15

IPv6 DEPLOYMENT AND IPv4 COEXISTENCE

15.1 INTRODUCTION

IPv6[*] was originally specified in the mid-1990s to address a then-urgent need to supplement the rapidly diminishing IPv4 address space. At the time work begun in earnest on defining IPv6 or IPng (IP next generation) as it was initially named, the Internet was just starting to catch on with the general public. More and more enterprises were expanding their internal networks to enable connection to the global Internet. Since every reachable host required a unique public IPv4 address, the demand for addresses skyrocketed.

While these events spurred the development of IPv6, they also stimulated the development of other technologies that prolonged the life expectancy of IPv4 address space. As we discussed in Chapter 1, classless interdomain routing (CIDR) enabled Regional Internet Registries and ISPs to allocate address space more efficiently than with former classful-only allocation methods.

Another IPv4 allocation strategy discussed in Chapter 1 that vastly reduced the amount of address space required by organizations from RIRs or ISPs was the allocation

[*] The material in this chapter is based on and adds more detail to Chapter 8 of Ref. 11 and on Ref. 147.

of private address space. Defined in RFC 1918, the allocation of private networks enabled every organization to use the same address space for their internal networks. Communicating to Internet hosts or among organizations still required public addresses, but the use of network address translation (NAT) firewalls provided private-to-public address translation for internal hosts accessing the Internet.

One could also argue that DHCP itself enabled better utilization of address space, with its ability to share addresses among a number of users on an as-needed basis. While predominantly configured with private address space within organizations having little impact on public space, DHCP is also used by broadband and wireless service providers to enable Internet access for their respective subscriber bases.

These growing subscriber bases are still driving increasing IPv4 address consumption, diminishing available capacity. And in many parts of the world, current allocations of IPv4 address space are inadequate. For example, Asia has been allocated about 20% of IPv4 allocations yet supports half of the world's population! Asia has been among the leaders in deployments of IPv6, followed by Europe. While North America has enjoyed relatively plentiful IPv4 address space, many organizations are evaluating a move to IPv6, especially among government and service provider organizations.

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

15.1.1 Why Implement IPv6?

On May 22, 2007, the American Registry of Internet Numbers (ARIN—one of the RIRs) board issued a public statement "advis[ing] the Internet community that migration to IPv6 numbering resources is necessary for any applications which require ongoing availability from ARIN of contiguous IP numbering resources." This in essence stated that IPv4 numbering resources (including IPv4 addresses) are becoming more scarce and that entities (LIRs, ISPs) requiring additional addresses over time need to plan for IPv6. All RIRs have also issued similar statements. Some estimates predict depletion of IPv4 space at the RIR level around 2012.* In fact Dr. Geoff Huston of APNIC has published enlightening analysis updated daily at www.potaroo.net/tools/ipv4 lending evidence to these dire predictions.†

Upon this final RIR allocation of IPv4 space in 2012 or whenever this occurs, ISPs will still have space to allocate; the last block allocated by an RIR will exhaust RIR space, but this allocated ISP block is still available for ISP allocation. ISP space will deplete when the ISP customer base fully consumes their allocatable space, which could take a year or so after that. Enterprises that haven't submitted nor have plans to submit requests

* While one can surmise the exhaustion date of IPv4 address space from this analysis, the intent of Dr. Huston's analysis focuses more on identifying when RIR policies must be in place to deal with the new RIR role for managing IPv4 address resources.

† Perhawps this "doomsday scenario" is what the Mayans had in mind!

for IP addresses in the foreseeable future may conclude that IPv6 won't affect them. But at the point in time where only IPv6 space is available, new ISPs or organizations seeking to expanding address space will generate end users with IPv6-only connectivity. This will drive IPv4-only organizations to implement IPv6 on external (Internet-facing) web, email, and other public servers as this IPv6-only population grows.

In addition, like many IP network changes such as those driven by wireless and PDA adoption within organizations, IPv6 may end up being driven by employees connecting to the network via next generation cell phones, PDAs, or even with Windows Vista (or 7) via network or home connections! In this case, the requirement to manage IPv6 address space may be thrust upon IT managers. Whether driven by external IPv6-only users or internal users, it's more prudent to proactively plan now for IPv6, whether you plan to fully migrate your network or plan to deal with individual IPv6 devices attempting to connect. Many service providers are already well along the path towards IPv6 deployment within their IP networks.

15.1.2 IPv4–IPv6 Coexistence Technologies

Numerous technologies are available to facilitate the migration of devices to IPv6. We use the term "coexistence" since the "migration" will likely be a very lengthy process. We'll discuss these approaches 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 or vice versa.
- *Translation*. IP header, address, and/or port translation such as that performed by gateway or NAT devices.

The selected strategy requires effective coordination of the following:

- IPv4 and IPv6 network and subnet allocations, existing and planned.
- Validating network infrastructure and application compatibility with 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 tunneling mode as appropriate.
- Deployment of translation gateway(s) as appropriate.

Example IPv6 implementation scenarios are provided later in this chapter for service providers and enterprises, but first, let's discuss the key coexistence technologies.

15.2 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, application servers, 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, which 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 implemented in Microsoft Vista and 7, as opposed to the XP implementation, which utilized dual transport and network layers, requiring in some cases, redundant configuration of each stack. Other approaches may span the entire stack, down to the physical layer requiring a separate network interface for IPv6 versus 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 intentionally support only one version or the other.

15.2.1 Deployment

Deployment of dual-stacked devices sharing a common physical 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 payload thanks to protocol layering. 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 15.1. This diagram can be extended beyond a physical LAN to a multihop network where routers support IPv4 and IPv6 and route IPv4 packets among native IPv4 hosts and IPv6 packets among IPv6-capable hosts.

While it's generally anticipated that routers would be among the first IP elements to be upgraded to support both protocols, RFC 4554 (193) 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

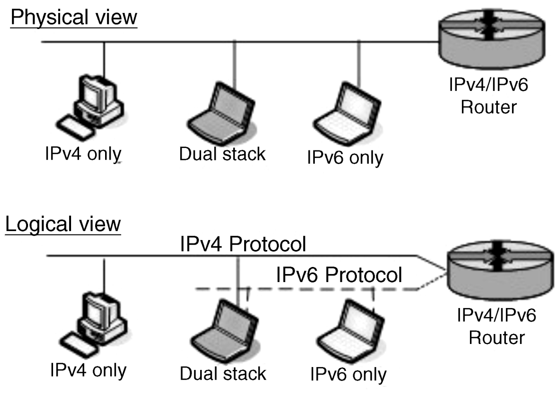

Figure 15.1. Dual-stacked network perspectives (11).

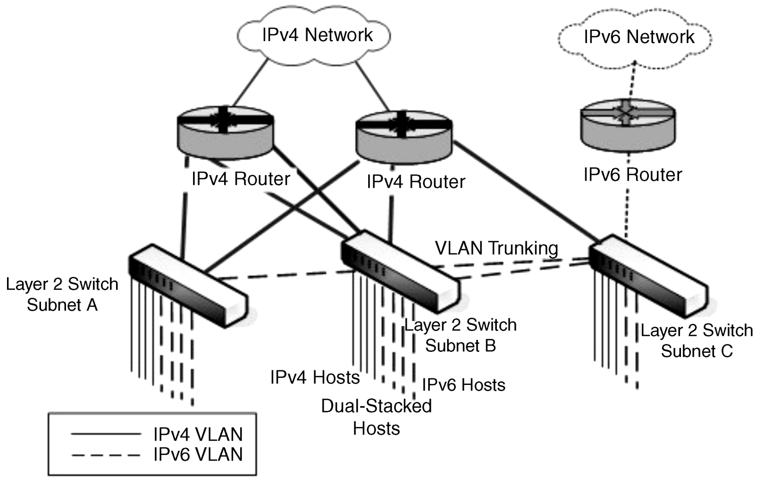

Figure 15.2. Dual-stack deployment using VLANs (147).

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, for example, the gateway to an IPv6 network, 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 IPv6 VLAN, and multiple such VLANs could be likewise configured. An example of this deployment is displayed in Figure 15.2.

15.2.2 DNS Considerations

As we shall see, DNS plays a crucial role in proper operation of each transition technology; after all, it provides the vital linkage between end user naming, for example, web site address at the application layer and the destination IP address, whether IPv4 or IPv6 at the network layer. 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 domain name corresponding to the device as per the following example.

```
dual-stack-host.ipamworldwide.com.86400 IN A    10.200.0.16
dual-stack-host.ipamworldwide.com.86400 IN AAAA 2001:DB8:2200::A
```

Resolution of IP-address-to-host domain 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.ipamworld-
  wide.com.
A.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.2.8.B.D.0.1.0.0.2.
  ip6.arpa. 86400
IN PTR dual-stack-host.ipamworldwide.com.
```

A 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. Some resolver configurations may enable definition of the preferred network protocol when both an A and 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, 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. Resolving corresponding PTR records requires traversal down the corresponding .arpa. tree, which in some cases can be tricky as we'll discuss.

In terms of IP version used in the transport of DNS queries and answers, RFC 3901 (Internet Best Current Practice 91) (148) recommends that each recursive DNS server should support IPv4-only or dual-stack IPv4/IPv6. The RFC also recommends that every DNS zone should be served by at least one IPv4-reachable authoritative DNS server. These recommendations were set forth to provide backward compatibility for IPv4-only resolvers which will be around for quite some time.

15.2.3 DHCP Considerations

The mechanism for using DHCP under a dual-stack implementation is simply that each stack uses its corresponding version of DHCP. That is, to obtain an IPv4 address, use DHCP; 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. For example, if DNS server addresses are provided by both DHCP transactions, preferences of IPv4, IPv6, or mixed preference ordering cannot be conveyed. This remains an ongoing area of concern, as documented in RFC 4477 (149), 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.

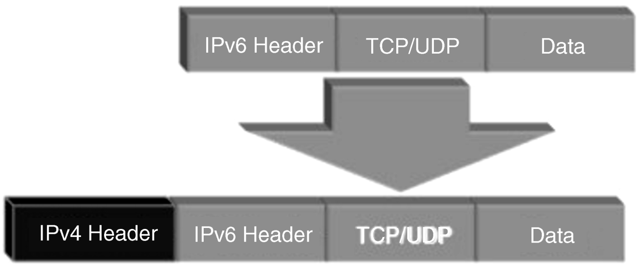

Figure 15.3. IPv6 over IPv4 tunneling (11).

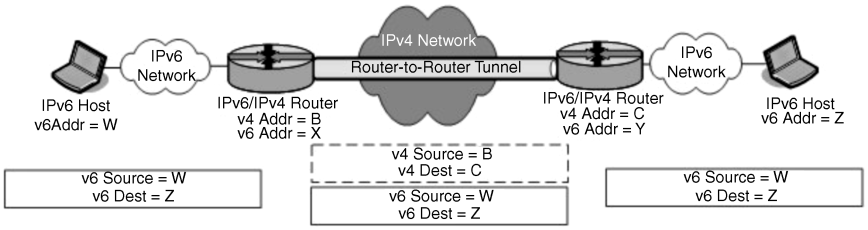

Figure 15.4. Router-to-router tunnel (11).

15.3 TUNNELING APPROACHES

A variety of tunneling technologies have been developed to support IPv4 over IPv6 and 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 over an IPv4 network entails prefixing an IPv6 packet with an IPv4 header as illustrated in Figure 15.3. This enables the tunneled packet to be routed over an IPv4 routing infrastructure; the IPv6 packet is simply considered payload within the IPv4 packet. The entry node of the tunnel, whether a router or host, performs the encapsulation. 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 routes the packet as appropriate to the ultimate destination via IPv6.

15.3.1 Tunneling Scenarios for IPv6 Packets Over IPv4 Networks

Using this basic tunneling approach, a variety of scenarios based on tunnel endpoints have been defined. Probably the most common configuration is a router-to-router tunnel, depicted in Figure 15.4, which is the most common approach for configured tunnels.

In this figure, the originating IPv6 host on the left has IPv6 address of W (for simplicity and brevity for now). A packet* destined for the host on the far end of the diagram with IPv6 address of Z is sent to a router serving the subnet. This router, with IPv4 address of B and 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 IPv4 address of C, as the destination address as depicted by the dashed rectangle beneath the IPv4 network in the center of Figure 15.4. The tunneled packets are routed like "regular" IPv4 packets to the destination tunnel

* This packet is crudely identified in the figure as the solid-line rectangle beneath the originating host displaying the packet's IPv6 source address of W and destination address of Z. The tunnel header is shown as the dotted-line rectangle in this and subsequent tunneling figures.

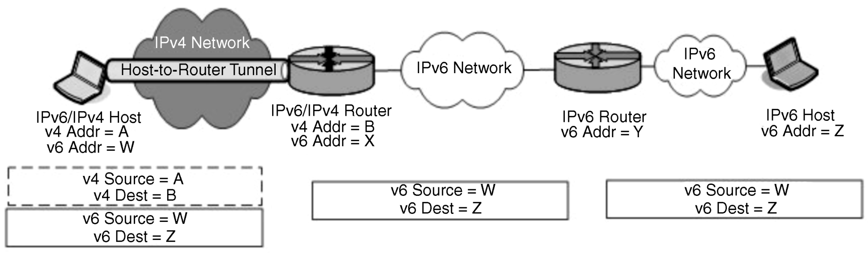

Figure 15.5. Host-to-router tunneling configuration (11).

endpoint router. This 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 15.5. The tunneling mechanism is the same as in the router-to-router case, but the tunnel endpoints are different.

The router-to-host configuration is also very similar, as shown in Figure 15.6. 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.

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 as shown in Figure 15.7. In this example, the communications is IPv4 from end-to-end.

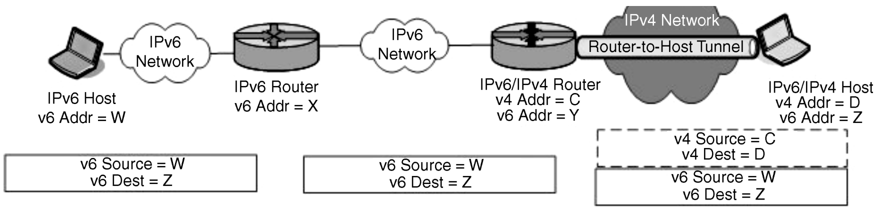

Figure 15.6. Router-to-host tunnel configuration (11).

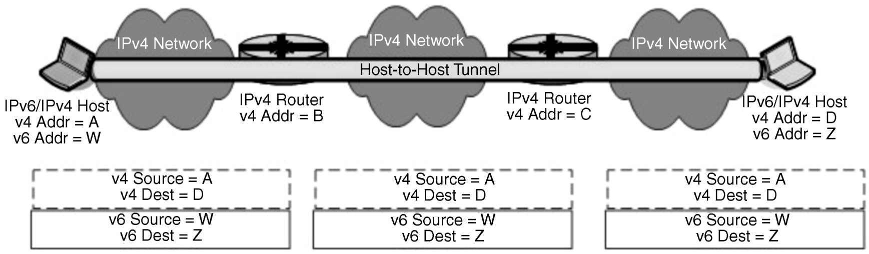

Figure 15.7. Router-to-host tunnel configuration (11).

15.3.2 Tunnel Types

As mentioned, tunnels are either configured or automatic. Configured tunnels are predefined by administrators in advance of communications. In the scenarios described above, configuration of the respective tunnel endpoints is required to configure the device regarding when to tunnel IPv6 packets, that is, based on destination, along with other tunnel configuration parameters that may be required by the tunnel implementation.

An automatic tunnel does not require tunnel preconfiguration, though enablement of tunneling configuration may be required. Tunnels are created based on information contained within 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 per the diagram below.

The unique IPv4 address, 192.0.2.131 in our example of Figure 15.8, 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.

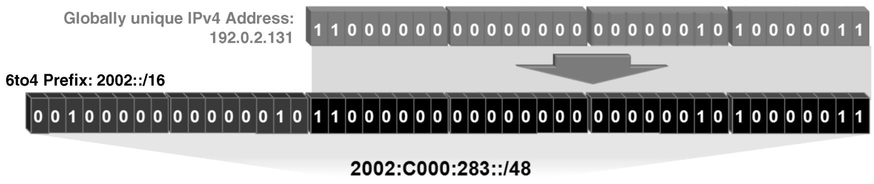

Figure 15.8. 6to4 address prefix derivation (147).

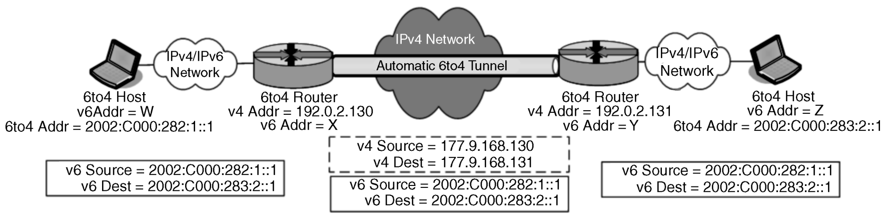
IPv4 Network
Automatic 6to4 Tunnel
IPv4/IPv6 Network
IPv4/IPv6 Network
6to4 Host
v6Addr = W
6to4 Addr = 2002:C000:282:1::1
6to4 Router
v4 Addr = 192.0.2.130
v6 Addr = X
6to4 Router
v4 Addr = 192.0.2.131
v6 Addr = Y
6to4 Host
v6 Addr = Z
6to4 Addr = 2002:C000:283:2::1
v6 Source = 2002:C000:282:1::1
v6 Dest = 2002:C000:283:2::1
v4 Source = 177.9.168.130
v4 Dest = 177.9.168.131
v6 Source = 2002:C000:282:1::1
v6 Dest = 2002:C000:283:2::1
v6 Source = 2002:C000:282:1::1
v6 Dest = 2002:C000:283:2::1

Figure 15.9. 6to4 tunneling example (147).

Let's consider an example, two sites containing 6to4 hosts desire 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 15.9 the IPv4 addresses of the routers' IPv4 interfaces (facing each other) are 192.0.2.130 and 192.0.2.131, respectively. Transforming these IPv4 addresses into 6to4 prefixes we arrive at 2002:C000:282::/48 and 2002:C000:283::/48, respectively. These prefixes now identify each site in terms of 6to4 reachability. Our 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:C000:282:1::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: C000:282:1::/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:C000:283:2::1.

The AAAA and PTR resource records corresponding to these 6to4 addresses should also be added to DNS within the appropriate domains. When tunneling through the Internet, the destination AAAA and PTR records are maintained by each organization managing the corresponding 6to4 devices and resolution may require traversal down each domain subtree. The AAAA record follows normal "forward domain" resolution, but the PTR record is less straightforward. Since the PTR domain tree is based on the corresponding IPv6 address, which in the 6to4 case is "self-configured" by an organization based on its IPv4 address space and not by an upstream IPv6 address registry, the ip6.arpa delegation is unlinked from an authoritative upstream domain parent. A special registrar was established to handle delegations from the 2.0.0.2.ip6.arpa zone: the Number Resource Organization (NRO). Our administrators for the ip6.arpa domain corresponding to the 2002:C000:283::/48 prefix in our example would register the 3.8.2.0.0.0.0.C.2.0.0.2.ip6.arpa zone with 6to4.nro.net along with corresponding authoritative name servers.

Getting back to the packet flow, when our host on the left wishes to communicate with the host on the right, a DNS lookup would resolve to its 6to4 address. 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) IPv4 addresses, respectively. The destination 6to4 router receiving the packet would decapsulate it and transmit the packet on its 2002:C000:283:2::/64 network to the destination 6to4 host.

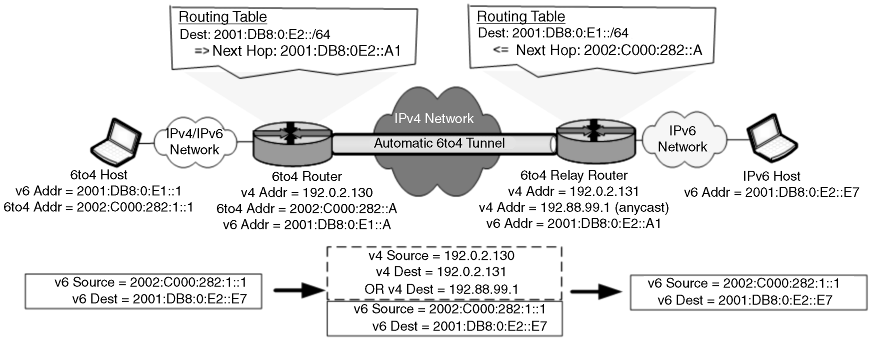
Routing Table
Dest: 2001:DB8:0:E2::/64
=> Next Hop: 2001:DB8:0E2::A1
Routing Table
Dest: 2001:DB8:0:E1::/64
<= Next Hop: 2002:C000:282::A
IPv4/IPv6 Network
IPv4 Network
Automatic 6to4 Tunnel
IPv6 Network
6to4 Host
v6 Addr = 2001:DB8:0:E1::1
6to4 Addr = 2002:C000:282:1::1
6to4 Router
v4 Addr = 192.0.2.130
6to4 Addr = 2002:C000:282::A
v6 Addr = 2001:DB8:0:E1::A
6to4 Relay Router
v4 Addr = 192.0.2.131
v4 Addr = 192.88.99.1 (anycast)
v6 Addr = 2001:DB8:0:E2::A1
IPv6 Host
v6 Addr = 2001:DB8:0:E2::E7
v6 Source = 2002:C000:282:1::1
v6 Dest = 2001:DB8:0:E2::E7
v4 Source = 192.0.2.130
v4 Dest = 192.0.2.131
OR v4 Dest = 192.88.99.1
v6 Source = 2002:C000:282:1::1
v6 Dest = 2001:DB8:0:E2::E7
v6 Source = 2002:C000:282:1::1
v6 Dest = 2001:DB8:0:E2::E7

Figure 15.10. 6to4 host communicating with a native IPv6 host (147).

6to4 can provide an efficient mechanism for IPv6 hosts to communicate over IPv4 networks. As IPv6 networks are incrementally deployed, *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 these relay routers can be configured

1. Configure routes to destination native IPv6 networks with the 6to4 relay router as the next hop. This is illustrated in Figure 15.10.
2. 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 15.10 is the "IPv6 internet," the following default route option is likely a better alternative.
3. Configure a default route to the 6to4 relay router to reach IPv6 networks. This scenario may apply where an IPv6 Internet connection is reachable only through a IPv4 network internally to the organization and few or no pure IPv6 networks exist within the organization.*

In walking through Figure 15.10, we have a 6to4 host on an IPv4/IPv6 network on the left of the diagram with a native IPv6 address and a 6to4 address. This host sets out to communicate with a native IPv6 host with IP address 2001:DB8:0:E2::E7 on the right side of the diagram. 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,

* A variant on this scenario calls for the definition of 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 (194) defines an anycast address for 6to4 relay routers: 2002:C058:6301::/48. This address corresponds to the IPv4 address of 192.88.99.1. This variant is also illustrated in Figure 15.10.

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 on the left of the IPv4 network cloud. The router would need to have a routing table entry in order to route to the destination 2001:DB8:0: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 Figure 15.10, the tunneled packet headers depicted below the IPv4 network cloud indicate that the destination IPv4 address encapsulating the IPv6 packet could be the IPv4 unicast address or the IPv4 address corresponding to the 6to4 anycast address. The 6to4 relay router, upon receiving the IPv4 packet, would decapsulate it, then 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 toward the IPv6 host.

Intrasite 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 192.0.2.131 is denoted as ::5EFE:192.0.2.131. 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:192.0.2.131.

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`," or using a DHCP vendor-specific 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 15.11, using the IPv4 address corresponding to the chosen router from the PRL.

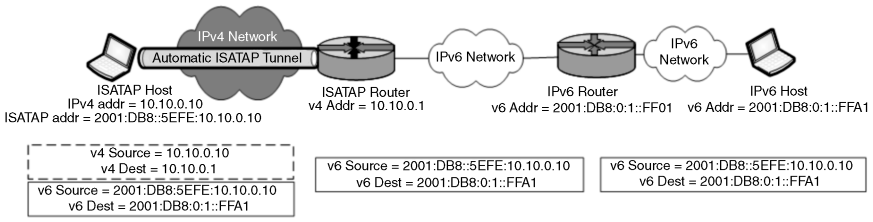
IPv4 Network
Automatic ISATAP Tunnel
IPv6 Network
IPv6 Network
ISATAP Host
IPv4 addr = 10.10.0.10
ISATAP addr = 2001:DB8::5EFE:10.10.0.10
ISATAP Router
v4 Addr = 10.10.0.1
IPv6 Router
v6 Addr = 2001:DB8:0:1::FF01
IPv6 Host
v6 Addr = 2001:DB8:0:1::FFA1
v4 Source = 10.10.0.10
v4 Dest = 10.10.0.1
v6 Source = 2001:DB8:5EFE:10.10.0.10
v6 Dest = 2001:DB8:0:1::FFA1
v6 Source = 2001:DB8::5EFE:10.10.0.10
v6 Dest = 2001:DB8:0:1::FFA1
v6 Source = 2001:DB8::5EFE:10.10.0.10
v6 Dest = 2001:DB8:0:1::FFA1

Figure 15.11. ISATAP host-to-router example (147).

ISATAP hosts can autoconfigure their ISATAP Interface IDs using configured IPv4 addresses, whether the IPv4 address is defined statically or is obtained via DHCP. Microsoft XP and 2003 server perform such autoconfiguration if configured with IPv6. Microsoft Vista and 7 clients and Windows 2008 servers support ISATAP autoconfiguration by default. The ISATAP interface ID is appended to a 64-bit global network prefix and subnet ID provided by solicited ISATAP routers in their router advertisements.

Following Figure 15.11, 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 tunnel 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 in the IP header 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 coming back 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 similar manner, the native IPv6 host to the right of Figure 15.11 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.

Host-to-host ISATAP tunnels, similar to that displayed in Figure 15.7, 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 15.7, 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 6over4. 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 its 6over4 Interface ID portion of its IPv6 address. For example, a 6over4 host with IPv4 address of 192.0.2.85 would formulate an IPv6 interface ID of ::C000:255, and thus, a 6over4 address of FE80::C000:255. 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 link layer, 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, for example, 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 (brokers) 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 to a tunnel broker web portal 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 a number of 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; and
- informs the client of its assigned tunnel server and associated tunnel and IPv6 parameters including address/prefix and DNS name.

Figure 15.12 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. RFC 5572 (150) formalizes the Tunnel Setup Protocol (TSP) to promote common tunnel setup messages and component interactions.

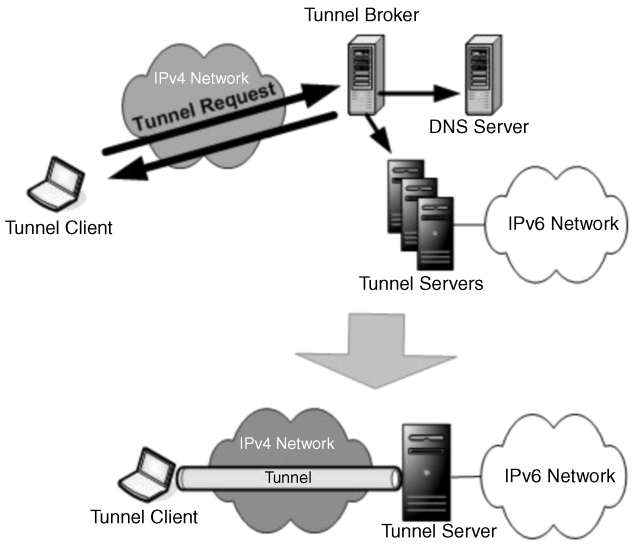

Figure 15.12. Tunnel broker interaction (147).

Teredo. Tunneling through firewalls that perform network address translation can be challenging if not impossible by design. Teredo is a tunnel broker technology that enables NAT traversal of IPv6 packets tunneled over UDP over IPv4 for host-to-host automatic tunnels. Teredo incorporates the additional UDP header (see Figure 15.13) in order to facilitate NAT/firewall traversal. Many NAT/firewall devices will not allow traversal of IPv4 packets with the packet header protocol field set to 41, which is the setting for tunneling of IPv6 packets as mentioned previously. 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 (151) 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 15.14.

- Teredo client
- Teredo server
- Teredo relay

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 NAT firewall that is in place. The Teredo relay is the Teredo tunnel endpoint serving the intended destination host. Teredo hosts must be preconfigured with this Teredo server IPv4 addresses to use, which help establish Teredo connections.

Determining the closest Teredo relay entails sending an IPv6 ping (ICMPv6 echo request) to the destination host. The ping is encapsulated with a UDP and IPv4 header and

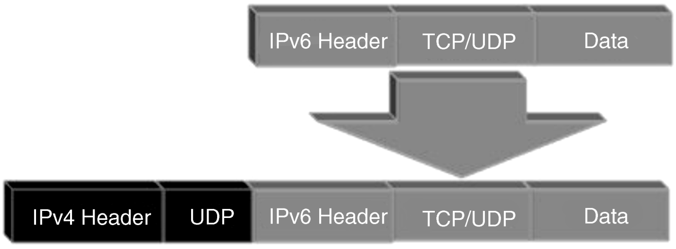

Figure 15.13. Teredo tunnels add UDP then IPv4 headers (151).

sent to the Teredo server, which decapsulates it and sends the native ICMPv6 packet to the destination. The destination host's response will be routed via native IPv6 to the nearest (routing-wise) Teredo relay, then back to the originating host. In this manner, the client determines the appropriate Teredo relay's IPv4 address and port by virtue of its IPv4 and UDP [tunnel] headers. Figure 15.14 illustrates the case with a Teredo client communicating to a native IPv6 host.

NAT Types. The type of NAT being traversed drives the need to perform an additional step to enable the NAT device to initialize table mappings for the data exchanged between the Teredo client and the IPv6 host. The following NAT types have been defined as follows:

- *Full Cone.* All IP packets from the same internal IP address and port are mapped to a corresponding external address and port. External hosts can communicate with the host by transmitting to the mapped external address and port.
- *Restricted Cone.* All IP packets from the same internal IP address and port are mapped 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.
- *Port Restricted Cone.* All IP packets from the same internal IP address and port are mapped 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.

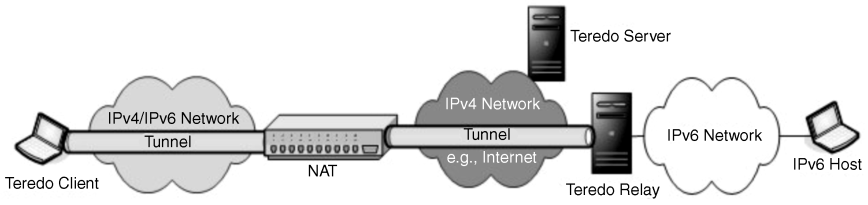

Figure 15.14. Teredo client to IPv6 host connection (147).

- *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.

Table 15.1 illustrates these different NAT types. Based on outgoing traffic shown for each case, the NAT will map the internal address and port to an assigned external address and port; this in turn drives the corresponding permitted incoming traffic shown in the right-hand column of the table. First considering the full cone example, an internal host with IP address 10.10.0.1 initiates a session through the NAT using source port 10081. The NAT maps the original 10.10.0.1 source IP address to 192.0.2.1 on the outgoing packet from the NAT. The NAT also assigns port number 43513 on the outgoing packet and assigns port number 42512 (any high numbered port) for the internal leg back to the IPv4 internal host. Hence, the NAT terminates the first connection and bridges it to another; the first connection is (10.10.0.1:10081 $\leftrightarrow$ NAT IP address:42512) while the second is (192.0.2.1:43513 $\leftrightarrow$ Destination IP address from original packet: Destination port from original packet).

Based on this internally initiated communiqué, any external host may initiate a packet to IP address 192.0.2.1 and thereby connect to the 10.10.0.1 host internally. This is

TABLE 15.1. NAT Types

Outgoing Traffic					Permitted Incoming Traffic	
Internal Host → External Host			NAT Mapping		External Host → Internal Host	
Full Cone	Source	Destination	Internal	⇔External	Source	Destination
IP Addr	10.10.0.1	Any	10.10.0.1	192.0.2.1	Any	192.0.2.1
Port	10081	Any	42512	43513	Any	Any
Restricted Cone	Source	Destination	Internal	⇔External	Source	Destination
IP Addr	10.10.0.1	203.0.113.8	10.10.0.1	192.0.2.1	203.0.113.8	192.0.2.1
Port	10081	Any	42512	43513	Any	Any
Port Restricted Cone	Source	Destination	Internal	⇔External	Source	Destination
IP Addr	10.10.0.1	203.0.113.8	10.10.0.1	192.0.2.1	203.0.113.8	192.0.2.1
Port	10081	80	42512	43513	80	Any
Symmetric	Source	Destination	Internal	⇔External	Source	Destination
IP Addr	10.10.0.1	203.0.113.8	10.10.0.1	192.0.2.1	203.0.113.8	192.0.2.1
Port	10081	80	42512	43513	80	43513

denoted on the right of the table under permitted incoming traffic. In the full cone case, an incoming packet from an external host may come from any IP address or port (source address and port = any) and the destination IP address of 192.0.2.1 on any port will be accepted.

Contrast this with the symmetric case at the bottom of the table, where an internal host on IP address 10.10.0.1 uses source port 10081 to initiate a connection to destination IP address 203.0.113.8 port 80. The NAT terminates this connection and initiates an external connection to the given destination address and port, mapping the source IP address to 192.0.2.1 and port to 43513. In this case, the only accepted incoming traffic permitted to traverse the NAT will have a source IP address and port of 203.0.113.8:80 and destination 192.0.2.1:43513, which the NAT had mapped based on the communications from the originating host.

The restricted cone configuration maps the original destination IP address as permitted as a subsequent incoming source IP address from the external space (203.0.113.8) when the corresponding destination IP address is the NAT-mapped address, 192.0.2.1 in this example. The port restricted scenario adds port validity checking by permitting the original destination address and port as permitted as subsequent incoming IP address and port from the external space (203.0.113.8:80) when the corresponding destination IP address and port match the NAT-mapped 192.0.2.1 address in this example.

To communicate using Teredo, the NAT type drives the Teredo initialization process in order to establish the firewall traversal. Identification of the NAT as full cone requires no further qualification because any external host may initiate communications to its external address. But either of the restricted cone scenarios do require further qualification to properly map the source and destination addresses corresponding to those within the NAT. To complete the mapping within the NAT of the internal host communiqué with the destination host, a *bubble packet* is sent by the Teredo client to the destination host. A bubble packet is an IPv6 header with no payload, itself encapsulated in the Teredo tunnel IPv4/UDP header. It 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 since this is an unsolicited external packet. In this case, the Teredo client times out and sends the bubble packet via the Teredo server, which is identified by the intended destination Teredo formatted IPv6 address, which encodes the Teredo IPv4 address. The Teredo server then forwards the packet over a Teredo tunnel to the destination host, which has its own IPv4 address also encoded in the Teredo IPv6 address.

Assuming the destination host is also a Teredo client, it will receive the packet, having been initialized by a prior ping it sent to this Teredo server during client configuration. The destination host will then respond to the originating host directly, completing the NAT mapping (on both sides). Figure 15.15 illustrates this scenario with two Teredo clients communicating via a common Teredo relay. Teredo does not support automatic traversal of symmetric NAT devices.

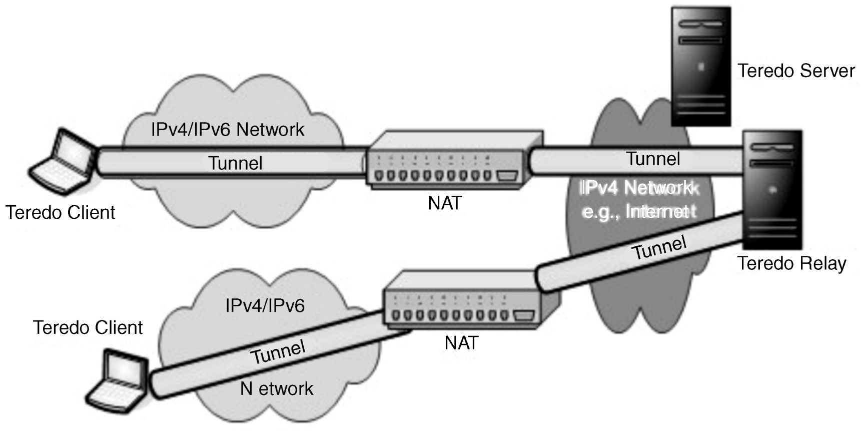

Figure 15.15. Two Teredo clients communicating via the IPv4 Internet (147).

As we've seen, the Teredo IPv6 address is formatted with the client and its server Teredo server IPv4 addresses. The Teredo IPv6 address is of the format depicted in Figure15.16.

The Teredo prefix is a predefined IPv6 prefix: 2001::/32. The Teredo server IPv4 address comprises the next 32 bits. Flags indicate the type of NAT as either full cone (hex value = 8000) or restricted or port restricted (hex value = 0000). The client port and client IPv4 address fields represent obfuscated values of these respective values by reversing each bit value.

15.3.3 Tunneling Scenarios for IPv4 Packets Over IPv6 Networks

During an IPv6 implementation, 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 with the IPv4 host would require a dual stack, as well as a DSTM client. Upon resolving the hostname of the intended destination host using DNS to only an IPv4 address, the client would initiate the DSTM process, which is very similar to the tunnel broker approach. The

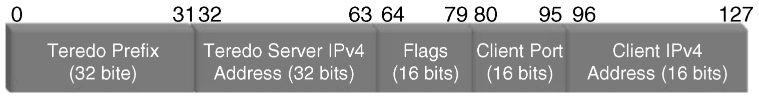

Figure 15.16. Teredo IPv6 address format (151).

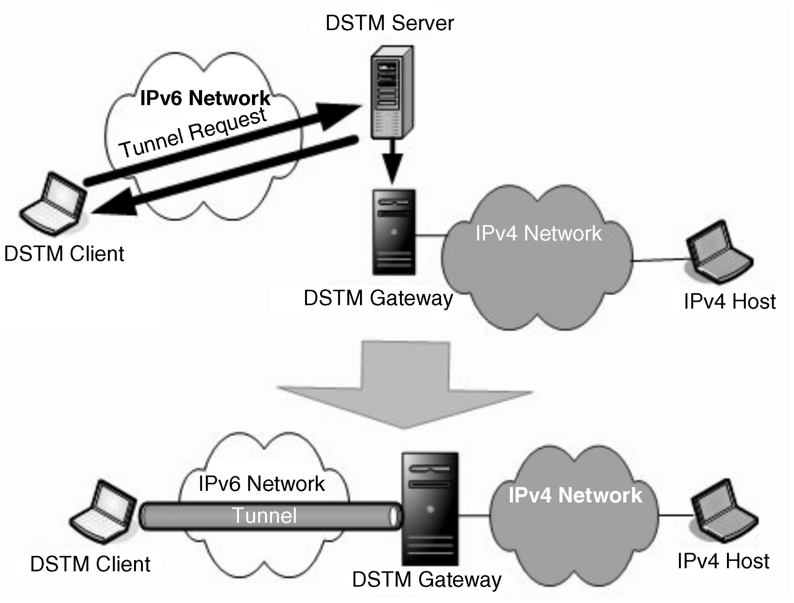

Figure 15.17. DSTM tunnel setup (147).

process begins with the DSTM client contacting a DSTM server to obtain an IPv4 address preferably via the DHCPv6 protocol,* 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 header 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, for example, 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.

15.3.4 Tunneling Summary

The following table summarizes the applicability of tunneling based on the source host capabilities/network type and the destination address resolution and network type.

* While the DSTM RFC drafts (152) 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.

From	To					
	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		Native IPv4 → IPv4-compatible		
Dual-stack client on IPv4 network	Native IPv4	Native IPv4	Host-to-host IPv6 over IPv4[a]	Native IPv4 → IPv4-compatible	Host-to-router IPv6 over IPv4[a]	Host-to-router IPv6 over IPv4[a]
Dual-stack client on IPv6 network	DSTM → Native IPv4	DSTM → Native IPv4	Native IPv6 → Router-to-host IPv6 over IPv4[a]	DSTM	Native IPv6	Native IPv6
IPv6 client on IPv6 network			Native IPv6 → IPv6 over IPv4[a]		Native IPv6	Native IPv6

[a]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 support of the corresponding technology.

The cells in the upper left and lower right of 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 tunnel or translated at each boundary using a translation technology, discussed in the next section.

The other cells indicate a tunneling scenario. The "$\rightarrow$" 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 blank cells indicate an invalid connection option via tunneling. However, translation technologies could be employed to bridge these gaps as we'll discuss next.

15.4 TRANSLATION APPROACHES

Translation techniques perform IPv4-to-IPv6 translation (and vice versa) at a particular layer of the protocol stack, typically network, transport, or application. Unlike tunneling, which does not alter the tunneled data packet but merely appends a header or two, 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; that is, for the blank cell scenarios in the summary table above. In dual-stack environments, native or tunneling mechanisms are preferable.*

15.4.1 Stateless IP/ICMP Translation (SIIT) Algorithm

The common algorithm for translating IPv4 and IPv6 packets is the Stateless IP/ICMP Translation (SIIT) algorithm. SIIT provides translation of IP packet headers between IPv4 and IPv6. SIIT resides on an IPv6 host or gateway and converts outgoing IPv6 packet headers into IPv4 headers, and incoming IPv4 headers into IPv6. To perform this task, the IPv6 host must be provided an IPv4 address, either configured on the host or obtained via a network service unspecified in RFC 2765 (153). When the IPv6 host desires to communicate with an IPv4 host, 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 15.18. 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.

Based on the presence of the IPv4-mapped address format as the destination IP address, SIIT performs header translation as described next to yield an IPv4 packet for transmission via the data link and physical layers. The source IP address for an IPv6 node uses a different format, that of the IPv4-translated format, shown in Figure 15.19, though RFC 2765 does not specify how this address is initially configured.

* Per RFC 2766 (156).

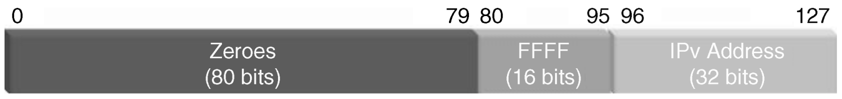

Figure 15.18. IPv4 mapped address format (12).

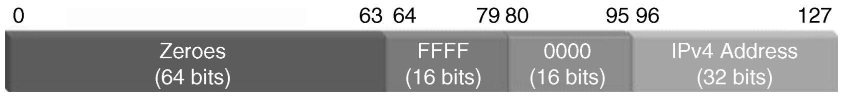

Figure 15.19. IPv4 translated address format used within SIIT (153).

A potential protocol stack view of the SIIT algorithm is shown in Figure 15.20.

The basic header translation process is summarized below for both directions.

IPv4 → IPv6 Header Translation	IPv6 → IPv4 Header Translation
Version = 6	*Version* = 4
Traffic Class = IPv4 header TOS bits	*Header Length* = 5 (no IPv4 options)
Flow Label = 0	*Type of Service* = IPv6 header Traffic Class field
Payload Length = IPv4 header total length value – (IPv4 header length + IPv4 options length)	*Total Length* = IPv6 header payload length field + IPv4 header length
	Identification = 0
Next Header = IPv4 header protocol field value	*Flags* = Don't fragment = 1, more fragments = 0
Hop Limit = IPv4 TTL field value – 1	*Fragment Offset* = 0
Source IP Address = 0:0:0:0:FFFF::/96 concatenated with IPv4 header source IP address	*TTL* = IPv6 hop limit field value – 1
	Protocol = IPv6 next header field
Destination IP Address = 0:0:0:0:0:FFFF::/96 concatenated with IPv4 header destination IP address	*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

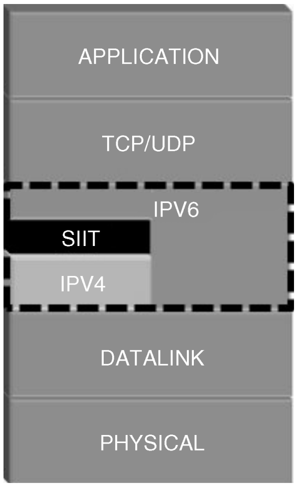

Figure 15.20. SIIT stack example (153).

Now let's look at some techniques that employ the SIIT algorithm to translate IPv4 and IPv6 packets.

15.4.2 Bump in the Stack (BIS)

BIS (154) 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 15.21.

The Translator component translates the IPv4 header into an IPv6 header according to the SIIT algorithm. The Extension Name Resolver snoops DNS queries for A record types; upon detecting such a query, the Extension Name Resolver component creates an additional query for the AAAA record type for the same host domain name (Qname) and class (Qclass). If no affirmative answer is received from the AAAA query, the communications ensues using IPv4; if the AAAA query is successfully 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 an internally 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 that IPv4 address are then translated by the Translator 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 internal pool and translate the IPv6 header into IPv4 for communication up the stack.

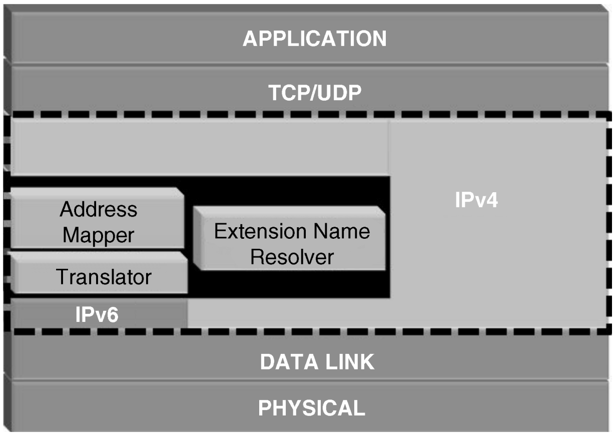

Figure 15.21. Bump- in-the-stack (BIS) components (154).

15.4.3 Bump in the API (BIA)

The bump-in-the-API (BIA) (155) 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 15.22.

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 an additional query requesting AAAA records. A DNS reply with an A record will provide the answer with

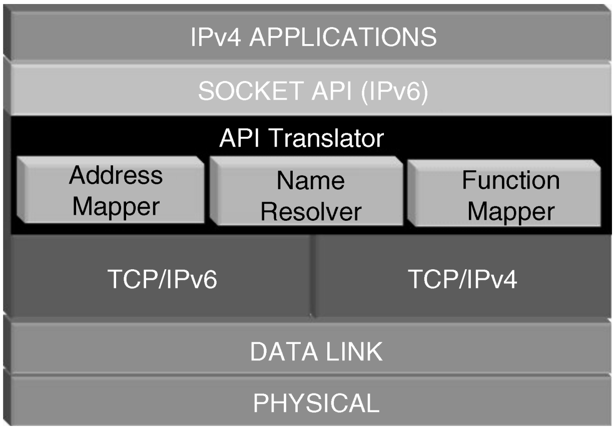

Figure 15.22. Bump in the API (BIA) (155).

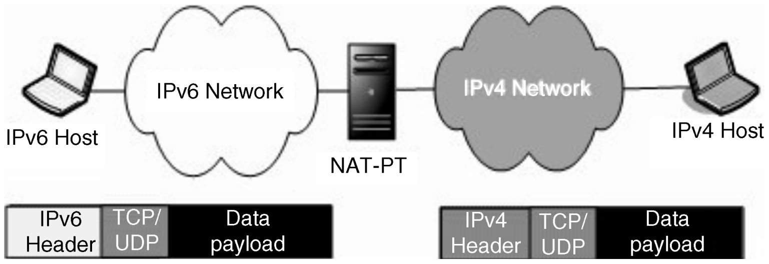

Figure 15.23. NAT-PT deployment (deprecated) (147).

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.

15.4.4 Network Address Translation with Protocol Translation (NAT-PT)—DEPRECATED

As the name implies, the NAT-PT (156) process not only entails translating IPv4 addresses into IPv6 addresses like a familiar IPv4 NAT, but also performs protocol header translation as described in the SIIT section.* 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 communications ensues. Figure 15.23 illustrates the architecture of a NAT-PT deployment. For numerous reasons enumerated in RFC 4966 (157), NAT-PT (and NAPT-PT described next) has been deprecated and should not be deployed.

15.4.5 Network Address Port Translation with Protocol Translation (NAPT-PT)—DEPRECATED

NAPT-PT enables IPv6 nodes to communicate with IPv4 nodes using a single IPv4 address. Thus, in Figure 15.23 above, instead of maintaining a one-to-one association of an IPv6 address and a unique IPv4 address as in NAT-PT, NAPT-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 minimizes the possibility of IPv4 address pool depletion under the NAT-PT scenario.

* With the exception of the source and destination IP address fields which are governed by associations within the NAT-PT gateway.

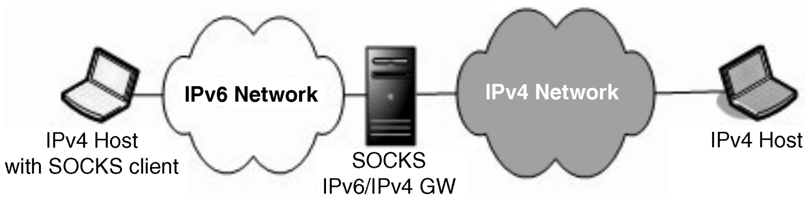

Figure 15.24. Basic SOCKS Gateway configuration (159).

15.4.6 SOCKS IPv6/IPv4 Gateway

SOCKS, defined in RFC 1928 (158), provides transport relay for applications traversing firewalls, effectively providing application proxy services. RFC 3089 (159) 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 15.24 illustrates the case of an IPv6 host configured with a SOCKS client connecting to an IPv4 host. A socksified IPv4 host could just as well communicate via the SOCKS gateway to an IPv6 host, from right-to-left.

15.4.7 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,* which acts as a DNS proxy. TRT is specified to enable IPv6 hosts to communicate with IPv4 destinations. 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 may ensue 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(160), 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 is required instead.

15.4.8 Application Layer Gateway (ALG)

ALGs perform protocol translation at the application layer and perform application proxy functions, similar to HTTP proxies. A client's application would typically need to be configured with the IP address of the proxy server, to which a connection would be

* Sometimes referred to as "trick or treat DNS-ALG" or totd.

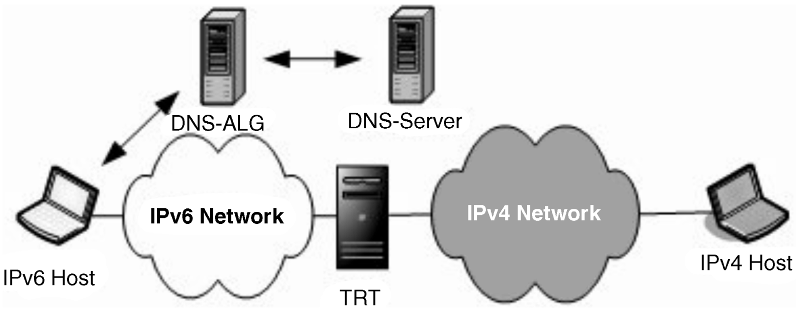

Figure 15.25. TRT configuration with DNS-ALG (160).

made upon opening the application, for example, 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.

15.5 APPLICATION MIGRATION

The *de facto* application programming 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 communicate over IP networks. Microsoft's Winsock API is also based on the sockets interface. Both sockets and Winsock interfaces have been modified to support IPv6's longer address size and additional features. In fact, most major operating system have implemented support for sockets or Winsock including Microsoft (XP SP1, Vista, 7, Server 2003 & 2008), 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. Check with your applications vendors for IPv6 compatibility and requirements.

15.6 PLANNING THE IPv6 DEPLOYMENT PROCESS

15.6.1 Service Provider Deployment Options

Service providers, residential broadband service providers in particular, can implement IPv6 within their networks and ultimately deploy IPv6 addresses to customers. Beyond applying the various techniques discussed in this chapter, two additional alternatives have been developed:

- *6rd.* IPv6 Rapid Deployment defines a modification to the 6over4 transition approach to enable provision of IPv6 addresses to end customers while continuing to support an IPv4 and IPv6 infrastructure.

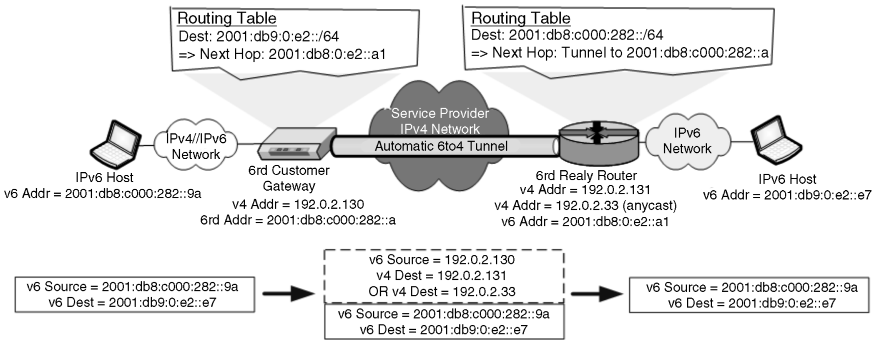
Routing Table
Dest: 2001:db9:0:e2::/64
=> Next Hop: 2001:db8:0:e2::a1
Routing Table
Dest: 2001:db8:c000:282::/64
=> Next Hop: Tunnel to 2001:db8:c000:282::a
Service Provider IPv4 Network
Automatic 6to4 Tunnel
IPv4//IPv6 Network
IPv6 Network
IPv6 Host
v6 Addr = 2001:db8:c000:282::9a
6rd Customer Gateway
v4 Addr = 192.0.2.130
6rd Addr = 2001:db8:c000:282::a
6rd Realy Router
v4 Addr = 192.0.2.131
v4 Addr = 192.0.2.33 (anycast)
v6 Addr = 2001:db8:0:e2::a1
IPv6 Host
v6 Addr = 2001:db9:0:e2::e7
v6 Source = 2001:db8:c000:282::9a
v6 Dest = 2001:db9:0:e2::e7
v6 Source = 192.0.2.130
v4 Dest = 192.0.2.131
OR v4 Dest = 192.0.2.33
v6 Source = 2001:db8:c000:282::9a
v6 Dest = 2001:db9:0:e2::e7
v6 Source = 2001:db8:c000:282::9a
v6 Dest = 2001:db9:0:e2::e7

Figure 15.26. 6rd deployment example.

- *Dual-Stack Lite*. Provides a means for service providers to better utilize increasingly scare IPv4 addresses while deploying IPv6 out to customers' premises.

6rd (IPv6 Rapid Deployment). RFC 5569 (161) defines "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)," a technique to enable a service provider to provision IPv6 addresses to end customers while maintaining an IPv4 infrastructure. This method calls for tunneling of customer IPv6 traffic from the customer premises to an IPv6 destination via modified 6to4 technique. The modification entails use of the service provider's IPv6 prefix (/32) in lieu of the 6to4 prefix, 2001::/16.

Like 6to4, the next 32 bits of the IPv6 prefix consists of the IPv4 address of the 6to4 gateway, in this case the customer premises broadband router. Hence, a 6to4 prefix is defined as 2001:<32-bit IPv4 address>::/48, while the 6rd prefix is <32-bit service provider IPv6 prefix>:<32-bit IPv4 address>::/64.* This enables the service provider to provision a /64 to each customer, which comprises a single IPv6 subnet. Thus, a service provider with an RIR allocated IPv6 block 2001:db8::/32 would provision a customer gateway device with IPv4 address 192.0.2.130 with a 6rd subnet address of 2001:db8:c000:282::/64 as shown in Figure 15.26.

A device within the residence requiring an IPv6 address would be assigned an address from this subnet. For example in Figure 15.26, a PC is assigned IPv6 address 2001:db8:c000:232::9a. The 6rd customer gateway tunnels native IPv6 packets over IPv4 to a 6rd gateway. The other address related change between 6rd and 6to4 is that the anycast 6to4 address is fixed (192.88.99.1), while the 6rd anycast address is defined by the service provider themselves within its own address space. Each customer router must be provisioned with the 6rd relay agent or anycast address(es).

The 6rd relay router terminates the IPv4 tunnel, then routes the IPv6 packet natively to its destination. The use of the service provider's prefix enables 6rd-reachable destinations to be advertised along with the service provider's native IPv6 traffic.

* Use of a portion of the IPv4 address, say 24 lower order unique bits of a common 10.0.0.0 address allows a longer service provider prefix if desired.

Dual-Stack Lite. Dual-stack lite is a technology that enables a service provider to more efficiently utilize the diminishing pool of available public IPv4 addresses, while facilitating long-term support of IPv4 addresses assigned to customer network devices (162). Service providers typically assign an address to a customer router or gateway which interfaces directly to the broadband access network. The customer gateway performs DHCP server functions in assigning IP addresses to IP devices in the home network. It is expected that such home network devices will support only IPv4 for quite some time.

The components comprising a dual-stack lite implementation include the following:

- Basic Bridging BroadBand (B4) element bridges the IPv4 home network with an IPv6 network; the B4 function may reside on the customer gateway device or within the service provider network.
- Softwire IPv4-in-IPv6 tunnel between the B4 and the AFTR.
- Address Family Translation Router (AFTR) terminates the IPv4-in-IPv6 softwire tunnel with the B4 element and also performs IPv4–IPv4 network address translation functionality.

Figure 15.27 illustrates the inter-relationship of these three components within an end-to-end IP connection. Starting on the left of the figure, the IPv4 host obtains an IPv4 address, 10.1.0.2, from the DHCP server function of the customer gateway. Let's say this IPv4 host desires to connect to a web site, which has been resolved to IP address 192.0.2.21. The IPv4 host formulates an IP packet with source address 10.1.0.2 and source port of 1000, for example, and destination address 192.0.2.21 port 80. The host transmits this packet to its default route, the customer gateway.

The customer gateway in this example includes the B4 element, which sets up the softwire IPv4-in-IPv6 tunnel if it is not already established. The customer gateway has been assigned an IPv6 address on its WAN port (facing the service provider network) and it is over this connection that the tunnel is established. The customer gateway has also been configured with the AFTR IPv6 address manually or via DHCPv6. As shown in Figure 15.27, the B4 element encapsulates the original IPv4 packet with an IPv6 header and transmits it to the AFTR.

The AFTR terminates the tunnel and removes the IPv6 header. The AFTR then performs an IPv4–IPv4 NAT function. This is required to translate the original packet's

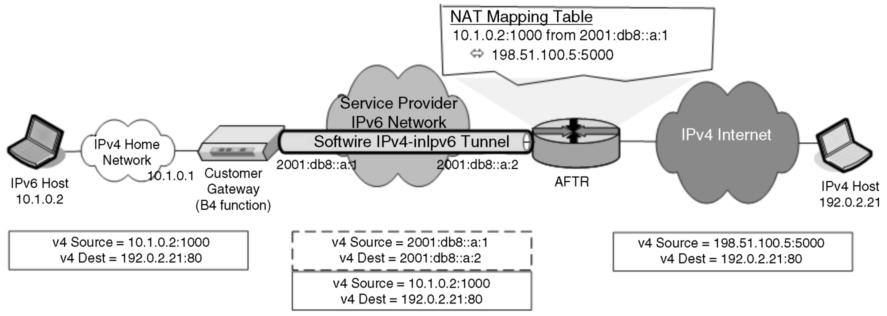

Figure 15.27. Dual-stack lite architecture (162).

private (RFC 1918) IPv4 source address into a public IPv4 address. Thus, the service provider must provision a pool of public IPv4 addresses which can be used as source IP addresses on packets destined for an IPv4 destination as in this case. This pooling enables the service provider to more efficiently utilize the increasingly scare public IPv4 address space. The AFTR also generally performs port translation as well and must track this mapping for each NAT operation in order to properly map IPv4 addresses and port numbers bidirectionally.

In Figure 15.27, the AFTR has mapped the customer's source IPv4 address and port, 10.1.0.2:1000 to 198.51.100.5:5000. Since all customers will utilize 10.0.0.0 address space, the NAT mapping table also tracks the tunnel over which the packet originated. The packet ultimately transmitted to the destination host includes this mapped IPv4 address and port, 198.51.100.5:5000. Return packets destined for this address/port are mapped to [destination] address 10.1.0.2:1000 and tunneled to 2001:db8::a:1.

Customers deploying native IPv6 or dual-stack hosts can have respective IPv6 addresses provided by DHCPv6 functionality implemented in the customer gateway or via autoconfiguration. IPv6 packets transmitted over the home network to the customer gateway would not utilize the softwire tunnel, but instead be routed natively over the service provider IPv6 access network.

15.6.2 Enterprise Deployment Scenarios

There's certainly no shortage of technology options when considering an IPv6 implementation approach. Having many options is good, but it can be intimidating. 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, as well as time frames. Given the proliferation of dual-stack support in leading operating systems and networking products, a dual-stack approach is likely to be the most prevalent approach. In this section, we'll review some basic IPv6 implementation scenarios to provide a flavor for various macro-level approaches. In reviewing these, let's use the basic diagram in Figure 15.28 as a baseline. In this figure, we have a client, in this case with IPv4 applications, IPv4 sockets API, and TCP/IPv4 stack, and a server with

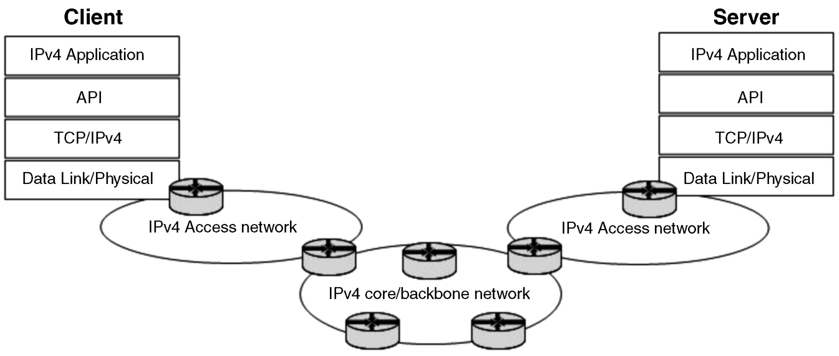

Figure 15.28. Base case for IPv4 network—initial state prior to migration (147).

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 access to 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.

15.6.3 Core Migration Scenario

The first scenario involves initially supplementing the backbone or core network with IPv6. This scenario requires upgrading of 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 properly implemented 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.

15.6.4 Server Side Migration

The next scenario assumes our base case as the initial state followed by 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

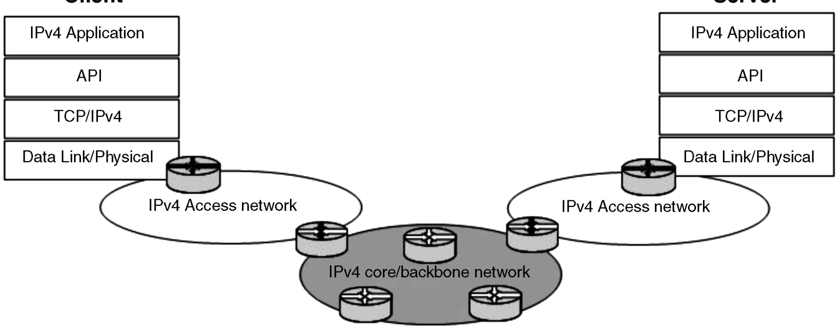

Figure 15.29. Core migration scenario (147).

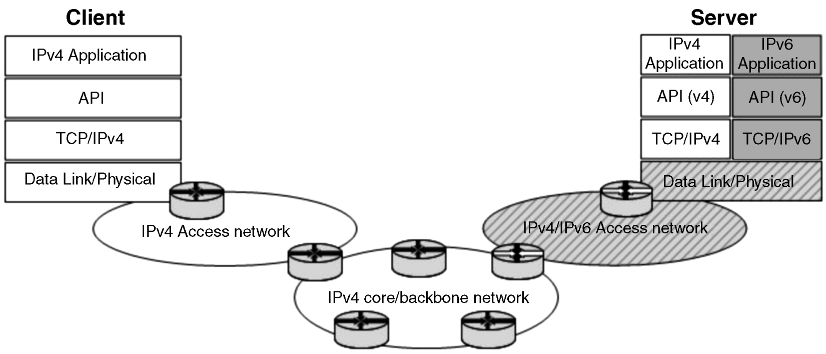

Figure 15.30. Server side migration scenario (147).

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 15.30 are within a common 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.
- This scenario may also reflect an interorganizational connection via an IPv4 network, for example, 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 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 interorganizational link.
- If we ignore the IPv4 portion of the dual stack on the server and consider the server IPv6-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 a host-based translation mechanism such as BIS or BIA to translate between IPv4 and IPv6 headers.

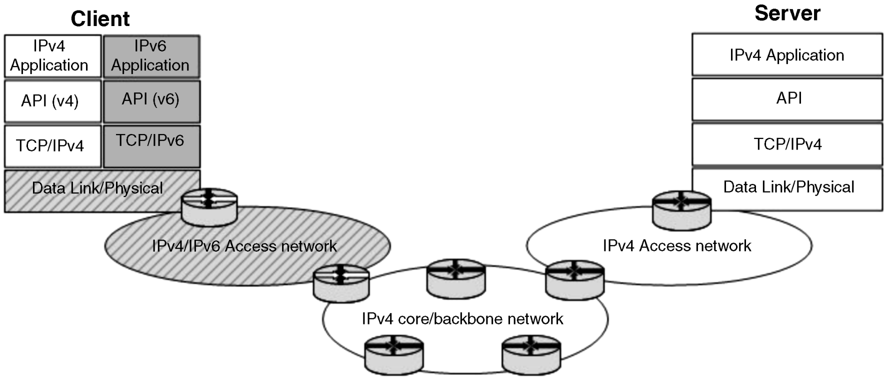

Figure 15.31. Client side migration scenario (147).

15.6.5 Client Side Migration

Figure 15.31 represents a scenario starting with our initial configuration of Figure 15.28, followed by the migrating of 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 or 7, or Mac OS X. This scenario represents the following example cases:

- After a particular organization's complete migration to IPv6, continued use of dual stack would support access to IPv4 web sites. Hence, this may be a post-transition 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 within an organization 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 access 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.

15.6.6 Client–Server Migration

A commonly anticipated approach to IPv6 migration within an organization features upgrading of clients and servers at about the same time or on a rolling basis, including applications as appropriate. The use of dual-stack deployments facilitates the deployment to clients and servers over time. Special consideration must be given to application

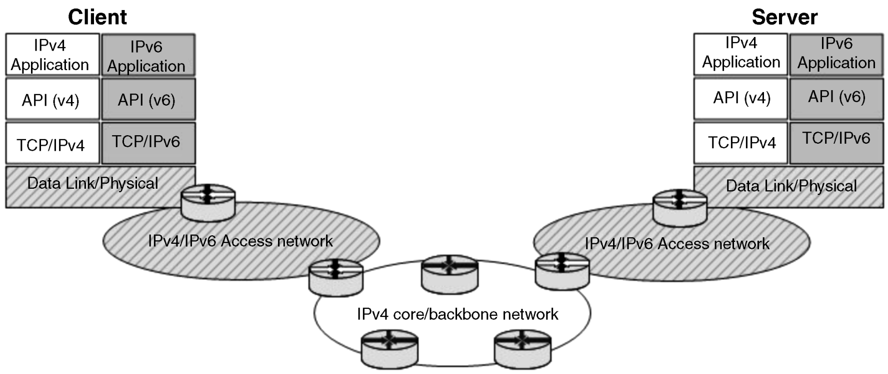

Figure 15.32. Dual-stack deployment scenario (147).

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.

This scenario reflects the intra-enterprise example described above, as well as a broader set of examples per the table below, considering also single-stack cases.

15.6.7 Overall IPv6 Implementation Planning

Despite substantial differences in address lengths and formats, among other aspects, IPv4 and IPv6 are both network layer protocols which enable application communications over a common layer 1 and 2 network. This facilitates coexistence of IPv4 and IPv6 during a transition interval over the same physical networks. However, layers above layer 3 will more likely be impacted, especially in terms of applications that display, utilize, or enable entry of IP addresses.

IPv6 implementation presents a number of challenges. Key among them are defining and organizing an overall plan for IPv6 network allocation and deployment. This is where the discipline of an IPAM system can help. While there are numerous detailed steps involved, the following four high-level steps are recommended for planning and performing such a transition.

1. Baseline your current environment; this step itself can be accomplished by implementing the practices we've discussed throughout this book.
   - Inventory and baseline the current IPv4 network environment in determining 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.
   - 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.

  - Identify and record associated Domain Name Server (DNS) configurations and resource records associated with IPv4 devices.
  - Inventory network devices (infrastructure and end user) to assess current and expected future IPv4/IPv6 level of support.
  - Analyze applications for potential migration impacts and if so, plan for addressing these impacts.

2. Plan your IPv6 deployment
   - Map out a strategy for IPv6 implementation including consideration of the following:
     - Application migration and required upgrades
     - Networking/router migrations or upgrades
     - Selection of coexistence technologies from tunneling, translation, and/or dual-stack approaches and planning corresponding DNS impacts
     - Selection of IPv6 device configuration strategy, for example, stateless autoconfiguration, stateful configuration, or hybrid
   - Consider time frame requirements if any, analyze dependencies across affected elements, and determine budgeting requirements to derive a migration plan.
3. Execute the deployment process
   - Begin execution of the deployment plan based on the prior step and project manage the process with respect to schedules, budget, dependencies, and contingency plans.
   - 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.
   - Update DNS and DHCP services to accommodate the transitioning environment, for example, to support both IPv4 and IPv6 hosts, as well as IPv4 and IPv6 transport.
4. Manage the IPv4–IPv6 Network
   - Manage IPv4 and IPv6 address space and corresponding DHCP and DNS services.
   - Based on your plan, some IPv4 space may be decommissioned. This can be done as portions of your network fully migrate to IPv6 or as a final decommissioning step across the network.
   - You may desire to leave IPv4 protocol operational throughout or on portions of your network depending on requirements to service external hosts which may be IPv4-only based on internal policies.