CYGNA LABS

IPv6 Addressing and Management Challenges

By Timothy Rooney

About Cygna Labs

Cygna Labs is a software developer and one of the top three global DDI vendors. Many Fortune 100 customers rely on Cygna Labs' DDI products and services, in addition to its industry-leading security and compliance solutions to detect and proactively mitigate data security threats, affordably pass compliance audits, and increase the productivity of their IT departments. For more information, visit https://cygnalabs.com.

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 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 several 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 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, global IPv4 address space availability has been exhausted for all intents and purposes. Organizations requiring new or additional address space these days are only able to obtain IPv6 space, without paying exorbitant fees for sparse IPv4 space, or using a service provider which shares IPv4 space behind carrier grade NATs. 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.

Google has been measuring the proportion of IPv6 traffic reaching its sites for years now . As of this writing, about 40% of Internet users accessing google sites are using IPv6. If you're not supporting IPv6 access to your Internet resources, a large proportion of potential visitors could be prohibited from viewing your content or suffering performance degradations if falling back to IPv4.

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.

[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.

[2] Statistics, IPv6 Adoption, Google, https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption, Accessed April 25, 2022.

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 232 addresses or 4.2 billion addresses. A 128-bit address field provides 2128 addresses or 340 trillion trillion trillion addresses or 340 undecillion [3] (3.4 x 1038). To put some context around this tremendously large number, consider that this quantity of IP addresses:

- Equals more than 4.5 X 1028 IP addresses per person on Earth
- Equals more than 4.3 X 1020 IP addresses per square inch of the Earth's surface
- Equals nearly 1.4 X 1013 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 addressing of DNS servers has proven successful in providing closest routing to the intended service.

[3] Using the American definition of undecillion of 1036, not the British definition which is 1066.

- *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 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 . 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

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.

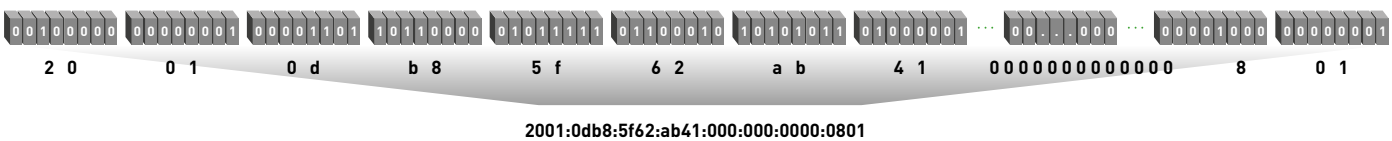


Figure 1: IPv6 address notation

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 hextet, i.e., between colons, may be dropped. Thus, the address above could be abbreviated: 2001:db8:5f62: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 zero hextets. Using this form of abbreviation, the address can be further abbreviated as 2001:db8:5f62:ab41::801. Note that only one double colon may be used within an address representation. Since there are always eight hextets 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: 2001:db8:0:56fa:0:0:0:b5. We can abbreviate this address as either: 2001:db8::56fa:0:0:0:b5 or 2001:db8:0:56fa::b5. We can easily calculate that the double colon denotes one hextet (8 total minus 7 hextets shown) in the first case and three (8 minus 5 shown) in the second notation. If we attempted to abbreviate this address as 2001:db8::56fa::b5, we could not unambiguously decode this, as it could represent any of the following possible addresses:

2001:db8:0:56fa:0:0:0:b5

2001:db8:0:0:56fa:0:0:b5

2001:db8:0:0:0:56fa:0:b5

Thus, the requirement holds that only one double colon may appear in an IPv6 address.

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

Address Structure

As with IPv4, the IPv6 address is composed of a network and a host portion; however, experience with VLSM and subnetting in general has led to addition of a subnet field The IPv6 address is generally divided into three fields shown in Figure 2.

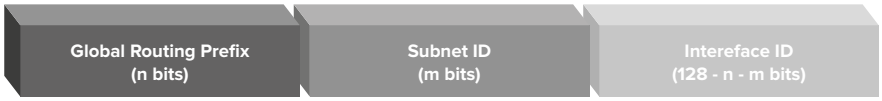
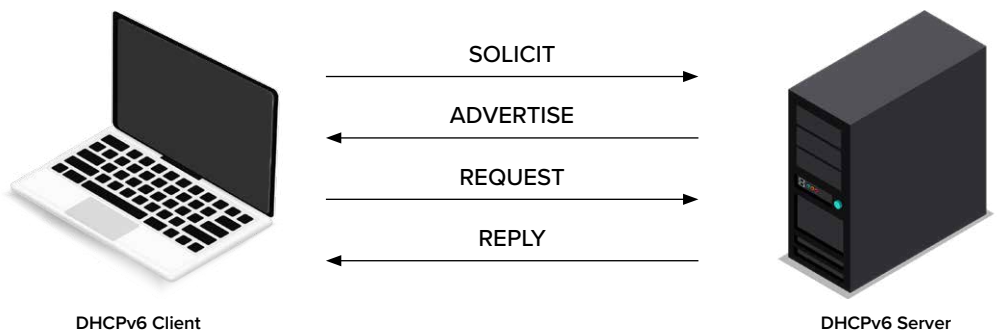


Figure 2: IPv6 address structure

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. Assuming our example IPv6 address, 2001:db8:5f62:ab41::801, resides within a /48 global routing prefix, this prefix address would be denoted as 2001:db8:5f62::/48. The network address is denoted with zero-valued bits beyond the prefix length (bits 49-128 in this case) as denoted by the terminating double colon. 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 216 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 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, though not without privacy concerns. 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 hextet 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 documentation IPv6 network 2001:db8::/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 2001:db8::/32, represented in binary as:

0010 0000 0000 0001 : 0000 1101 1011 1000: 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, 2001:db8::/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.

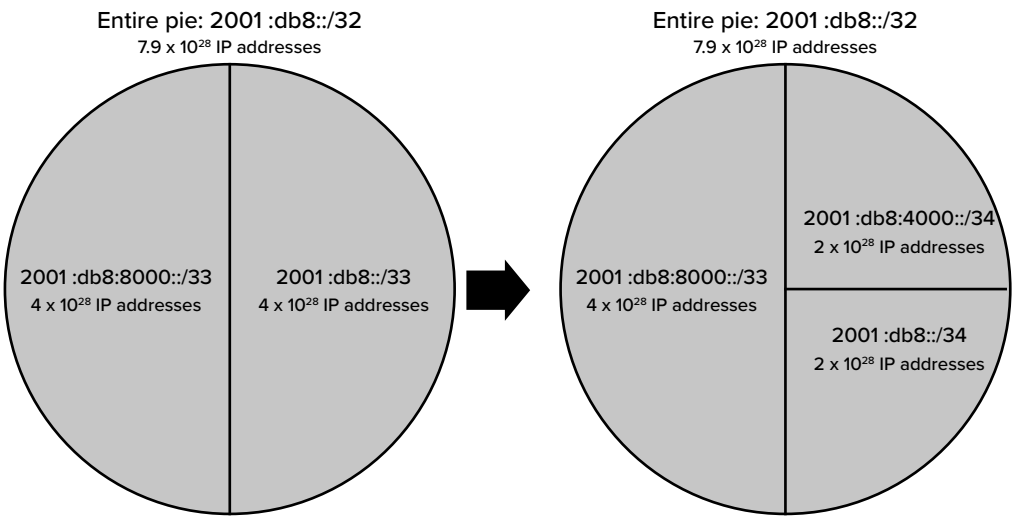


Figure 2: IPv6 address structure

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 six /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 2001:db8::/32

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

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.

0010 0000 0000 0001 : 0000 1101 1011 1000 : 1000 0000 0000 0000 : 0000... 2001:db8:8000::/33

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0100 0000 0000 0000 : 0000... 2001:db8:4000::/34

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0010 0000 0000 0000 : 0000... 2001:db8:2000::/35

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0001 0000 0000 0000 : 0000... 2001:db8:1000::/36

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 1000 0000 0000 : 0000... 2001:db8:800::/37

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0100 0000 0000 : 0000... 2001:db8:400::/38

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0010 0000 0000 : 0000... 2001:db8:200::/39

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0001 0000 0000 : 0000... 2001:db8:100::/40

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0000 0000 0000 : 0000... 2001:db8::/40

Through this successive halving of address space, we readily have two /40 networks available (highlighted above): 2001:db8::/40 and 2001:db8:100::/40. Next, using the best fit approach, we can take the next smallest available network, in this case 2001:db8:200::/39, shown right above the two highlighted /40 networks just allocated, and split it into two /40s:

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0010 0000 0000 : 0000... 2001:db8:200::/40

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0011 0000 0000 : 0000... 2001:db8:300::/40

This yields 2001:db8:200::/40 and 2001db8:300::/40. We need two more /40s, so our next smallest unallocated block is now 2001:db8:400::/38, two networks above the shaded blocks in the hierarchy above. To derive two /40s from this /38, we split this into two /39s, then split one /39 to arrive at 2001:db8:400::/40 and 2001:db8:500::/40, leaving 2001:db8:600::/39 unallocated. Figure 6.3 illustrates this successive halving in a pie chart form.

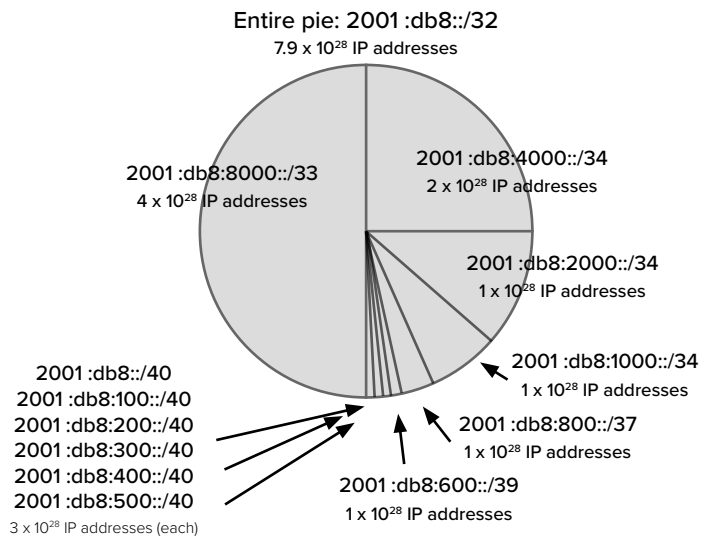


Figure 4: Best fit allocation of six /40s from a /32

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:

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0000 0000 0000 : 0000... 2001:db8::/40

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0001 0000 0000 : 0000... 2001:db8:100::/40

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0010 0000 0000 : 0000... 2001:db8:200::/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 some 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, using a fixed length subnet ID field adds uniformity and simplifies tracking, so the sparse algorithm may make sense at the top level of your tiered allocation strategy. 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 contiguous 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 2001:db8::/32 space would look like:

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0000 0000 0000 0000 : 0000... 2001:db8::/40

0010 0000 0000 0001 : 0000 1101 1011 1000 : 1000 0000 0000 0000 : 0000... 2001:db8:8000::/40

0010 0000 0000 0001 : 0000 1101 1011 1000 : 0100 0000 0000 0000 : 0000... 2001:db8:4000::/40

These translate as 2001:db8::/40, 2001:db8:8000::/40 and 2001:db8:4000::/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.

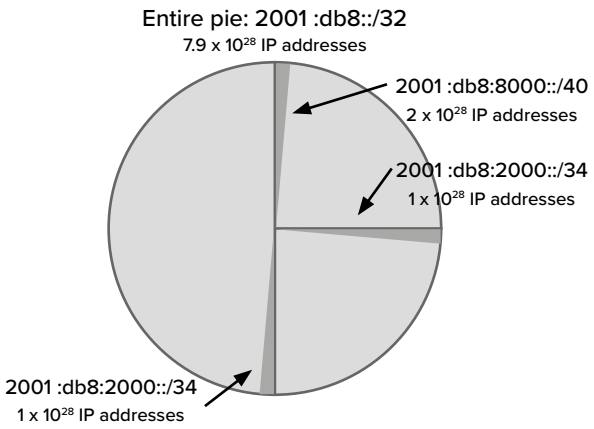


Figure 5: Sparse Allocation Example

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 28 – 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.

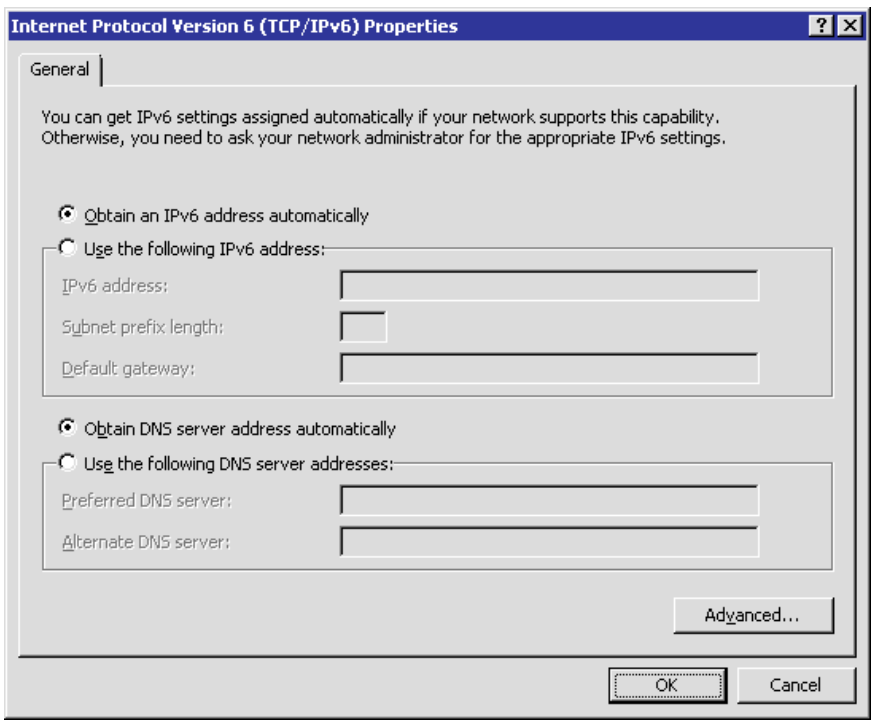


Figure 6: Example Microsoft Windows IPv6 Configuration

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. Three basic forms of IPv6 address autoconfiguration are defined:

- **Stateless** – This process is "stateless" in that it is 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. This form is abbreviated as SLAAC (stateless address autoconfiguration).
- **Stateful** – The stateful process relies solely an external address assignment mechanism such as DHCPv6.

- **Combination Stateless and Stateful** – This process commonly entails a device autoconfiguring an IPv6 address using the stateless method, then utilizing DHCPv6 to obtain additional parameters or options such as which Network Time Protocol (NTP) servers to query for time resolution on the given network.

At the most basic level, the autoconfiguration of an IPv6 unicast address was intent on concatenating the address of the network to which the device is connected (where you are) and the device's interface ID (who you are). But as we touched on earlier, the latter portion introduces glaring privacy concerns. Internet conglomerates such as content delivery networks (CDNs) and popular social media websites with access to large quantities of user IP traffic could easily track the source locations and destination endpoints for individuals' Internet sessions. Privacy extensions for the derivation of the Interface ID have been introduced to assuage such privacy concerns.

Neighbor Discovery

The process of neighbor discovery in IPv6 enables a node to discover the IPv6 subnet address on which it is connected. Neighbor discovery in general also enables identification of other IPv6 nodes on the subnet, to identify their link layer addresses, to discover routers serving the subnet and to perform duplicate address detection. Discovery of routers enables IPv6 nodes to automatically identify routers on the subnet, negating the need to configure a default gateway manually within the device's IP configuration. This discovery process enables a device to identify the network prefix(es) and corresponding prefix length(s) assigned to the link.

Neighbor discovery entails each router periodically sending advertisements on each of its configured subnets 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, as well as other configuration parameters.

The router advertisement also indicates whether a DHCPv6 server is available for address assignment or other configuration. The M bit (Managed address configuration flag) in the router advertisement indicates that DHCPv6 services are available for address and configuration settings. The O bit (other configuration flag) indicates that configuration parameters other than the IP address are available via DHCPv6; such information may include which DNS servers to query for devices on this link. Nodes can also solicit router advertisements using Router Solicitation messages, addressed to the link local routers multicast address (ff02::2). The following table summarizes the interpretation of these flags.

Flag	O = 0	O = 1
M=0	No DHCPv6	DHCPv6 for configuration only
M=0	DHCPv6 for address and configuration	DHCPv6 for address and configuration

Interface Identifiers

Initial IPv6 standards called for the derivation of a device's interface ID based on the device's MAC address, namely via the modified EUI-64 algorithm. However, privacy concerns arose given the ease with which this static Interface ID may be tracked as we'll discuss in this section.

Modified EUI-64 Interface Identifiers

Once a node identifies the subnet to which it is attached, it may complete the SLAAC process by formulating its Interface ID. The IPv6 addressing architecture stipulates that all unicast IPv6 addresses, other than those beginning with binary [000] must use a 64-bit Interface ID. Interface IDs derived from the device's MAC address must be generated using the modified EUI-64 algorithm. The "unmodified" extended unique identifier-64 algorithm entails concatenating the 24-bit company identifier issued by the IEEE to each network interface hardware manufacturer (e.g., the initial 24 bits of an Ethernet MAC 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 into a modified EUI-64 identifier calls for 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 where a MAC address of AC-62-E8-49-5F-62 yields an interface ID of ae62:e8ff:fe49:5f62.

Interface Identifiers

Initial IPv6 standards called for the derivation of a device's interface ID based on the device's MAC address, namely via the modified EUI-64 algorithm. However, privacy concerns arose given the ease with which this static Interface ID may be tracked as we'll discuss in this section.

Modified EUI-64 Interface Identifiers

Once a node identifies the subnet to which it is attached, it may complete the SLAAC process by formulating its Interface ID. The IPv6 addressing architecture stipulates that all unicast IPv6 addresses, other than those beginning with binary [000] must use a 64-bit Interface ID. Interface IDs derived from the device's MAC address must be generated using the modified EUI-64 algorithm. The "unmodified" extended unique identifier-64 algorithm entails concatenating the 24-bit company identifier issued by the IEEE to each network interface hardware manufacturer (e.g., the initial 24 bits of an Ethernet MAC 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 into a modified EUI-64 identifier calls for 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 where a MAC address of AC-62-E8-49-5F-62 yields an interface ID of ae62:e8ff:fe49:5f62.

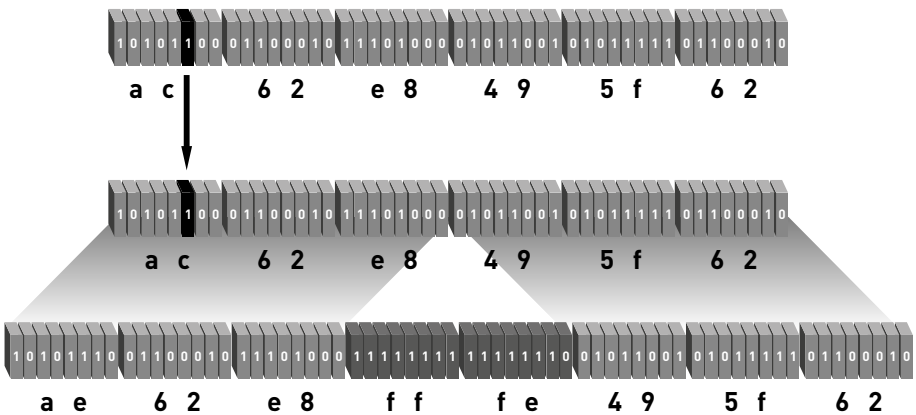


Figure 5: Example Modified EUI-64 IID Derivation

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.

Opaque Interface IDs

As mentioned above, use of an Interface ID derived from a device's MAC address raises several privacy and security concerns:

- **Location tracking** — IPv6 addresses containing a deterministic (or static in general) interface ID could allow the tracking of the user's location based on the network prefixes to which the user's Interface ID is associated
- **Activity tracking** — User activity may likewise be tracked by analyzing destinations accessed by the corresponding source IPv6 address containing the user's Interface ID.
- **Device targeting** — Decoding of the interface ID may allow an attacker to identify the manufacturer of the device's interface and thereby enable exploitation of known vendor vulnerabilities.
- **Address reconnaissance** — Determination of MAC vendors used within a network could aid attackers in reducing the search space from an entire /64 (1.8 X 1019 addresses) for a subnet to those containing portions of the Interface ID associated with known MAC vendors when identifying possible attack targets

Several alternative approaches to the modified EUI-64 algorithm have been devised to address these concerns, including the following.

- Cryptographically Generated Addresses (CGA) utilize an Interface ID derived as a one-way hash of a public key and auxiliary parameters, cryptographically binding the public key to the corresponding IPv6 address.
- Temporary (privacy) addresses are supplemental addresses periodically generated using a random interface ID. These are supplemental in the sense that these addresses are used only on outbound connections in order to reduce device identity exposure, so another stable form of interface ID derivation is also required. Temporary addresses have been found by some network managers as exacerbating the challenges of defining access control, auditing addresses, and troubleshooting network problems.
- Constant Interface IDs are not based on the modified EUI-64 algorithm but on some other form of pseudo random generation but is appended to its relevant prefix.
- Stable (non-temporary) semantically opaque addresses are based on a pseudo-random Interface ID that is the same every time the device connects to a given subnet, but is different for each subnet visited. In addition, they are not correlated with the device's MAC address. This method is the IETF recommended approach to Interface ID generation for SLAAC instead of the modified EUI-64 method.

The following table offers a comparison of the resilience of each of these address assignment schemes against the major privacy and security concerns outlined at the beginning of this section.

Mechanism	Vulnerability based on the IPv6 address*			
	Location Tracking	Activity Tracking	Device targeting	Address reconnaissance
Modified EUI-64	During device lifetime	During device lifetime	Possible	Possible
Static (manual)	During address lifetime	During address lifetime	Depends on address derivation method	Depends on address derivation method
CGA	Not vulnerable	During address lifetime, e.g., until regenerated	Not vulnerable	Not vulnerable
Temporary	Not vulnerable	During temporary address lifetime	Not vulnerable	Not vulnerable
Constant	During address lifetime	During address lifetime	Not vulnerable	Not vulnerable
Stable	Not vulnerable	During use within a given subnet	Not vulnerable	Not vulnerable
DHCPv6	Not vulnerable	During address (lease) lifetime	Not vulnerable	Depends on address assignment method

* A device may be exposed to these vulnerabilities based on aspects other than its IPv6 address.

Methods which utilize an unchanging Interface ID are more susceptible to location tracking given the deterministic association of the Interface ID with the user/device. Activity tracking of a session between two IP addresses is certainly possible while both addresses are in use, e.g., during the user/device address lifetime. And any method that is derived from a device's MAC address opens the door to device targeting as well as address reconnaissance. DHCPv6 assignment methods could also simplify address reconnaissance attempts if addresses are assigned monotonically if not otherwise deterministically.

Reserved Interface IDs

Upon derivation of an interface ID using any of the above algorithms, a node must confirm that it does not overlap with the set of Interface IDs reserved by IANA:

Reserved Interface Identifier Range	Reserved for
0000:0000:0000:0000	Subnet-Router anycast address
fdff:ffff:ffff:ff80 – fdff:ffff:ffff:ffff	Reserved subnet anycast addresses

The Interface ID may not be unique, especially if not derived from a unique 48-bit MAC address. Thus, the device must also perform duplicate address detection (DAD) prior to committing the new address. Prior to completing the DAD process, the address is considered tentative.

Duplicate Address Detection (DAD)

DAD is performed using the neighbor discovery process, which entails the device sending an IPv6 Neighbor Solicitation packet to the IPv6 address it just derived (or obtained from DHCPv6) 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 this address as well.

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 may be required to assign an alternate interface ID. 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.

IPv6 addresses have a lifetime during which they are valid as illustrated in Figure 6. In some cases, the lifetime is infinite, but the concept of address lifetime applies to both DHCPv6 leased addresses as well as autoconfigured addresses. This is useful in easing the process of network renumbering. Routers are configured with and advertise a preferred lifetime and a valid lifetime value for each network prefix in their Router Advertisement messages. IP addresses 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 differentiation provides a means for upper layer protocols (e.g., TCP, UDP) to select an IP address that will likely not change during the ensuing session.

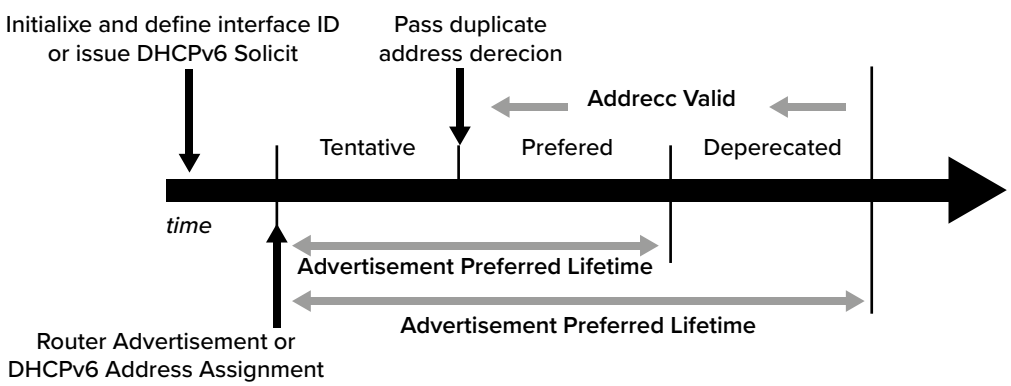


Figure 6: IPv6 Address Lifetimes

A device refreshes the preferred and valid lifetimes with each Router Advertisement message or lease renewal. 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. For example, to renumber a subnet, the router can be configured to advertise the new prefix, and devices on the network would undergo the autoconfiguration process using the new prefix as the lifetime of the old prefix expires.

DHCPv6 Overview

DHCP for IPv6 addresses is referred to as DHCPv6 and is defined in RFC 8415 . 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

[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
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
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.

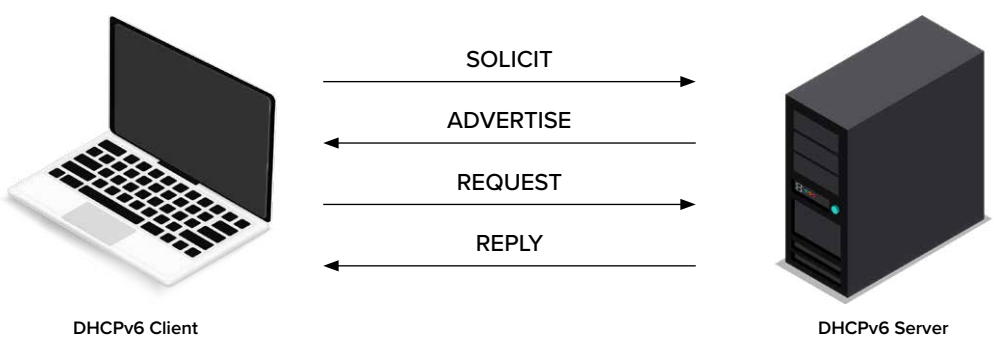


Figure 6: DHCPv6 Address Assignment

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.

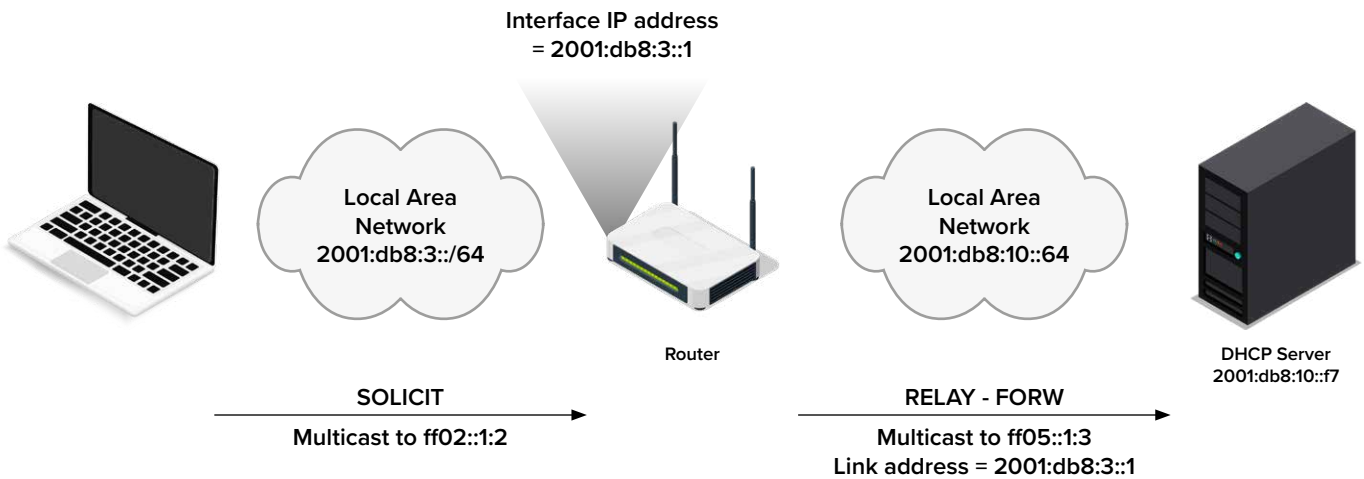


Figure 7: DHCPv6 Relay

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.

Device Unique Identifiers (DUIDs)

Like DHCPv4, DHCPv6 servers must track the availability and assignment of IP addresses within its configured address pools, and identify requestors and holders of IP addresses. DHCPv6 utilizes the Device Unique Identifier, or DUID to identify clients. DUIDs are used not only for servers to identify clients, but for clients to identify servers. The DUID is analogous to the client-identifier concept in that DUIDs are intended to be globally unique for a device, not an interface. DUIDs should not change over time, even if the device undergoes changes in network interface hardware. DUIDs are constructed in various manners automatically by IPv6 nodes. They consist of a two-octet type code followed by a variable number of octets based on the type. The following DUID type codes are currently defined:

- Type = 1 - Link layer address plus Time (DUID-LLT)
- Type = 2 - Vendor-assigned unique ID based on Enterprise Number (DUID-EN)
- Type = 3 - Link-layer based DUID (DUID-LL)
- Type = 4 – Universally Unique Identifier (DUID-UUID)

For those based on link layer address, they are to be used for all device interfaces, even if the hardware from which the link layer address was obtained is removed. The DUID is a device identifier, not an interface identifier.

Identity Associations (IAs)

While DUIDs are associated with all interfaces of a device and IP addresses are assigned to interfaces, you may be wondering how the device and server identify particular interfaces for a given DUID. The concept of the Identity Association (IA) provides this linkage between a DHCPv6 server and a client interface for individual address or prefix assignment. IAs are differentiated by type between those for temporary addresses (IA_TA), which are short-leased, non-renewable addresses, those for non-temporary addresses (IA_NA), and those for prefix delegation (IA_PD).

Temporary address assignments assuage privacy concerns associated with auto-configured addresses based on hardware addresses (i.e., modified EUI-64 interface IDs), which do not change over time. The concern is that a given Interface ID within an IPv6 address does not change unless the underlying hardware interface changes. Thus, even if the network upon which a device is connected changes from day-to-day, the interface ID does not. The ability to track the location of a device and thus its user, becomes relatively easy; hence the concern with privacy. The use of short-lived, non-renewable address assignments via DHCP can be one approach to address this concern, hence the concept of temporary addresses. We'll discuss this privacy issue in detail later in the chapter.

For individual address assignment, temporary or non-temporary, each client interface has an IA, identified by an IA Identifier, or IAID. The IAID is represented as four octets in client-server DHCPv6 communications and is chosen by the client. The IAID must be unique among all IAIDs associated with the client and must be stored persistently across client reboots or consistently derivable upon each reboot. The client specifies its DUID and IAID for which an address is being requested from the DHCPv6 server. The DHCPv6 server assigns an IPv6 address to the IAID, along with the corresponding T1 (renew) and T2 (reboot) timer values.

IA_PDs are not necessarily associated with a device interface. Recall that the requesting router is using DHCPv6 to obtain an IPv6 network delegation. The requesting router must derive one or more IA_PDs for use within DHCPv6, and it must be persistent across reboots or consistently derivable.

DNS Support of IPv6

The Berkeley Internet Name Domain (BIND) distribution version 9 (BIND9), supports IPv6 resource records and runs on server operating systems that may be configured with IPv6 communications capabilities. 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 2001:db8:200::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.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.8.b.d.0.1.0.0.2.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 2001:db8::/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 8.b.d.0.1.0.0.2.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 to enable multiple allocation forms at different hierarchy levels. Allocating sparsely in the top tiers can reduce routing overhead, allocating as best firt within middle layers can optimize allocation and allocating randomly at the subnet level can help reduce the risk of trivial network scanning otherwise eased by monotonic allocations.
- 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 and DHCP options.
- 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 Cygna Labs Diamond IP can help you manage your IPv4 and IPv6 address space, as well as associated DHCP and DNS services. Only IPControl from Diamond IP enables you to manage IPv4 and IPv6 together, holistically, i.e., without flipping through different tabs on the user interface. IPControl also enables single click (or single API call) allocation of multiple blocks and/or subnets to facilitate new virtual private cloud (VPC) or branch office provisioning, including dual stack networks. IPControl also simplifies configuration of DHCPv6 for our Sapphire appliances as well as stock ISC DHCPD/Kea, Microsoft and Cisco Prime Network Registrar (PNR) implementations. DNS configuration is likewise streamlined with support for these multi-vendor implementation including such IPv4-IPv6 features as DNS64 and AAAA filtering. 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

Leveraging expertise gained through more than two decades of experience working with customers, industry analysts, and various software implementations, this white paper recommends numerous best practices for effectively managing your foundational IPAM services. Building on this experience and these recommendations, Cygna Labs Diamond IP has developed IPControl software and appliance products as well as a managed IPAM service to simplify the fulfillment of these recommendations.

Cygna Labs Diamond IP products and services offer an advanced, comprehensive IPAM solution you need to automate many tedious, error-prone, yet critical IPAM functions. Diamond IP provides unsurpassed extensibility and user-definability to enable you to manage your IP address space the way you want to manage it, all at an affordable price. Please email us at sales@cygnalabs.com to learn more about how IPControl can automate more of the IPAM functions you need at an exceptional return on investment (ROI).

Toll Free: **(844) 442-9462**
International: **+1 (305) 501-2430**
Fax: **+1 (305) 501-2370**

Sales: **sales@cygnalabs.com**
Support: **support@cygnalabs.com**
Billing: **finance@cygnalabs.com**

cygnalabs.com

CYGNA LABS