PART TWO

MOBILE IPv6

03Soliman.qrk 3/10/04 3:20 PM Page 81

T H R E E

Mobile IPv6

Consider a scenario where you had to change your place of residence on a semi-permanent basis, for instance, due to relocation of your company. One problem (of many) that you would need to solve is how you would continue to be reachable by your family, friends, bank, among other organizations that need to send you mail. In most countries, the post office can redirect your mail to your new residence for a fee. After the redirection request is processed by the post office, all mail addressed to your original residence will be forwarded to your new residence. On a very high-level, this is essentially how Mobile IPv6 provides reachability. Mobile IPv6 allows devices to be reachable and maintain ongoing connections (e.g., FTP, video, or audio) while moving within the Internet topology. To be reachable, mobile nodes are provided with a permanent IP address, called the *home address*. To maintain ongoing connections while moving, Mobile IPv6 uses the redirection function provided by the post office in our example; the Mobile IPv6 equivalent of the post office is called the *home agent*. The home agent redirects packets addressed to a mobile node's home address to its current location. Mobile nodes always update their home agent with their current location (i.e., every time they move).

In addition to reachability and maintaining ongoing connections, the protocol allows for optimal routing between mobile nodes and other nodes they are communicating with. That is, Mobile IPv6 allows a mobile device to communicate with another peer using the shortest possible path provided by IP routing.

In this chapter, we discuss the design details for Mobile IPv6 and the reasons behind the choices that were made during the protocol design. The following chapters discuss the security aspects of the protocol and the reasons for the current design. We start by discussing the basic operations required to allow mobile nodes to change their point of attachment within the Internet without dropping ongoing connections and while being reachable by potential correspondents. Following this, we show how mobile nodes can maintain knowledge about their home links (where the home address is located) while located anywhere on the Internet. Such features are intended to reduce the amount of manual configuration required in mobile nodes. Finally, we discuss how mobile nodes can optimize their communication with correspondent nodes by ensuring that packets take the shortest possible path between a mobile and a correspondent node. Before we get into the various details, we first discuss the terminology used in Mobile IPv6.

3.1 Mobile IPv6 Terminology

Mobile Node (MN): A mobile node is a node that changes its location within the Internet topology. A node's mobility could be a result of physical movement or of changes within the topology. That is, movement can be due to a device moving from one link to another because of its physical movement (e.g., a device in a car or a train) or because of changes in the topology that cause the device to be attached to a different router (e.g., router failure) while being in the same place physically. Mobile IPv6 is dedicated to host mobility; therefore, the term "mobile host" would have been more appropriate in this context. However, the Mobile IPv6 specification talks about "mobile nodes" (deceptively referring to hosts only), so we use the same term in this book to avoid further confusion.

Correspondent Node (CN): Any node that communicates with the mobile node. Note that the terms "mobile" and "correspondent" nodes refer to certain functions within an IPv6 node. Therefore, a mobile node can also be a correspondent node, and vice versa, depending on the context. For instance, a host can be seen as a correspondent node by the mobile (moving) node it communicates with. At the same time, the correspondent node might also move, which makes it a mobile node (because it is moving) while being seen as a correspondent node by its mobile peer.

Home address: A stable address that belongs to the mobile node and is used by correspondent nodes to reach mobile nodes. Like all IPv6 addresses, the home address is based on the 64-bit prefix assigned to the *home link* combined with the mobile node's interface identifier. A mobile node can have more than one home address. IP packets addressed to the home address are routed to the home link using standard routing protocols.

Home link: A link to which the home address prefix is assigned.

Home Agent (HA): A router located on the home link that acts on behalf of the mobile node while away from the home link. The *home agent* redirects packets addressed to a mobile node's home address to its current location (care-of address) using IP in IP tunneling.

Foreign link: Any link (other than the home link) visited by a mobile node.

Care-of address: An address that is assigned to the mobile node when located in a foreign link. This address is based on the prefix of the foreign link combined with the mobile node's interface identifier. There is no special format for a care-of address; it is a normal unicast IPv6 address that is configured through the stateless or stateful means described in Chapter 2. This address identifies the current location of the mobile node.

Binding: The association of the mobile node's home address with a care-of address for a certain period of time. That is, between the stable home address and the mobile node's current location. This allows the home agent (post office) to forward packets to the mobile node's current location. The binding is refreshed (if the timer expires) or updated when the mobile node gets a new care-of address (because it moved to a new link).

Binding cache: A cache stored in volatile memory containing a number of bindings for one or more mobile nodes. A binding cache is maintained by both the correspondent node and the home agent. Each entry in the binding cache contains the mobile node's home address, care-of address, and the lifetime that indicates the validity of the entry. When the binding cache is maintained by correspondent nodes, it also contains some security parameters, discussed in Chapter 5.

Binding Update List (BUL): A list maintained by the mobile node in volatile memory. This list contains all bindings that were sent to the mobile node's home agent and correspondent nodes. This list is maintained in order for the mobile node to know when a binding needs to be refreshed and is also used for selecting the right care-of address when communicating directly with a correspondent node. The details of the use of the binding update list are presented in this chapter.

3.2 Overview of Mobile IPv6

Mobile IPv6 was designed to allow nodes to be reachable and maintain ongoing connections while changing their location within the topology. To meet the requirements discussed in Chapter 1, mainly transparency to upper layers, Mobile IPv6 uses a stable IP address, assigned to mobile nodes (home address). The home address is used for two reasons: first, to allow a mobile node to be reachable by having a stable entry in the DNS, and second, to hide the IP layer mobility from upper layers. We already saw in Chapter 1 that DNS caching implies that a node that frequently changes its IP address will not have the same IP address in all DNS servers because some of them will cache an old address until the caching period for such address expires. Therefore, in order for nodes to be reachable at the right address, the address should be stable and should not change every time they move. Hence, the need for the home address provided by Mobile IPv6. A consequence of keeping a stable address independently of the mobile node's location is that all correspondent nodes try to reach the mobile node at that address, without knowing the actual location of the mobile node. That is, whether the mobile node is physically located on the home link or not, packets are forwarded to the home link. If the mobile node is not at its home link, its home agent is responsible for tunneling packets to the mobile node's care-of address (i.e., its real location).

Since correspondent nodes attempt to connect to the mobile node's home address, sockets (in the mobile and correspondent nodes) use the home address to record such connections, so it is necessary that applications (on both nodes) see only the home address for the mobile node. Therefore, the IP layer is responsible for presenting the home address to applications running on the mobile node as a source address regardless of the mobile node's location. That is, the IP layer hides the mobility (address change) from upper layers to maintain ongoing connections while the mobile node changes its address. If the address actually changed, the information in the sockets would become invalid and connections would automatically terminate.

The home address is formed by appending an interface identifier to the prefix advertised on the home link. Like any IPv6 node, a mobile node can be assigned several addresses of different scopes. This applies to both the home and care-of addresses.

While at home, a mobile node operates like any other IPv6 node. It receives packets addressed to any of its home addresses and delivered via normal routing. Mobile IPv6 is only invoked when a mobile node is not "at home"—that is, when the mobile node is located on a foreign link.

When a mobile node moves from its home link to a foreign link, it first forms a care-of address based on the prefix of the foreign link. The care-of address can be formed based on stateless or stateful mechanisms. However, this book focuses on stateless address autoconfiguration, as it is expected to be the most widely used mechanism for mobile nodes. Following address configuration, the mobile node informs its home agent of such movement by sending a *Binding Update* (BU) message. The binding update message is one of several Mobile IPv6 messages that are encoded as options in a new header called the *mobility header*. The mobility header is the last header in the chain of extension headers and appears as an upper layer protocol (see section 3.2.1).

The binding update message contains the mobile node's home address and its care-of address. The home address is included in a new option called the *home address option*, and the care-of address is included either in the source address in the IP header or in a new option called the *alternate-care-of address option*. The home address option is part of the *destination options* extension header, while the alternate care-of address option is included in the mobility header. Chapter 5 and Chapter 7 discuss some of the typical uses of the alternate-care-of address option. For now, we only need to know that the mobile node's care-of address can be included in the source address in the IPv6 header or the alternate-care-of address option.

The purpose of the binding update is to inform the home agent of the mobile node's current address (i.e., care-of address). Therefore, the home agent needs to store this information in order to forward packets addressed to the mobile node's home address. The home agent contains a *binding cache*, which contains all bindings for the mobile nodes it serves. Each entry in the binding cache stores a binding for one home address. When the home agent receives the binding update, it performs a number of actions to validate the message; if the binding update is accepted, the home agent searches its binding cache to see if an entry already exists for the mobile node's home address. If an entry is found, the home agent updates that entry with the new information received in the binding update. Otherwise, if no entry is found, a new one is created. From this point onward, the home agent acts as a proxy for the mobile node on the home link. That is, it represents the mobile node on the home link. The home agent essentially becomes the post office offering a forwarding service to the mobile mode. To ensure this representation is understood by all nodes on the home link, the home agent sends a *proxy neighbor advertisement* addressed to the all-nodes multicast address on the link. The advertisement includes the mobile node's home address in the *target address* field and the home agent's link-layer address. Hence, the home agent ensures that any IP packet addressed to the mobile node is forwarded to the home agent's link-layer address. Since the recipients of this message cannot confirm reception to the home agent, it may resend the neighbor advertisement more than once to reduce the probability of the message being lost. The home agent also defends the mobile node's addresses in case another node configures an address that collides with a mobile node's home address (or addresses). Hence, if another node tentatively configures one of the mobile node's addresses and attempts to test it using DAD, the home agent replies to the message, indicating that the address is already assigned to another node.

At this stage, the home agent starts receiving all packets addressed to the mobile node. Upon receiving a packet addressed to the mobile node's home address, the home agent checks its binding cache to see whether an entry for the mobile node exists. The home agent searches the cache by using the mobile node's home address (the destination address in the received packet) as a key to identify each entry. When an entry is found, the packet is tunneled to the mobile node's care-of address, which is included in the mobile node's binding cache entry. The *tunnel entry point* is the home agent (source address in the outer header), and the *tunnel exit point* is the mobile node's care-of address. The tunnel is bidirectional. That is, when the mobile node sends any IP packets, it tunnels them first to the home agent, which decapsulates the packet and forwards the original one to its destination. Figure 3–1 illustrates the binding update messages and data forwarding (for ongoing connections) between the mobile node and its home agent.

The Mobile IPv6 specification prohibits the home agent from tunneling packets addressed to the mobile node's link-local address. Furthermore, the specification recommends that the default behavior of the home agent should be set to not tunnel packets addressed to any of the mobile node's site-local addresses. The only exception to these rules are packets sent by or to the mobile node that are needed for multicast group membership. The Multicast Listener Discovery (MLD) protocol requires nodes to use their link-local addresses when attempting to join multicast groups. Therefore, MLD packets need to be tunneled by the home agent.

Tunneling is required to ensure the transparency of the service provided by the home agent. This is needed to preserve the end-to-end nature of IP packets exchanged between the mobile node and correspondent nodes. Recall that routers must not modify the content of the source or destination addresses in the IP header, thereby preserving the integrity of the packet and allowing for end-to-end integrity checks (e.g., Authentication Header). Furthermore, tunneling is essential to maintain transparency for upper layers. Consider the case where, instead of tunneling, the home agent rewrites the destination address in the packet (i.e., replace the home address with the care-of address). A direct result of this action is that the packet's integrity is compromised, causing the Authentication Header (if present) to fail. In addition, the mobile node receives the packet and passes it to upper layers (e.g., TCP), which process the packet assuming that the care-of address is used to identify the connection. However, the correspondent node identifies the connection by its address, source port number, the mobile node's **home** address, the mobile node's port number, and the protocol number. Hence, if the mobile node replies to the correspondent node directly, using its care-of address, the correspondent node is unable to find the connection and consequently drops the packets.

The home agent may choose to secure the tunnel to the mobile node using an Authentication Header or ESP, depending on local policies within the home network. For instance, within a corporate network containing a home agent, other nodes within the corporate will be unaware of the mobile node's movement away from the home link, as they continue to send packets to a mobile node's home address. Nodes would typically assume that within a wired corporate network, IPsec is not required (door lock security!) due to the difficulty involved in penetrating corporate security (physically) and the typical use

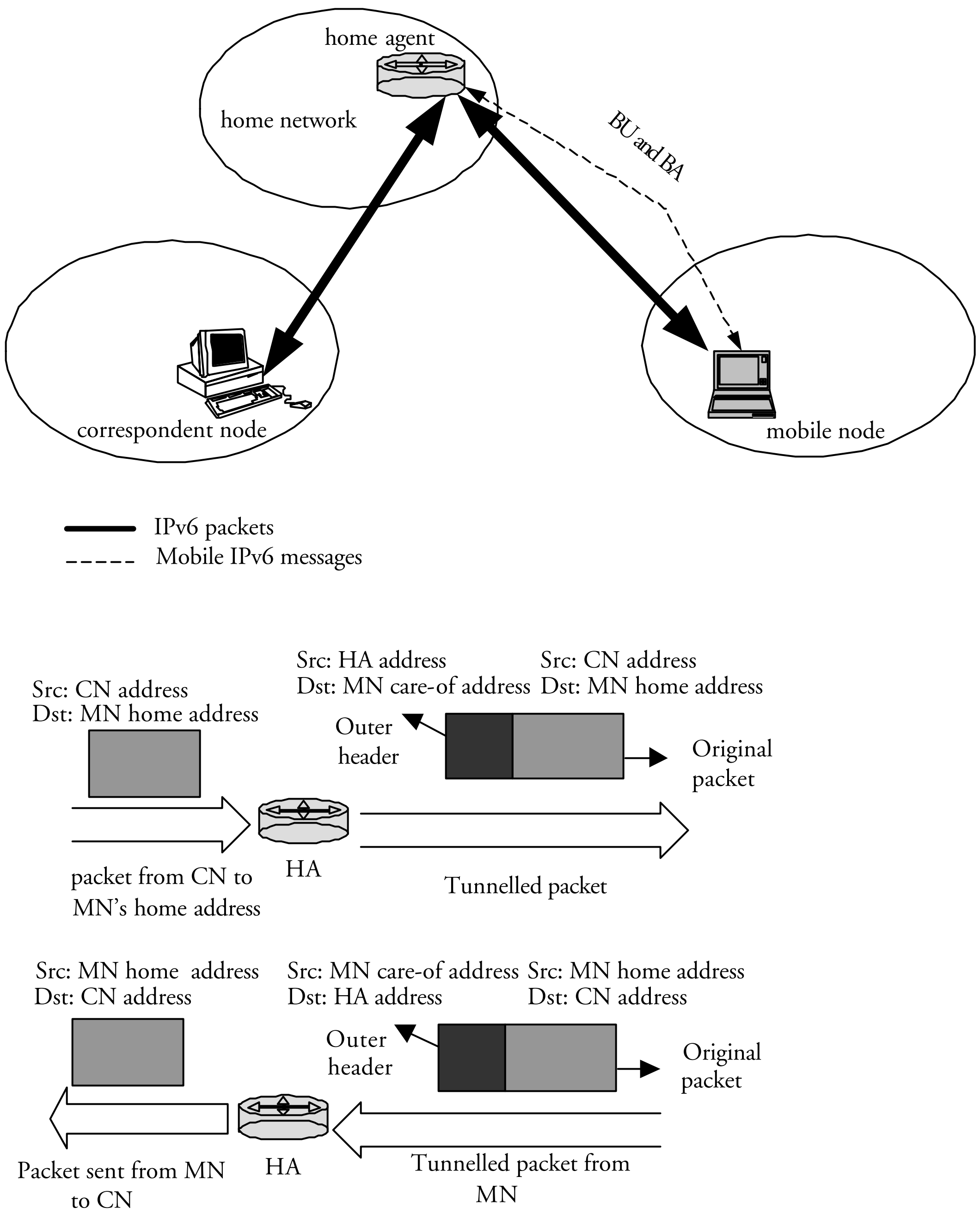

Figure 3–1 *Mobile IPv6 operation and data forwarding.*

of Firewalls on the edge of the network (more on Firewalls in Chapter 4). In this case, a home agent can be configured to use ESP to protect all traffic tunneled to a mobile node that moved away from the home link located within the corporate network.

3.2.1 Binding Updates and Acknowledgments

It is crucial for the mobile node to ensure that it has a binding with its home agent. If the binding is lost or the binding update message was not received, the mobile node will be unreachable while away from home. Hence, a reliable protocol is required to install a binding in the home agent's binding cache. The home agent is required to acknowledge the binding update message sent by the mobile node. Figure 3–2 shows the binding update message format.

Mobile IPv6 defines a new IPv6 extension header, the *mobility header*. The mobility header is used to carry all Mobile IPv6 messages. It contains four permanent fields (included in all messages): *payload proto, header length, mobility header type,* and *checksum*.

The *payload proto* field indicates the type of the following header, which makes you wonder what the difference is between this field and the next header field in all IPv6 extension headers: there is no difference, it was just named differently! According to the current specification, this field is always set to the decimal value 59, indicating that the mobility header is the last header.

The header length field includes the length of this extension header in 8-octet units, excluding the first 8 octets.

The *mobility header type* field is used as a switch to indicate which message is included in the mobility header. When a binding update is included in the mobility header, this field is set to 5. It should be noted that like all extension headers, the mobility header may carry other options after the message indicated by the mobility header type field. For instance, message type 5 (the binding update) can also include the alternate-care-of address option after the binding update message.

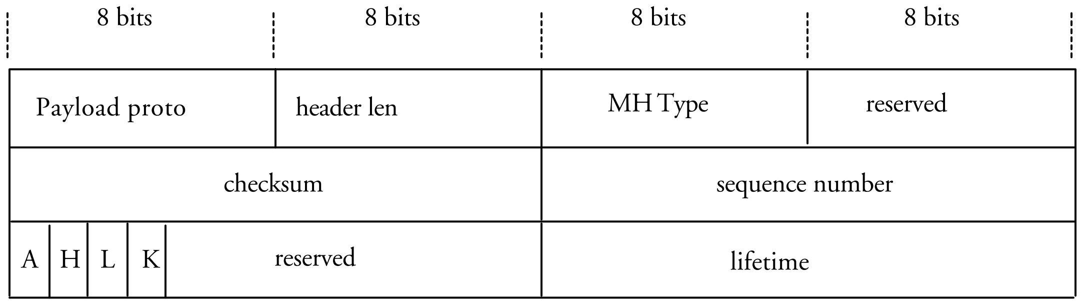

Figure 3–2 *The binding update message format.*

The *checksum* field includes a checksum value for the entire mobility header in addition to the IPv6 header. The calculation of the checksum is slightly different when the *home address option* is included in the destination options header. We discuss this when we discuss the home address option in section 3.3.

The *sequence number* field contains a 16-bit integer used to ensure that binding updates are received and acknowledged in order. The mobile node stores the value of the last used sequence number per destination. Whenever a binding update is sent to a particular destination, the sequence number is incremented. Hence, this number represents a cyclic counter used to allow a window for the expected sequence number. That is, since the sequence number space is finite, it is inevitable that it will be repeated. However, this field ensures that consecutive binding updates are received in the correct order.

The *A* flag indicates whether an acknowledgment is expected for this binding update. When sending the binding update to the home agent, the *H* and *A* flags are set to indicate that the binding update is sent to a home agent (the *H* flag) and that an acknowledgment is required (the *A* flag).

The *L* flag is used by the mobile node to indicate (when set) to the home agent that its link-local address' interface identifier is the same as that included in its home address. Hence, the home agent can defend the link-local address by appending the interface identifier to the well-known link-local prefix.

Binding updates need to be integrity-protected to avoid Bad Guy from stealing a mobile node's traffic by sending a binding update on its behalf. We discuss the security issues in detail in chapters 4 and 5. The *K* flag indicates whether the protocol used to establish a security association between the mobile node and its home agent must be rerun every time the mobile node moves. If the *K* flag is cleared, the protocol will not survive movement; that is, the security association will need to be reestablished when the mobile node gets a new care-of address. Otherwise, if the flag is set, the protocol will survive movement.

The *lifetime* is a 16-bit field that indicates the requested lifetime (in 4-second units) for the binding cache entry created by the receiver of the binding update. When this value is set to zero, it indicates a request to delete the binding cache entry for a home address. A binding cache entry remains valid for the duration of the lifetime field; if the mobile node does not refresh the binding cache entry, the entry is deleted after the lifetime expiry.

When the home agent receives the first binding update from a mobile node, it performs DAD for the mobile node's home addresses included in the binding update. If DAD fails, the mobile node is informed (when receiving the binding acknowledgment) and is unable to use the home address. Duplicate address detection is a useful tool for avoiding address collisions while the mobile node is away. For instance, while the mobile node is moving away from its home link, another node might be arriving at that home link and forming an address that collides with the mobile node's home address. Although this scenario is highly unlikely (considering the probability of collision for a 64-bit interface identifier), DAD would detect this case. A more likely case of address collision could take place if the mobile node's binding has expired and not been updated for some time (for instance, the mobile node is turned off). During this period, another node could visit the mobile node's home link and generate an address that collides with the mobile node's home address.

As a side note, it is worth mentioning that DAD is not performed when the mobile node is removing a binding cache entry (i.e., lifetime is set to zero). In this case the mobile node is sharing the link with the home agent; therefore, DAD can cause an infinite loop because both the mobile node and the home agent will defend the same addresses. We revisit this scenario in section 3.2.5.

After a successful DAD operation, and provided that there is nothing stopping the home agent from accepting the binding (e.g., wrong message format or lack of memory resources), the home agent copies the contents of the message into an existing binding cache entry or creates a new one if this was the first binding update. If a binding cache entry already exists, the home agent updates it with the contents of the new message. Hence, a binding update message overwrites an existing binding, resulting in a single entry per home address. Table 3–1 shows an example of a binding cache.

Table 3–1 *Binding Cache Entries in the Home Agent*

Home address	Care-of address	Sequence no.	Lifetime	Flags
`3ffe:200:8:1:A:B:C:D`	`3ffe:200:1:5:A:B:C:D`	11	250	A/H/K/L
`3ffe:200:8:1:D:E:F:9`	`3ffe:100:3:1:D:E:F:9`	2000	400	A/H/L

For the home agent to be able to tunnel packets to the right care-of address, the address must be contained in the binding cache entry associated with the mobile node's home address. The care-of address can be retrieved through two different fields: the source address included in the IP header or an alternate-care-of address option included in the mobility header containing the binding update. Retrieving the address from the source address field in the header is clear: the home agent can simply copy the source address in the IPv6 header, to the binding cache entry. However, in some cases (discussed in the chapters 5 and 7), a mobile node may wish to associate the binding with a care-of address other than the one included in the source address field in the IP header. For those cases, Mobile IPv6 allows the mobile node to include the care-of address in a separate option, the alternate-care-of address option. The option format is shown in Figure 3–3.

The alternate-care-of address option simply consists of an IPv6 address that the mobile node wishes to use as a care-of address. The *option type* is a number that informs the receiver about the content and format of the option. The *option length* indicates the number of bytes following this field.

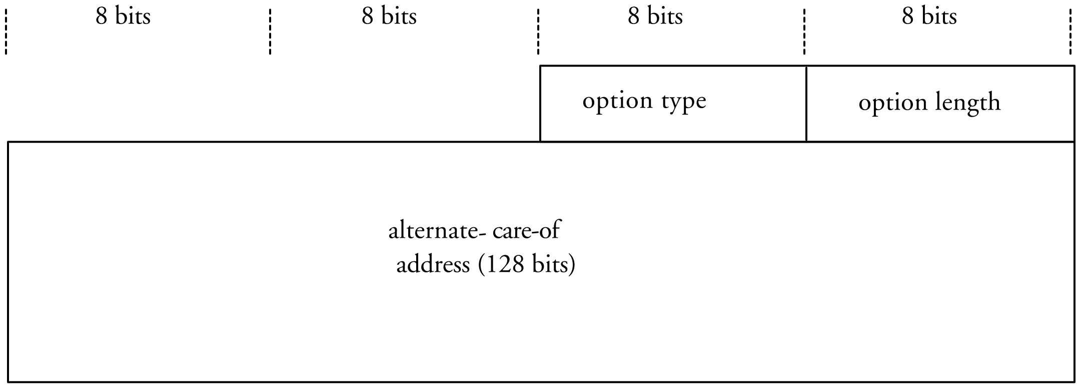

Figure 3–3 *The alternate-care-of address option format.*

Binding updates sent to the home agent must request an acknowledgment to ensure that a mobile node receives confirmation for the bindings it has established with the home agent. The *binding acknowledgment* message provides this information to the mobile node by including a *status* field that indicates the success or failure of the binding. Values below 128 indicate success; the rest of the values indicate the reasons for failure. The message also contains a *K* flag that indicates (when cleared) that the protocol run between the home agent and mobile node to secure binding updates does not survive movement.

The binding acknowledgment (Figure 3–4) synchronizes the lifetime of the binding between the mobile node and its home agent. A home agent might reduce the lifetime proposed in the binding update, subject to loading or other policies. Hence, the lifetime fields in the binding update and acknowledgment messages may not have the same values. If the lifetime of the binding is changed in the binding acknowledgment, the mobile node must store the new value returned by the home agent and ignore its own suggestion sent in the binding update message.

Finally, the binding acknowledgment contains the sequence number of the binding update being acknowledged in order to allow the mobile node to track which binding update is being acknowledged. For instance, the mobile node may have sent more than one binding update to its home agent and therefore would need to know which one is being acknowledged in order to be able to store the right values and send the next binding update with the correct *sequence number*.

Mobile nodes continue to retransmit the binding update (increasing the sequence number for each retransmission) to the home agent until an acknowledgment is received. However, such retransmission is done based on an exponential back-off algorithm. When using this algorithm, the mobile node doubles the time between retransmissions every time a binding update message

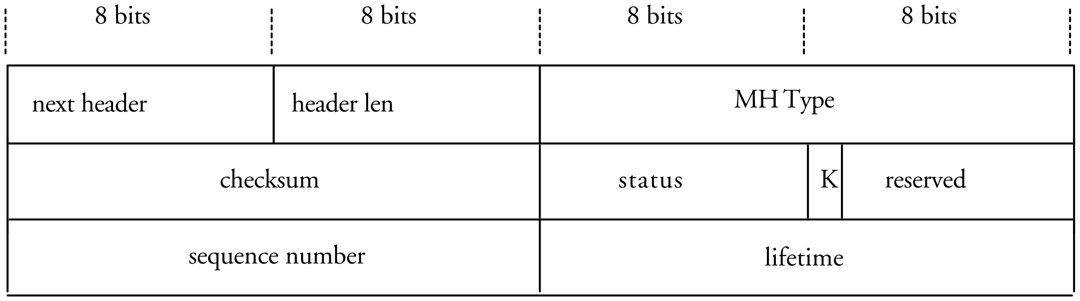

Figure 3–4 *Binding acknowledgment message format.*

is sent. The initial timeout value depends on whether the binding update sent was the first one to the home agent or a refresh of an existing binding. The first binding update would require more time, since the home agent performs DAD for the mobile node's home address (possibly 1 to 2 seconds). If the binding update were sent to refresh an existing binding, the initial timeout period is set to a default value of 1 second. The retransmission with exponential back-off (1 second, 2 seconds, 4 seconds, and so on) continues until the mobile node receives a binding acknowledgment or until a maximum timeout value (256 seconds) is reached. If the maximum timeout value is reached, a mobile node's implementation may repeat the entire algorithm or select another home agent using the Dynamic Home Agent Address Discovery (DHAAD) mechanism (described in section 3.2.7).

This retransmission algorithm is needed to minimize congestion in the network or unnecessary waste of limited bandwidth. The assumption behind this approach is that binding acknowledgments were not received because the binding update was lost (most likely due to congestion) or due to a home agent failure. Hence the need for back-off to avoid increasing the potential congestion.

After receiving the binding acknowledgment from the home agent, the mobile node needs to ensure that such information is stored to be able to refresh the binding before it expires. For this purpose, the mobile node maintains a data structure called the *binding update list*. This is equivalent to the binding cache maintained by the receivers of the binding update and contains the same information. Hence, the lifetimes in the mobile node's binding update list and the home agent's binding cache are synchronized (through the binding update and acknowledgment messages). Using the binding update list, the mobile node monitors the lifetimes of its bindings and refreshes them before they expire.

3.2.2 Refreshing Bindings

As described earlier, bindings are valid for a lifetime included in the binding update message. Mobile nodes should refresh bindings by sending another binding update before they expire or when the mobile node's care-of address changes. Mobile IPv6 allows the receiver of the binding update to request that the mobile node update its binding entry. This is done using the *binding refresh request*. This message requests that the mobile node update its current binding because it's about to expire. This message can be sent by the mobile node's home agent or a correspondent node communicating directly with the mobile node (section 3.3 talks about direct communication between the mobile and correspondent node).

The benefit of this message is not clear. Since the mobile node already knows (from its own binding update list) that it has a binding entry in its home agent's (or correspondent node's) binding cache, it should not need a reminder unless the mobile node's implementation contains bugs!

Like all Mobile IPv6 messages, the binding refresh request is carried inside the mobility header.

3.2.3 Why Reverse Tunneling?

So far we have discussed the need for tunneling from the home agent to the mobile node. To maintain the transparency of mobility to upper layers, *reverse tunneling* (from the mobile node to the home agent) is also needed. Recall that maintaining transparency to upper layers requires the mobile node to use the home address as the source address. If the mobile node sends packets directly to the correspondent node using the home address as a source address, ingress filtering causes the mobile node's packets to be dropped by the default router, since the source address is not derived from the foreign link's prefix. Ingress filtering problems are eliminated when reverse tunneling to the home agent is used. This is because the outer header (inspected by the default router in the foreign link) contains the mobile node's care-of address, which is topologically correct. When the home agent decapsulates the mobile node's packets, the source address in the inner header (the home address) belongs to the home agent's link's topology (prefix), and normal forwarding takes place.

3.2.4 Movement Detection

As discussed earlier, the mobile node should always inform its home agent of its current location (the care-of address). This is done using the binding update/acknowledgment messages. Consider a mobile node moving from one link to another; its home agent continues to forward packets to its previous care-of address until the mobile node updates it about its movement. This results in packet losses, as illustrated by Figure 3–5. Therefore, it is important for the mobile node to update the home agent as soon as movement to a new link

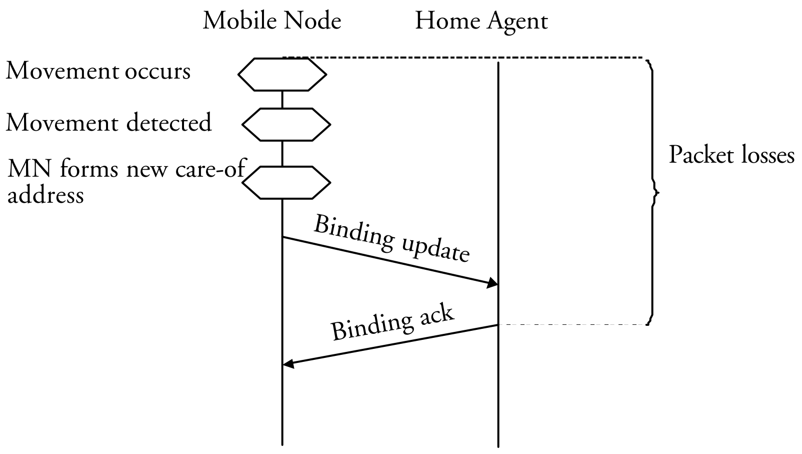

Figure 3–5 *Movement detection and associated packet losses.*

is detected. However, the mobile node needs to configure a new care-of address before it can update its home agent. This can be efficiently done using stateless address autoconfiguration. Hence, the following actions are required from the mobile node:

1. Detect movement.
2. Form a new care-of address.
3. Inform the home agent by sending a binding update containing the new care-of address.

To minimize packet losses, the first two steps need to be executed as quickly as possible. We now discuss some of the steps taken by Mobile IPv6 to accelerate movement detection.

Humans are aware of their physical movement from one location to another by simply looking around and noticing the changes in the surroundings, street names, addresses, and buildings. In a similar manner, a movement within the Internet topology implies a change of addresses. Each link has at least one unique prefix that identifies it and is advertised by one or more default routers on that link. Hence, if a mobile node notices a change in the link's prefix based on a new router advertisement, it can use that as a hint indicating that topological movement may have occurred. However, it is important to note that a new prefix advertised does not necessarily imply movement. The Neighbor Discovery specification allows routers to advertise more than one prefix on the same link. Furthermore, having one prefix option appear while another one disappears does not imply that the old prefix is no longer valid. Routers may be trying to reduce the size of a router advertisement by advertising a subset of the options whose timers are about to expire; this is perfectly legal according to Neighbor Discovery. Therefore, a mobile node can only be certain about movement when two events have taken place:

1. A new prefix has appeared on link, and
2. The current default router has disappeared.

The first step is achieved when the mobile node receives a router advertisement containing a new prefix option. However, the amount of time that passes before the reception of the router advertisement clearly depends on the frequency of the advertisements. To minimize this time (and the resulting packet losses), Mobile IPv6 relaxes the minimum interval between router advertisements to 0.05 seconds (as opposed to 3 seconds in standard Neighbor Discovery). In addition, the specification defines a new *advertisement interval* option (see Figure 3–6) for inclusion in router advertisements. This option informs mobile nodes about the maximum interval between router advertisements in milliseconds.

A mobile node receiving this option with an *advertisement interval* value of 0.1 seconds would realize that it may have moved if the router advertisement was not received on time and send a router solicitation for a new advertisement.

After noticing that a new prefix has appeared on link, the mobile node must check if it still shares a link with its default router. At this point another complication is raised: routers typically use their link-local address as a source address when sending router advertisements, which is only unique within the link. Therefore, different network operators may configure routers with the same link-local address, causing movement detection to fail (i.e., the mobile node would think that it is still attached to the same router). For this reason, the *R* flag was added to the prefix option (see section 2.6.1.2). The *R* flag informs the receivers of the option that the option includes the global address of the router (i.e., the last 64 bits are set to the router's interface identifier). Since

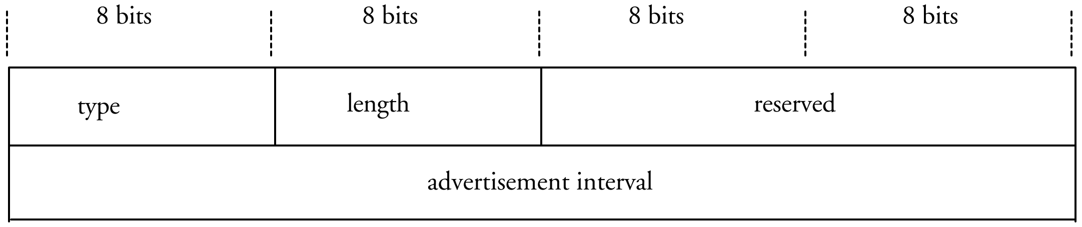

Figure 3–6 *Advertisement interval option.*

the global address is globally unique, the mobile node knows when it has changed default routers.

However, the *R* flag does not solve the problem for all configurations. Consider Figure 3–7, where a mobile node is located on a link that has more than one default router, router_1 and router_2, connected. IPv6 specifications do not require all routers on a link to advertise the same prefixes. That is, router_1 may advertise one prefix and router_2 may advertise another prefix. If the mobile node initially configures its care-of address based on router_1's advertised prefix, then later receives an advertisement from router_2, it might think that it moved when in fact it did not. The *R* flag would not help, since each router has a different interface identifier. There are two ways to avoid this scenario:

1. Configure all routers on a link to advertise the same prefixes.
2. Design a new option that should be added to the router advertisement (e.g., link identifier option). This option could include a globally unique address that identifies the link. All routers on a link would have to include this option regardless of the prefixes they advertise. This would allow movement detection to work when several routers are connected to a link.

Another issue to consider when addressing movement detection is the possibility of gaining hints from lower layers. This can be discussed in light of Figure 3–8.

In Figure 3–8 a mobile node moving from Cell_1.2 to Cell_2.1 might get an indication from lower layers that an "intercell" movement has taken place before it detects its movement from router advertisements. While such an indication does not necessarily imply a topological movement, it can prompt the

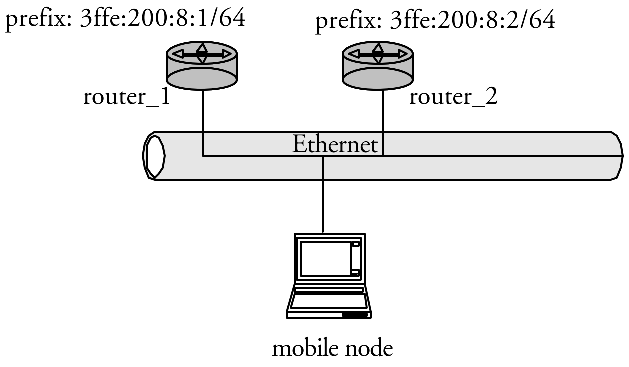

Figure 3–7 *Different routers advertising different prefixes on the same link.*

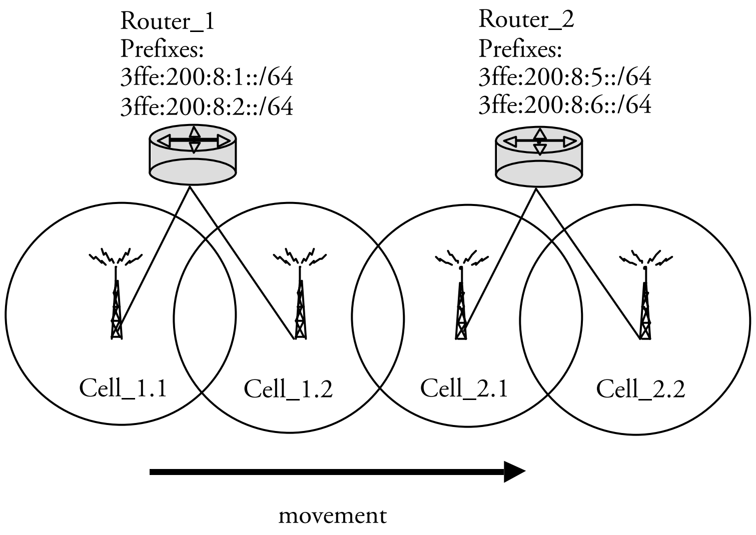

Figure 3–8 *Movement detection.*

mobile node to solicit for a router advertisement. Generic lower-layer hints should not replace the need for movement detection on the IP layer; however, they can provide useful optimizations. In Figure 3–8, a mobile node might receive a lower-layer indication when moving from Cell_1.1 to Cell_1.2 or when moving from Cell_2.1 to Cell_2.2. Those cells are attached to the same router. Therefore, movement between those two cells does not constitute topological movement and should not invoke IP layer mobility. Clearly a link layer may be integrated with the IP layer so as to provide specific triggers that indicate IP layer movement (e.g., when a link is configured such that intercell movement is always associated with IP movement). However, this a special optimization (i.e., not a generic assumption that can be made in an IP layer protocol, which applies to all link layers).

3.2.5 Returning Home

We have discussed the operation of the protocol between the mobile node and its home agent when the mobile node is located in a foreign network or is moving between two foreign networks. When the mobile node returns to its home network, it needs to inform the home agent that it should stop receiving its packets on its behalf. To achieve this, the mobile node must send a binding update to the home agent with a lifetime of zero and a care-of address (source address) equal to the mobile node's home address, indicating that the home agent should no longer receive the mobile node's traffic. Here we face a chicken-and-egg problem: the mobile node needs to configure its home addresses to be able to send the binding update. However, the home agent still assumes that it should defend the mobile node's home addresses, which would cause DAD to fail and prevent the mobile node from configuring its home address. The mobile node also needs to learn the home agent's MAC address (on link layers with MAC addresses, like Ethernet). This is done by sending a *neighbor solicitation* with a *target address* field set to the home agent's global IP address. The destination address is set to the home agent's *solicited node multicast address* and the source address is set to the unspecified address (::). This message looks like a DAD message, causing the home agent to respond with a neighbor advertisement to the *all-nodes multicast address* (since the solicitor's address was not provided) with its unique address and its link-layer address.

To avoid this problem, the mobile node sends a binding update with its home address as a source and the home agent's address as a destination. This is done without performing DAD on the mobile node's home address (an exception to address autoconfiguration). The mobile node then waits for an acknowledgment from the home agent. As soon as the home agent processes the binding update, it will no longer defend the mobile node's home address. Therefore, it will be able to send a binding acknowledgment to the mobile node's home address. Note that in order for the home agent to send the binding acknowledgment, it may need to send a neighbor solicitation to the mobile node's home address. The mobile node must respond to this solicitation from the home agent, despite not performing DAD on its home address—that is, another exception from normal address autoconfiguration. This process is shown in Figure 3–9.

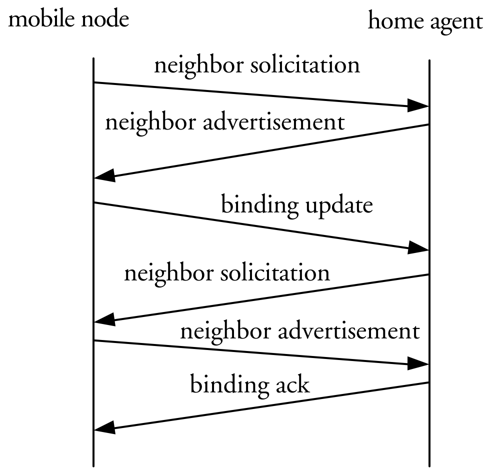

Figure 3–9 *Returning home*

When the binding acknowledgment is received, the mobile node sends a neighbor advertisement to the *all-nodes multicast address* with the *O* flag set. The purpose of this advertisement is to inform all nodes that they should send traffic directly to the mobile node (i.e., override the home agent's earlier proxy advertisements). To minimize the probability of such advertisement being lost, the mobile node may send it more than once. At the end of this procedure, the mobile node should receive all its traffic from other nodes on-link directly without passing through the home agent.

3.2.6 Source Address Selection in Mobile Nodes

Mobile IPv6 provides transparent mobility to upper layers by providing a fixed home address to applications, independent of the mobile node's location within the Internet topology. To achieve this, an anchor point within the home network (the home agent) is needed to provide the forwarding service to the mobile node's care-of address.

However, in some cases there may be a need to provide a care-of address to applications. For instance, a mobile node would typically need to communicate with a DNS located in the same network (a foreign network). If the mobile node uses a home address as a source, all DNS queries and replies would pass through the home agent. This can cause unnecessary delays. Furthermore, since such queries represent a very short-lived connection, during which the mobile node is not expected to move, the mobile node is likely to be able to use its care-of address to communicate directly with the DNS. The DNS is only one example, and there may be many types of short connections that can use the care-of address successfully (e.g., Web browsing). The choice between the use of home and care-of addresses as a source address is left to the discretion of the implementation. However, the use of a care-of address for connections that are known to be short (one way of knowing this is by looking at well-known port numbers for applications, e.g., DNS) would provide better performance.

It is also possible to add some information in the sockets API to allow applications to request the type of address needed for their connections. However, an implementer cannot assume that all applications will have such intelligence (to request a care-of address or a home address). Therefore, a good default source address selection algorithm is needed (see [2] for further details).

3.2.7 Dynamic Home Agent Discovery

One of the details that was bypassed in our earlier analogy with the mail-forwarding service was how to locate the post office. One would need to locate the post office to be able to request a forwarding service. In a similar manner, a mobile node needs to locate a suitable home agent. The home agent's address can be configured in the mobile node (in a nonvolatile memory to survive reboots). However, this approach has some drawbacks. Addresses change, or the home agent may fail or simply get overloaded if too many mobile nodes register with it. To avoid these problems, Mobile IPv6 provides a *Dynamic Home Agent Address Discovery* mechanism that allows mobile nodes to discover home agents' addresses. In addition to address discovery, DHAAD allows home agents to share the load between them, in cases where multiple home agents are located on the same link, by utilizing a *preference* parameter. This parameter is included in a new option, the *home agent information option*, which is included in router advertisements sent by home agents and shown in Figure 3–10.

The option includes a *preference* field with a default value of zero. Larger values indicate higher availability of the home agent. This option can only be included if the router is also a home agent (i.e., when the *H* flag is set in the router advertisement).

The *home agent lifetime* is used to indicate how long a router can serve as a home agent. This field must never be set to zero. If a router cannot serve as a home agent, it should clear the *H* flag in the router advertisement and stop advertising this option.

When other home agents receive this option, they store the home agent's address, its preference, and its lifetime. If a home agent sends router advertisements that do not include this option, its preference is assumed to be zero. Hence, every home agent on a link keeps a list (the *home agents list*) containing an IP address of each home agent on the link and its preference. Home agents can change their preference values dynamically and communicate them in router advertisements, depending on their availability or other load-related parameters.

The next step is to communicate the information in the home agents list to mobile nodes; this is done using DHAAD messages. DHAAD messages are carried in two ICMP messages and consist of a *DHAAD request* and *DHAAD reply* message. The mobile node sends a message requesting a list of global IP addresses for possible home agents on its home link. The message is sent to the *home agents' anycast address*; the anycast address is formed by appending the special interface identifier reserved for home agents' anycast addresses (see

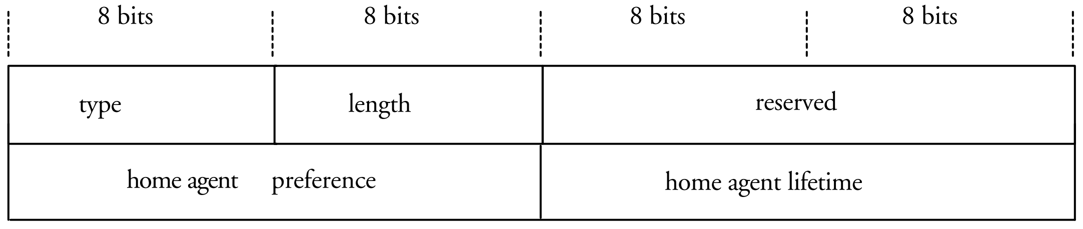

Figure 3–10 *Home agent information option.*

section 2.5.4) to the prefix of the home link. Hence, the message will be delivered to one of the home agents on the home link. Since every home agent keeps a home agents list for all home agents on-link, the receiver of the DHAAD request message would be able to send a DHAAD reply message containing a list of IP addresses for the different home agents on-link, as shown in Figure 3–11. The home agents' IP addresses are arranged in order of preference, with the most preferred home agent on top of the list. If two or more home agents have the same preference, their order is randomized between different messages. Hence, some equality can be achieved between different home agents that have equal preferences.

The *identifier* field includes a number sent from the mobile node in the DHAAD requested and echoed in the DHAAD reply sent from the home agent. The actual value has no meaning, but it allows the mobile node to keep a record of its attempts to discover a home agent in cases where the mobile node had to retransmit the DHAAD request because the original message was lost.

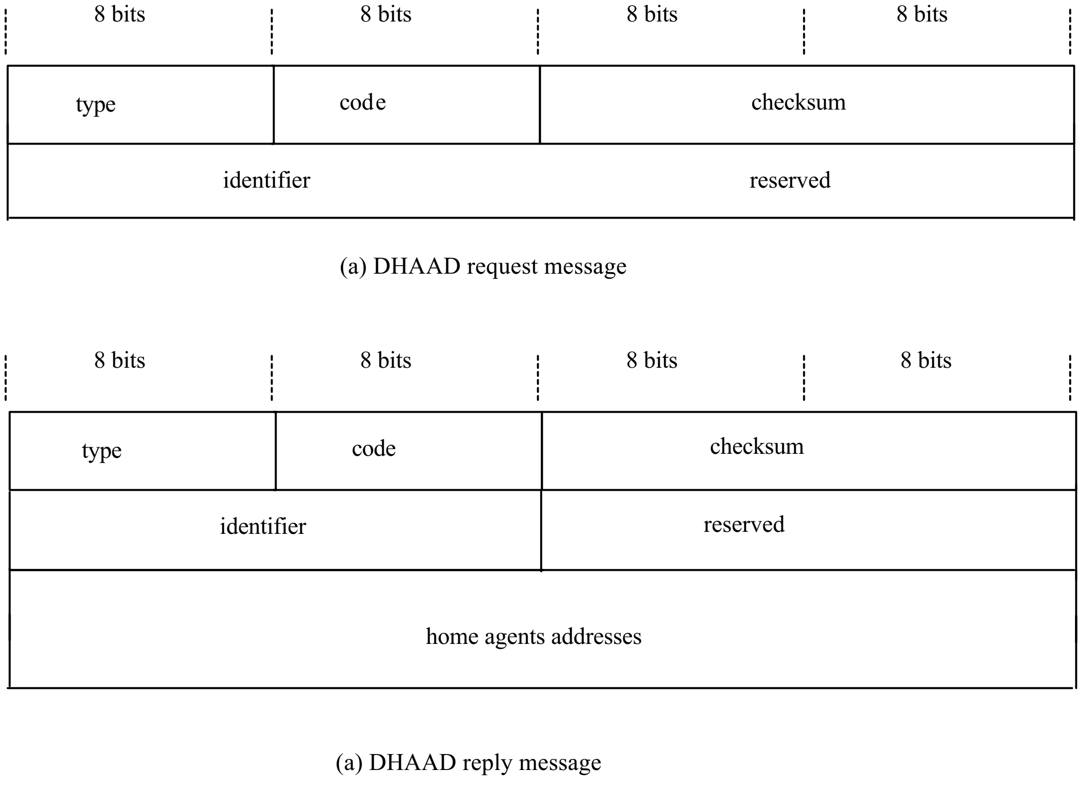

Figure 3–11 *DHAAD request and reply messages.*

Clearly DHAAD does not completely eliminate manual configuration in the mobile node. Mobile nodes still need to know the home link's prefix in order to construct the home agent's anycast address. However, this mechanism is needed to allow load-sharing between different home agents on the same home link.

Why can't we rely on anycast addresses only for load-sharing? Why do we require home agents to maintain a list of all other home agents on-link? The simple answer is that anycast addresses may allow a router to pick a different anycast destination for each packet, but that does not necessarily mean that the load will be shared fairly between home agents. Home agents are routers that can have very different capabilities, including capacity, processing power, and throughput. Furthermore, not all home agents will experience the same amount of traffic from the mobile nodes that they serve. Therefore, even if all home agents were the same make and model, they could still experience different amounts of traffic from mobile nodes. Hence, load-sharing based on anycast routing alone is not sufficient. The preference value is still needed to give a more accurate view of the home agent's load.

3.2.7.1 DETECTING CHANGES ON THE HOME LINK

Apart from discovering a home agent's address, mobile nodes often need to be updated with any changes on the home link. For instance, the home link may be renumbered with new prefixes, or new prefixes may be advertised in addition to the existing ones. These additional prefixes may imply new home addresses for the mobile node. It is important for the mobile node to detect such changes and detect new home addresses or deprecate the old ones (in case of renumbering). Neighbor Discovery allows nodes to detect such changes when located on the link in question. However, Neighbor Discovery messages cannot be sent beyond a link. For this reason, Mobile IPv6 defined two new messages: *Mobile Prefix Solicitation (MPS)* and *Mobile Prefix Advertisement (MPA)*. The first message is sent from the mobile node to its home agent, while the second is sent from the home agent to the mobile node. Just like the router solicitation and advertisement messages, these messages are carried in ICMP headers (with different ICMP types from those used in Neighbor Discovery). However, Mobile IPv6 requires that these messages are authenticated by IPsec. This is done to prevent a Bad Guy located between the mobile node and its home agent from intercepting these messages and modifying them in a malicious manner (e.g., setting a prefix's lifetime to zero, which would deprecate all addresses formed by the mobile node based on that prefix). More discussions on the security of these messages follow in Chapter 5.

3.2.7.2 WHAT IF THE HOME AGENT FAILED?

The home agent is crucial for the mobile node's reachability and session continuity. It maintains state about the mobile node's location, possibly for long periods of time. Like all nodes that keep state, the home agent reduces reliability.

If the home agent fails, the mobile node becomes unreachable. Furthermore, any connections being tunneled through that home agent are lost. It might be possible to transfer the binding cache between home agents to allow other agents to take over when the home agent fails, but no protocol is currently defined to solve this problem.

When the home agent fails, a mobile node does not receive any indication. The mobile node, however, loses any connections routed through the home agent (if it had any connections with correspondent nodes). If no connections are going through the home agent, the mobile node becomes unreachable without knowing it. In any case, the mobile node eventually tries to refresh its binding with the home agent. If no response is received after several attempts, as described earlier, the mobile node can perform DHAAD and attempt to register with a different home agent.

This approach does not lead to quick detection of home agent failures. However, the use of route optimization (described in section 3.3) reduces the reliance on the home agent. In addition, further developments in Mobile IPv6 may result in a protocol that transfers state between home agents to avoid its being a single point of failure. One example of an existing standard can be found in [5], which is used for IP routers' redundancy.

3.2.8 Challenges Associated with Plug and Play for Mobile Nodes

We now know how a mobile node can discover a suitable home agent and remain updated about any changes in the home link. While DHAAD provides a dynamic mechanism for detecting changes on the home link (including the possibility of a home agent failing), such mechanisms still require some manual configuration inside the mobile node. The mobile node needs to know at least one of its home link's prefixes to be able to send the DHAAD request to the home agents' anycast address. The knowledge of a home link's prefix essentially gives the mobile node one of its home addresses (except for saving 64 bits of storage, there is no point in configuring a prefix and not the interface identifier, i.e., the entire home address). Furthermore, there is a need for a security association between the mobile node and its home agent to be able to secure the *Mobile Prefix Solicitation* and *Mobile Prefix Advertisement* messages. There is also a stronger need for mobile nodes to secure binding updates to the home agent, as described in Chapter 5.

You can imagine scenarios where a mobile node is out of coverage for long periods of time due to lack of wired or wireless coverage (or simply because you're trying to enjoy a holiday without being reminded of how much work you will do when the holidays are over!). If the home link were renumbered during such period, a mobile node would not be able to contact its home network (which was renumbered) and, as a result, would not be able to send a binding update to its home agent. Hence, removing the reliance on manual configuration would allow for better resilience to changes in the home link. However, removing such dependency is not a trivial task. In this section we discuss the challenges associated with removing manual configuration.

Achieving pure plug and play for mobile nodes would entail the following:

1. The mobile node discovers its own home address.
2. The mobile node uses the knowledge of the previous step to construct the home agents' anycast address and perform DHAAD.
3. The mobile node selects a home agent and sends a binding update.

DHAAD was described earlier and does not pose any special requirements. However, step 1 poses a more difficult challenge, as shown below. Step 3 requires mutual authentication between the mobile node and its home agent. This is described in detail in Chapter 5.

3.2.8.1 HOW CAN A MOBILE NODE DISCOVER ITS HOME ADDRESS?

To remove the dependency on manual configuration of the home address, mobile nodes need to have an abstract identifier that can be mapped to a home address, for example, by storing that identifier in a database whose lookup would show the corresponding address. Such database already exists on the Internet: the DNS. Hence, one could configure a mobile node with its domain name. When starting the mobile node, it could query the DNS for the IP address corresponding to its own domain name and obtain its home address, then proceed with steps 2 and 3 above. However, DNS lookups are not usually secure. Bad Guy may intercept a DNS lookup and act as a DNS, providing the mobile node with a false home address. Currently the IETF is developing a DNS security protocol (DNSSEC) [3], which can allow mobile nodes to authenticate the DNS replies to ensure their authenticity. However, DNSSEC is currently not widely deployed and will take some time to get to wide deployment. In the worst-case scenario, Bad Guy can deny the mobile node access to its home agent by returning a false home address in the DNS reply. This is a form of a Denial of Service (DoS) attack, as it denies the mobile node access to its home agent. However, Bad Guy cannot pretend to be a home agent, since binding updates between the mobile node and its home agent are authenticated.

The next challenge with this approach is to be able to authenticate binding updates to the home agent. To avoid manual configuration, mobile nodes need to be able to establish security associations with the home agent in a dynamic manner. This requires the use of a key exchange mechanism (e.g., the Internet Key Exchange -IKE- protocol). Chapter 5 discusses these issues in more detail.

3.2.8.2 CAN MOBILE NODES GENERATE A RANDOM HOME ADDRESS?

The concept of a permanent home address is extremely useful for mobile nodes that need to be reachable. However, some users may be concerned about being tracked when using the same home address all the time. For instance, someone snooping traffic near the home link may be able to track a home address (which can be attributed to human if the machine's owner is known) and find out which correspondent nodes it communicates with (e.g., Hesham.example.com reads news on newschannel.com and uses mailserver.net for his email service). This may lead to undesirable profiling of users, which raises privacy concerns.

RFC 3041 attempts to solve this problem for IPv6 nodes by providing an algorithm that they can use to generate random numbers and use them as interface identifiers. It also assumes that these addresses will be relatively stable (e.g., usable for several days). If a mobile node generates a home address, sends a binding update to its home agent, and then goes out of coverage until its binding expires, another mobile node may use this address with the same home agent, resulting in binding update failure due to home address duplication. The probability of this happening is low, but it can be significant for a large home agent and a bad random number generator in the mobile node. Hence, such address may not be stable for the period required by RFC 3041. This may not be a significant problem if the mobile node is ready for such change (i.e., does not allow applications to use this address after receiving a notification from the home agent that the binding update cannot be accepted) and can generate another address. However, another problem might be that a socket was already open with the previously generated home address before the mobile node went out of coverage. If an application tries to reuse that socket, it will fail, since the address no longer exists. In this case the application would need to open a new socket.

A more significant problem is due to the reachability requirement on the home address. For a home agent to accept a binding from a mobile node, it needs to make sure that this address is not allocated to another mobile node on a permanent basis. Simply checking that no other binding exists in the home agent's binding cache is not sufficient, as the real owner of the address may not have registered with the home agent yet (e.g., it is turned off or out of coverage). In chapters 4 and 5 we see how Cryptographically Generated Addresses (CGAs) can help with solving this problem.

3.2.9 Can a Mobile Node Have More than One Home Agent?

Mobile nodes can have several home addresses. The binding for each home address must be stored in **one** home agent only. A mobile node cannot send two different home registrations simultaneously (binding updates with the *H* flag set) to two different home agents. If it did, two home agents would attempt to defend the same home address on the same link, which would not work.

On the other hand, a mobile node can have several home addresses with one or more home agents. Whenever the mobile node moves, it must send a binding update for each home address to the corresponding home agent. That is, the same procedures described above are done for each home address. After the binding is established, the mobile node has several tunnel entry points, one for each home address.

3.2.10 Virtual Home Links

We saw how a mobile node can register and deregister a binding with its home agent. When creating a new binding, the home agent needs to perform DAD and subsequently defend the mobile node's home addresses and possibly its link-local address (if the *L* flag were set). Currently, there is no deployed mechanism defined to secure Neighbor Discovery messages (more on that in chapters 4 and 5). Moreover, the mechanisms currently being developed in IETF for this purpose have not yet provided a way of securing *proxy neighbor advertisements*.

It is possible for Bad Guy, sharing a link with the home agent, to launch a DoS attack by pretending to own the mobile node's home address and therefore cause the binding establishment to fail, rendering the mobile node unreachable. These problems can be avoided if the home link is a virtual link—that is, if the home prefix is not associated to any physical link. The home prefix can be configured on the home agent and mobile nodes only. The home agent would not send any router advertisements for this prefix. Consequently, the mobile node would never "return home"; it would always move from a foreign link to another. In this scenario, DAD would be done much quicker than if there were a physical link associated with the home prefix. The home agent would only need to make sure that the home address is not allocated to another node. Furthermore, no proxy neighbor advertisement would be sent on the wire, since no one else is located on the home link except mobile nodes and the home agent. Therefore, Bad Guy cannot launch DoS attacks on the mobile node's home address.

This configuration is possible with Mobile IPv6, but clearly not mandatory. The Mobile IPv6 specification does not restrict the home link configuration, but home agent implementers and network operators implementers are certainly free to allow for virtual home links, as they do have significant benefits.

3.3 Route Optimization

Going back to our post office forwarding service analogy once more, consider the case where you are planning to communicate with a certain person or organization on a regular basis after changing your address temporarily to another country. For example, you might be receiving a monthly credit card bill that needs to be paid within one week. If forwarding the bill to the new location takes a week, it is likely that your payment will be overdue and possibly cost more. The logical solution for this problem would be to inform the credit card company of your new address and avoid the additional delays.

Mobile nodes experience similar problems. Routing packets through the home agent adds additional delays. This can be clearly seen in Figure 3–1. As shown in Figure 3–12, the worst-case scenario would occur if the mobile node and the correspondent node share the same link. On the other hand, the best-case scenario (least possible delay) would occur if the correspondent node were on the mobile node's home link. However, in all cases there will be some additional delay when compared to direct communication.

Another problem with forcing traffic through the home agent is the introduction of a single point of failure in the network. That is, while it may be possible to eventually detect the failure of a home agent, its failure would cause the mobile node to lose all ongoing connections. In the future this problem may be avoided at the cost of introducing new protocols to synchronize binding cache states between home agents. However, it would be beneficial to remove this dependency for the general case. Another argument for eliminating the home agent from the path between the mobile node and correspondent node is that such routing uses more network bandwidth than direct communication and therefore makes inefficient utilization of the network bandwidth. Hence, if all mobile nodes' traffic is forwarded through the home agent, operators would need to be careful about the location of the home agent and the capacity of the links attached to it, which adds a burden on network design.

Route optimization is about routing packets between a mobile node and a correspondent node, using the shortest possible path (as it is normally done between two communicating hosts relying on normal routing). The mobile node is aware when packets are routed through the home agent when it receives tunneled

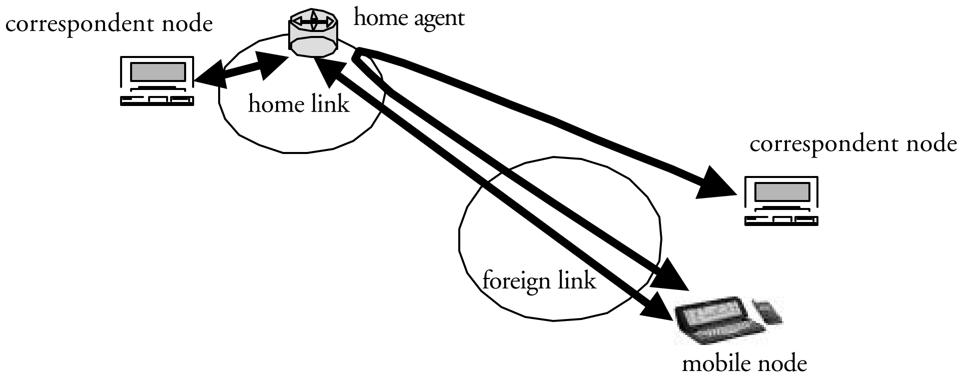

Figure 3–12 *Worst-case and best-case scenarios for routing through the home agent.*

packets addressed to its home address. Depending on the nature of communication (long or short term), the mobile node can decide whether it should attempt to optimize the route between itself and the correspondent node. How does a mobile node know whether the communication with a correspondent node will last for a short or a long time? There is, unfortunately, no concrete answer for this question. Obviously, a user may configure the mobile node to treat some addresses in a special way, but this is far from being a realistic solution; most users know nothing about IP addresses or IP in general. Also, this does not cover communication with other nodes of which the user had no prior knowledge. A "smart" mobile node's implementation may make some assumptions about the type of applications used between the mobile and correspondent node (e.g., Web browsing usually involves short-lived connections), the duration of a connection, the RTT, and the amount of data being sent between the two nodes to decide whether route optimization is needed. On the other hand, using a simpler approach, a mobile node's implementation may also decide to always use route optimization or never use route optimization.

When a mobile node receives a packet tunneled from the home agent, it must decide whether route optimization is needed. If so, the mobile node informs the correspondent node of its current location. This is done using the same binding update message shown in Figure 3–2. The correspondent node maintains a binding cache similar to the one maintained by the home agent. However, a binding update sent to a correspondent node must not set the *H*-, *K*-, or *L*-bits, as they are only usable when communicating with the mobile node's home agent. Route optimization is shown in Figure 3–13.

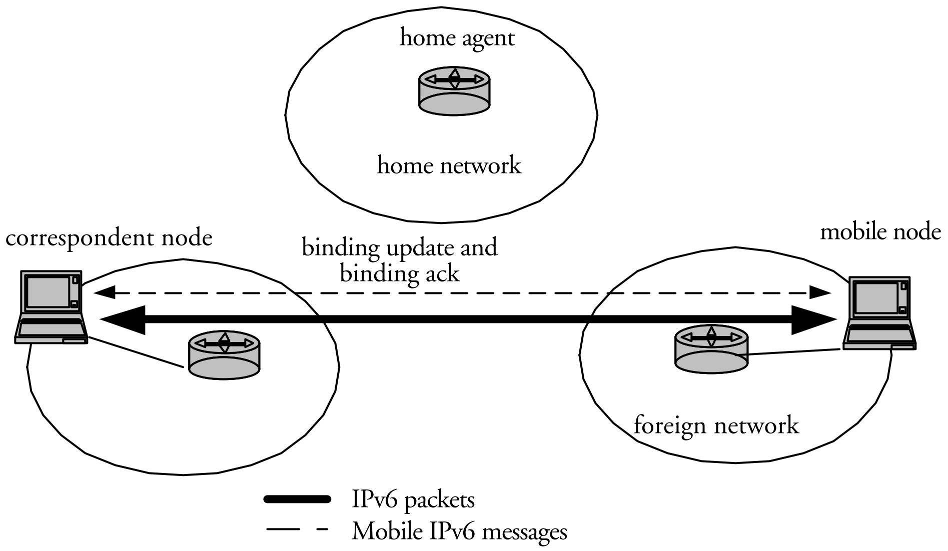

Figure 3–13 *Route optimization signaling.*

When a correspondent node receives a binding update from a mobile node, it creates a new entry in the binding cache or updates the existing one with the new location of the mobile node. Following this step, the correspondent node can communicate directly with the mobile node by sending packets to the mobile node's care-of address.

The aim of the binding update is to achieve two goals: first, allow packets to be sent directly between the mobile and correspondent nodes without going through the home agent, and second, maintain ongoing connections in the meantime by allowing applications to keep using the home address as a source address (in the mobile node) and destination address (in the correspondent node). The first step can be achieved if both nodes ensure that the packets transmitted contain the mobile node's care-of address in the source and destination fields when being sent from and to the mobile node respectively. This is needed to allow Internet routing to deliver packets directly to the mobile node (for packets sent to the mobile node). In addition, this allows the mobile node to send outgoing packets with a topologically correct address. However, to maintain ongoing connections, the mobile node's home address must somehow be included in packets sent to and from the mobile node to allow the IP layer (in the mobile and correspondent nodes) to present it to upper layers. To do this, Mobile IPv6 defines two messages, a new *routing header type 2* and a new destination option called the *home address option*, that are included in packets sent to and from the mobile node respectively. We first discuss the home address option, then the new routing header. The security considerations for route optimization are discussed in detail in Chapter 5.

3.3.1 Sending Route Optimized Packets to Correspondent Nodes

When the mobile node sends a binding update to a correspondent node, it needs to indicate the home address for which the binding is sent. The home address is included in the home address option, which is included in the destination options extension header. After accepting the binding update, the mobile node's home address is stored in the correspondent node's binding cache with the rest of the contents of the binding update. If the *A* flag is set in the binding update, the correspondent node sends a binding acknowledgment to the mobile node. After receiving a binding acknowledgment, the mobile node updates its binding update list to include the information sent in the binding update to the correspondent node. This includes the correspondent node's IPv6 address, the mobile node's home and care-of addresses, the sequence number used, and the lifetime for the binding.

After successfully installing a binding in the correspondent node's binding cache, the mobile node uses the home address option in every packet sent that includes data from an application using the home address as a source and communicating with the correspondent node (destination). The home address option is essentially a disguised form of tunneling. When a correspondent node receives a packet containing a home address option, it replaces the source address in the packet's header with the address included in the home address option before passing the packet to upper layers, which effectively makes the packet appear to be received from the mobile node's home address when viewed by upper layers. Hence, mobility is kept transparent to upper layers.

It is important to note that the mobile node includes the home address option only in packets sent directly to correspondent nodes (i.e., to which a binding update was sent and accepted). This is done for security reasons that are discussed in Chapter 5. In order for mobile nodes to know which correspondent nodes have a binding cache entry for the mobile node (and hence will accept the home address option), the mobile node checks the content of the binding update list before sending packets. The binding update list contains an entry for each correspondent node's address. Each entry contains the details of the binding update sent to the correspondent node: the lifetime, sequence number, home address, and correspondent node's address, which is used as a search key for each entry. The authentication credentials for binding updates are also included in the binding update list. Finally, the binding update list contains a flag, set by the mobile node, to indicate whether a mobile node should attempt route optimization with a particular correspondent node in the future. The aim of this flag can be explained in light of one of the requirements on Mobile IPv6: backward compatibility. Since not all hosts on the Internet are expected to support Mobile IPv6, some correspondent nodes might not understand the mobility header at all and would send an ICMP error ("host unreachable") informing the mobile node that the mobility header is not supported. Hence, the mobile node can avoid sending future binding updates to those correspondent nodes by setting the flag mentioned above. How long should this entry be kept? This is left up to the mobile node implementer and primarily depends on the size of memory allocated for binding update lists.

Figure 3–14 shows how the home address option is added in the mobile node's IP layer implementation before sending the packet to the correspondent node.

Figure 3–14 shows three steps: first, the original packet as expected by the application is formed, then the home address option is added in the destination options header. This option includes the care-of address. We intended to show this step first to clarify exactly how the packet is processed. The reason for including the care-of address in the home address option is to allow any operations on the packet (between constructing the packet and sending it), like IPsec processing, to assume that the source address is in fact the home address. This is done to avoid changing, for example, an IPsec security association based on the mobile node's source address every time the mobile node changes its care-of address.

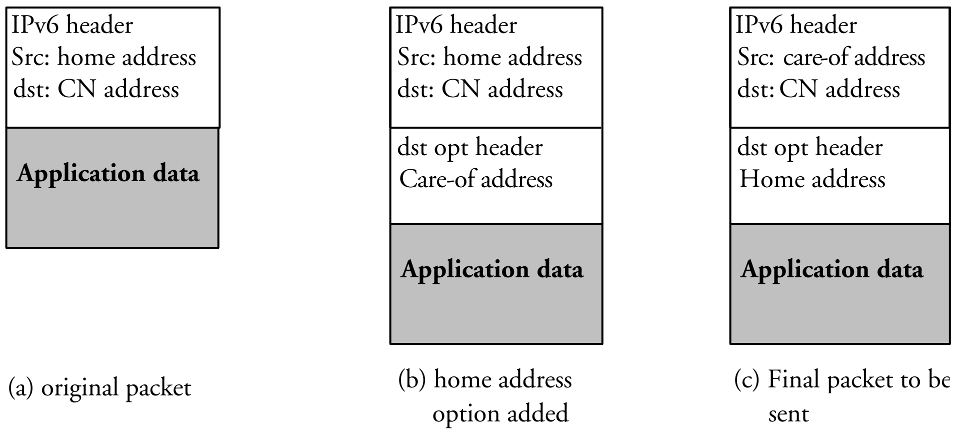

Figure 3–14 *Adding the home address option before sending a packet.*

When the packet is ready to be sent, the content of the home address option is swapped with the source address field. Hence, the packet leaving the mobile node will contain the care-of address in the source address field and the home address option in the destination options extension header.

When the packet is received by the correspondent node, the same operations are done in the reverse order. Hence, the packet seen by upper layers in the correspondent node will look like the original packet (seen in step a) in Figure 3–14.

3.3.2 Receiving Route Optimized Packets from Correspondent Nodes

When a correspondent node sends packets to a mobile node for which it has a binding cache entry, it must include a new routing header (with a *type* field set to 2). This routing header is identical to the one shown earlier in Chapter 2, with the exception that it can only include the mobile node's home address, that is, the number of segments cannot be larger than 1. The reason for this restriction becomes clear in section 5.1.2. When receiving the packet, the mobile node processes the routing header. This results in replacing the destination address in the packet (care-of address) with the address in the routing header (the home address). Since this address also belongs to the mobile node, the mobile node is essentially forwarding the packet to itself. Following this step the packet is passed to upper layers with the mobile node's home address in the destination address field, hiding the address change from upper layers.

3.3.3 Acknowledging Binding Updates Sent to Correspondent Nodes

The process of route optimization involves three distinct steps:

1. Detecting that packets are tunneled by the home agent.
2. Sending a binding update to the correspondent node.
3. Sending packets directly to the correspondent node and including the home address option in those packets.

The final step can only be done after the binding update is received and processed by the correspondent node. If the mobile node sends packets directly to correspondent nodes, including the home address option, when the correspondent node has not accepted the binding update, the correspondent node discards them. This act causes packets to be lost and consequently disrupts communication. Hence, the mobile node needs to ensure that the binding update was received and accepted by the correspondent node before routing packets directly to the correspondent node's address. To ensure the acceptance of the binding update, the mobile node can request an acknowledgment by setting the *A* flag in the binding update. If the binding is accepted, the correspondent node responds with a binding acknowledgment containing the appropriate status. After receiving the binding acknowledgment, the mobile node can be sure that the correspondent node will accept packets containing the home address option.

3.3.4 What if the Correspondent Node Failed?

The mobile node should perform route optimization signaling in a reliable manner to ensure that no packets are lost due to the premature inclusion of the home address option. However, it is possible for a correspondent node to lose the information in its binding cache (e.g., due to a reboot, which causes a node to lose the contents of its volatile memory that includes the binding cache, or because the cache exceeded the allocated memory and some entries were deleted) some time after accepting a binding update. The mobile node would not be aware of such reboot and would continue to send packets containing the home address option. The correspondent node receives these packets, checks if it has a binding cache entry corresponding to the home address, but finds none and consequently drops the packet. To inform the mobile node of such failure, the correspondent node can send a *binding error* message indicating that no binding cache entry exists for this home address. The binding error message may also be sent due to other errors in the mobility header. This message (see Figure 3–15) is sent to the source address in the packet (i.e., the mobile node's care-of address).

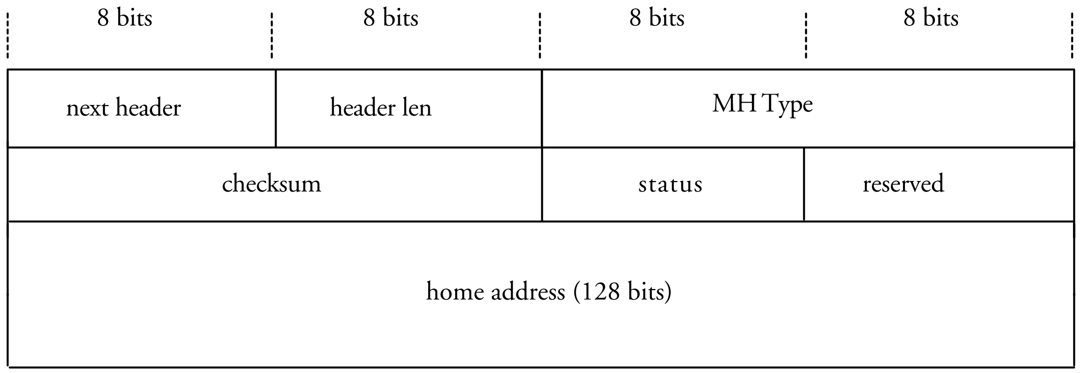

Figure 3–15 *Binding error message.*

The binding error message has a *type* field value of 7. The *status* field has two possible values (and more may be defined in future): when set to 1 it indicates that there is no binding cache entry associated with the home address in the binding error message. This may be the case when the contents of the binding cache are lost or the binding update for this home address was never received. A value of 2 indicates that the type of mobility header received (i.e., the message included in the mobility header) is not known to the correspondent node. This value ensures that new mobility signals can be added in the future, in a backward-compatible manner. That is, new functions can be added without affecting legacy nodes, as they will inform mobile nodes if they are unable to support such functions.

3.3.5 Why Not IP in IP Tunneling for Route Optimization?

The basic idea behind route optimization is to hide the mobile node's home address inside the packets (using the routing header and the home address option) to avoid breaking ingress filtering (when the packets are sent by the mobile node) or to route packets to the right location in the topology (when they are sent by the correspondent node). However, the same effect could have been achieved by using IP in IP tunneling between the mobile node and the correspondent node. Following the binding update processing, a tunnel can be established between the mobile and correspondent nodes. Packets originating from the mobile node would contain the mobile node's home address in the source address field within the inner header; the destination address would be the correspondent node's address. The outer header would contain the mobile node's care-of address in the source address field and the correspondent node's

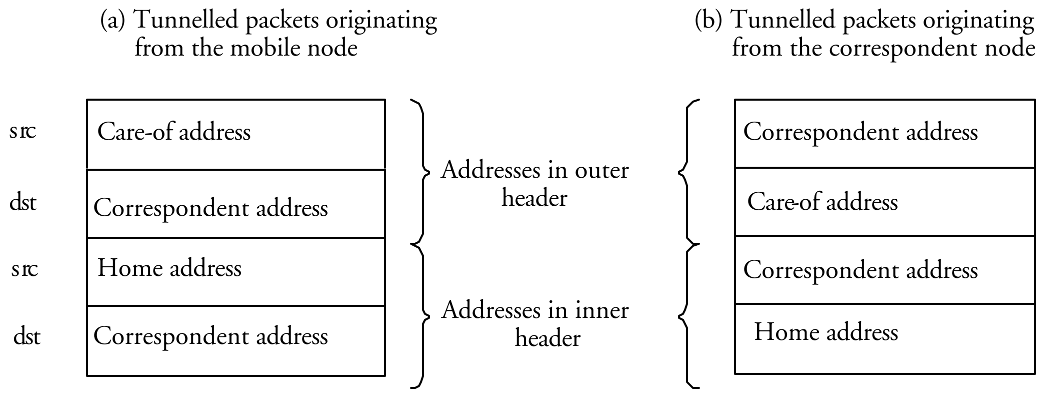
(a) Tunnelled packets originating from the mobile node
(b) Tunnelled packets originating from the correspondent node
src
dst
src
dst
Care-of address
Correspondent address
Home address
Correspondent address
Addresses in outer header
Addresses in inner header
Correspondent address
Care-of address
Correspondent address
Home address

Figure 3–16 *Using tunneling between the mobile node and the correspondent node.*

address in the destination field. The opposite can be done for packets originating from correspondent nodes, as shown in Figure 3–16.

While this approach has the benefit of using existing mechanisms (tunneling) instead of defining new ones (i.e., the routing header and the home address option), it has the disadvantage of using more network bandwidth when compared to using the routing header and the home address option. This is due to having four addresses (i.e., two IP headers) in the tunnel when compared to three when either the home address option or the routing header is used. In [1] it was proposed to compress the redundant address in the tunnel (the correspondent node's address). However, this approach was proposed at a later stage of Mobile IPv6's development and was viewed as a "good approach that came too late."

3.4 What if the Mobile Node Failed?

The mobile node maintains state about the nodes with which it has bindings in its binding update list. The loss of such information results in having different sets of information in the mobile node, its home agent, and correspondent nodes. If the mobile node reboots and loses this information, it immediately attempts to update its home agent. To update the current binding, the mobile node needs to use a sequence number larger than the one used in its last binding update message. Since the mobile node is not aware of the last value used, it picks a random value. If the chosen value is lower than the last one received by the home agent (or correspondent node), a binding acknowledgment is received with the status value set to 135, indicating that the sequence number is incorrect and providing the last used sequence number to the mobile node. The mobile node can then attempt to send the binding update with an appropriate value. Obviously, if the mobile node picks a random sequence number higher than the last used number, the binding update will be accepted the first time.

3.5 Site-Local Addresses and Mobile IPv6

One of the problems with site-local addresses is that a mobile node has no way of knowing whether it is in its home site or another one. That is, if a mobile node moves from one link to another, it may or may not have left its home site (the IPv6 site that was the home agent is located). There is no information in the router advertisement that tells nodes how large the site is. This will cause some confusion in the mobile node regarding its ability to use site-local care-of addresses. We can illustrate this confusion by demonstrating a couple of different scenarios where site-local addresses are used.

The first scenario to consider is a home site, including the mobile node's home agent, which uses site-local addresses only. A mobile node in this site will have a site-local home address. When the mobile node moves away from its home link, it does not know whether it has moved to a new site or within the same site. However, it does know that it moved from its home link. The mobile node then forms a site-local care-of address and attempts to send a binding update to its home agent. If the mobile node is in fact in the same site, the binding update is received by the home agent, and Mobile IPv6 will work. However, if the mobile node is in a different site, two different outcomes are possible:

- Another node in the same site is configured with the same site-local address as the home agent; therefore, this node will receive the binding update from the mobile node. However, since binding updates and acknowledgments are protected by IPsec there is no chance of confusion in this case; the node that receives the binding update will silently discard the packet.
- No node in this site is configured with the home agent's site-local address; therefore, the mobile node will receive an ICMP error "Destination unreachable."

From this discussion we can conclude that sites configured only with site-local addresses allow mobile nodes to move within the site; however, once the mobile node moves to another site, normal routing practices will inform the mobile node that its home agent is not reachable. We can also see that the mobile node's care-of address' scope should be equal to, or larger than, its home address' scope. In general, mobile nodes should use site-local care-of addresses only if no global care-of address exists.

3.6 A Communication Example

In this section we use the knowledge gained from this chapter to draw a simplified flowchart for the mobile and correspondent nodes' operation. This flowchart can be compared to the one used in Chapter 2 for normal IPv6 hosts. We do not consider the binding update security details in this section. Security issues are discussed in detail in chapters 4 and 5. Our flowchart for the mobile node's sending implementation is shown in Figure 3–17.

When the mobile node's IP layer implementation receives upper-layer data, it will have already supplied the home address as a source address for applications. Hence, an IP header can be constructed with the home address being the source address and the correspondent node's address in the destination address. Next, the binding update list is checked to see if the mobile node has already sent a binding update to the correspondent node and get the care-of address from the correspondent's node's entry. If it has, the mobile node can include its home address in the home address option. However, for now the mobile node places its care-of address in the home address option instead. We see why this is done when we analyze the following steps in the flowchart. The mobile node then checks its binding cache to see if the correspondent node has sent it a binding update. Note that *correspondent node* refers to a functional entity in an IPv6 node and does not exclude a correspondent node from being a mobile node as well. If the correspondent node had sent a binding update to the mobile node, a binding cache entry is found. Hence, the mobile node constructs a routing header type 2 and places the correspondent node's care-of address inside it.

Assuming that the mobile node does not need to add any other extension headers, the next step is to check whether IPsec is needed. This is done by checking the SPD to see whether the packet should be protected. When IPsec is used, the destination options header containing the home address option needs to be placed before the IPsec header (AH or ESP). This is done because IPsec identifies a security association in the SAD by *selectors*. Selectors include the source and destination addresses of the nodes sharing the security association. To avoid changing the security association every time the mobile node moves, it is best to use the home address (not the care-of address) as a selector identifying the mobile node. When the packet is received by the correspondent node, the headers are processed in the order they appear in the packet; therefore, IPsec headers are processed last, before the upper layer. Hence, an IPsec implementation in the correspondent node always sees the home address in the source address field of a received packet. Now we can see the reason for keeping the mobile node's home address in the source address field and its care-of address in the home address option until IPsec has added its header. We need to perform the IPsec operations on the header seen by the correspondent node after verifying the IPsec header. That is, the packet needs

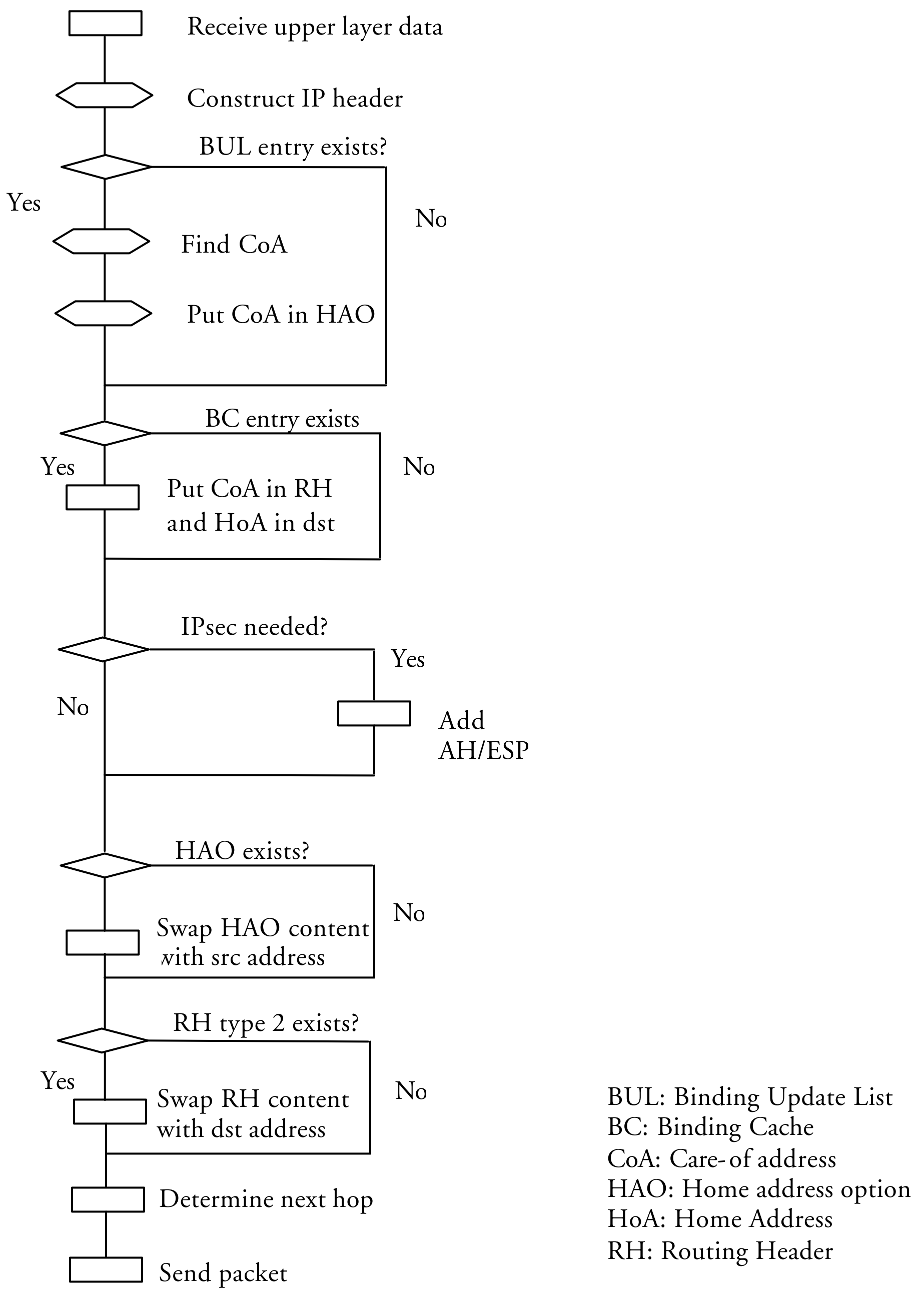

Figure 13–17 *Simplified flowchart for mobile node operations (sending packets).*

to look exactly the same when passed to the correspondent node's IPsec implementation as it did when passed to the mobile node's IPsec implementation. Otherwise, to IPsec, the packet will seem to have been modified and will be dropped.

After IPsec processing is done, the mobile node swaps the contents of the home address option and the routing header with the source and destination addresses respectively. The next hop is determined based on the earlier discussion in Chapter 2 and whether the packet is tunneled to the home agent or sent directly to the correspondent node. As discussed in Chapter 2, addresses are tied to interfaces. When the mobile node is on a foreign link, its home address is associated with a *tunnel interface* to the home agent. When it moves back home, the tunnel is deleted and the home address is associated with a physical interface. The tunnel interface can be treated by the IP layer implementation as another physical interface that is one hop away from the home agent. Hence, if the packet contains the home address in the source address field, it is immediately tunneled to the home agent; otherwise, it is sent on the interface associated with the care-of address.

Now let's consider the steps taken by the correspondent node when it receives packets from the mobile node, as shown in Figure 3–18.

When the correspondent node receives a packet, it processes the headers in the order that they were sent. If a routing header type 2 is included, and provided that there is one address in it and that it is the correspondent node's home address, it swaps that address with the destination address in the packet; otherwise, the packet is dropped. The node then processes other extension headers. In Figure 3–18 we skip this processing until it gets to the destination options header. If a home address option is found, the binding cache is searched for a corresponding entry. If nothing is found, a binding error message is sent to the source address in the IPv6 header. If the correspondent node does not implement any Mobile IPv6 messages, then it does not understand the home address option and sends an ICMP error message to the source address in the packet. If a binding cache entry is found for this home address, the home address is swapped with the care-of address in the source address field.

If an IPsec header is included, it is verified as usual. Note that the packet would now look exactly the same when seen by IPsec implementations at both ends. Finally, the information is passed to upper layers.

3.7 Summary

This chapter described the operation of Mobile IPv6, starting with the role of each element: the mobile node, the correspondent node, and the home agent. We also saw how mobile node–home agent and mobile node–correspondent

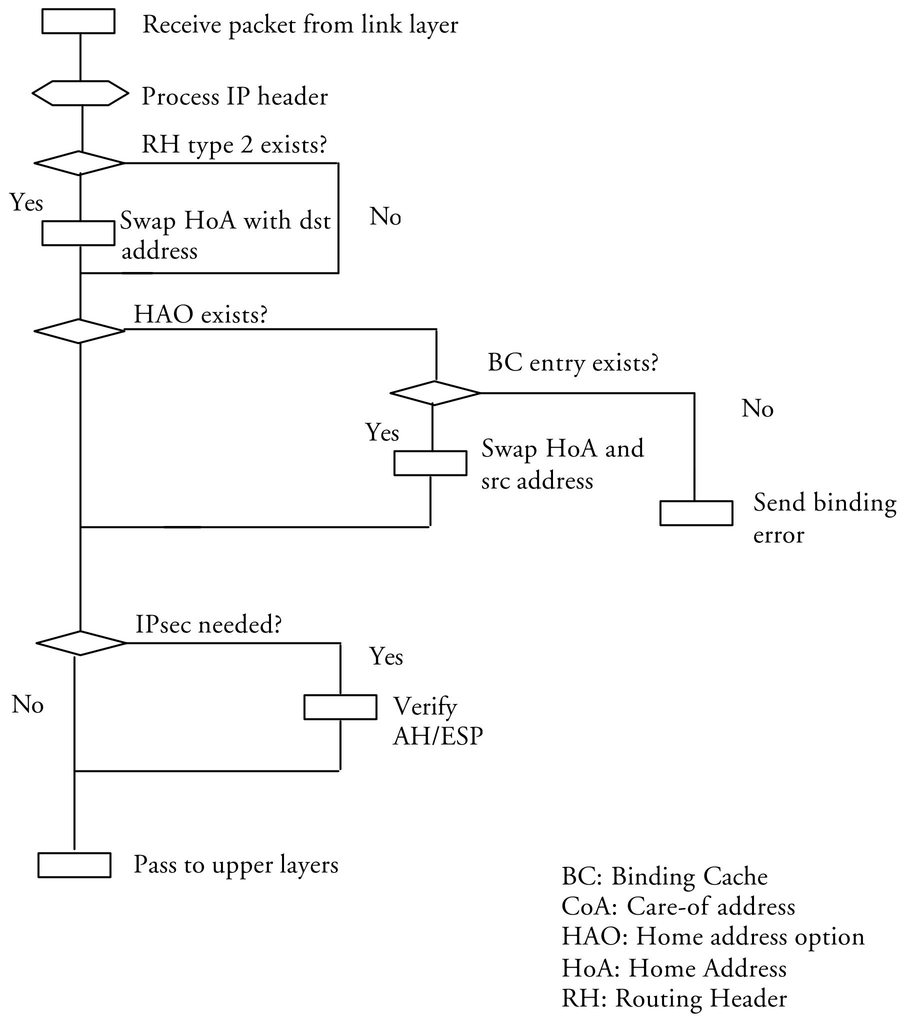

Figure 13–18 *Processing received packets at the correspondent node.*

node signaling is performed. Two different communication scenarios between the mobile and correspondent nodes were described: communication through the home agent and direct communication (route optimization). The processing within each node was described for both communication scenarios.

We also discussed the various failure cases for the mobile node, correspondent node, and home agent; and we saw how the protocol can cope with each one. Finally, the knowledge gained in this chapter was used to build simplified flowcharts that describe the mobile and correspondent nodes' operations.

The next chapter introduces a new topic: security, an interesting topic required to set the scene for the Mobile IPv6 security discussion.

Further Reading

[1] Deering, S., and B. Zill, "Redundant Address Deletion when Encapsulating IPv6 in IPv6," draft-deering-ipv6-encap-addr-deletion-00, work in progress, November 2001.

[2] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)," RFC 3484, February 2002.

[3] Eastlake, D., "Domain Name System Security Extensions," RFC 2535, March 1999.

[4] Johnson, D., C. Perkins, and J. Arkko, "Mobility Support in IPv6," draft-ietf-mobileip-ipv6-24, work in progress, June 2003.

[5] Knight, S., et al., "Virtual Router Redundancy Protocol," RFC 2338, April 1998.

[6] Nordmark, E., "MIPv6: From Hindsight to Foresight?" draft-nordmark-mobileip-mipv6-hindsight-00, work in progress, November 2001