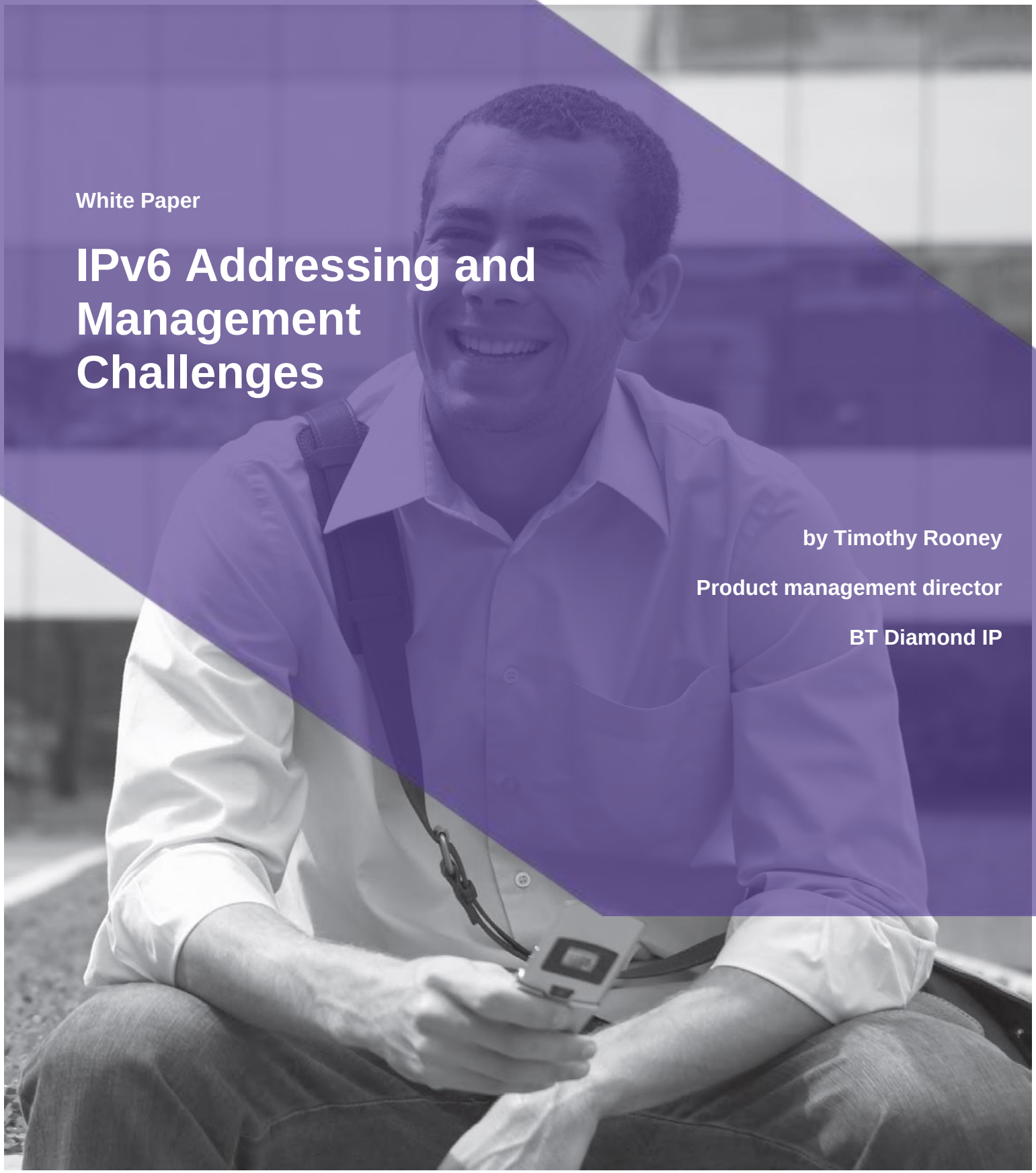


White Paper

IPv6 Addressing and Management Challenges

by Timothy Rooney

Product management director

BT Diamond IP

BT Diamond IP

IPv6 Addressing and Management Challenges

By Tim Rooney, Director, Product Management

Introduction

IPv6 was originally specified in the mid-1990's to address a then-urgent need to supplement the rapidly diminishing IPv4 address space. At the time work was begun in earnest on defining version 6 of the Internet Protocol[1] or IPng as it was initially called, the Internet was just starting to catch on with the general public. More and more enterprises were supplementing their internal networks to support the TCP/IP protocol suite in order 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 the IPv4 address space. Classless Inter-Domain Routing (CIDR) enabled Regional Internet Registries (RIRs) and Internet Service Providers (ISPs) to allocate address space more efficiently than with former classful-only allocation methods. For example, instead of allocating an entire "class B" /16 network to a constituent requiring 2000 IP addresses, the RIR or ISP could allocate a /21 network with CIDR. Given that there are $2^{(21-16)} = 2^5 = 32$ /21s in a /16, 32 similarly sized customer requests could be handled from the address space that may have formerly been distributed to one.

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

Despite these schemes to better utilize IP addresses, growing subscriber bases and the availability of other IP-enabled devices continue to rapidly increase IPv4 address consumption, while diminishing available capacity. And in many parts of the world, current allocations of IPv4

[1] IP version 5 was never implemented as an official version of IP. The version number of '5' in the IP header was assigned to denote packets carrying an experimental real-time stream protocol called ST, the Internet Stream Protocol. To learn more about ST, refer to RFC 1819.

address space are inadequate. For example, Asia has been allocated only nine percent of IPv4 space but contains half of the world's population. Given this imbalance, it's easy to understand why 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.

Nevertheless, IPv4 address space availability is diminishing throughout the world. Every Regional Internet Registry (RIR) has issued notifications that IPv4 space availability is limited and will be exhausted within "a few years." RIRs are responsible for IP address allocation to Internet Service Providers, who in turn allocate space to enterprises, service providers and any organization requiring IP address space. Ultimately, this exhaustion will impact organizations requiring public IP address space.

Even if your organization currently has more than ample public and private IPv4 address space for the foreseeable future, consideration of IPv6 is recommended. Such consideration should include planning for IPv6 implementation on external resources such as web and email servers. After IPv4 space has been depleted, organizations requiring address space will only be able to obtain IPv6 space, thereby enabling them to communicate on the Internet via IPv6 only. To enable these organizations to access web and email services on your site, implementation of IPv6, at least externally, will be required. While your particular implementation may not be necessary for "a few years," it's prudent to begin learning about and planning for IPv6 now.

This white paper is intended to help you learn about IPv6. Our companion white paper, *IPv4-IPv6 Transition and Co-Existence Strategies* provides input on the planning process. We'll begin with an overview of the key benefits of IPv6, then delve into the details of IPv6 addressing and allocation. Next, we'll discuss IPv6 address assignment strategies using autoconfiguration or DHCPv6. With this foundation of IPv6 addressing information, we'll then review the key IP management challenges related to IPv6.

IPv6 Addressing Overview

The Internet Engineering Task Force (IETF) has attempted to develop IPv6 as an evolution of IPv4 to enable IPv6 to provide many new features while building on the foundational concepts that made IPv4 so successful. Key IPv6 features include:

- *Expanded addressing* – 128 bits hierarchically assigned with address scoping (e.g., local link vs. global) to improve scalability
- *Routing* – Strongly hierarchical routing supporting route aggregation
- *Performance* – Simple (unreliable) datagram service
- *Extensibility* – New flexible extension headers provide built-in extensibility for new header types and more efficient routing
- *Multimedia* – Flow-label header field facilitates QoS support
- *Multicast* – Replaces broadcast and is compulsory
- *Security* – Authentication and encryption are built-in and mandatory

- *Autoconfiguration* – Stateless and stateful address configuration by IP devices with duplicate address detection
- *Mobility* – Mobile IPv6 support eliminates triangular routing of Mobile IPv4

IPv6 Address Capacity

The most striking difference between IPv4 and IPv6 is the tremendous expansion of IP address space size. Whereas IPv4 uses a 32-bit IP address, IPv6 uses 128 bits. A 32-bit address field provides a maximum of 2^{32} addresses or 4.2 billion addresses. A 128-bit address field provides 2^{128} addresses or 340 trillion trillion trillion addresses or 340 undecillion[2] (3.4×10^{38}). To put some context around this tremendously large number, consider that this quantity of IP addresses:

- Equals more than 5×10^{28} IP addresses per person on Earth
- Equals more than 4.3×10^{20} IP addresses per square inch of the Earth's surface
- Equals nearly 1.4×10^{13} IP addresses per millimeter from Earth to the nearest galaxy, Andromeda, 2.5 million light years away

Address Types

Three types of IPv6 addresses have been defined. Like IPv4, these addresses apply to interfaces, not nodes. Thus, a printer with two interfaces would be addressed by either of its interfaces. The printer can be reached on either interface, but the printer node does not have an IP address per se, though one could consider the loopback address of a device as the node address. Of course, for end users, DNS can hide this subtlety by enabling a host name to map to one or more interface IP addresses. The address types supported within IPv6 are:

- *Unicast* – the IP address of a single interface analogous to the common interpretation of an IPv4 host address (non-multicast/non-broadcast IPv4 address).
- *Anycast* – a single IP address for a set of interfaces usually belonging to different nodes, any *one* of which is the intended recipient. An IP packet destined for an anycast address is routed (according to routing table metrics) to the nearest interface configured with the anycast address. The concept is that the sender doesn't necessarily care which particular host or interface receives the packet, but that one of those sharing the anycast address receive it. Anycast addresses are assigned from the same address space from which unicast addresses have been allocated. Thus, one cannot differentiate a unicast address from an anycast address by sight; however, anycast addresses may be assigned a different network prefix so routers can distinguish them and route them to the nearest interface configured with the anycast address, but this is not required. Anycast in IPv4 networks has recently created a buzz in providing similar *closest routing to the intended service*, such as for DNS servers by using a shared unicast IPv4 address.
- *Multicast* – an IP address for a set of interfaces typically belonging to different nodes, *all* of which are intended recipients. This of course is similar to IPv4 multicast. Unlike IPv4, IPv6

[2] Using the American definition of undecillion of 10^{36}, not the British definition which is 10^{66}.

does not support broadcasts. Instead, applications that utilized broadcasts in IPv4, such as DHCP, use multicast. For example, the well-known DHCP_Servers_and_Relay_Agents multicast group is used in IPv6 in lieu of broadcasts and relay agent configurations. IPv6 multicast addresses also have scope, a useful feature for constraining the breadth of multicast members to whom to transmit the message to within the same link, site, organization or globally.

A device interface may have multiple IP addresses of any or all address types[3]. While typically these IP address are globally unique or of a global scope, IPv6 also defines a *link local* scope of IP addresses to uniquely identify interfaces attached to a particular link, such as a LAN. IPv6 packets destined to link local addresses are not routed and are communicated only to hosts on the same link.

Address Notation

Recall that IPv4 addresses are represented in dotted decimal format where the 32-bit address is divided into four 8-bit segments, each of which is converted to decimal, then separated with "dots." If you think that is difficult, IPv6 addresses are represented using hexadecimal. The 128-bit IPv6 address is divided into eight 16-bit segments, each of which is converted to hexadecimal, then separated by colons. Each hexadecimal "digit" represents four bits per the mapping of each hex digit (0-F) to its four-bit binary mapping shown below. Each hex digit corresponds to four bits with possible values of:

0 = 0000	4 = 0100	8 = 1000	C = 1100
1 = 0001	5 = 0101	9 = 1001	D = 1101
2 = 0010	6 = 0110	A = 1010	E = 1110
3 = 0011	7 = 0111	B = 1011	F = 1111

After converting 128-bit IPv6 address from binary into hex, we group sets of four hex digits and separate them with colons. We'll use the term *quad-hex* to represent a grouping of four hex digits or 16 bits; thus, we have eight quad-hex values separated by colons, rendering an IPv6 address appearing as shown in Figure 1

Figure 1: IPv6 Address Notation

[3] Including IPv4 addresses as well for dual-stack deployments.

Unlike IPv4 addresses with their four decimal values, each between 0 and 255 and separated by dots, IPv6 addresses consist of up to eight quad-hex values, each between 0 and FFFF, separated by colons. Two forms of abbreviation are permitted when writing IPv6 addresses. First, leading zeroes within a quad-hex section, i.e., between colons, may be dropped. Thus, the address in Figure 1 could be abbreviated 4C62:E849:5F62:AB41:0:0:0:801.

The second form of abbreviation is the use of a double colon to represent one or more consecutive sets of quad-hex zero strings. Using this form of abbreviation, the address can be further abbreviated as 4C62:E849:5F62:AB41::801 Note that only one double colon may be used within an address representation. Since there are always eight quad-hex segments in the address, it is easy to calculate how many of them are zero with one double-colon notation; however, it would be ambiguous with more than one.

Consider the address: 4C62:0:0:56FA:0:0:0:B5. We can abbreviate this address as either 4C62::56FA:0:0:0:B5 or 4C62:0:0:56FA::B5. We can easily calculate that the double colon denotes two quad-hex segments (eight total minus six quad-hex segments shown) in the first case and three (eight minus five shown) in the second notation. If we attempted to abbreviate this address as 4C62::56FA::B5, we cannot unambiguously decode this, as it could represent any of the following possible addresses:

4C62:0:56FA:0:0:0:0:B5

4C62:0:0:56FA:0:0:0:B5

4C62:0:0:0:56FA:0:0:B5

4C62:0:0:0:0:56FA:0:B5

Thus the requirement holds that only one double colon may appear in an IPv6 address. The IPv6 address is generally divided into three fields shown in Figure 2.

Figure 2: IPv6 address structure

The global routing prefix is akin to an IPv4 network number and would generally be used by routers to route packets to the router(s) locally serving networks or subnets associated with the prefix. For example, a customer of an ISP would typically be assigned a /48 global routing prefix and all packets destined to the customer would contain the corresponding global routing prefix value. Referring to Figure 2, n = 48 bits in this case. When denoting a network, the global routing prefix is written followed by the network size, called the prefix length. For example, if our previous IPv6 address above, 4C62:E849:5F62:AB41::801, resided within such a /48 network, the network notation (zeroing out bits 49-128) would be 4C62:E849:5F62::/48. Note that we use the double colon to signify the remaining bits are zero when specifying the network address. This network notation is analogous to that for IPv4, where we zero out the non-network bits to denote the network or subnet address, e.g., 172.16.0.0/12.

The subnet ID portion of an IPv6 address provides a means to denote particular subnets within the organization. Our ISP customer with a /48 may choose to use 16 bits for the subnet ID, providing 2^{16} or 65,536 subnets. Thus, in Figure 2 m would equal 16. This leaves 128 – 48 – 16 = 64 bits for

the interface ID. The Interface ID denotes the interface address of the source or intended recipient for the packet. The IPv6 addressing architecture[4] requires that the global unicast address space that has been allocated for use so far requires a 64-bit interface ID field. One of the unique aspects of this IPv6 address structure in splitting a network ID consisting of the global routing prefix and subnet ID, from an Interface ID, is that a device can retain the same Interface ID independently of the network to which it is connected, effectively separating "who you are," your interface ID, from "where you are," your network address. As we'll see, this convention facilitates address autoconfiguration. But before diving into individual address assignment, let's start at the top and consider IPv6 address block allocations.

IPv6 Address Block Allocation

Though IPv6 addresses are represented differently than IPv4 addresses, the allocation process works essentially the same way. The main difference is converting hexadecimal to binary and back versus decimal to binary and back. Though most allocations will be made on hex digit if not quad-hex segment boundaries, nullifying the need for binary conversion, the binary breakdown is still needed for managing blocks leftover after the allocation is made. This will become clear in the example in the following section using a process of allocation of the smallest available free block as is commonly used for IPv4 address space today. Due to the vast difference in available address space, RIRs allocating IPv6 address space may utilize a best-fit algorithm or a sparse allocation algorithm. We'll outline each of these algorithms in this section along with a random version, using the example IPv6 network 4FFE:0320::/32.

Best Fit Allocation

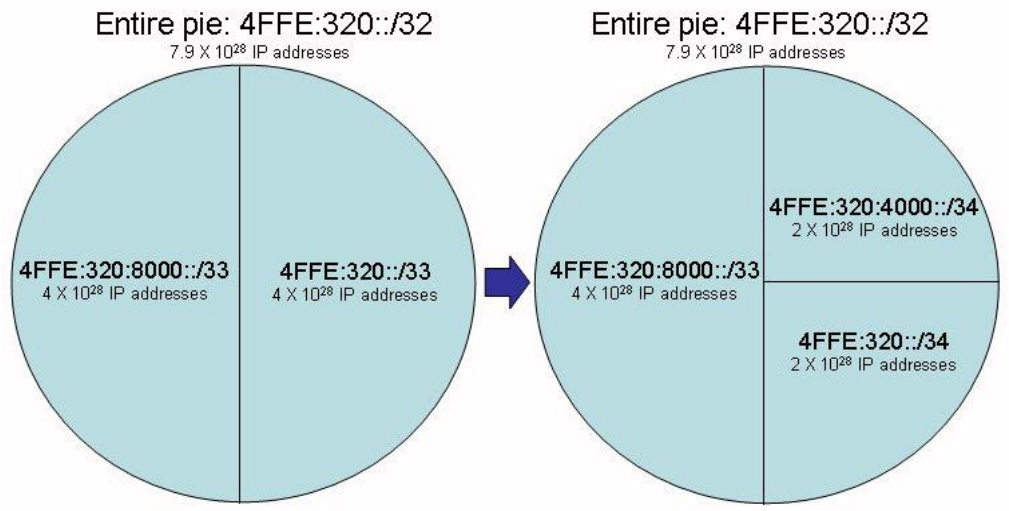
Illustrating the best fit approach, we'll follow the same basic algorithm used for IPv4 networks today; namely, allocate the smallest available block that meets the need of the block size requested. This approach optimizes address utilization efficiency by successively halving the address space to the size required. This enables retention of larger remaining (unallocated "halves") blocks of address space available for future requests and alternative allocations. After converting the hexadecimal to binary, the process is identical in terms of successive halving by seizing the next bit for the network portion of the address. For example, consider our example network 4FFE:0320::/32, represented in binary as:

```
0100 1111 1111 1110 :0000 0011 0010 0000 :0000 0000 0000 0000:0000...
```

One way to visualize this best-fit halving process from an overall allocation perspective is to view the address space as a pie chart as illustrated in Figure 3. If the pie represents the base network, 4FFE:320::/32, then cutting it in half renders two /33s as shown on left of Figure 3. We can then leave one of the /33s as "free" (left half) and slice the other /33 (right half) into two /34s as shown on the right.

[4] Hinden and Deering, IP Version 6 Addressing Architecture, RFC 4291, February, 2006.

Figure 3: Splitting IPv6 Networks



So as space is halved, one of the halves is still free or available for allocation, while the other half can be further split or ultimately allocated for the need at hand. Thus, large free blocks are retained for future allocations. To continue this example, let's say we'd like to allocate three /40 networks from this space. In following the best fit allocation example from a binary perspective, by successively halving the address space down to a /40 size by taking the next bit, assigning the "1" value as free and the "0" value as allocated for further halving. This process of successive halving of our 4FFE:0320::/32 network is illustrated table below by the state of the bold (red) bits in the IPv6 address. The resulting set of networks shown are available for further allocation or splitting.

0100 1111 1111 1110 : 0000 0011 0010 0000 : **1**000 0000 0000 0000 : 0000…
4FFE:320:8000::/33

0100 1111 1111 1110 : 0000 0011 0010 0000 : **01**00 0000 0000 0000 : 0000…
4FFE:320:4000::/34

0100 1111 1111 1110 : 0000 0011 0010 0000 : **001**0 0000 0000 0000 : 0000…
4FFE:320:2000::/35

0100 1111 1111 1110 : 0000 0011 0010 0000 : **0001** 0000 0000 0000 : 0000…
4FFE:320:1000::/36

0100 1111 1111 1110 : 0000 0011 0010 0000 : **0000 1**000 0000 0000 : 0000…
4FFE:320:800::/37

0100 1111 1111 1110 : 0000 0011 0010 0000 : **0000 01**00 0000 0000 : 0000…
4FFE:320:400::/38

0100 1111 1111 1110 : 0000 0011 0010 0000 : **0000 001**0 0000 0000 : 0000…
4FFE:320:200::/39

```
0100 1111 1111 1110 : 0000 0011 0010 0000 : 0000 0001 0000 0000 : 0000...
4FFE:320:100::/40

0100 1111 1111 1110 : 0000 0011 0010 0000 : 0000 0000 0000 0000 : 0000...
4FFE:320::/40
```

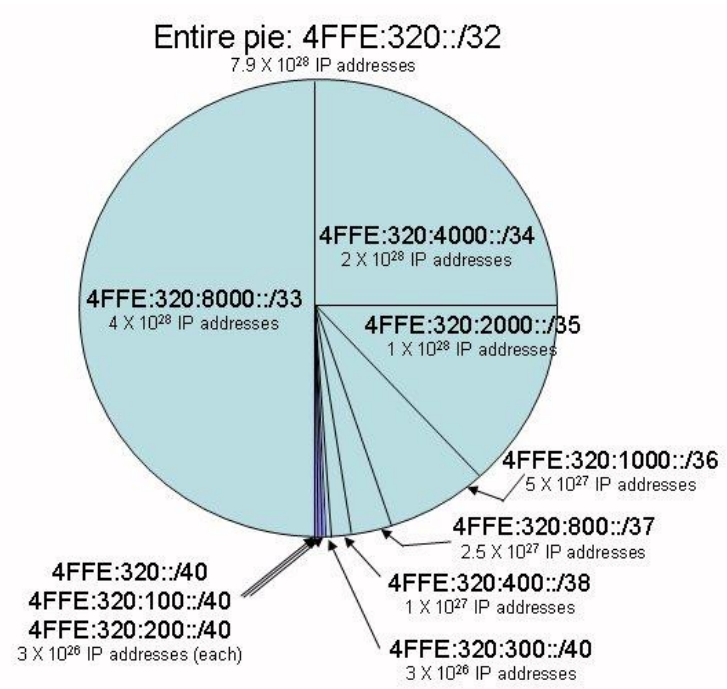
Through this successive halving of address space, we readily have two /40 networks available (highlighted above): 4FFE:320::/40 and 4FFE:320:100::/40. In order to allocate our third /40 using the best fit approach, we can take the next smallest available network, in this case 4FFE:320:200::/39, shown right above the two highlighted /40 networks just allocated, and split it into two /40s:

```
0100 1111 1111 1110 : 0000 0011 0010 0000 : 0000 0010 0000 0000 : 0000...
4FFE:320:200::/40

0100 1111 1111 1110 : 0000 0011 0010 0000 : 0000 0011 0000 0000 : 0000...
4FFE:320:300::/40
```

Following the same algorithm, we split this in half by assigning the next bit, one to 0 (assigned) and one to 1 (free or available for future assignment) yielding two /40s. Thus we can choose to allocate 4FFE:320:200::/40, and 4FFE:320:300::/40 will remain free for future assignment. Our resulting three /40s for allocation are 4FFE:320::/40, 4FFE:0320:100::/40 and 4FFE:0320:200::/40. The other /40, 4FFE:0320:300/40, is available for future assignment. Figure 4 illustrates the results of the process graphically.

Figure 4: Allocation Results of Carving /40s from a /32



Sparse Allocation Method

You may notice from the prior algorithm that by allocating a /40 from a /32, we incrementally extend the network length to the 40th bit. We then assign the network by assigning a 0 or 1 to the 40th bit as our first two /40 networks. In essence, we process each bit along the way, considering "1" the free block and "0" the allocated block. However, if we step back and consider the eight subnet ID bits that extend the /32 to a /40 as a whole, instead of incrementally halving the network, we observe that we've actually allocated our subnets by simply numbering or counting within the subnet ID field as denoted by the bold (red) bits below:

0100 1111 1111 1110 : 0000 0011 0010 0000 : **0000 0000** 0000 0000 : 0000…
4FFE:320::/40

0100 1111 1111 1110 : 0000 0011 0010 0000 : **0000 0001** 0000 0000 : 0000…
4FFE:320:100::/40

0100 1111 1111 1110 : 0000 0011 0010 0000 : **0000 0010** 0000 0000 : 0000…
4FFE:320:200::/40

Thus, if you knew in advance that the original /32 network would be carved into equal-sized /40 blocks, a simpler allocation method would be to simply increment the subnet ID bits. The next allocation would use subnet ID values of 0000 0011, 0000 0100, 0000 0101 and so on. In many networks, this uniformity policy of allocating /40 blocks may not apply, in which case the method of successive halving may be more appropriate.

On the other hand, if you are a Local Internet Registry or ISP, a sparse allocation method may be more appropriate. The sparse allocation method seeks to spread out allocations to provide room for growth within each allocated network by allocating with the maximum space between allocations. If space originally allocated to a constituent proves inadequate, a subsequent allocation cont*iguous with the prior allocation* is more likely using this sparse allocation approach. Contiguous allocations are always preferable as they vastly simplify route aggregation and routing table and route advertisement packet overhead.

Like the best-fit approach, the sparse algorithm also features halving of the available address space. But instead of continuing this process down to the smallest size, it calls for allocating the next block on the edge of the new half. This results in allocations being spread out and not optimally allocated. The philosophy is that this provides room for growth of allocated networks by leaving ample space between allocations in the plentiful IPv6 space. For example, our allocation of three /40's from our 3FFE:0320::/32 space would look like:

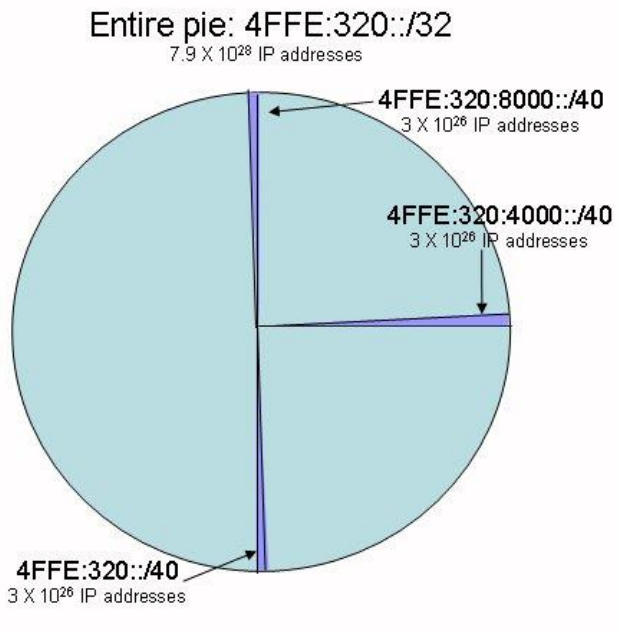
0100 1111 1111 1110 : 0000 0011 0010 0000 : **0000 0000** 0000 0000 : 0000…
4FFE:320::/40

0100 1111 1111 1110 : 0000 0011 0010 0000 : **1000 0000** 0000 0000 : 0000…
4FFE:320:8000::/40

0100 1111 1111 1110 : 0000 0011 0010 0000 : **0100 0000** 0000 0000 : 0000…
4FFE:320:4000::/40

These translate as 4FFE:0320::/40, 4FFE:00320:8000::/40 and 4FFE:0320:4000::/40, respectively. This allocation enables spreading out of address space as illustrated in Figure 5. Note that our subnet ID bits are effectively incremented from left to right, instead of the conventional right-to-left method normally used for counting.

Figure 5: Sparse Allocation Example



RFC 3531 describes the sparse allocation method. Because network allocations are expected to follow a multi-layered allocation hierarchy, several sets of successive network bits can be used by different entities for successive allocation. For example, an Internet Registry may allocate the first macro block to a regional registry, which in turn will allocate from that space to a service provider, which may in turn allocate from that subspace to customers, who can further allocate across their networks. RFC 3531 recommends the higher level allocations, e.g., from the registries, utilize the leftmost counting or sparse allocation, the lowest level allocations use the rightmost or best-fit allocation and others in the middle use either of those options or a center-most allocation. This flexibility in allocation enables plans for growth without necessarily allocating the expected fully-grown network to another organization on day one.

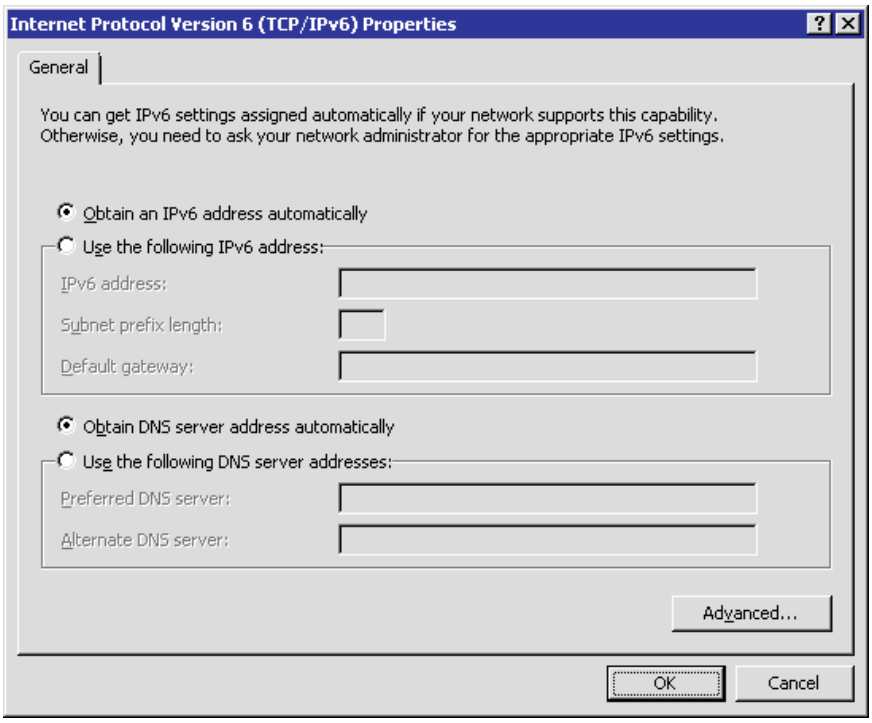
Random Allocation Method

The random allocation method selects a random number within the sizing of the subnetwork bits to allocate subnetworks. Using our /40 allocations from a /32, a random number would be generated between 0 and $2^8 - 1$ (i.e., 0-255) and allocated assuming it's still available. This method provides a means for randomly spreading allocations across allocated entities, e.g., as privacy provision, and generally works best for "same-size" allocations.

IPv6 Address Assignment

Once block and subnets have been allocated, individual IP addresses can be assigned to hosts based on which subnets those hosts are connected to. IPv6 hosts can be assigned addresses statically (e.g., entered manually by an administrator), dynamically through DHCPv6 or dynamically through autoconfiguration. Entering an address statically would be based on the particular host IPv6 stack and the mechanisms for entering this information via the interface. An example Microsoft Windows screen shot below illustrates one such user interface.

Figure 6: Example Microsoft Windows Interface for Defining IPv6 Addressing Configuration



IPv6 Address Autoconfiguration

One of the major benefits of IPv6 is the ability for devices to automatically configure their own IPv6 address that will be unique and relevant to the subnet to which it is presently connecting[5]. The process is facilitated by the implementation of *neighbor discovery* in IPv6. Neighbor discovery enables nodes to discover the presence of other IPv6 nodes, to identify their link layer addresses, to discover routers serving the subnet and to perform duplicate address detection.

[5] Note that some IPv4 protocol stacks, such as those provided with Microsoft Windows 2000 and XP, among others, perform address autoconfiguration utilizing the IPv4 "link local" address space, 169.254.0.0/16.

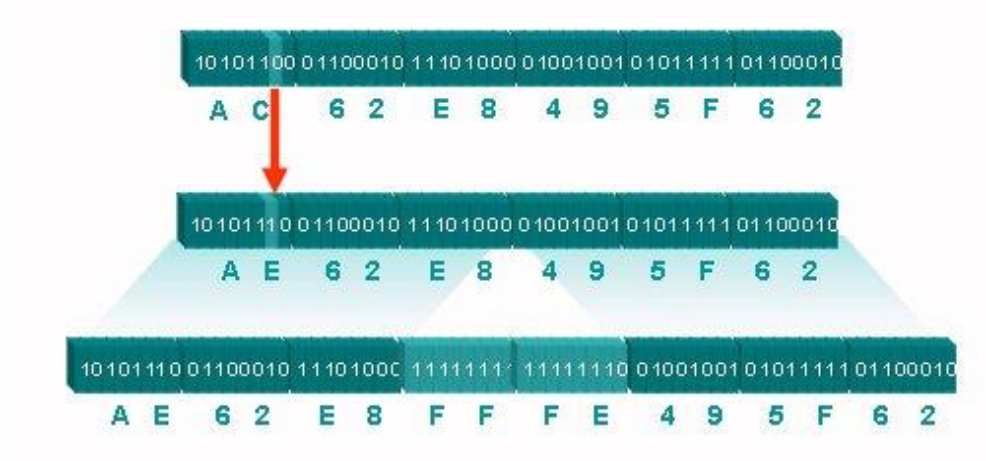
Discovery of routers enables IPv6 nodes to automatically identify routers on the subnet, negating the need to configure a default gateway in the device's IP configuration. *Prefix discovery* enables a device to identify the network prefix(es) and corresponding prefix length(s) assigned to the link. This information is used in defining unicast addresses during the autoconfiguration process. The discovery process entails a router periodically issuing advertisements on the link indicating its IP address, its ability to provide default gateway functionality, its link layer address, the network prefix(es) served on the link including corresponding prefix length and valid address lifetime (described later), as well as other configuration parameters. The advertised network prefix provides the global routing prefix and subnet ID portion of the IPv6 address per Figure 2; all that is needed is the Interface ID.

Interface Identifier

The IPv6 addressing architecture RFC stipulates that all unicast IPv6 addresses, other than those beginning with binary 000, must use a 64-bit Interface Identifier (ID) derived using the modified EUI-64 algorithm. The "unmodified" EUI-64 algorithm entails concatenating the 24-bit company identifier issued by IEEE to each network interface hardware manufacturer (e.g., the initial 24 bits of an Ethernet address) with a 40-bit extension identifier. For 48-bit Ethernet addresses, the company identifier portion of the Ethernet address (first 24 bits) is followed by a 16-bit EUI label, defined as hexadecimal FFFE, followed by the 24-bit extension identifier, i.e., the remaining 24 bits of the Ethernet address.

The modification required to convert an unmodified EUI-64 identifier into a modified EUI-64 identifier entails inverting the "u" bit (universal/local bit) of the company identifier field. The "u" bit is the seventh most significant bit in the company identifier field. Thus the algorithm for a 48-bit MAC address is to invert the "u" bit and insert the hexadecimal value FFFE between the company identifier and the interface identifier. This is illustrated in Figure 5 using a MAC address of AC-62-E8-49-5F-62. The resulting interface ID is AE62:E8FF:FE49:5F62.

Figure 5: Example of Modified EUI-64 Algorithm Generating an Interface ID from an Ethernet MAC Address



For non Ethernet MAC addresses, the algorithm uses the link layer address as the Interface ID, zero padding (from the "left"). For cases where no link layer address is available, e.g., on a dial-

up link, a unique identifier utilizing another interface address on the box, a serial number of the device or other device specific identifier is recommended.

Completing the Autoconfiguration Process

The next step in the autoconfiguration process involves concatenating the network prefix information received from the router and the device's Interface ID. However, there are some additional considerations in this process. Obtaining an IP address is critical for communicating on the network, but identifying the location or IP addresses of required services such as DNS or NTP are equally critical for device initialization on the IPv6 network.

There are three basic forms of IPv6 address autoconfiguration:

- *Stateless* – not dependent on the state or availability of external assignment mechanisms, e.g., DHCPv6. The device attempts to configure its own IPv6 address(es) without external or user intervention.
- *Stateful* – relies solely an external address assignment mechanism such as DHCPv6. The DHCPv6 server would assign the 128-bit IPv6 address to the device in a manner similar to DHCPv4 operation. We'll discuss DHCPv6 in the next section.
- *Combination Stateless and Stateful* – involves a form of stateless address autoconfiguration used in conjunction with stateful configuration of additional IP parameters. This commonly entails autoconfiguring an IPv6 address statelessly but utilizing DHCPv6 to obtain additional parameters or options such as which DNS servers to query for name resolution on the given network.

The stateless or combination autoconfiguration of an IPv6 unicast address involves concatenating the address of the network to which the device is connected (where you are) and the device's interface ID (who you are). The link local network prefix is fixed according to the IPv6 addressing architecture, so once an interface ID is generated, the device can autoconfigure its link local address by appending the interface ID to the pre-defined FE80::/64 prefix. Leveraging our earlier example using a MAC address of AC-62-E8-49-5F-62, and resulting interface ID of AE62:E8FF:FE49:5F62, the link local address would be FE80::AE62:E8FF:FE49:5F62. However, due to the fact that interface IDs may not be globally unique, especially if not derived from a unique 48-bit MAC address, the device must next perform *duplicate address detection* prior to committing the new address. Hence, at this point, the address is considered *tentative*. Duplicate address detection is performed through a process called neighbor discovery, which entails the device sending an IPv6 Neighbor Solicitation packet to the IP address it just derived in order to identify a pre-existing occupant of the IP address. After a slight delay, the device also sends a Neighbor Solicitation packet to the solicited node multicast address associated with the tentative address.

If another device is already using the IP address, it will respond with a Neighbor Advertisement packet, and the autoconfiguration process will stop; i.e., manual intervention or configuration of the device to use an alternate interface ID is required. If a Neighbor Advertisement packet is not received, the device can assume uniqueness of the address and assign it to the corresponding interface. Participation in this process of Neighbor Solicitation and Advertisement is required not only for autoconfigured addresses but even for those statically defined or obtained through DHCPv6.

After the link local address has been assigned, the device attempts to configure a global address. In IPv6, routers advertise network prefixes and associated configuration parameters for locally

connected devices. A device may issue a Router Solicitation message to explicitly request a Router Advertisement message. These Router Advertisement messages can be used by the device to identify the global unicast network prefix that can be used for this subnet. The device appends its Interface ID to the network prefix and issues a Neighbor Solicitation packet to this tentative address. If no Neighbor Advertisement reply is received, the device can assign the address to its corresponding interface.

DHCPv6 Overview

DHCP for IPv6 addresses is referred to as DHCPv6 and is defined in RFC 3315[6]. According to this RFC, DHCPv6 is not integrated with DHCPv4. This means that DHCPv6 will support IPv6 addresses and configurations, but not additional IPv4 addresses and parameters. It's left to future development to define this should demand dictate.

DHCP Comparison: IPv4 vs. IPv6

DHCPv6 uses different message types and packet formatting but is similar in many ways to DHCPv4. The following table highlights these similarities and differences.

Table 1: DHCPv4 and DHCPv6 Comparison

Feature	DHCPv4	DHCPv6
Destination IP address of initial message	Broadcast (255.255.255.255)	Multicast to link-scoped address: All-DHCP-Agents address (FF02::1:2)
DHCP Relay Support	Yes – relay agents use pre-configured relay (DHCP server) addresses	Yes – relay agents use All_DHCP_Servers site-scoped multicast address (FF05::1:3)
Relay Agent forwarding	Same message type code but the relay agent inserts the *giaddr* field into the DHCP packet and unicasts to DHCP server(s)	The relay agent encapsulates the client message within a RELAY-FORW message to the DHCPv6 server(s) and receives a RELAY-REPL message from the server(s)
Message to locate server to obtain IP address and configuration	DHCPDISCOVER	SOLICIT
Server message to engage client	DHCPOFFER	ADVERTISE
Client message to accept parameters	DHCPREQUEST	REQUEST
Server acknowledgement of lease binding	DHCPACK	REPLY
Client message to leasing DHCP server to extend lease	DHCPREQUEST (unicast)	RENEW
Client message to any DHCP server to extend lease	DHCPREQUEST (broadcast)	REBIND
Client message to relinquish a lease	DHCPRELEASE	RELEASE
Client message to indicate that an offered IP address is already in use	DHCPDECLINE	DECLINE

[6] The *v6* suffix on *DHCPv6* refers not to the sixth version of DHCP, but the version of DHCP compatible with IPv6. This provides a convenient association to the address type served by DHCP and is now also used to suffix IPv4 DHCP as *DHCPv4*.

Feature	DHCPv4	DHCPv6
Server message to instruct client to obtain a new configuration	DHCPFORCERENEW	RECONFIGURE
Request IP configuration only, not address	DHCPINFORM	INFORMATION-REQUEST

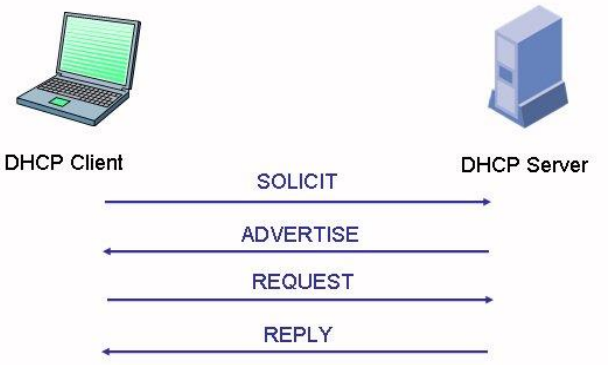
DHCPv6 Address Assignment

The DHCPv6 process begins with a client issuing a SOLICIT message, in essence requesting a "bid" from DHCP servers that can provide an IP address on the particular subnet to which the client is connected. Instead of broadcasting this initial packet as in IPv4, the SOLICIT message is sent by the client to the All_Relay_Agents_and_Servers multicast address, FF02::1:2. Any routers configured as relay agents will receive the SOLICIT packet, encapsulate it within a RELAY-FORW packet and forward it to the site-scoped All_DHCP_Servers multicast address, FF05::1:3.

DHCPv6 servers on this subnet will receive the SOLICIT packet directly, and others responding to the site-scoped All_DHCP_Servers multicast address will receive the SOLICIT packet encapsulated within a RELAY-FORW packet. In either case, the DHCPv6 server will respond with an ADVERTISE packet, indicating a preference value. The preference value is intended to enable the client to select the server advertising the highest preference as configured by administrators. The server will also indicate if it has no addresses available on the subnet. The ADVERTISE packet will be unicast to the client if the SOLICIT had been received directly using the client's source IP address from the SOLICIT packet (most likely the client's link local address). If the SOLICIT had been received by the server via a RELAY-FORW packet from a relay agent, the ADVERTISE message will be encapsulated in a RELAYREPL packet and unicast to the corresponding relay agent.

The client analyzes the advertisements received, selects a server from which to request an IP address, preferably with the highest preference, and issues a REQUEST unicast message to the server as illustrated in Figure 6. The server will then record the address assignment and reply to the client with a REPLY message.

Figure 6: DHCPv6 Address Assignment

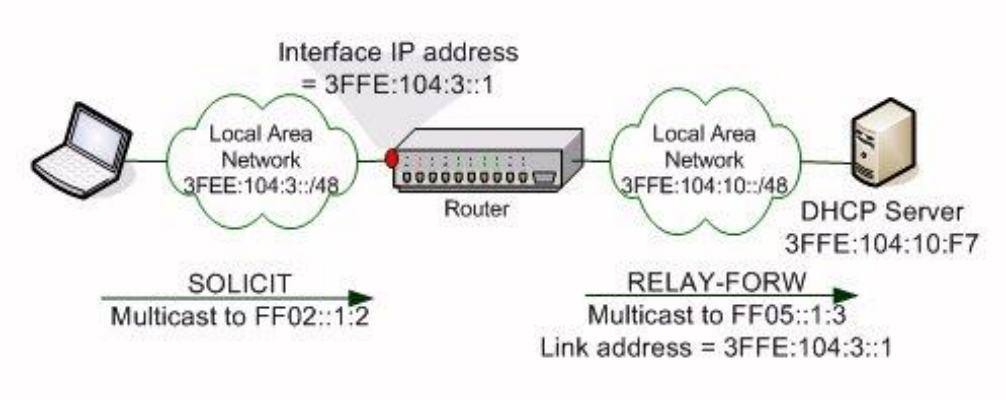


As in the IPv4 case, if the DHCP server resides on the same subnet as the DHCP client, the server can easily identify from which address pool to assign an address to the client. If the DHCP server

is deployed remotely and reachable via a relay agent, the relay agent must forward the message to the DHCP server. This process works as follows.

- The client multicasts the Solicit packet to the All-DHCP-Agents address, FF02::1:2.
- All local (on the same link) DHCP relay agents and servers should receive the Solicit packet.
- IPv6 relay agents do not require configuration of DHCP Relay addresses as in the IPv4 case, though they may enable such configuration. Instead, relay agents encapsulate the original Solicit packet within a Relay-Forw packet, which is then multicast to the site-scoped All-DHCP-Servers multicast address (FF05::1:3). The Link Address field of the Relay-Forw packet indicates the link on which the client requesting an IP address currently resides.
- This information is used by the DHCP server in assigning an appropriate IP address for this link, in a manner similar to the DHCPv4 GIAddr field. This process is illustrated in Figure 7.

Figure 7: DHCPv6 Relay



After the DHCPv6 process ends with the client receiving a Reply packet to confirm the address assignment, the client should perform duplicate address detection, just to ensure no other device is already using the IP address due to autoconfiguration or manual configuration. If another device is detected using the assigned IP address, the client would send a Decline message to the DHCPv6 server, indicating that the address is in use. The client can then reinitiate the DHCPv6 process to obtain a different IP address.

In addition to the four-packet exchange outlined above, DHCPv6 features a *rapid commit* option. This halves the messaging requirements by enabling the server to simply REPLY to a SOLICIT packet. The client requests rapid commit via an option setting in its SOLICIT message. Servers responding with an address assignment would issue a REPLY packet directly, also including the rapid commit option. Note that each server responding will assume the address it assigned is leased, so rapid commit should be used with either short lease times or for support by a limited number of servers serving the same subnet.

DHCPv6 Prefix Delegation

DHCPv6 is used not only to provide individual host IP addresses and/or associated IP configuration information to hosts, but can also be used to delegate entire networks to requesting router devices. This form of delegation via DHCPv6 is called *prefix delegation*. The original motivation for prefix delegation arose from broadband service providers seeking to automate the process of delegating IPv6 subnets (e.g., /48 to /64 networks) to broadband subscribers in a hierarchical, aggregate manner. A [requesting] router device at the edge of the service provider network, facing subscribers, would issue a request for address space via the DHCPv6 protocol to a delegating router. Note the terminology: this is intended to be an inter-router protocol though a DHCPv6 server could perform the functions of the delegating router.

The prefix delegation process utilizes the same basic DHCPv6 message flow described above for address assignment per Figure 6: Solicit, Advertisement, Request and Reply. Additional information within the corresponding DHCPv6 messages can be used to determine an appropriate network for delegation.

DHCPv6 Support of Address Autoconfiguration

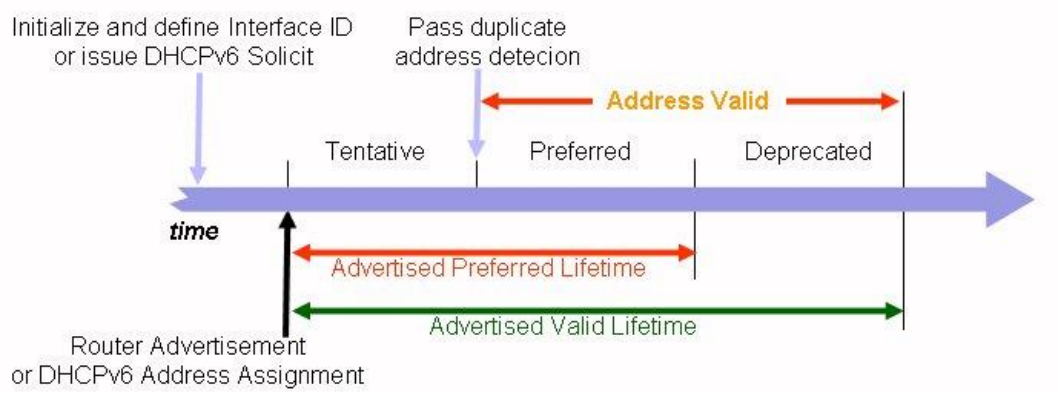
When we discussed IPv6 autoconfiguration earlier, we defined three types of autoconfiguration: stateless, stateful and combination stateless and stateful.

This third form of autoconfiguration leverages DHCPv6 not for IPv6 address assignment, but for assignment of additional parameters, encoded as DHCPv6 options. The client can request configuration parameters via the Information-Request message, indicating which option parameter values it is seeking. One or more servers configured to supply the desired configuration parameters will respond with a Reply message including the corresponding option parameters.

IPv6 Address Lifetimes

Each IPv6 address has a lifetime during which it is valid. In some cases the lifetime is indefinite, however, the lifetime concept always applies to DHCPv6 leased addresses and prefixes, as well as to autoconfigured addresses. Address lifetimes provide a useful mechanism to ease the process of network renumbering should it become necessary. Routers are configured with and advertise a *preferred* lifetime and a *valid* lifetime value for each network prefix in the Router Advertisement messages (Figure 6). DHCPv6 servers are likewise configured with preferred and valid address lifetimes for client addresses or prefixes. Addresses obtained via DHCPv6 or autoconfigured that have successfully proven unique through the duplicate address detection process described above can be considered either *preferred* or *deprecated*. In either state, the address is valid, but this provides a means for upper layer protocols (e.g., TCP, UDP) to select an IP address that will not likely change during the session.

Figure 6: IPv6 Address Lifetime



A device refreshes the preferred and valid time with each Router Advertisement message or by sending DHCPv6 RENEW messages in accordance with the values advertised. When time expires on a preferred prefix, the associated address(es) will become deprecated, though still valid. Thus, the deprecated state provides a transition period during which the address is still functional but should not be used to initiate new communications. Once the valid lifetime of the address expires, the address is no longer valid for use. Should a subnet be reassigned a different network prefix, the router will advertise the new prefix, and devices on the network would undergo the autoconfiguration process using the new prefix. Likewise, the DHCPv6 server can be configured with new address pools to reflect the renumbering plan.

DNS Support of IPv6

The Berkeley Internet Name Domain (BIND) distribution version 9 (BIND9), supports IPv6 resource records and runs on servers configured with IPv6 communications capabilities such as Solaris, Linux, Microsoft and others. DNS administrators can define AAAA resource records to resolve hostnames to IPv6 addresses, and PTR resource records to resolve IPv6 addresses to hostnames. Thus end users attempting to access an IPv6-addressed device will query DNS for a AAAA record, which will return the device's IPv6 address. The following example contrasts an IPv4 A record with a AAAA record.

```
v4-host.example.com.  86400   IN  A       10.200.0.16

v6-host.example.com.  86400   IN  AAAA    4FFE:320:200::A
```

Resolution of IP addresses to host names can also be configured in DNS to enable lookup of a valid hostname given an IP address. The example below contrasts IPv4 and IPv6 PTR records corresponding to the A and AAAA records above:

```
16.0.200.10.in-addr.arpa.      86400   IN   PTR     v4-host.example.com.

A.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.0.2.3.0.E.F.F.4.ip6.arpa.
86400   IN   PTR   v6-host.example.com.
```

While abbreviated IPv6 address notation is permitted for AAAA records, it cannot be specified unambiguously for IPv6 PTR records. This is due to the fact that only a 32-hex character string, dot-separated and reversed, represents a complete IPv6 address. Notation with fewer than 32-hex characters in the `ip6.arpa.` domain reflects a subdomain. For example, when an RIR allocates our 4FFE:0320::/32 example IPv6 network to a user, the user can generally sub-allocate and assign the address space as desired within the rules of the RIR. The user also takes on responsibility for the associated resolution of addresses within this space to subdomains or hosts. Thus the user would typically operate two or more DNS servers with authoritative data for the `0.2.3.0.E.F.F.4.ip6.arpa.` domain and may delegate subdomains of this domain according to address sub-allocations.

IPv6 Address Management Challenges

While IPv6 is positioned as an evolution of IPv4 by providing similar network layer functionality within a protocol stack, IPv6 has many unique features, most notably its vastly larger address space and address structure. As IP addresses are among the set of critical pre-requisites for communicating on an IP network, they must be properly managed. Proper management of IPv6 addresses can be accomplished with the use of an IP address management system that can address the following IPv6 address management challenges:

- IPv6 address blocks must be hierarchically allocated as IPv6 is hierarchical by design. The challenge is not only to accurately allocate address space and configure associated routers with proper network notation according to the allocation, but to do so hierarchically.
- Flexible address allocation strategies may be required to enable best-fit allocations within enterprise networks and sparse allocations among service provider networks.
- Support of prefix delegation and its proper configuration in routers and/or DHCPv6 servers.
- Support of address autoconfiguration and configuration of network prefixes and associated parameters for advertisement.
- Accurately inventorying IPv6 addresses assigned statically by DHCPv6, or autoconfigured.
- Configuring DHCPv6 servers with address pools and associated configuration options including client classes.
- Configuring DNS servers with AAAA resource records for resolution of IPv6 hosts, as well as maintenance of the reverse domain tree (ip6.arpa.) and PTR resource records.
- Planning and management of a transition from IPv4 to IPv6. This is a major topic in and of itself, so please see the companion white paper, *IPv4-IPv6 Transition and Co-Existence Strategies*.

The IPControl™ system from BT Diamond IP can help you manage your IPv4 and IPv6 address space, as well as associated DHCP and DNS services. Contact us to learn more about how IPControl can help you plan and manage IPv6 address space in your network, while effectively managing your IPv4 address space.

Conclusion

With the continued proliferation of IP devices, the existing IPv4 address space is becoming an increasingly scarce resource. More organizations are considering deployment of IPv6 to address this issue and to leverage other benefits of IPv6. Managing the IPv4/IPv6 address space resource requires an advanced tool that intelligently allocates, assigns and tracks the address blocks and individual IP addresses deployed on modern IP networks. Utilizing a tool like IPControl which provides a logical, intuitive user interface, with reports and alerts on the current and projected address utilization, enterprises and service providers can quickly and cost-effectively plan and implement their deployment of IPv6 address space, while effectively managing the ever-changing IP address needs that they face every day. IPControl is a highly scalable tool for managing both IPv4 and IPv6 address space and associated DHCP and DNS configurations, providing a solid management foundation for IPv4 networks, as well as mixed IPv4-IPv6 and IPv6-only networks.

About BT Diamond IP

BT Diamond IP is a leading provider of software and appliance products and services 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 and Sapphire Appliances for DNS/DHCP services deployment. 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 more information, please contact us directly at +1-610-321-9000 worldwide, email to btdiamondip-sales@bt.com or consult www.diamondipam.com.

IPControl and ImageControl are trademarks of BT Americas, Inc.