NEIGHBOR DISCOVERY and Stateless Autoconfiguration in IPv6

THOMAS NARTEN
IBM Corporation

The prospect of an exhausted supply of address space in the current Internet Protocol, IPv4, prompted the IETF to start a next-generation IP project in the early 1990s. The resulting IPv6 offers a 128-bit address space and other upgrades that support the autoconfiguration of new hosts.

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 and e-mail, all use TCP and UDP, which in turn are built on top of IP. The current version, IPv4, uses 32-bit addresses.[1] 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 percent utilization. In the early 1990s, the rate of address consumption was great enough to indicate an exhausted pool of addresses early in the 21st century.[2]

Depletion of the IPv4 address space was the core motivation for IPv6,[3] which emerged from the Internet Engineering Task Force's Next Generation Internet Protocol (IPng) project and became a draft standard. IPv6 has 128-bit addresses, which precludes running out of unique addresses in the foreseeable future. However, the larger address space required a change in the basic format of IP packets. Because such changes are not backward-compatible, it also became appropriate to consider other changes that would correct known deficiencies in IPv4. Autoconfiguration is one such change. New machines that plug into an IPv6 network for the first time can autoconfigure a usable address and, if one or more routers is present, find and use them as appropriate. Hosts can also switch automatically from one router to another when a connectivity failure occurs.

This article begins with background on IPv6 to help understand its autoconfiguration facilities, and then describes the neighbor discovery protocols that implement them.

BACKGROUND

In response to concerns about the possible depletion of IPv4 addresses, registries—such as Reseaux IP Européens (RIPE) and Asia-Pacific Network Information Center (APNIC)—and Internet service providers (ISPs) became stingy about giving out addresses in the mid 1990s. Obtaining a

 1089-7801/99/$10.00 ©1999 IEEE

block of addresses now requires a customer to justify that all addresses will be used immediately. In addition, the introduction and deployment of Network Address Translation devices[4] (see the sidebar, "NAT Devices: When an address is not an address") has slowed the apparent consumption of IPv4 address space, though it has not lessened the need for IPv6.

IPv6 Improvements on IPv4

In addition to the upgrade to 16-byte addresses, IPv6 redesigned the packet format to account for lessons learned from almost 20 years of experience with IPv4. The following summarizes some improvements incorporated into the new protocol:

- IPv6 has built-in security support. All IPv6 nodes support cryptographic-based authentication and encryption. An IPv6 application can assume that IPsec is present on all nodes running IPv6. With IPsec,[5] each node can send and receive packets with encrypted contents, and have a strong degree of assurance about the correct identity of its peers.
- IPv6 eliminates the checksum from the IP header. This reduces the amount of work a router must do to forward a packet, which in turn improves router performance and reduces costs.
- IPv6 offers more flexible and extensible option support than IPv4. For example, options in IPv4 are limited to a total of 44 bytes in length; no such restriction exists in IPv6.
- IPv6 facilitates efficient renumbering of sites by explicitly supporting multiple addresses on an interface. In addition, all addresses are leased; when a lease expires, the address is no longer usable. A site can renumber itself by having a transition period in which both old and new addresses work equally.
- IPv6 supports plug-and-play operation. A new machine generates its own addresses automatically and can immediately communicate with other nodes. Autoconfiguration is targeted both at small sites, such as a home network without a router or DHCP server, and at large sites, such as 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 DHCPv6 to obtain addresses and other configuration information.[6]

NAT boxes intercept all packets leaving the network, and change the internal private address into a global address. This makes it possible for a site with thousands of nodes to use only a small number of global addresses. Thus, although the Internet continues to grow exponentially, the rate at which globally unique addresses are being consumed has been significantly reduced since the early 1990s.

Despite the popularity of NAT-based solutions, they introduce their own problems. Although many applications work transparently through a NAT device, others do not. The latter thus require Application-Level Gateways. ALGs terminate one TCP connection and set up a second TCP connection to the actual desired server. Alternatively, it may be necessary to change the application protocol itself, usually a painful process if it can be done at all.

NAT also poses security problems. By definition, NAT involves taking a packet and replacing one of its addresses with another. Security, on the other hand, concerns itself with ensuring that a packet carried across a network has not been inappropriately modified, for example, by a malicious intruder. The IPsec security standard, for instance, does not work when a NAT device resides along the path between secured endpoints. When a NAT device changes an address, the IPsec authentication check performed by the receiver fails; a receiver is unable to distinguish the case where a packet is modified by a NAT device from the case where an intruder is attempting an attack.

There are numerous other issues related to NAT. NAT places hard state in the network; if NAT device restarts, all connections traveling through the NAT device are lost and must be restarted. Scaling NAT beyond a single NAT device (e.g., a site with multiple ISP connections or even multiple egress routers to the same ISP) is problematic. If a site's internal routing topology changes, the NAT device a particular TCP connection travels through may change. This typically results in the failure of a connection, much like if a NAT device restarts.

In summary, although NAT devices have reduced the rate at which globally unique addresses are being assigned, they have not eliminated the desirability of having such addresses. As security concerns become more important, and NAT operational problems become more evident, the large globally unique address space available in IPv6 will become necessary.

Some technologies originally motivated by IPv6, such as IPsec, have been retrofitted as options in IPv4. However, it is not possible to retrofit all of them. The most obvious exception is IPv6's large pool of globally unique addresses. Features like stateless address autoconfiguration, which depend on large addresses, are another.

Conceptual Bases for IPv6 Autoconfiguration

IPv6 is designed to run over virtually any physical media. The two standard 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 media*, which connect exactly two nodes. Links are assumed to support bi-directional connectivity, in order to obtain the full benefits of autoconfiguration. IPv6 also supports other link types, including non-broadcast multiple access (NBMA) links like ATM[7,8] and Frame Relay, though specialized servers may be needed.

Once the new addresses are in place and working, the old addresses can be quietly phased out.

Multiple addresses. IPv6 supports multiple address scopes:[9]

- *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, such as 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 (for example, a home Ethernet) or during bootstrap operations as part of obtaining other addresses.
- *Site-local* addresses are unique only within a particular site or organization. Site-local addresses are analogous to IPv4's private addresses.[10] They can be used only within a site, and routers do not forward packets containing site-local addresses past site boundaries.

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. It can then use this address to obtain 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 (that is, the site has two independent connections), it could obtain addresses from both ISPs, and each node in the site could have two addresses. Multiple addresses also support situations where an entire site needs to be renumbered, for example, if the ISP is changed. IPv6 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.

Address lifetimes. 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 the address will no longer be forwarded properly.
- A *preferred lifetime* indicates the period during which the address can safely be used for all communication. The preferred lifetime must be shorter than the valid lifetime.

An address is said to be *preferred* if its preferred lifetime has not yet expired. An address is said to be *deprecated* if its preferred lifetime has expired, but it is still valid. Deprecated addresses can still be used for communication, but applications opening new TCP connections, for instance, would use a preferred address instead of one that is deprecated. Doing so ensures that few, if any, applications are using a deprecated address at the time its valid lifetime finally expires.

Having both a valid and preferred lifetime facilitates renumbering by gracefully overlapping the phase out of an existing address with the introduction of a new one.

Multicasting. IPv6 does not support broadcasting at the IP layer; 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 off the link.
- Site-local addresses are used for communication within a site, and packets containing site-local addresses are not forwarded beyond the site.
- Global-scope addresses are used for global communication.

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 IPv6 node on a link.

NEIGHBOR DISCOVERY IN IPV6

The IPv6 protocols known collectively as Neighbor Discovery replace a number of IPv4 protocols that offer some autoconfiguration facilities. ND defines new functionality as well.[11,12] In general, ND performs three major functions:

- It provides address resolution service, updating and expanding IPv4's address resolution protocol (ARP).[13]
- It allows hosts to discover what neighboring routers are present and provides a mechanism for obtaining certain configuration information from them.
- It 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 the Internet Control Message Protocol (ICMP),[14] making all services (including address resolution) independent of link technologies. In addition, IPsec services (for example, cryptographic authentication) can be applied to ND if desired.

ND defines two main pairs of messages:

- Neighbor solicitation (NS) and neighbor advertisement (NA) messages 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 resolves the address 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. As shown in Figure 1, the solicited-node address is formed by appending the rightmost 24 bits of the target IP address to the well-known solicited-node multicast prefix. By sending NS messages to a solicited-node multicast address, only the few nodes that have joined the specific multicast group associated with the requested address will actually process the NS request. In practice and by design, it is 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 like Ethernet.

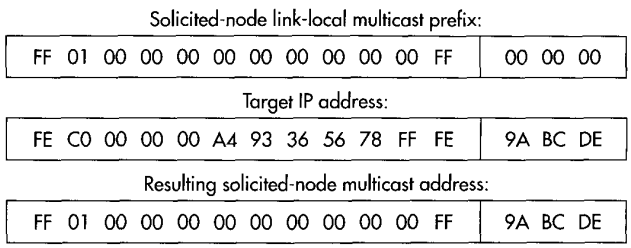

Figure 1. The solicited-node multicast address in IPv6 is formed by appending the rightmost 24 bits of the target IP address to the solicited-node multicast prefix.

NUD also uses NS and NA messages to determine whether a neighbor has become unreachable. In those cases where a neighbor's link-layer address is already known, NS "Are You There" messages are unicast directly to that neighbor; the returned NA provides a "Yes, I'm Here" acknowledgment, indicating that the path and neighbor are still reachable. When ND determines that a neighbor has become unreachable, it attempts to switch to another path. In the case where the neighbor is a router, an alternate router can be tried.

Hosts send RS messages to find routers, and routers send RA messages to advertise their presence, together with other configuration information. 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. RAs are sent both periodically and in response to RS messages.

Neighbor Discovery extends and improves on IPv4's ARP. First, ND uses link-layer multicast, when supported on the underlying media, and consequently scales better than ARP as the number of nodes on a link increases. Moreover, the NS/NA

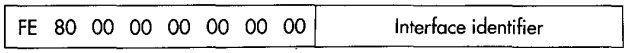

Figure 2. The link-local address is formed by appending the interface identifier to the link-local prefix, fe80::/64.

message pairs are also used for NUD, whereas ARP is purely for address resolution.

ND's RS/RA message pairs subsume the functionality provided by IPv4's router discovery.[15] In addition, NUD and its associated cache coherence protocol ensure that reachability failures are detected and acted upon before higher layer protocols, such as TCP, give up and abort a connection. This is a key aspect of fault recovery that was not part of IPv4's router discovery specification.

Forming a Link-Local Address

A node starts the bootstrap process by generating a link-local address and assigning it to its interface. Because a node may sometimes be the only node on a 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, such as MAC addresses. The identifiers 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 for forming a unique IPv6 address.

IEEE identifiers used for Ethernet MAC addresses are 48 bits long. IEEE also defines a 64-bit Extended Universal Identifier (EUI-64).[16] The longer identifier will be used in next-generation devices and ensures continued global uniqueness when 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, such as a 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 network type, such as Ethernet[17] or Token Ring.[18]

In the Ethernet case, the interface identifier is formed by first converting the 48-bit MAC address into a 64-bit EUI-64 identifier, and then converting the result into a 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, there is a bit position for which a value of "0" indicates that the identifier is globally unique. In IPv6, the meaning of that bit is reversed. The difference simplifies configuration of IPv6 addresses in those rare cases where manual configuration is needed (for example, the automatically generated address is a duplicate).

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 standard link-local prefix of fe80::/64 (that is, the hex value of 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 specified when duplicate addresses occur, it is important that duplicate link-local addresses occur rarely in practice. This is why IEEE identifiers are used in forming addresses—they are by design globally unique. However, DAD is still needed to detect duplicates that arise from misconfiguration 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 no NS queries other than the ones it sends), the sending node determines whether another node is already using the address.

Finding Neighboring Routers

Once a host has a link-local address and verifies 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 to receive RAs. The time between successive periodic multicasts is controlled by the system administrator and can vary from a few seconds to as long as 30 minutes. To ensure that 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 wait until the next periodic multicast scheduled for transmission.

In some environments, such as 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 assumes that all destinations reside on-link.

Routers send RAs containing information a host needs to configure its interfaces. Two DHCP bits in the RA header indicate whether a host should invoke DHCPv6 to obtain configuration information. A *managed* bit indicates whether a host should use DHCPv6 to obtain addresses in addition to those generated via stateless autoconfiguration. The *other* bit indicates that DHCPv6 should be used to obtain other information, such as the address of one or more servers, via DHCP. If both bits are zero, DHCPv6 should not be used. (For details on DHCPv6, see Bound and Perkins.[6])

Independent of the DHCP bits, a host examines RAs for one or more *prefix information options*. These options contain information on the prefixes associated with the link. The information is used in two ways:

- First, a node needs to know the subnet number and mask of the link to which it attaches. All traffic to destinations on the subnet is 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, DAD can safely be skipped. Duplicate addresses occur when two interfaces use the same interface identifier; however, 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 DAD 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 uniquely identify a specific link, and the remaining bits uniquely specify the interface on that link.

RAs provide a mechanism that makes it straightforward to change the IP addresses a site is using.

Routers typically send periodic RAs 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 expire if not updated by a successive RA.

RAs 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 to include in RAs and what lifetimes to use.

Lifetimes can be configured in two ways:

- in a delta form, meaning that each successive RA contains the same lifetime value, or
- in an absolute time (in the future), in which case each successive RA includes a lower lifetime than its predecessor, eventually reaching zero.

An absolute lifetime is useful when an address is to be phased out at a specific known time, for example, when a switch from one ISP to another is scheduled to complete. The delta form, however, will likely be more common. 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 lifetimes being advertised.

It is possible to specify a very long—even infinite—lifetime, but 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; for example, 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, such as a few days or weeks. This makes it easy to phase out the use of a particular prefix on relatively short notice.

Keeping Track of Routers

Hosts build and maintain a list of available routers. All routers are deemed equivalent for the purposes of RAs. If one router is "better" than another, redirects can be used to send traffic to the more appropriate router.

The time between successive periodic RAs depends on local configuration options, but is typically 4 to 10 minutes. This is too much time for hosts to go without knowing that a router (or the path leading to it) is no longer functioning properly. Hence, the need for the NUD mechanism.

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.

Fitting the Pieces Together

To handle routing issues, ND makes use of four conceptual data structures:

- The *available routers list* contains an entry for each neighboring router.
- The *prefix list* indicates which destinations are on-link (that is, on the attached subnet). Destinations not covered by prefixes on the prefix list must be forwarded to a router for delivery.
- The *destination cache* contains entries for all destinations being communicated with.
- The *neighbor cache* contains entries for all neighbors to which packets are being forwarded.

The destination cache, prefix list, and available routers list collectively form 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, with which a host currently 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 benefiting from the sharing of information.

The neighbor cache contains entries for every neighbor (host or router) on an attached link. The neighbor cache contains information necessary to communicate with neighboring nodes (for example, link-layer addresses). 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 benefit of having the two-level cache is that it becomes easy to switch from one router to another when failures occur. In particular, NUD will quickly determine if a neighboring router is no longer reachable (for example, 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, ND creates one, and the node consults the prefix list to determine whether the destination is on-link or behind a router. If the former, ND consults the neighbor cache, 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 takes place. 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.

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, it reaches the neighbor. Path failures include the link going down (possibly only in the forward direction) or a crash of the neighboring node itself. NUD provides a way to actively monitor the path to a neighbor and to attempt recovery when a problem is identified.

NUD is implemented by augmenting the neighbor cache with state information about the current reachability state of each neighbor. A node unicasts an NS probe to each active neighbor whose reachability is unknown. The receipt of a response NA indicates that the path is working properly. Reachability is verified quite frequently, roughly every 30 seconds. It is important to detect path failures quickly, so that an alternate path can be tried before higher layers, such as TCP, give up.

It might seem that sending NS probe messages every 30 seconds would generate a lot of network traffic. However, probes are sent only for active entries. If an entry isn't being used, no probes are sent. In addition, it turns out that sending explicit NS probes is often unnecessary. Higher layer transport protocols such as TCP retransmit messages that are not acknowledged. New acknowledgments indicate that data is reaching a peer. Thus, TCP can update the reachability information in the neighbor cache whenever it receives such acknowledgments, and NS probes are only needed for UDP-based protocols.

CONCLUSION

IPv6 (including neighbor discovery and stateless address autoconfiguration) has been promoted by the IETF to Draft Standard. Draft Standard status is granted to protocols for which multiple interoperable implementations have been demonstrated and for which the technology is believed to be mature and stable. Work on the core IPv6 protocols is now finished, and IPv6 is mature enough for widescale implementation and deployment.

IPv6 hosts can plug into the network and start communicating without requiring special configuration, whether the connection is to isolated standalone networks (for example, a home network) or to a large corporate network. A link-local address is autoconfigured from the embedded EUI-64 identifier of the interface and tested to ensure that it does not duplicate an address already in use by another node on the link. Once the address is deemed unique, the node sends out solicitations to locate routers and obtain additional configuration, including prefixes needed to generate global-scope

Mobile IP allows a node to move from one part of the Internet to another without an interruption of communication to higher-layer protocols (for example, TCP). Specifically, a node that leaves one part of the network retains the address it was using there and arranges to have traffic forwarded from its previous address to its current address. Mobile IP exploits a number of powerful capabilities in IPv6.[1] It also leverages ND to build a robust mobility environment.

In mobile IP, *home agents* proxy on 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*—that is, the mobile node's current location.

When a mobile node is visiting a link, it obtains a temporary care-of address, using the IPv6 autoconfiguration mechanisms. Packets sent to the mobile node are forwarded through the care-of address, either via tunneling (if intercepted by the home agent) or via the IPv6 routing header (if the sender has learned of the mobile node's care-of address). Should the node then move elsewhere, 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 act temporarily as a home agent for the previous care-of address.

For the interaction with a previous router to work, a mobile node must know an appropriately scoped address for the router—in other words, the address cannot be a link-local address. By default, however, ND includes only the link-local address of a sender. To handle this case, a special bit in the prefix information option indicates that the prefix is actually the complete globally scoped address of the sending router.

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 for RAs with the home-agent bit set. In those cases where a home agent is configured differently from a standard configuration, IPv6 includes a New Home Agent Information option that allows a home agent to advertise itself as having a different preference level and home-agent lifetime.

REFERENCE

1. D. Johnson and C. Perkins, *Mobility Support in IPv6*, IETF, Nov. 1998, work in progress; available online at http://www.ietf.org/internet-drafts/draft-ietf-mobileip-ipv6-07.txt.

addresses. Equipped with one or more addresses, the node can begin communicating.

IPv6 is being designed in the IPng Working Group of the IETF. More information on IPv6, including information on implementations and ongoing work, is available at http://playground.sun.com/pub/ipng/html/ipng-main.html. ■

REFERENCES

1. J. Postel, *Internet Protocol*, RFC 791, Sept. 1981; available online at http://www.rfc-editor.org/rfc791.txt.
2. P. Gross and P. Almquist, *IESG Deliberations on Routing and Addressing*, RFC 1380, Nov. 1992; available online at http://www.rfc-editor.org/rfc1380.txt.
3. S. Deering and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 2460, Dec. 1998; available online at ftp://ftp.isi.edu/in-notes/rfc2460.txt.
4. K. Egevang and P. Francis, *The IP Network Address Translator (NAT)*, RFC 1631, May 1994; available online at http://www.rfc-editor.org/rfc1631.txt.
5. S. Kent and R. Atkinson, *Security Architecture for the Internet Protocol*, RFC 2401, Nov. 1998; available online at http://www.rfc-editor.org/rfc2401.txt.
6. J. Bound and C. Perkins, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, work in progress, Feb. 1999; available online at http://www.ietf.org/internet-drafts/draft-ietf-dhc-dhcpv6-14.txt.
7. G. Armitage et al., *IPv6 over Non-Broadcast Multiple Access (NBMA) Networks*, RFC 2491, Jan. 1999; available online at http://www.rfc-editor.org/rfc2491.txt.
8. G. Armitage, P. Schulter, and M. Jork, *IPv6 over ATM Networks*, RFC 2492, Jan. 1999; available online at http://www.rfc-editor.org/rfc2492.txt.
9. R. Hinden and S. Deering, *IP Version 6 Addressing Architecture*, RFC 2373, July 1998; available online at http://www.rfc-editor.org/rfc2373.txt.
10. Y. Rekhter et al., *Address Allocation for Private Internets*, RFC 1918, Feb. 1996; available online at http://www.rfc-editor.org/rfc1918.txt.
11. D.C. Plummer, *Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*, RFC 826, Nov. 1982; available online at http://www.rfc-editor.org/rfc826.txt.
12. T. Narten, E. Nordmark, and W. Simpson, *Neighbor Discovery for IP Version 6 (IPv6)*, RFC 2461, Dec. 1998; available online at http://www.rfc-editor.org/rfc2461.txt.
13. S. Thomson and T. Narten, *IPv6 Stateless Address Autoconfiguration*, RFC 2462, Dec. 1998; available online at http://www.rfc-editor.org/rfc2462.txt.
14. A. Conta and S. Deering, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, RFC 2463, Dec. 1998; available online at http://www.rfc-editor.org/rfc2463.txt.
15. S. Deering, *ICMP Router Discovery Messages*, RFC 1256, Sep. 1991; available online at http://www.rfc-editor.org/rfc1256.txt.
16. *IEEE Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority*, Mar. 1997, available online at http://standards.ieee.org/db/oui/tutorials/EUI64.html.
17. M. Crawford, *Transmission of IPv6 Packets over Ethernet Networks*, RFC 2464, Dec. 1998; available online at http://www.rfc-editor.org/rfc2464.txt.
18. M. Crawford, T. Narten, and S. Thomas, *Transmission of IPv6 Packets over Token Ring Networks*, RFC 2470, Dec. 1998; available online at http://www.rfc-editor.org/rfc2470.txt.

Thomas Narten works for IBM in Research Triangle Park, North Carolina. From 1989-1994 he was on the faculty at State University of New York-Albany. His current focus is on networking protocols. Narten has a BA from the University of Tennessee and an MS and PhD in computer science from Purdue University. He is active in the Internet Engineering Task Force, where he is an area director in the Internet area.

Narten can be reached at IBM Corp., 3039 Cornwallis Ave., PO Box 12195 - BRQA/502, Research Triangle Park, NC 27709-2195; email narten@raleigh.ibm.com.