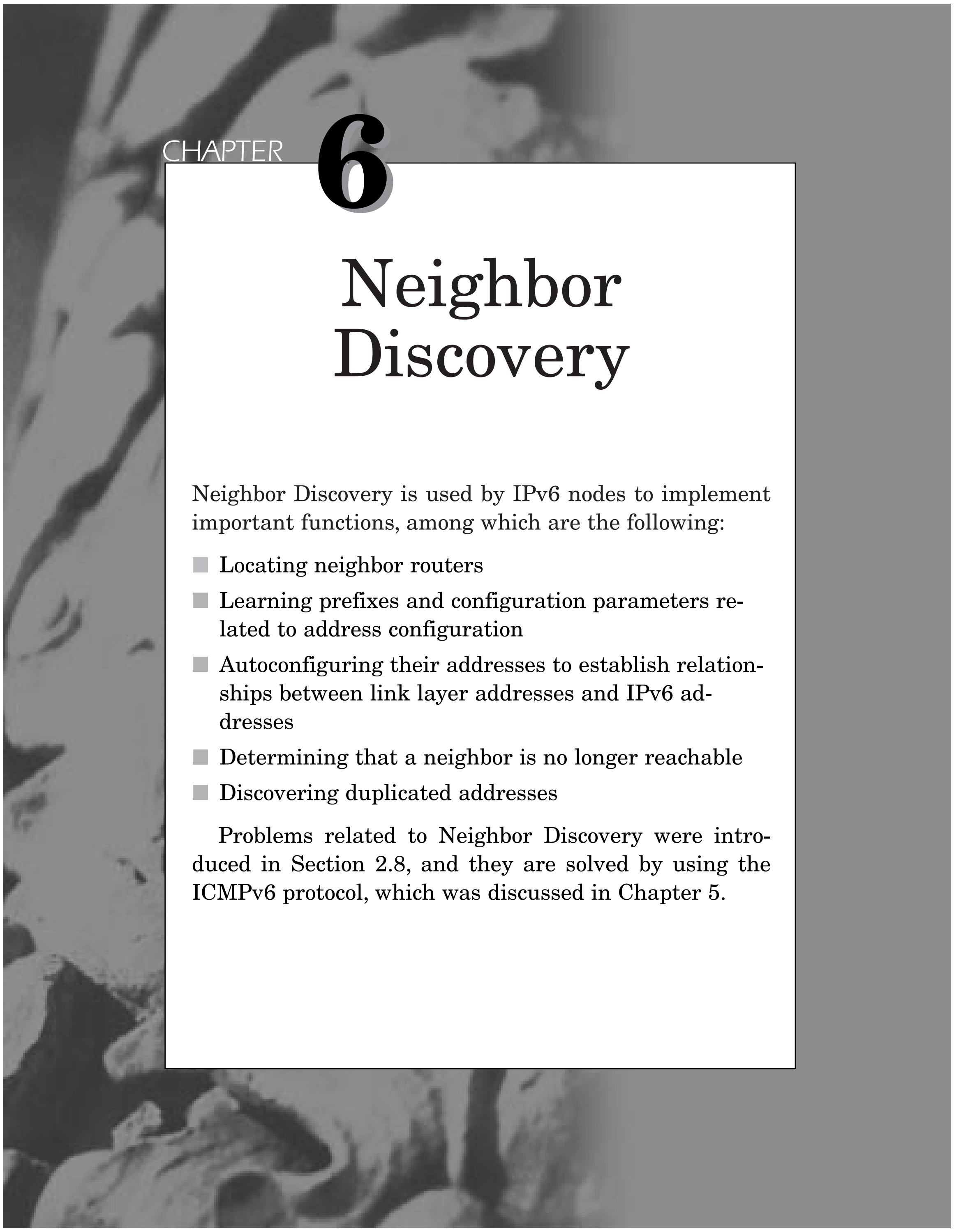

CHAPTER 6

Neighbor Discovery

Neighbor Discovery is used by IPv6 nodes to implement important functions, among which are the following:

- Locating neighbor routers
- Learning prefixes and configuration parameters related to address configuration
- Autoconfiguring their addresses to establish relationships between link layer addresses and IPv6 addresses
- Determining that a neighbor is no longer reachable
- Discovering duplicated addresses

Problems related to Neighbor Discovery were introduced in Section 2.8, and they are solved by using the ICMPv6 protocol, which was discussed in Chapter 5.

6.1 Terminology

The following definitions are excerpted from the RFC 1970[1]; they are in addition to those reported in Section 2.2:

- *prefix:* The initial part of an IPv6 address, common to all nodes connected to the same link.
- *link layer address:* The layer 2 address of an interface.
- *on-link:* An IPv6 address that is assigned to an interface on a specified link.
- *off-link:* An IPv6 address that is not assigned to any interfaces on the specified link.
- *longest prefix match:* The process of determining which prefix includes a given IPv6 address. When multiple prefixes cover an address, the longest prefix is the one that matches.
- *next hop:* The next node toward which to transmit a packet. The node must be on-link and therefore must be a neighbor.
- *reachability:* Whether the one-way "forward" path to a node is functioning properly.
- *target:* An address searched through a process of address resolution or the address of the first hop obtained through the redirection process.
- *proxy:* A router that responds to Neighbor Discovery query messages on behalf of another node—for example, in the case of mobile nodes.
- *random delay:* A delay introduced before the transmission of a packet to prevent multiple nodes from transmitting at exactly the same time.
- *cache:* A small memory area that contains information stored on a node for a given period of time.
- *global address:* A unique worldwide address.
- *tentative address:* An address whose uniqueness is verified within a link before assigning it to an interface.
- *preferred address:* An address associated with an interface whose use by upper layer protocols is allowed without limitation.
- *deprecated address:* An address associated with an interface whose use by upper layer protocols is discouraged.
- *valid address:* A preferred or a deprecated address.
- *invalid address:* An address not assigned to any interface.

- *preferred lifetime:* The period an address remains preferred—that is, the time before it becomes deprecated.
- *valid lifetime:* The period of validity of an address.
- *interface token:* A link layer interface identifier that is unique (at least) at the link layer; usually derived from the interface MAC address.
- *relay:* A node that acts as an intermediate device in the transmission of a packet between two other nodes—for example, between client and server.
- *agent:* A server or a relay.

6.2 Link Types

Neighbor Discovery problems are strictly related to links belonging to one of the following classes:

- *point-to-point:* A link that connects exactly two interfaces. The Neighbor Discovery protocol deals with these links as a particular case of multicast links.
- *multicast:* A multiple access link that supports a native mechanism for sending packets to all nodes (or to a subset) by a single link layer transmission. The Neighbor Discovery protocol is implemented on this type of link according to the specifications of RFC 1970[1], which is discussed in this chapter.
- *Non Broadcast Multiple Access (*NBMA*):* A multiple access link that does not support the transmission of a packet to all stations using multicast or broadcast modalities; examples of NBMA links are X.25, Frame Relay, and ATM. This type of link supports only Redirect, Neighbor Unreachability Detection, and next hop identification functions. Other functions are specified by other standards; see, for example, Chapter 9 about IPv6 on ATM networks.
- *shared media:* A type of link that allows direct communication among a number of nodes. Attached nodes are configured without a complete list of prefixes; for this reason, different nodes, connected to the same shared medium, can ignore neighbors. Examples of shared media are SMDS and B-ISDN. The Neighbor

Discovery protocol exploits the extended semantics of the ICMPv6 Redirect message (see Section 6.3.3).

- *variable MTU:* A type of link that does not have a well-defined MTU. The Neighbor Discovery protocol simplifies its management by standardizing the MTU of all nodes connected to the same link.
- *asymmetric reachability:* A type of link in which packets from node A reach node B, but packets from node B don't reach node A. At present, the Neighbor Discovery protocol limits itself to identifying asymmetric reachability situations, and IPv6 does not use those links.

Note that all the types of links cited here (also NBMA ones) must provide IPv6 with a multicast service; if they cannot support the service natively, they can emulate it through a server (see Figure 6-1). Moreover, it is not yet clear whether IPv6 will use the emulated multicast service on nonmulticast links or whether it will prefer other alternative solutions to implement the Neighbor Discovery service (see Chapter 9).

6.3 Neighbor Discovery Service

The Neighbor Discovery Service uses five types of ICMPv6 messages: Router Solicitation (see Section 5.5.4), Router Advertisement (see Section 5.5.5), Neighbor Solicitation (see Section 5.5.6), Neighbor Advertisement (see Section 5.5.7), and Redirect (see Section 5.5.8).

6.3.1 Router and Prefix Discovery

The Router Discovery function is used to identify which routers are connected to a given link, and to learn prefixes related to the link and parameters to be used in the node's autoconfiguration procedure described in Section 6.7.

Routers periodically, or in response to a solicitation, send multicast Router Advertisement messages to announce their reachability to the nodes on the link (see Figure 6-2). Each host receives Router Advertisement messages from all routers connected on its links and builds a list of default routers (routers to be used when the path to a destination is unknown). Routers generate Router Advertisement messages frequently enough so that hosts learn of their presence within a few minutes but not so frequently to be used to detect unreachability problems (for example,

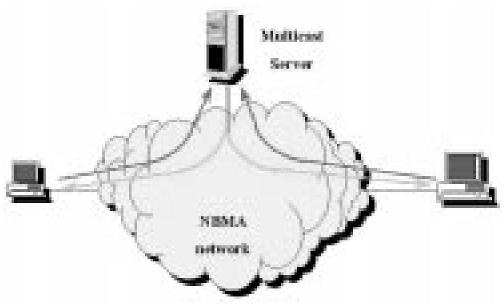

Figure 6-1
Emulation of a multicast service through a server

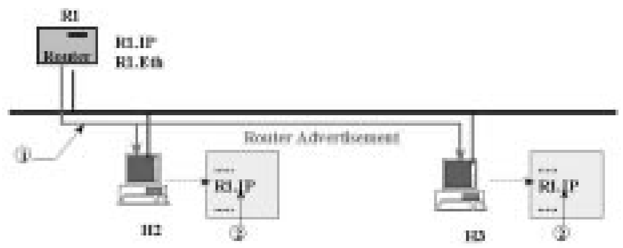

Figure 6-2
Router Advertisement

in the case of errors). Unreachability problems are handled by the Neighbor Unreachability Detection procedure, which is discussed in Section 6.6.

Router Advertisement messages contain a list of prefixes used to determine the on-link reachability. Hosts use prefixes extracted from Router Advertisement messages to determine whether a destination is on-link and can therefore be directly reached, or whether it is off-link and can therefore be reached only through a router. Note that a destination can be on-link even if it is not covered by prefixes learned through Router Advertisement messages; in this case, the host considers the destination as off-link, and the router sends a Redirect message to the sender.

Router Advertisement messages contain a set of flags that allow routers to inform hosts how to perform the address's autoconfiguration. For example, routers can specify whether hosts must use a stateful (based on DHCP servers) or a stateless—that is, autonomous—autoconfiguration procedure without resorting to servers. These procedures are described in Section 6.7.

Moreover, Router Advertisement messages contain parameters to facilitate a centralized administration of the network—for example, the default value for the Hop Limit parameter to be used in packets generated by hosts, or the link MTU.

Hosts can request routers to transmit Router Advertisement messages immediately through a Router Solicitation message, speeding up the configuration phase in this way.

6.3.2 The Address Resolution

IPv6 nodes accomplish the resolution of IPv6 addresses into link layer addresses through Neighbor Solicitation and Neighbor Advertisement messages. In IPv4, this problem is separately treated by the ARP protocol[2], which doesn't exist any longer in IPv6 and whose functions are included in ICMPv6.

A node activates the address resolution procedure (see Figure 6-3) by multicasting a Neighbor Solicitation packet that requests the target node to return its link layer address. Neighbor Solicitation messages are multicast to the solicited-node multicast address (see Section 4.8.1) associated with the target address. Starting from this multicast address, IPv6 algorithmically computes a multicast link layer address; this process happens in different ways depending on the type of link (see, for example, Section 6.8.4). The target returns its link layer address in a unicast Neighbor Advertisement message. A pair of messages is sufficient for both the initiator and the target to resolve each other's link layer address. In fact, the initiator node includes its link layer address in the Neighbor Solicitation message.

The Neighbor Solicitation message is also used in the Duplicate Address Detection procedure (see Section 6.7.4) to determine whether the same unicast address has been assigned to more than one node.

Moreover, the Neighbor Solicitation is used in the Neighbor Unreachability Detection procedure (discussed in Section 6.6) to detect whether a node is reachable. This process requires the positive confirmation that packets have been received by the node. This confirmation can be provided from upper layer protocols that confirm that a connection is "progressing" —that is, that transmitted data has been correctly delivered. When positive confirmation is not generated from upper-layer protocols, a node sends Neighbor Solicitation messages to the target node, which has to confirm its reachability through a Neighbor Advertisement message.

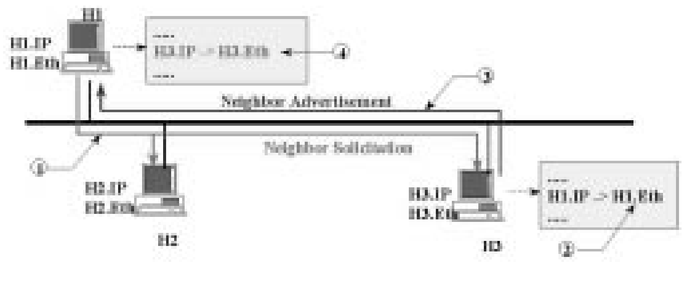

Figure 6-3
Redirect function

6.3.3 Redirect Function

When a packet must be transmitted to an off-link destination, choosing the router through which the packet will be routed is necessary. The choice cyclically falls on all reachable routers. The chosen router is the next hop to which the message will be transmitted. The chosen next hop is not necessarily the best one. For this reason, the router can generate a Redirect message to inform the source node of the presence of a router that represents a better next hop toward a specific destination.

Let's consider, for example, the network shown in Figure 6-4. Suppose that node A must transmit a packet to node H. Node A examines its Default Router List and randomly chooses the R2 router to which it transmits the packet. R2 routes the packet toward R1 and then on toward R3, which delivers the packet.

Node R2 has, however, retransmitted the packet on the same link on which it received the packet. This process causes the generation of a Redirect message, which R2 sends to A, identifying the router R1 as the best next hop toward H; A uses this information for next packets addressed to H, which are directly sent to R1.

In IPv6, the Redirect message has another use if compared to IPv4; it is similar to the one specified in the XRedirect proposal[4]. When an IPv6 node receives a Redirect message, it always assumes that the next hop is on-link; therefore, it executes a procedure to translate the IPv6 address into a link layer address. This capability allows, for example, hosts belonging to different subnets on the same link to exchange messages directly, passing through the router only for the first packet. This capability is particularly important for shared media links.

6.3.4 Other Functions

The Neighbor Discovery procedure also handles the following situations:

- *Link layer address change:* A node that knows its link layer address has changed can send a few unsolicited Neighbor Advertisement messages to update information quickly in hosts' cache memories. This function is used to improve the network performances because, as time passes, the change will be learned anyhow through the Neighbor Unreachability Detection procedure.
- *Inbound load balancing:* Nodes with multiple interfaces can balance the load among different interfaces on the same link.

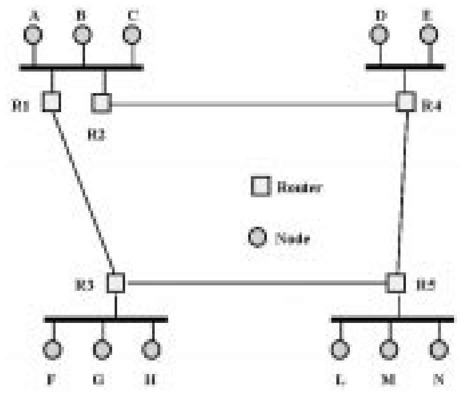

Figure 6-4 Example of a network

- *Anycast addresses:* The Neighbor Discovery procedure is ready to receive multiple Neighbor Advertisement messages for the same target because many different nodes can be configured as belonging to the same anycast address on the same link. Neighbor Advertisement messages for anycast addresses are marked as being "nonoverride" advertisements; a procedure specifies the Neighbor Advertisement to be considered valid.
- *Proxy advertisement:* A router willing to accept packets on behalf of another node that is unable to respond can issue non-Override Neighbor Advertisement messages. At present, the use of proxy advertisements is not standardized, but clearly, this use may have applications—for example, for mobile hosts.

6.4 Data Structures of a Host

One of the principles on which IPv6 design is based is that hosts must correctly work even if they have a very limited vision of the network. In fact, hosts, unlike routers, do not store the routing table (see Section 2.6) and may not have any permanent configuration. This means that, during the bootstrap, they must autoconfigure themselves; then they must learn a minimum set of information only about destinations with which they exchange data. This information is stored in memory in a set of small data structures called *caches*. These data structures are technically arrays of records, and each record will be referred to as an *entry* in the following text. Information stored in each entry has a limited period of validity, and obsolete entries are periodically purged from caches to limit the sizes of caches themselves.

RFC 1970[1] describes a possible implementation based on the four types of caches, specifying that actual implementations can choose different organizations of the cache (for example, by aggregating two or more of them in a unique cache). Caches are partly present also on routers, where, however, the main data structure remains the routing table. In the following subsections, we will analyze the different roles of the four caches, which we already mentioned in Section 2.8.

6.4.1 Neighbor Cache

The Neighbor Cache contains one entry for each neighbor to which the node has recently sent any traffic. Each entry contains an on-link unicast IPv6 address, the associated link layer address, a flag indicating whether the neighbor is a router, and a pointer to packets waiting to be transmitted. Moreover, each entry contains the state information used by the Neighbor Unreachability Detection algorithm (see Section 6.6).

6.4.2 Destination Cache

The Destination Cache contains one entry for each destination to which the node has recently sent some traffic. Entries contain a unicast IPv6 address and a pointer to the entry of Neighbor Caches that contain the address of the node that is the next hop toward the destination. Destination Cache entries are updated by Redirect messages sent by routers. Nodes can store additional information such as the Path MTU.

The main difference between this cache and the one previously described is the fact that the Destination Cache contains one entry for each destination, either on-link or off-link; whereas the Neighbor Cache contains entries only for on-link destinations.

6.4.3 Prefix List

The Prefix List contains one entry for each on-link prefix, and it is used to define whether an address is on-link or off-link. Prefix List entries are created from information received in Router Advertisements. These messages also specify the temporal validity that can be either limited or unlimited. The link local prefix belongs to the Prefix List with an unlimited validity.

6.4.4 Default Router List

The Default Router List contains one entry for each router that can be used as a default router. Default Router List entries contain a pointer to Neighbor Cache entries that contain IPv6 and link layer addresses of default routers and state flags. The algorithm for selecting a default router can favor these entries whose state indicates that the router is known to be reachable. Moreover, Default Router List entries have an invalidation time value extracted from Router Advertisement.

6.4.5 An Example of a Cache

Figure 6-5 shows a simplified example of the content of a host's caches. In particular, with reference to Figure 6-4, host A's caches are shown.

The Default Router List contains two pointers to the Neighbor Cache relevant to the two routers present on LANs R1 and R2.

The Destination Cache contains, for all destinations with which communications are active, the next hop—that is, the pointer to a neighbor. In the case of destination C (on-link), the neighbor is C itself; in the case of other destinations (off-link), the neighbor is R1 or R2. In the case of off-link destinations, Destination Cache pointers have been optimized by Redirect messages.

The Neighbor Cache contains associations between IPv6 addresses and link layer addresses for neighbors toward which communications are active. Note that node B has no association; in fact, even if it is connected to the same LAN as A, B is not exchanging packets with A.

The Prefix List contains prefixes associated with the LAN on which node A is connected. The first prefix is the link local one (see Section 4.6.4); the second one is the site local relevant to subnet 3 (see Section 4.6.5); and the third one is a provider-based address (see Section 4.6.2).

6.4.6 Possible States Associated with Entries

Neighbor Cache entries have associated states that can be one of the following:

- *Incomplete:* The entry has been created, but the link layer address

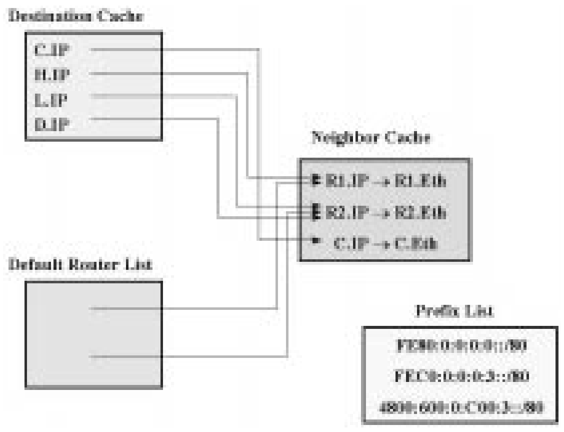

Figure 6-5
Example of a cache

has not yet been determined because the address resolution is in progress.

- *Reachable:* The entry is known to have been reachable recently.
- *Stale:* The entry is unknown to have been reachable recently, but until traffic is sent to the neighbor, no attempt should be made to verify its reachability.
- *Delay:* The entry is unknown to have been reachable recently, and the traffic has been sent to the neighbor. In this state, Neighbor Solicitation packets (called *probes*) are delayed for a short time to give upper layer protocols a chance to provide neighbor reachability confirmation.
- *Probe:* The neighbor reachability is very uncertain, and probe messages have been sent to verify reachability.

6.5 Transmission Algorithm of a Packet

When a node is sending a packet to a destination, it uses a combination of the Destination Cache, the Prefix List, and the Default Router List to determine the IP address of the appropriate next hop. After this operation, the node consults the Neighbor Cache to determine the link layer address of that neighbor.

The next hop determination for an IPv6 unicast address operates as follows. The sender performs a longest prefix match by using prefixes stored in the Prefix List to determine whether the packet destination is on-link or off-link. If the destination address is on-link, the next hop address is the same as the destination address; otherwise, the sender selects a router from the Default Router List as the next hop. If the Default Router List is empty, the sender assumes that the destination is on-link.

The next hop determination is stored in the Destination Cache and used for next packets. In particular, when a node has a packet to send, it first examines the Destination Cache, and only if no relationship exists in the Destination Cache, it activates the procedure for the next hop determination.

After the IPv6 address of the next hop node is known, the sender examines its Neighbor Cache to determine the link layer information, mainly the link layer address. If no entry exists for the IPv6 address of the next hop, the node does the following:

- Creates a new entry and sets its state to Incomplete
- Initiates the address resolution (see Section 6.3.2)
- Queues packets for transmission

When the address resolution ends, the link layer address is available and can be stored in the Neighbor Cache. At this point, the entry assumes the new Reachable state, and queued packets can be transmitted.

For multicast packets, the next hop is always considered to be on-link. The procedure for determining the link layer address corresponding to a multicast IPv6 address depends on the type of the link; for example, the case of Ethernet networks will be described in Section 6.8.4.

Each time a Neighbor Cache entry is accessed to transmit a unicast packet, the sender checks related reachability information according to the Neighbor Unreachability Detection algorithm presented in Section 6.6. This check might result in the transmission of probes to verify the neighbor reachability.

In case a neighbor becomes unreachable, the next hop determination procedure may be performed again to verify whether another path toward the destination is available. For an off-link destination in a partially meshed network, this is possible. For example, let's consider a case in which the LAN's egress router has an error, but an alternative backup router is present on the LAN. Another example is represented by the possibility of rerouting traffic destined for mobile nodes.

6.6 Neighbor Unreachability Detection

A node can be unreachable for numerous reasons. These reasons range from hardware failure, to the lack of power, to network problems, and so on. If the problem concerns the end nodes of the communication, no recovery is possible, and the communication fails. On the other hand, if the problem concerns the path between two nodes, then an alternative path may exist, and it allows the communication to be continued without upper layers detecting any change. For this reason, all nodes continuously check the reachability of neighbors to which packets are sent by using the Neighbor Unreachability Detection procedure.

This procedure is used between hosts and hosts, between hosts and routers, and between routers and hosts. It can also be used between routers and routers, but it turns out to be useless because protocols to compute routing tables such as RIP[6] and OSPF[7] already perform equivalent functions.

Neighbor Unreachability Detection is also called *black hole* detection because it is used to identify particular nodes on the network that discard packets without signaling it in any way.

We have already seen that the main source of reachability confirmations are upper layer protocols, and in particular the TCP[8] protocol that, being a connection-oriented protocol, is able to probe whether the connection continues to transmit data. However, in many cases, the reachability information cannot come from upper layers. For example, all UDP-based applications[9] cannot provide these confirmations because UDP is a connectionless protocol. Another example is represented by a router sending messages to a host because the router doesn't process upper layer packets.

When the confirmations described in the preceding paragraph are not available, a node must send *probes*, or Neighbor Solicitation packets. Let's suppose that node A has doubts about the reachability of node B. A sends probes to B and waits to receive Neighbor Advertisement packets with the flag S (Solicited) set (see Section 5.5.7). These packets are considered reachability confirmations for node B. Note that possible Neighbor Advertisement messages with the flag S clear received by B indicate only that the transmission from B to A is working properly, but these messages give no information about the transmission from A to B; therefore, they cannot be considered reachability confirmations.

Neighbor Unreachability Detection operates in parallel to packet transmission, and it is activated only in the presence of traffic. When one entry of the Neighbor Cache is Reachable, but 30 seconds have elapsed since the last reachability confirmation, that entry enters the *Stale* state. In the Stale state, at the moment of the transmission of the first packet toward the neighbor associated with the entry, a 5-second period begins (the *Delay* state), at the end of which the entry enters the *Probe* state. In the Probe state, three probe packets are sent, one per second. At the end of the transmission of the probe packets, if the reachability has not been confirmed, the entry is deleted and will be created again by the first packet through the Address Resolution procedure (see Section 6.3.2).

Receipt of a reachability confirmation brings the entry back to the Reachable state. The time periods cited here can be changed; the times reported are default values.

6.7 Address Autoconfiguration

The two types of Address Autoconfiguration procedures are *stateless* and *stateful*. The *stateless* type is the integrating part of IPv6 and is described in RFC 1791[3]. The other type, stateful, is based on the *Dynamic Host Configuration Protocol* (DHCP) and is described in a Draft of the IETF[10]. The purpose of these procedures is to solve two problems, better known by the following expressions:

- *Dentist's office:* Dentists are supposed to be rich enough to afford the purchase of many computers, but they don't know anything about computer networks. Therefore, they simply take PCs out of their boxes, connect cables, and expect the network to work.
- *Thousand computers on the dock:* The recurring nightmare of network administrators is that a thousand new PCs are delivered on Friday, and they have to be installed as soon as possible (spoiled weekend!). In this case, the know-how isn't lacking, but to meet the strict deadline, the network must nearly autoconfigure itself.

IPv6 has, among its specifications, the capability to succeed in solving both the situations described here by using the Address Resolution procedures.

6.7.1 Stateless Autoconfiguration

The stateless autoconfiguration procedure has been designed to meet the following requirements:

- Before connecting an IPV6 host to the network, no form of manual configuration must be required. A mechanism must be in place to create a unique address automatically for each interface, starting from the interface token (almost always the interface link layer address).
- Small LANs consisting of some hosts connected to a link must not require the presence of a stateful server (DHCP) or of a router to communicate. They must be able to configure with link local addresses automatically and to use these addresses for communications.
- Big company networks consisting of hosts and routers must not require the presence of stateful servers (DHCP) to communicate. Hosts must be able to derive site local or global addresses from Router Advertisements that must contain lists of prefixes associated with links.
- The stateless configuration procedure must simplify renumbering operations (change of addresses). In fact, renumbering operations will be periodically executed because global addresses are usually provider based (see Section 4.6.2). Transition periods will be required to allow the coexistence of new addresses with old addresses to make the migration painless.
- Network administrators should be allowed to specify if they will use the stateless configuration, the stateful configuration, or both of them.

After we consider these requirements, let's discuss how an interface can autoconfigure itself. First, we must determine that only multicast-capable interfaces (those able to transmit multicast packets) can autoconfigure themselves, and therefore the autoconfiguration is present only on links that support multicast traffic. When an interface is activated (either just turned on or following a reset), the host generates a link local address for the interface (see Section 4.6.4) by deriving it from the interface token. The generated address is not immediately assigned to the interface, but it is set in a *tentative* state, and a Duplicate Address Detection is started (see Section 6.7.4) to check that the link local address is not already in use. If the procedure confirms that the address is unique, the address will be assigned to the interface.

At this point, the interface has a link local address, and the first autoconfiguration step, which is executed both by hosts and by routers, ends. The following steps will be executed only by hosts.

The following step consists of obtaining a Router Advertisement message or of verifying that no routers are available on the network. We have already seen that routers periodically send Router Advertisements (see Section 6.3.1 and Section 5.5.5), but the interval between two Router Advertisements is very long. Therefore, the interface can decide to send one or more Router Solicitations to the All-Router (FF02::2) multicast address.

Router Advertisements contain two flags (see Section 5.5.5) that indicate the type of autoconfiguration to be executed. The flag M (Managed address configuration) indicates whether the host must use the stateful autoconfiguration for addresses. The flag O (Other stateful configuration) indicates whether the host must use the stateful autoconfiguration for other information (except addresses).

Moreover, Router Advertisements contain prefixes to be used for the stateless autoconfiguration of site local and global addresses. Remember that stateless and stateful procedures are not mutually exclusive; they can be used in parallel by a host to autoconfigure both stateless derived addresses and stateful derived addresses.

As Router Advertisements are also periodically generated, a host's addresses are continually updated. New addresses can be added as a consequence of the proliferation of new prefixes, and old addresses can become invalid as they are no longer announced by any router.

6.7.2 Site Renumbering

Site renumbering is an unwelcome operation for network administrators because the process is usually complicated and susceptible to faults. This operation is undoubtedly very simplified in IPv6.

At present, upper layer protocols such as TCP identify connections by also using the IP address; therefore, a change of address cannot be executed without interrupting connections in progress. To understand this point, we must know that addresses are divided into two categories: valid addresses and invalid addresses. Valid addresses are further subdivided into two subcategories: preferred addresses and deprecated addresses.

When upper layer protocols have to open a new connection, they must always use a preferred address. When network administrators start a renumbering procedure, they first insert new prefixes in routers (prefixes that will be used to form new addresses), and then they wait for the DNS to propagate these prefixes in the whole network (an operation that can require several days). At this point, the administrators remove old prefixes (prefixes of addresses that will be used no longer). This operation creates new preferred addresses on all interfaces and transforms some addresses that were previously preferred into deprecated addresses. An address remains in the deprecated state for a reasonable period of time (for example, for several days) to allow all connections that were open when the address was preferred to be closed correctly. Note that a deprecated address is valid anyhow, and it can be used with the only limitation being that new connections cannot be opened by using a deprecated address. Eventually, the deprecated address becomes invalid, and the transition from old addresses to new ones ends. In the transition phase, routers must announce both addresses (see Chapter 7).

6.7.3 DHCPv6 and Stateful Autoconfiguration

At the time this book was written, the stateful autoconfiguration procedure is not yet a standard; however, an Internet Draft on DHCPv6[10] is in an advanced state, and a standard version of DHCP for IPv4 is already available[11].

Dynamic Host Configuration Protocol version 6 (DHCPv6) is designed to provide clients (IPv6 nodes) with configuration information that is stored on a server. In the following text, DHCPv6 will be indicated as *DHCP*, omitting the version number. The information provided by DHCP mainly concerns IPv6 addresses, but other parameters can be provided, too. Therefore, DHCP is based on a client-server paradigm, in which servers manage the addresses and network parameters database and provide them to clients that choose a stateful configuration procedure.

Because, in a complex network, having a server for each link is impossible, DHCP introduces the concept of *relay*—that is, of a node that operates as an intermediary in the transmission of a packet between a client and a server. It also introduces the concept of *agent*, which can be a server or a relay. Moreover, storing the configuration information on many DHCP servers must be possible, in order to increase the reliability and the performance of the network itself.

The DHCP protocol is based on a *User Datagram Protocol* (UDP)[9] transport. In particular, DHCP agents transmit all messages to clients by using the port UDP 546 and receive all messages from the port UDP 547. Messages exchanged by the DHCP protocol are subdivided into the following six types:

- *DHCP Solicit:* This type is a message sent by the client to the multicast address of all DHCP Server/Relay agents (FF02::C); it is used when a new client doesn't know any DHCP server or it wants to locate a new server.

- *DHCP Advertise:* This type is a unicast message sent by an agent to a client in response to a DHCP Solicit message.
- *DHCP Request:* This type is a unicast message from a client to a server to request parameters for the network configuration.
- *DHCP Reply:* This type is a unicast message sent by a server to a client in response to a DHCP Request message. It contains the indication of resources (for example, addresses and parameters) that the server allocated for the client.
- *DHCP Release:* This type is a unicast message sent by the client to the server to inform that the client released certain resources (which, therefore, the server can reallocate to other clients).
- *DHCP Reconfigure:* This type is a unicast message sent by the server to notify the client about some modifications on the network. The client must acquire modifications by sending a DHCP Request message and waiting for the DHCP Reply message.

The types of messages listed here reveal the DHCP protocol to be a Request / Response protocol on an unreliable communication channel like the one supplied by UDP on IPv6. A pair of DHCP Request and DHCP Reply messages is also indicated with the term *transaction*.

When a client decides to use a stateful configuration procedure (for example, because doing so is specified in the Router Advertisement message), it first has to discover the address of a DHCP server (that can be on another link). To discover this address, the client sends a DHCP Solicit multicast message on its link, and a server or a relay responds with a DHCP Advertise message. The DHCP Advertise message contains one or more IPv6 unicast addresses of DHCP servers.

At this point, the server can acquire configuration parameters by sending to the selected DHCP server a DHCP Request message and obtaining in response a DHCP Reply message. Note that, because the communication channel is unreliable, both the request message and the reply message can be lost or can be delivered corrupted. In this case, the client simply retransmits one or more DHCP Request messages until it obtains a valid DHCP Reply message.

If the server must reconfigure the client, it doesn't do so directly, but it requests the client to start a transaction through a DHCP Reconfigure message. In this way, reliability mechanisms supplied by the Request/Response philosophy become valid again.

6.7.4 Duplicate Address Detection

The Duplicate Address Detection procedure is used for all unicast addresses, either written manually or obtained through a stateful or stateless procedure. However, it must never be used for anycast or multicast addresses.

The Duplicate Address Detection procedure uses a Neighbor Solicitation packet sent to the tentative address to check whether the tentative address is already present on the link. In fact, if the address is unique, no response will be made to the Neighbor Solicitation message; whereas, if the address is already being used, the node using the address will respond with a Neighbor Advertisement packet.

In this second case, the tentative address will not be used, and the network administrator will have to resolve the conflict manually, typically by configuring a different interface token on one of the two nodes.

The default configuration sees that only one Neighbor Solicitation packet will be sent and that it will wait for the Neighbor Advertisement for one second. Default values can be different for different types of links.

6.8 IPv6 on Ethernet

In this section, we will analyze some problems related to the transmission of IPv6 packets on Ethernet. This case has been chosen as extremely important based on the current market and because it is already standardized by RFC 1972[5].

6.8.1 Frame Format

The first thing to be standardized, when we want to decide how to transport IPv6 on a certain type of link, is the enveloping of the IPv6 packet within the frame (link layer envelope). In our case, we must standardize how to envelope IPv6 packets in the Ethernet frame. The solution adopted by RFC 1972[12] was presented in Section 2.9 and shown in Figure 2-6(a). As the data field length in an Ethernet envelope must be less than or equal to 1500 octets, the IPv6 MTU on Ethernet is by default equal to 1500, and it can be decremented through manual configurations.

6.8.2 Link Local Addresses

A second point to be standardized is what to use as an interface token—that is, as a unique identifier of the node (at least) inside the link. In the case of Ethernet, the choice is obvious: The 48-bit MAC address is used as the interface token. At this point, the construction of the link local address can be standardized, and it turns out to be

```
FE80::XXXX:YYYY:ZZZZ
```

where `XXXX:YYYY:ZZZZ` indicates the interface MAC address. This result is based on the description in Section 4.6.4, illustrated by Figure 4-8.

6.8.3 Link Source/Target Addresses

Neighbor Discovery packets need to include link layer addresses (in this case, MAC addresses) within the Link Source/Target Address option (see Section 5.5.10).

These addresses are used, for example, in the Neighbor Advertisement message during address resolution. In this case, the solution is obvious; it is shown in Figure 6-6. Here Type = 1 indicates a Source Link Layer Address, and Type = 2 indicates a Target Link Layer Address. The value of the Length field is 1.

6.8.4 Multicast Addresses

In conclusion, deciding how to map IPv6 multicast addresses and Ethernet addresses is necessary[12]. With this aim, the IETF registered at IEEE all OUIs beginning with 33-33 (hexadecimal). The mapping is implemented as shown in Figure 6-7, where DST13, DST14, DST15, and DST16 are the octets 13, 14, 15, and 16, respectively, of the IPv6 destination address.

Note that this type of mapping considerably improves performance as network boards can filter almost all the multicast traffic that doesn't interest the node, and this process wouldn't be possible if all the multicast traffic were sent to the same link layer address.

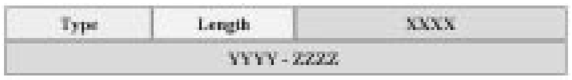

Figure 6-6
Link Source/Target Address Option

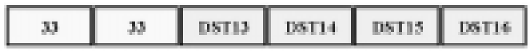

Figure 6-7
IEEE 802 Multicast address obtained from an IPv6 address

REFERENCES

[1]T. Narten, E. Nordmark, W. Simpson, *RFC 1970: Neighbor Discovery for IP Version 6 (IPv6)*, August 1996.

[2]D.C. Plummer, *RFC 826: Ethernet Address Resolution Protocol: On converting network protocol addresses to 48 bit Ethernet address for transmission on Ethernet hardware*, November 1982.

[3]S. Thomson, T. Narten, *RFC 1971: IPv6 Stateless Address Autoconfiguration*, August 1996.

[4]B. Braden, J. Postel, Y. Rekhter, *RFC 1620: Internet Architecture Extensions for Shared Media*, May 1994.

[5]M. Crawford, *RFC 1972: A Method for the Transmission of IPv6 Packets over Ethernet Networks*, August 1996.

[6]G. Malkin, *RFC 1723: RIP Version 2—Carrying Additional Information*, November 1994.

[7]J. Moy, *RFC 1583: OSPF Version 2*, March 1994.

[8]J. Postel, *RFC 793: Transmission Control Protocol*, September 1981.

[9]J. Postel, *RFC 768: User Datagram Protocol*, August 1980.

[10]IETF Draft, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, August 1996.

[11]S. Alexander, R. Droms, *RFC 1533: DHCP Options and BOOTP Vendor Extensions*, October 1993.

[12]S. Gai, P.L. Montessoro, P. Nicoletti, *Reti Locali: dal Cablaggio all'Internetworking*, SSGRR (Scuola Superiore G. Reiss Romoli), 1995.