domains via IPv4 Clouds (aka 6to4)
3089	Informational	A SOCKS-based IPv6/IPv4 Gateway Mechanism
3142	Informational	IPv6 to IPv4 Transport Relay Translator
3315	Proposed Standard	DHCP for IPv6 (DHCPv6)
3338	Experimental	Dual-Stack Hosts Using "Bump-in-the-API" (BIA)
3596	Draft Standard	DNS Extensions to Support IPv6
3964	Informational	Security Considerations for 6to4
4007	Proposed Standard	IPv6 Scoped Address Architecture
4038	Informational	Application Aspects of IPv6 Transition
4213	Proposed Standard	Basic Transition Mechanisms for IPv6 Hosts and Routers
4214	Experimental	Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)
4291	Draft Standard	IP Version 6 Addressing Architecture
4380	Proposed Standard	Teredo
4477	Informational	DHCP: IPv4 and IPv6 Dual-Stack Issues
4554	Informational	VLANs for IPv4-IPv6 Coexistence

About BT Diamond IP

BT Diamond IP is a leading provider of software and appliance products that help customers effectively manage complex IP networks. Our next-generation IP management solutions help businesses more efficiently manage IP address space across mid-to-very large sized enterprise and service provider networks. These products include IPControl™ for comprehensive IP address management, Sapphire Appliances for DNS/DHCP services deployment and NetControl™ for full-cycle IP address block management and utilization. Our cable firmware management product, ImageControl™, helps broadband cable operators automate and simplify the process of upgrading and maintaining firmware on DOCSIS devices in the field. Our customers include regional, national and global service providers and enterprises in all major industries. For additional information, please visit bt.diamondip.com or contact BT Diamond IP at 1-800-390-6295 in the U.S. or 1-610-423-4770 worldwide.

IPControl is a trademark of BT INS, Inc.