Autoconfiguration in IP Version 6

Thomas Narten
IBM Corporation
3039 Cornwallis Ave.
PO Box 12195 - BRQA/502
Research Triangle Park, NC 27709-2195
narten@raleigh.ibm.com

Abstract

This paper describes the autoconfiguration facilities in the Next-Generation Internet Protocol, IP Version 6 (IPv6). IPv6 Hosts can plug into the network and start communicating without requiring special configuration. This includes hosts connecting to isolated stand-alone networks (e.g., at a home) or those connected to a large corporate network. This paper describes how hosts generate addresses, find and keep track of routers, and how they dynamically switch to backup routers should the one they are using fail.

1 Introduction

The Internet Protocol (IP) is the basic building block on which all Internet protocols are built. Applications that we take for granted today, such as web browsers, e-mail, etc., all use TCP and UDP, which in turn are built on top of IP. Without IP, the Internet as we know it would not exist.

The current version of IP, IPv4, uses 32-bit addresses [Pos81]. In theory, 32 bits is enough to address over 4 billion nodes. In practice, the interaction between routing and addressing makes it impossible to assign addresses compactly enough to come anywhere near 100% utilization. Consequently, there is concern that the continuing exponential growth of the Internet will lead to an exhaustion of IPv4 addresses early in the next century.

IP Version 6 (IPv6) is a new version of IP [DH98]. IPv6 uses 128-bit addresses, addresses so large that it will be impossible to run out of addresses in the foreseeable future. The use of larger addresses, however, necessitated changing the basic format of IP packets. Once the packet format changed, it also became appropriate to consider other changes that improved known deficiencies in IPv4. One of the areas of improvement is autoconfiguration. New machines that plug into an IPv6 network for the first time autoconfigure a useable address and if one or more routers is present, find and use them as appropriate. Although IPv4 and DHCP might appear to give a similar capability, no DHCP server is needed in IPv6, and machines that attach to an isolated LAN (e.g., a home Ethernet) will be able to immediately communicate with each other with no manual configuration at all. Moreover, hosts automatically switch from one router to another when a connectivity failure occurs.

The rest of this paper reviews important features of IPv6 with an emphasis on the autoconfiguration features it provides. Section 2 provides background on why IPv6 is needed. Section 3 summarizes some of the major improvements of IPv6 over IPv4. Section 4 provides background on IPv6 needed to explain some of the details of autoconfiguration. Section 5 describes the neighbor discovery protocols in detail. Section 6 concludes with a summary.

2 Why IPv6

The core motivation for IPv6 is the eventual depletion of the IPv4 address space. In the early 1990s, the rate of address consumption appeared great enough that the pool of unused addresses would become exhausted early in the 21st century [GA92]. In response to that fear, however, address registries (e.g., RIPE, APNIC, etc.) and Internet Service Providers (ISPs) became stingy about giving out addresses. Obtaining a block of addresses now requires that a customer justify that all will be used essentially immediately (i.e., not a few years in the future). In addition, *Network Address Translation* (NAT) devices [EF94] make it possible to number a site's internal nodes using private (i.e., net 10) addresses [RMK$^+$96] and still be able to access the Internet. NAT has slowed the apparent consumption of IPv4 address space, but has not lessened the need for IPv6 (see side bar on NAT).

3 IPv6 Improvements

Switching to 16 byte addresses in IPv6 necessitates changing the format of IP packets. Because packet changes are non-backwards compatible, it makes sense to redesign the entire packet from scratch, taking into account what has been learned from IPv4's experience. The following summarizes some of IPv6's improvements:

- IPv6 has built-in support for security. All IPv6 nodes support cryptographic-based authentication and encryption. In IPv6, IPsec [KA98] can be assumed to be present on all nodes running IPv6. With IPsec, nodes have the ability to send and receive packets whose contents are encrypted, and have a strong degree of assurance that it knows the correct identity of the peers it communicates with.

- The IP header contains no checksum, reducing the amount of work a router must do to forward a packet. Reducing the per-packet processing overhead improves router performance and reduces their cost.

- Option support in IPv6 is more flexible and extensible than in IPv4. For example, options in IPv4 are limited to a total of 44 bytes in length; no such restriction exists in IPv6.

- IPv6 supports efficient renumbering of sites by explicitly supporting multiple addresses on an interface. In addition, all addresses are leased. When a lease expires, the address no longer is usable. A site can renumber itself by having a transition period in which both old addresses and new addresses work equally; once the transition to the new address is near completion, the old address expires.

- IPv6 supports plug-and-play "out of the box" operation. A new machine generates its own addresses automatically and can immediately communicate with other nodes. Autoconfiguration is targeted both at small sites (e.g., a home network without a router or DHCP server) and large sites (e.g., a corporate intranet). There are two flavors of address autoconfiguration. In stateless autoconfiguration, a node forms addresses by determining the subnet prefixes on the links to which it attaches and then forming an address on that subnet. Alternatively, using stateful autoconfiguration, a node can use DHCP [BP99] to obtain addresses and other configuration information.

It should be noted that some of the technologies originally motivated by IPv6 have also been retrofitted into IPv4 (e.g., IPsec). However, it is not possible to retrofit them all. As discussed earlier, IPv6's large globally-unique address cannot be provided by IPv4. Likewise, stateless address autoconfiguration (discussed in detail below) requires IPv6's large addresses and cannot easily be retrofitted into IPv4. Finally, the new features are required of all IPv6 implementations, while they are optional in IPv4.

4 IPv6 Background/Tutorial

Before presenting the details of autoconfiguration, we first review some of the important IPv6 concepts.

IPv6 is designed to run over virtually any media type. The two standard media types include broadcast media, which support the efficient delivery of a packet to all or a subset of nodes on a specific link, and point-to-point links. Links are also assumed to support bi-directional connectivity, in order to obtain the full benefits of autoconfiguration. This does not mean that IPv6 cannot run over other link types. *non-broadcast multiple access*(NBMA) links (e.g., ATM [ASJH99, ASJ99] and Frame Relay) are also supported though specialized servers may be needed.

IPv6 supports multiple address scopes [HD98]. *Global-scope addresses* are globally unique and can be used anywhere in the Internet. *Link-local addresses*, on the other hand, are unique only on a specific link (e.g., a LAN). Link-local addresses can be used for communication among nodes attached to the same link. They are used on isolated networks that do not have connections to the outside world (e.g., a home Ethernet) or during bootstrap operations as part of obtaining other addresses. IPv6 also has *site-local addresses*, which are unique only within a particular site or organization. Site-local addresses are analogous to IPv4's private addresses [RMK+96]. They can be used only within a site, and routers do not forward packets containing site-local addresses off of the site.

An IPv6 interface can have multiple addresses assigned to it. When a node first boots, it creates and assigns a link-local address to the interface. Using its link-local address, a node obtains additional addresses (either site-local, global or both). Having multiple addresses provides flexibility. For example, if a site is *multi-homed* to the Internet via two ISPs (i.e., it has two independent connections), it could obtain addresses from both, and each node in the site could have two addresses. Multiple addresses also support the situation where an entire site needs to be renumbered, for example, if it changes ISPs. IPv6's provides facilities that make it possible to start using a new address while simultaneously continuing to use an existing address. Once the new addresses are in place and working, the old addresses can be quietly phased out.

Unicast addresses are leased to a node. Every address has two lifetimes associated with it. A *valid lifetime* defines the period of time during which the address can be used. Once the valid lifetime expires, the address can no longer be used. Indeed, it may well be that packets sent to or from that address will no longer be forwarded properly. A *preferred lifetime* indicates the period of time during which the address can safely be used for all communication. The preferred lifetime must be shorter than the valid lifetime, and an address is said to be preferred if its preferred lifetime has not yet expired. An address whose preferred lifetime has expired, but that is still valid is called *deprecated.* Deprecated addresses can still be used for communication, but there use should be avoided if possible because they will become invalid soon. Applications opening new TCP connections, for instance, use a preferred address over one that is deprecated. Doing so ensures that few if any applications are using a deprecated address at the time the valid lifetime finally expires.

Having a valid and preferred lifetime facilitates renumbering by gracefully overlapping the phase out of an existing address with the introduction of a new one. When a new address is introduced, the preferred lifetime is set far into the future. In contrast, the preferred lifetime of the address being phased out is allowed to expire. For some time period (say several days or weeks), the old address is still usable, but because it is in a deprecated state, all new TCP connections will use the newer address. As existing applications using the old addresses terminate, the usage of the old address goes down. Finally, the old address is made invalid. At that point, only those (few) applications that make use of very long running connections are affected.

IPv6 does not support broadcasting; in cases where IPv4 uses broadcasting, IPv6 uses multicast instead. As with unicast addresses, multicast addresses have a number of scopes. Link-local multicast addresses can only be used on a single link, and routers do not forward them elsewhere. Site-local addresses are used for communication within a site and packets containing site-local addresses are not forwarded outside of the site. For global communication, global scope addresses are used. A number of multicast addresses have been assigned for specific purposes. For example, the (link-local) all-routers multicast address is used to send packets to the routers on a link, whereas the (link-local) all-nodes multicast address is used to send a packet to every node on a link.

5 Neighbor Discovery

5.1 Overview

The rest of this paper discusses the protocols known collectively as *neighbor discovery* (ND) [NNS98, TN98]. ND subsumes a number of IPv4 protocols and defines new functionality as well. ND performs three major functions. First, it provides address resolution services, updating and expanding IPv4's *address resolution protocol* (ARP) [Plu82]. Second, it allows hosts to discover what neighboring routers are present and provides a mechanism for obtaining configuration information from them. Finally, ND defines *neighbor unreachability detection* (NUD), a mechanism that determines when a neighbor becomes unreachable. If the neighbor is a router, the node invokes a recovery procedure to find an alternate router. ND is implemented within ICMP [CD98], making all services (including address resolution) independent of link technologies. In addition, the IPsec security services (e.g., cryptographic authentication) can be applied to ND if desired.

ND has two main sets of message pairs. *Neighbor solicitations* (NSs) and Neighbor Advertisements (NAs) are used to determine the link-layer addresses of neighbors, as well as to verify that a neighbor is reachable. *Router solicitation* (RS) and *router advertisement* (RA) messages are used to locate and obtain information from routers.

When a node doesn't know the link-layer address corresponding to the IP address of a neighbor, it performs address resolution by sending out a multicast NS message. The neighbor with the requested *target* IP address responds with an NA containing its link-layer address. In IPv4, address resolution is performed by broadcasting an ARP request to all nodes. In contrast, NS messages are sent to a *solicited-node* multicast address, an

Solicited-Node Link-Local Multicast Prefix:

FF	01	00	00	00	00	00	00	00	00	00	FF	00	00	00

Target IP Address:

FE	C0	00	00	00	A4	93	36	56	78	FF	FE	9A	BC	DE

Resultant Solicited-Node Multicast Address:

FF	01	00	00	00	00	00	00	00	00	00	FF	9A	BC	DE

Figure 1: The Solicited node multicast address for a target address is formed by appending the rightmost 3 bytes of the target address to the well-known solicited-node multicast prefix.

address derived from the target address as shown in Figure 1. The right-most 24 bits of the target IP address are appended to the well-known solicited-node multicast prefix to obtain the solicited-node multicast address. By sending NS messages to a solicited-node multicast address, only the node that has joined the multicast group for the requested address will actually process the NS request. In contrast, ARP messages are broadcast and must be processed by all nodes. In practice, it is (by design) unlikely that more than one node on a link will use the same solicited-node multicast address, making the cost of sending a multicast NS message as cheap as unicast on common link types (e.g., Ethernet).

NS and NA messages are also used by neighbor unreachability detection (NUD). In those cases where a neighbor's link-layer address is known, NS "Are You There?" messages are unicast directly to that neighbor. The returned NA provides an "Yes, I'm Here" acknowledgment indicating that the path and neighbor are still reachable. When NUD determines that a neighbor has become unreachable, an attempt is made to switch to another path. In the case where the neighbor is a router, an alternate router can be tried.

Hosts send *router solicitation* (RS) messages to find routers and routers send *router advertisement* (RA) messages to advertise their presence together with other configuration information. When a host first boots, it sends out an RS message. Routers respond by sending RA messages advertising their presence. Hosts use received RAs to build a list of default routers to which they may forward traffic. RAs also contain information used to configure addresses and such information as what Maximum Transmission Unit (MTU) value to use on the link, in those cases where the default isn't appropriate. Router advertisements are sent both in response to RS messages and periodically.

Neighbor discovery extends and improves on the IPv4 protocols ARP [Plu82] and router discovery [Dee91]. As described above, ARP uses link-layer broadcast, while ND uses link-layer multicast, when supported on the underlying media. Consequently, ND scales better than ARP as the number of nodes on a link increases. Moreover, the NS/NA message pairs are also used for NUD, whereas ARP is used purely for address resolution. ND's RS/RA message pairs subsume the functionality provided by IPv4's router discovery. In addition, NUD and the algorithm for flushing stale information from a node's cache in a timely manner insure that reachability failures are detected and acted upon before higher-layer protocols (e.g., TCP) give up and abort a connection. This key aspect of fault recovery was not part of the router discovery specification.

FE 80 00 00 00 00 00 00	Interface Identifier

Figure 2: The link-local address is formed by appending the interface identifier to the well-known link-local prefix fe80::/64.

5.2 Link-Local Address Formation

When a node starts, it begins the bootstrap process by generating a link-local address and assigning it to its interface. Because (in some cases) a node may actually be the only node on the network, it cannot rely on other servers for help. Thus, it must create an address using only information that it has. Almost all network interfaces today contain embedded IEEE identifiers (e.g., MAC addresses), which are to be globally unique. They consist of two parts, an *organization identifier* and a *local identifier*. Typically, the organization identifier identifies the manufacturer of the device, while the local identifier is a sort of serial number identifying a specific instance of the device. Thus, IEEE identifiers are designed to be globally unique and are good candidates to use in forming a unique IPv6 address.

IEEE identifiers used for Ethernet MAC addresses are 48 bits long. IEEE also defines an a 64-bit *Extended Universal Identifier* (EUI-64) [IEE97]. The longer identifier will be used in next-generation devices and ensures continued global uniqueness once the 48-bit IEEE identifier space becomes depleted. The current 48-bit identifier space is grandfathered into the EUI-64 space to prevent collisions.

IPv6 addresses are formed by concatenating a 64-bit address prefix with a 64-bit *interface identifier*. The address prefix identifies a specific link (e.g., LAN) within the Internet, while an interface identifier identifies an interface on a particular link. IPv6 interface identifiers are (in most cases) derived directly from EUI-64 identifiers. The exact details of how an interface identifier are formed depend on the specific device type (e.g., Ethernet [Cra98] vs. Token Ring [CNT98], etc.).

In the Ethernet case, the interface identifier is formed as follows. The 48-bit MAC address is converted into a 64-bit EUI-64 identifier, and then converted into an 64-bit IPv6 interface identifier. The only difference between the EUI-64 identifier and the Interface identifier is the flipping of the *universal/local* bit of the identifier. In an EUI-64 identifier, a bit of 0 indicates that that the identifier is globally unique. In IPv6, the meaning of the bit is reversed.

When initializing an interface, a node begins by assigning a link-local address to the interface. The link-local address is formed by appending the device's 64-bit interface identifier to the well-known link-local prefix of fe80::/64 (i.e., the hex value 0xfe80 followed by enough zeros to make a prefix a total of 64 bits) as shown in Figure 2. To ensure that the newly formed address is not already in use by another node on the attached link, the node performs a *duplicate address detection* (DAD) check. If another node is already using the address, autoconfiguration halts and manual configuration is required. Because there is no automatic recovery procedure when duplicate addresses occur, it is important that duplicate link-local addresses occur rarely in practice. It is for this reason that IEEE identifiers are used in forming addresses — they are by design globally unique. However, DAD is still needed to detect duplicates that arise arise from misconfigurations or other errors.

The DAD protocol is straightforward. A node sends out an NS message asking for the link-layer address of the target address it is proposing to use. If another node is already using the address, it will respond. By checking that there are no responses to its message (and that there are no additional NS queries other than the ones it sends), the sending node determines whether another node is already using the address.

5.3 Finding Neighboring Routers

Once a host has a link-local address and verified its uniqueness, it is ready to find neighboring routers. Because periodic RA messages are sent to the all-nodes multicast group, hosts must join the all-nodes multicast group in order to receive RAs. The length of time between successive periodic multicasts is controlled by the system administrator and can vary from a few seconds to as long as thirty minutes. To insure a booting host finds routers quickly, it sends out an RS message, using its link-local address as a source address. If there are any routers present, they will respond immediately, rather than waiting until the next periodic multicast has been scheduled for transmission.

In some environments (e.g., a home or small office) no routers will be present. If a host receives no RAs after retransmitting a small number of RS messages, it concludes that no routers are present and that all destinations should be assumed on-link. That is, all destinations will be assumed to reside on the attached link.

When routers are present, the RAs they send contain information a host needs to configure its interfaces. Two DHCP bits in the RA header indicate whether a host should invoke DHCP to obtain configuration information. A *managed* bit is used to indicate whether a host should use DHCP to obtain addresses in addition to those generated via stateless autoconfiguration. The *other* bit indicates that DHCP should be used to obtain other information (e.g., the addresses of one or more servers) via DHCP. If both bits are zero, DHCP should not be used. (The details of DHCP [BP99] usage are beyond the scope of this paper.)

Independent of the DHCP bits, a host examines RAs for one or more *prefix information options*. Prefix information options contain information on the prefixes associated with the link. The prefix information is used in two ways. First, a node needs to know the subnet number(s) and mask(s) of the link(s) to which it attaches. All traffic sent to destinations on the subnet are sent directly rather than forwarded through a router. Second, a node may want to generate addresses corresponding to the subnets to which it attaches.

To satisfy these requirements, prefix information options contain a subnet prefix together with preferred and valid lifetimes. If the *on-link* bit of the option is set, the node treats the prefix as being on-link for the next valid lifetime seconds. Packets sent to destinations matching the on-link prefix are sent directly rather than being forwarded through a router. If the *address bit* of the option is set, the node generates an address by combining the supplied prefix with its interface identifier. The preferred and valid lifetimes for the address are initialized from values in the option. If multiple prefix information options are supplied, each one is processed appropriately. Thus, a node can be assigned multiple addresses and/or on-link prefixes as appropriate.

For addresses generated from prefix information options, duplicate address detection can safely be skipped. Duplicate addresses occur when two interfaces use the same interface identifier. But since a node generates all of its addresses using the same interface identifier, the existence of duplicate identifiers should be detected at the time the link-local address is formed. Note also that although duplicate address detection only checks for uniqueness on the attached link, addresses formed from prefix information options should be unique within their scope. For example, global-scope addresses will be globally unique; the first 64-bits of the address identify a specific link (uniquely), and the remaining bits specific the interface on that link (uniquely).

Routers typically send periodic router advertisements every few minutes. Each new RA updates the information in earlier RAs. Because the information in prefix information options has an associated lifetime, it will timeout if not updated by a successive RA.

Router advertisements provide a mechanism that makes it straightforward to change the IP addresses a site is using. As part of configuring routers, a system administrator specifies which prefixes should be included in router advertisements and what lifetimes should be used. Lifetimes can be configured in two ways. First, they can be specified in a delta form, meaning that each successive RA contains the same lifetime value. Lifetimes can also be specified to indicate an absolute time in the future. In this case, each successive RA includes a lower lifetime than its predecessor, eventually reaching zero. Using an absolute lifetime is useful when an address is to be phased out at a specific known time in the future (e.g., when a switch from one ISP to another is scheduled to complete). The delta method will likely be the more common form. It allows the same prefixes to be advertised essentially indefinitely, yet should renumbering be required, it can be initiated on relatively short notice, the exact time bounded by the the lifetimes being advertised. Note that while it is possible to specify a very long (or even infinite) lifetime, doing so has the disadvantage that it is not necessarily possible to retroactively make the lifetime shorter in the future. Once a node has received information with a certain lifetime value, there is no guarantee that it will ever receive a future update that shortens the lifetime (e.g., the node might be disconnected from the network when shorter lifetimes are advertised). Thus, it is suggested that the delta method be used with moderately short lifetimes (a few days or weeks). That way, one can easily phase out the use of a particular prefix on relatively short advance notice.

5.4 Keeping Track of Routers

Hosts build and maintain a list of the available routers. Each available router sends out an RA, and the RA includes a lifetime field specifying how long a host should keep that router on its list. All routers are deemed equivalent for the purposes of RAs. If one router is "better" than another, redirects can be used to redirect traffic to the more appropriate router.

The time between successive periodic RAs depends on local configuration options, but is typically 4-10 minutes. Thus, hosts cannot use lack-of-received RAs as a timely enough indication that a router (or the path leading to it) is no longer functioning properly. The neighbor unreachability detection mechanism, described later, addresses this issue.

RAs contain information used for configuring hosts. Some information is required to appear in every RA, while some is optional. For example, each RA message includes a *router lifetime* field that defines how long a host may keep that router on the available router list. An RA may also (but is not required to) contain one or more prefix information options as described earlier.

5.5 Neighbor Discovery

We have already described the basic neighbor discovery messages and how they are processed. The remaining step is to describe how the pieces fit together so that hosts can perform optimal routing without needing to participate in routing protocols.

To handle routing issues, neighbor discovery makes use of four conceptual data structures, the *Destination Cache*, the *Neighbor Cache*, a *Prefix List*, and an *Available Routers List*. The available routers list contains an entry for each neighboring router. The prefix list indicates which destinations are on-link (i.e., are on the attached subnet). Destinations not covered by prefixes on the prefix list must be forwarded to a router for delivery. The destination and neighbor caches contain entries for all destinations being communicated with, and for all neighbors to which packets are being forwarded respectively. The destination cache, prefix list, and available routers list collectively comprise what is commonly thought of as the routing table. The neighbor cache serves a similar purpose to that of the ARP cache, though it contains more information.

The destination cache is keyed on complete IPv6 addresses and contains an entry for every destination (whether a neighbor or remote) to which a host is (or has recently been) communicating. The idea behind the destination cache is that all applications communicating with a particular destination share the same destination cache entry. All information common to communication, such as the current Path MTU value or retransmission timer estimates, could be stored and maintained in the same place, with all applications benefitting from the sharing of information.

The neighbor cache contains entries for every neighbor (hosts and routers) on an attached link. The neighbor cache contains information necessary to communicate with neighboring nodes. For example, the link-layer address of a neighbor is stored here. Entries in the neighbor cache are pointed to by entries in the destination cache. Specifically, when sending traffic to a destination, the destination cache points to the specific neighbor to which traffic should be forwarded. When the ultimate destination is a neighbor, it might appear that having a two-level cache is unnecessary. But in many cases, the destination cache will point to a neighbor cache entry belonging to a router. Indeed, many entries in the destination cache will point to the same router in the neighbor cache.

One of the main benefits of having the two-level cache is that it becomes easy to switch from one router to another, in the case of failures. In particular, neighbor unreachability detection will quickly determine if a neighboring router is no longer reachable (e.g., if it crashes). Once NUD detects a connectivity problem with a router, the sending node will go to the list of available routers and pick a new one. All applications sending traffic through the "failed" router will then quickly switch to an alternate one.

When a node initiates communication to a particular destination, it consults the destination cache. If there is no entry for the destination, one is created and the node consults the prefix list to determine whether the destination is on-link or behind a router. If the former, the neighbor cache is consulted, and the node consults appropriate neighbor cache entry (creating it if necessary). If the destination is behind a router, the node chooses a router from the available router list. The policy for selecting routers is quite flexible. If multiple routers are present, the entries can be selected in a round-robin fashion, so that load sharing of traffic among routers is possible. Alternatively, all traffic could be forwarded to the same router. Once a router has been selected, an appropriate entry is created (if needed) in the neighbor cache.

5.6 Neighbor Unreachability Detection

NUD deals with the issue of verifying that a path to a neighbor is functioning properly. *Properly* means that if a packet is forwarded to a neighbor, the packet does reach the neighbor. Path failures include the link going down (possibly only in the forward direction) or a crash of the neighboring node itself. By actively monitoring the path to a neighbor, NUD provides a way to detect potential problems. If a problem is identified, attempts at recovery can be initiated (e.g., find an alternate router).

NUD is implemented by augmenting the neighbor cache with state information about the current reachability state of each neighbor. For each active neighbor, if it is not known whether the neighbor is actually reachable, a NS probe is unicast to it. The receipt of a response NA indicates that the path is working properly. Reachability is verified quite frequently, roughly every thirty seconds. It is important to detect path failures quickly, so that an alternate path can be tried (e.g., if multiple routers are available) before higher layers such as TCP give up.

At first glance, it might appear that sending NS probe messages every thirty seconds generates a lot of network traffic. However, probes are only sent for active entries, those through which traffic is actually being sent. If an entry isn't being used, no probes are sent. In addition, it turns out that explicit NS probes are usually not sent. Higher layer transport protocols such as TCP retransmit messages that are not acknowledged. New acknowledgments (i.e, acknowledgements that have not been received before) indicate that data is reaching a peer. Thus, TCP can update the reachability information in the neighbor cache whenever it receives such ACKs. Consequently, the sending of NS probes are only needed for UDP-based protocols.

5.7 Neighbor Discovery Support For Mobility

IPv6 [DH98] has a number of powerful capabilities that Mobile IP [JP98] exploits. Mobile IP also leverages off of ND to build a robust mobility environment. In mobile IP, *home agents* proxy on the behalf of nodes that consider a particular link their "home", but are currently located elsewhere. *Home agents* intercept packets sent to a mobile node's *home address* and resend them to the mobile node's *care-of address* (i.e., the mobile node's current location).

Home agents use the special *override flag* in ND messages to proxy for nodes that are away from home. Where the override flag comes in is when the mobile node is back home again, but the home agent still believes it should be proxying for the mobile node. A home agent always clears the override flag in NA messages it sends, meaning that the advertisement shouldn't replace existing information in a node's cache. However, if a node is requesting information and has no entry in its cache, an NA with the override bit cleared will update the entry. When both the home agent and the mobile node respond to a NS, the mobile node's NA will have override bit set, causing it to take precedence over the home agent's advertisement. This ensures that communication goes through the home agent only when the mobile node is not connected to its home link.

In order to provide robust home agent service, the home agents attached to the same link need to know of each other's presence as well as some characteristics about the service they provide. Rather than invent a new protocol for this purpose, a single bit in the ND header indicates whether a router is also a home agent. Since all home agents are routers, they can advertise themselves and learn of each other's presence by sending and listening RAs with the home agent bit set. In those cases where a home agent is configured differently from a standard configuration, a *New Home Agent Information option* allows a home agent to advertise itself as having a different preference level and home agent lifetime.

When a mobile node is visiting a link, it obtains a temporary care-of address that it uses while visiting the link. The care-of address is obtained via the autoconfiguration mechanisms described earlier. Packets sent to the mobile node are forwarded through the care-of address, either via tunneling (if intercepted by the home agent) or by the IPv6 *routing Header* [DH98] (if the sender has learned of the mobile node's care-of address). Should the node then move elsewhere, however, packets sent to the mobile node's *previous* care-of address would not normally be delivered to the node's *current* care-of address. To cover this case, mobile nodes send *binding updates* to the router at their previous location asking them to (temporarily) act as a home agent for the previous care-of address. In order for the interaction with a previous router to work, a mobile node must know an appropriately-scoped (i.e., not link-local) address for the router. ND, by default however, only includes the link-local address of a sender. To handle this case, a special bit in the prefix information option indicates that the prefix included in the option is actually the complete globally-scoped address of the sending router.

6 Summary & Conclusion

This paper presented an overview of the facilities provided by IPv6 with an emphasis on its autoconfiguration facilities. IPv6 hosts can plug into the network and start communicating without requiring special configuration. This includes hosts connecting to isolated stand-alone networks (e.g., at a home) or those connected to a large corporate network.

Nodes autoconfigure a link-local address by deriving an address from the embedded IEEE identifier of the interface. Duplicate address detection is then performed to ensure that the new address is not already in use by another node on the link. Once the address has been deemed non-unique, the node sends out router solicitations to locate routers and obtain additional configuration, including prefix information needed to generate global-scope addresses. Equipped with one or more addresses, the node can begin communicating.

Nodes use neighbor discovery to perform address resolution and neighbor unreachability detection. NUD ensures that cached information such as the link-layer addresses of neighbors, next-hop router addresses, etc. is updated quickly should it become stale. Specifically, NUD ensures that cached information about a path is revalidated in a timely fashion should communication begin to fail. Attempts to find alternate paths happen quickly enough that higher layers (e.g., TCP) won't time out and abort connections due to excessive retransmissions in those cases where an alternate path is viable.

Neighbor discovery is a flexible protocol designed to allow future protocols to leverage off of it. Indeed, Mobile IP is in the process of defining new options that demonstrate the viability of this approach.

IPv6 is a protocol designed in the IPng Working Group of the IETF. For more information on IPv6, including information on implementations and ongoing work, see the web page at http://playground.sun.com/pub/ipng/html/ipng-main.html.

References

[ASJ99] G. Armitage, P. Schulter, and M. Jork. RFC 2492: IPv6 over ATM Networks, January 1999.

[ASJH99] G. Armitage, P. Schulter, M. Jork, and G. Harter. RFC 2491: IPv6 over Non-Broadcast Multiple Access (NBMA) Networks, January 1999.

[BP99] J. Bound and C. Perkins. Dynamic Host Configuration Protocol for IPv6 (DHCPv6) (work in progress), February 1999.

[CD98] A. Conta and S. Deering. RFC 2463: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, December 1998.

[CNT98] M. Crawford, T. Narten, and S. Thomas. RFC 2470: Transmission of IPv6 Packets over Token Ring Networks, December 1998.

[Cra98] M. Crawford. RFC 2464: Transmission of IPv6 Packets over Ethernet Networks, December 1998.

[Dee91] S. Deering. RFC 1256: ICMP Router Discovery Messages, September 1991.

[DH98] S. Deering and R. Hinden. RFC 2460: Internet Protocol, Version 6 (IPv6) specification, December 1998.

[EF94] K. Egevang and P. Francis. RFC 1631: The IP Network Address Translator (NAT), May 1994.

[GA92] P. Gross and P. Almquist. RFC 1380: IESG Deliberations on Routing and Addressing, November 1992.

[HD98] R. Hinden and S. Deering. RFC 2373: IP Version 6 Addressing Architecture, July 1998.

[IEE97] IEEE. Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority, March 1997. http://standards.ieee.org/db/oui/tutorials/EUI64.html.

[JP98] D. Johnson and C. Perkins. Mobility Support in IPv6 (work in progress), November 1998.

[KA98] S. Kent and R. Atkinson. RFC 2401: Security Architecture for the Internet Protocol, November 1998.

[NNS98] T. Narten, E. Nordmark, and W. Simpson. RFC 2461: Neighbor Discovery for IP Version 6 (IPv6), December 1998.

[Plu82] D. C. Plummer. RFC 826: Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware, November 1982.

[Pos81] J. Postel. RFC 791: Internet Protocol, September 1981.

[RMK+96] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. RFC 1918: Address Allocation for Private Internets, February 1996.

[TN98] S. Thomson and T. Narten. RFC 2462: IPv6 Stateless Address Autoconfiguration, December 1998.