IP Options

4.1. IPV4 OPTIONS

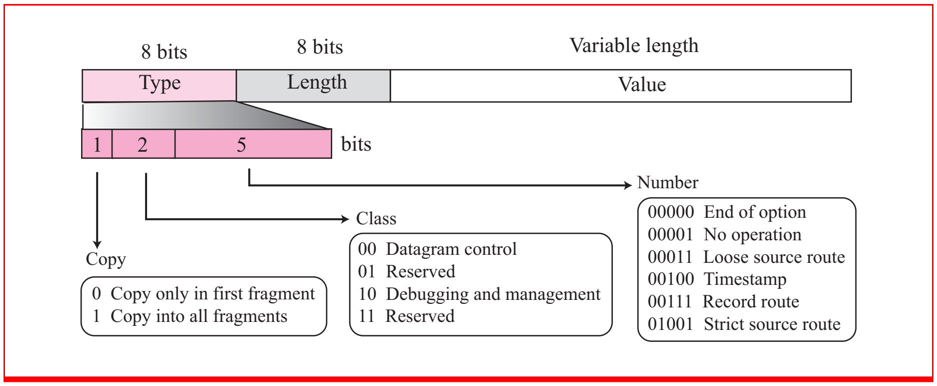
The header of the IP datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes (as mentioned in the text). The variable part comprises the options, which can be a maximum of 40 bytes.

Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging. Although options are not a required part of the IP header, option processing is required of the IP software. This means that all implemen- tations must be able to handle options if they are present in the header.

4.1.1 Format

Figure 4-1 shows the format of an option. It is composed of a 1-byte type field, a 1-byte length field, and a variable-sized value field. The three fields are often referred to as type-length-value or TLV.

Figure 4-1 *Option format*



Type

The **type field** is 8 bits long and contains three subfields: copy, class, and number. ■ **Copy.** This 1-bit subfield controls the presence of the option in fragmentation.

When its value is 0, it means that the option must be copied only to the first frag- ment. If its value is 1, it means the option must be copied to all fragments.

- **Class.** This 2-bit subfield defines the general purpose of the option. When its value is 00, it means that the option is used for datagram control. When its value is 10, it means that the option is used for debugging and management. The other two possi- ble values (01 and 11) have not yet been defined.
- **Number.** This 5-bit subfield defines the type of option. Although 5 bits can define up to 32 different types, currently only 6 types are in use.

Length

The **length field** defines the total length of the option including the type field and the length field itself. This field is not present in all of the option types.

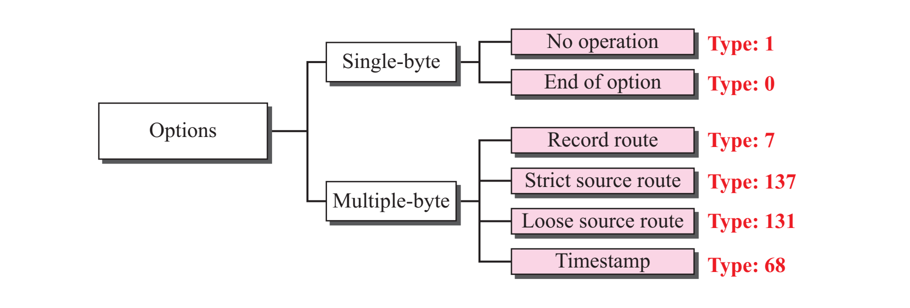
Value

The **value field** contains the data that specific options require. Like the length field, this field is also not present in all option types.

4.1.2 Option Types

As mentioned previously, only six options are currently being used. Two of these are 1-byte options, and they do not require the length or the data fields. Four of them are multiple-byte options; they require the length and the data fields (see Figure 4-2).

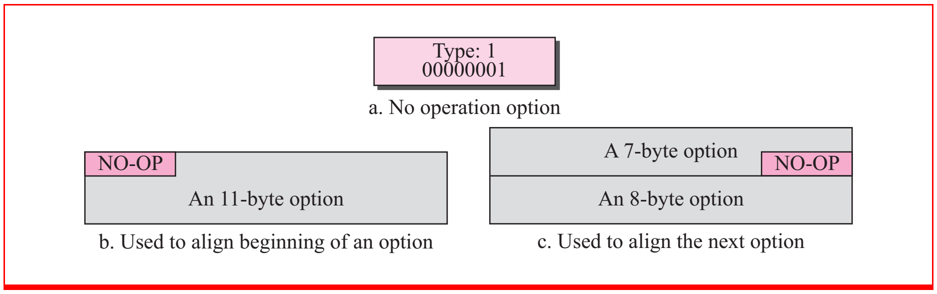
Figure 4-2 *Categories of options*



No-Operation Option

A **no-operation option** is a 1-byte option used as a filler between options. For example, it can be used to align the next option on a 16-bit or 32-bit boundary (see Figure 4-3).

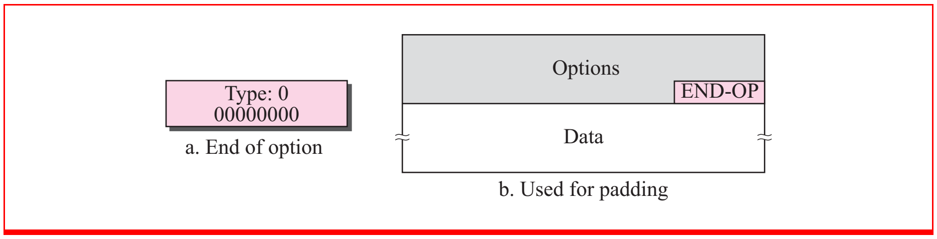
Figure 4-3 *No operation option*



End-of-Option Option

An **end-of-option option** is also a 1-byte option used for padding at the end of the option field. It, however, can only be used as the last option. Only one end-of-option option can be used. After this option, the receiver looks for the payload data. This means that if more than 1 byte is needed to align the option field, some no-operation options must be used, followed by an end-of-option option (see Figure 4 -4).

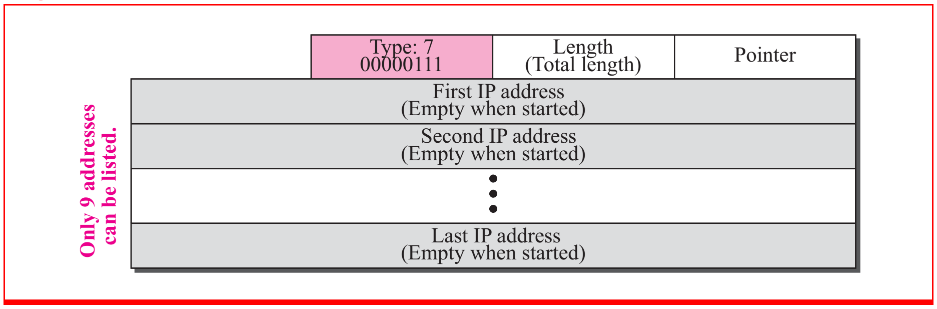
Figure 4-4 *End-of-option option*



Record-Route Option

A **record-route option** is used to record the Internet routers that handle the datagram. It can list up to nine router IP addresses since the maximum size of the header is 60 bytes, which must include 20 bytes for the base header. This implies that only 40 bytes are left over for the option part. The source creates placeholder fields in the option to be filled by the visited routers. Figure 4-5 shows the format of the record route option.

Figure 4-5 *Record-route option*

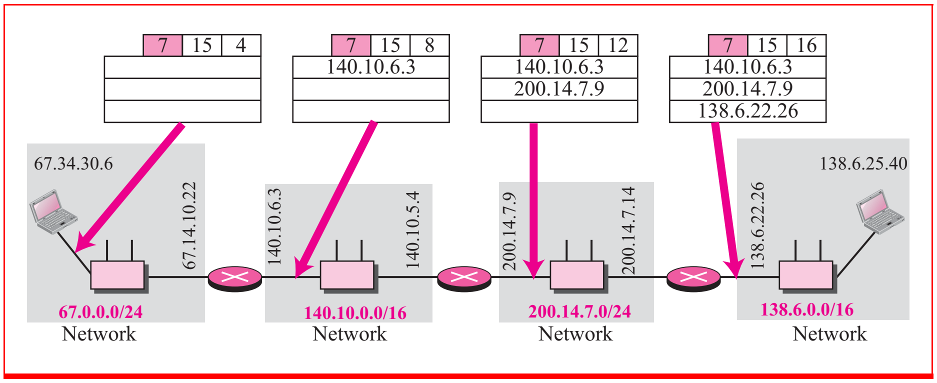


Both the code and length fields have been described above. The **pointer field** is an offset integer field containing the byte number of the first empty entry. In other words, it points to the first available entry.

The source creates empty fields for the IP addresses in the data field of the option. When the datagram leaves the source, all of the fields are empty. The pointer field has a value of 4, pointing to the first empty field.

When the datagram is traveling, each router that processes the datagram compares the value of the pointer with the value of the length. If the value of the pointer is greater than the value of the length, the option is full and no changes are made. However, if the value of the pointer is not greater than the value of the length, the router inserts its out- going IP address in the next empty field (remember that a router has more than one IP address). In this case, the router adds the IP address of its interface from which the data-gram is leaving. The router then increments the value of the pointer by 4. Figure 4-6 shows the entries as the datagram travels left to right from router to router.

Figure 4-6 *Record-route concept*



Strict-Source-Route Option

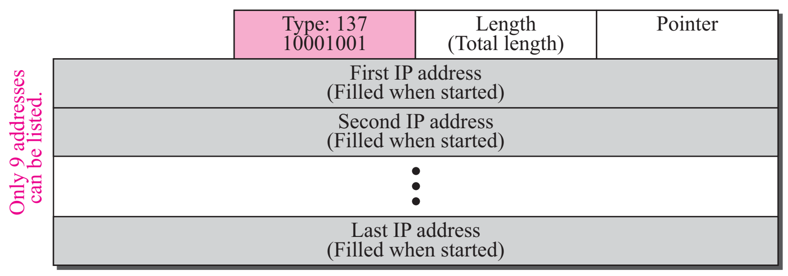
A **strict-source-route option** is used by the source to predetermine a route for the data- gram as it travels through the Internet. Dictation of a route by the source can be useful for several purposes. The sender can choose a route with a specific type of service , such as minimum delay or maximum throughput . Alternatively, it may choose a route that is safer or more reliable for the sender's purpose. For example, a sender can choose a route so that its datagram does not travel through a competitor's network.

If a datagram specifies a strict source route, all of the routers defined in the option must be visited by the datagram. A router must not be visited if its IP address is not listed in the datagram. If the datagram visits a router that is not on the list, the datagram is dis- carded and an error message is issued. If the datagram arrives at the destination and some of the entries were not visited, it will also be discarded and an error message issued .

Regular users of the Internet, however, are not usually aware of the physical topol- ogy of the Internet. Consequently, strict source routing is not the choice of most users. Figure 4-7 shows the format of the strict source route option.

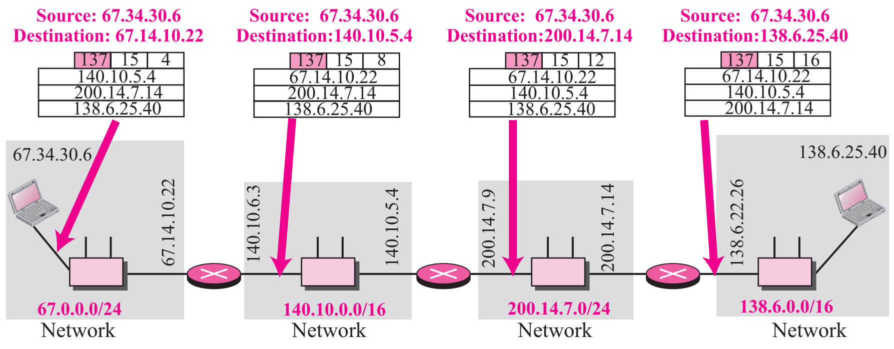
The format is similar to the record route option with the exception that all of the IP addresses are entered by the sender. When the datagram is traveling, each router that

Figure 4-7 *Strict-source-route option*



processes the datagram compares the value of the pointer with the value of the length. If the value of the pointer is greater than the value of the length, the datagram has vis- ited all of the predefined routers. The datagram cannot travel anymore; it is discarded and an error message is created. If the value of the pointer is not greater than the value of the length, the router compares the destination IP address with its incoming IP address: If they are equal, it processes the datagram, swaps the IP address pointed by the pointer with the destination address, increments the pointer value by 4, and forwards the datagram. If they are not equal, it discards the datagram and issues an error mes- sage. Figure 4-8 shows the actions taken by each router as a datagram travels from source to destination.

Figure 4-8 *Strict-source-route concept*



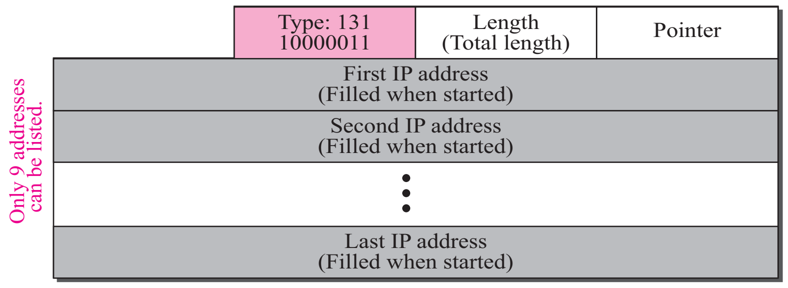
Loose-Source-Route Option

A **loose-source-route option** is similar to the strict source route, but it is more relaxed. Each router in the list must be visited, but the datagram can visit other routers as well. Figure 4-9 shows the format of the loose source route option.

Timestamp

A **timestamp option** is used to record the time of datagram processing by a router. The time is expressed in milliseconds from midnight, Universal Time. Knowing the time a

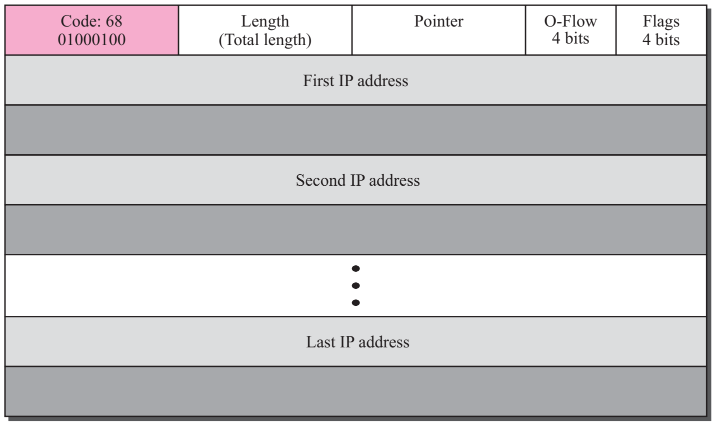
Figure 4-9 *Loose-source-route option*



datagram is processed can help users and managers track the behavior of the routers in the Internet. We can estimate the time it takes for a datagram to go from one router to another. We say *estimate* because, although all routers may use Universal Time, their local clocks may not be synchronized.

However, non-privileged users of the Internet are not usually aware of the physical topology of the Internet. Consequently, a timestamp option is not a choice for most users. Figure 4-10 shows the format of the timestamp option.

Figure 4-10 *Timestamp option*



In this figure, the definitions of the code and length fields are the same as before. The overflow field records the number of routers that could not add their timestamp because no more fields were available. The flags field specifies the visited router responsibilities. If the flag value is 0, each router adds only the timestamp in the pro- vided field. If the flag value is 1, each router must add its outgoing IP address and the timestamp. If the value is 3, the IP addresses are given, and each router must check the given IP address with its own incoming IP address. If there is a match, the router overwrites the IP address with its outgoing IP address and adds the timestamp (see Figure 4-11).

Figure 4-11 *Use of flag in timestamp*

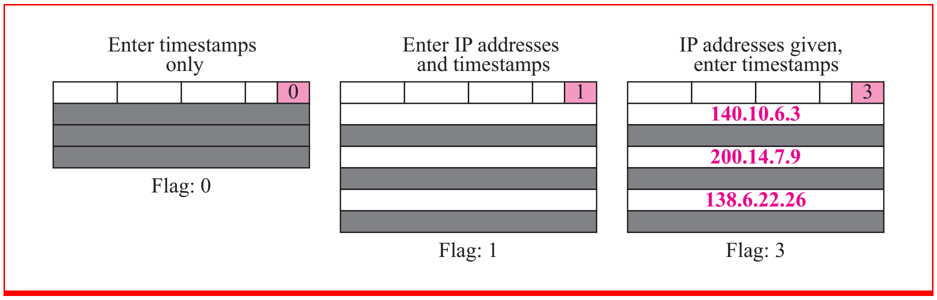


Figure 4-12 shows the actions taken by each router when a datagram travels from source to destination. The figure assumes a flag value of 1.

Figure 4-12 *Timestamp concept*

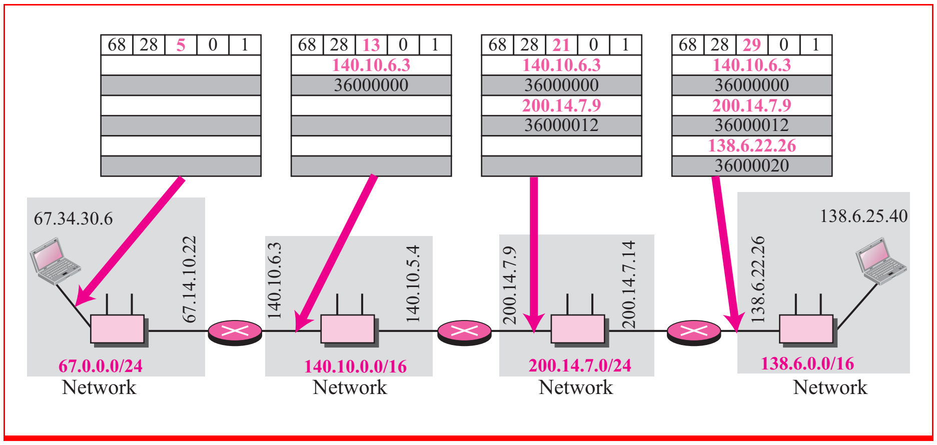
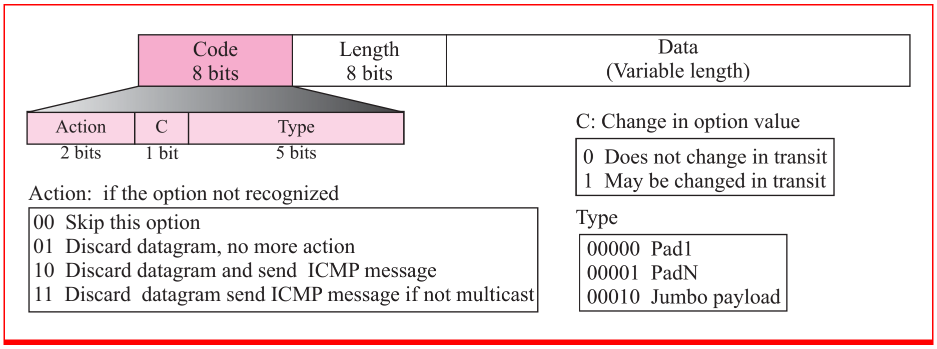
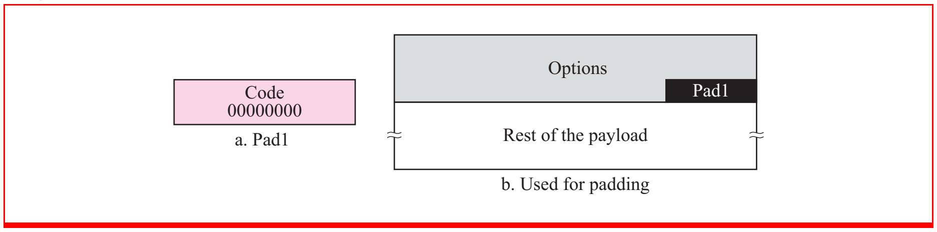


Figure 4-15 *The format of options in a hop-by-hop option header*


Code
8 bits
Length
8 bits
Data
(Variable length)
Action
2 bits
C
1 bit
Type
5 bits
C: Change in option value
0 Does not change in transit
1 May be changed in transit
Action: if the option not recognized
00 Skip this option
01 Discard datagram, no more action
10 Discard datagram and send ICMP message
11 Discard datagram send ICMP message if not multicast
Type
00000 Pad1
00001 PadN
00010 Jumbo payload

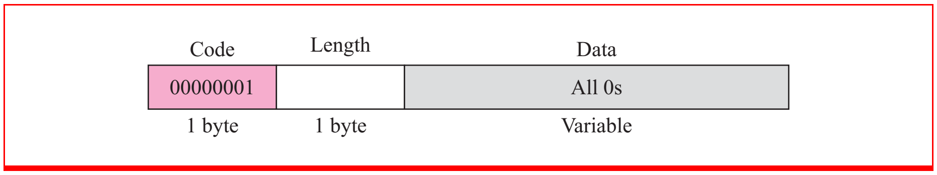
- **Pad1.** This option is 1 byte long and is designed for alignment purposes. Some options need to start at a specific bit of the 32-bit word (see the jumbo payload description to come). If an option falls short of this requirement by exactly one byte, Pad 1 is added to make up the difference . Pad 1 contains neither the option length field nor the option data field. It consists solely of the option code field with all bits set to 0 (action is 00, the change bit is 0, and type is 00000). Pad1 can be inserted anywhere in the hop-by-hop option header (see Figure 4-16).

Figure 4-16 *Pad1*



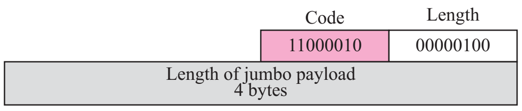
- **PadN.** PadN is similar in concept to Pad1. The difference is that PadN is used when 2 or more bytes are needed for alignment. This option consists of 1 byte of option code, 1 byte of the option length, and a variable number of zero padding bytes. The value of the option code is 1 (action is 00, the change bit is 0, and type is 00001). The option length contains the number of padding bytes. See Figure 4-17.

Figure 4-17 *PadN*



- **Jumbo payload.** Recall that the length of the payload in the IP datagram can be a maximum of 65,535 bytes. However, if for any reason a longer payload is required, we can use the jumbo payload option to define this longer length. The jumbo payload option must always start at a multiple of 4 bytes plus 2 from the beginning of the extension headers. The jumbo payload option starts at the $(4n + 2)$ byte, where n is a small integer. See Figure 4-18.

Figure 4-18 *Jumbo payload*



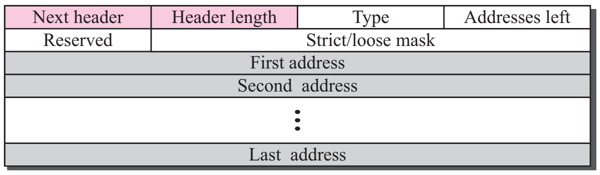
Destination Option

The **destination option** is used when the source needs to pass information to the desti- nation only. Intermediate routers are not permitted access to this information . The format of the destination option is the same as the hop-by-hop option. So far, only the Pad1 and PadN options have been defined.

Source Routing

The source routing extension header combines the concepts of the strict source route and the loose source route options of IPv4. The source routing header contains a mini- mum of seven fields (see Figure 4-19). The first two fields, next header and header length, are identical to that of the hop-by-hop extension header. The type field defines loose or strict routing. The addresses left field indicates the number of hops still needed to reach the destination. The strict/loose mask field determines the rigidity of routing. If set to strict, routing must follow exactly as indicated by the source. If, instead, the mask is loose, other routers may be visited in addition to those in the header.

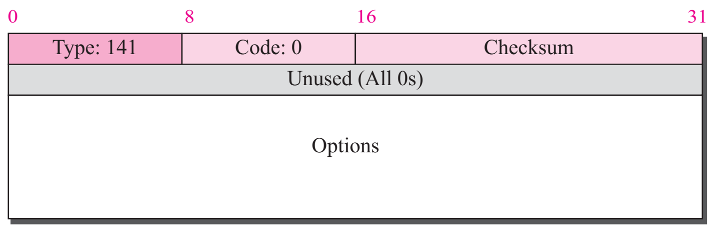
Figure 4-19 *Source routing*



The destination address in source routing does not conform to our previous defini- tion (the final destination of the datagram). Instead, it changes from router to router. For example, in Figure 4-20, Host A wants to send a datagram to Host B using a spe- cific route: A to R1 to R2 to R3 to B. Notice the destination address in the base headers. It is not constant as you might expect. Instead, it changes at each router. The addresses in the extension headers also change from router to router.

lowing two pieces of information in the option field: its link-layer address and the link-layer address of the target node. The sender can also include its IP address and the MTU value for the link. Figure 22-51 shows the format of this message.

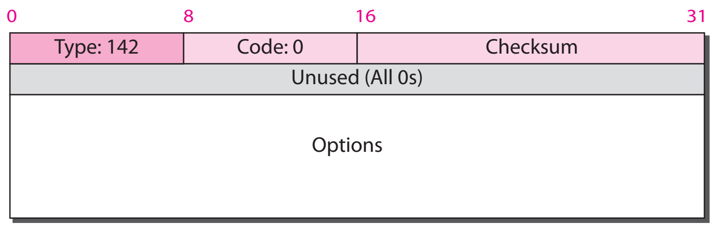
Figure 22-51 *Inverse-neighbor-solicitation message*



Inverse-Neighbor-Advertisement Message

The **inverse-neighbor-advertisement message** is sent in response to the inverse-neighbor-discovery message. The sender of this message must include the link layer address of the sender and the link layer address of the target node in the option section. Figure 22-52 shows the format of this message.

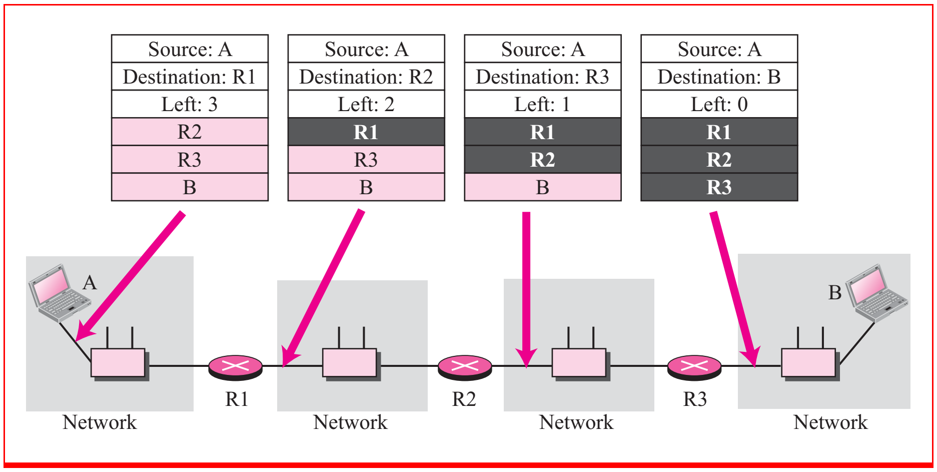
Figure 22-52 *Inverse-neighbor-advertisement message*



22.4.11 Group Membership Messages

The management of multicast delivery handling in IPv4 is given to the IGMPv3 protocol. In IPv6, this responsibility is given to the **Multicast Listener Delivery** protocol. MLDv1 is the counterpart to IGMPv2; MLDv2 is the counterpart to IGMPv3. The material discussed in this section is taken from RFC 3810. The idea is the same as we discussed in IGMPv3, but the sizes and formats of the messages have been changed to fit the larger multicast address size in IPv6. Like IGMPv3, MLDv2 has two types of messages: *membership-query message* and *membership- report message*. The first type can be divided into three subtypes: *general*, *group-specific*, and *group-and-source specific*.

Figure 4-20 *Source routing example*

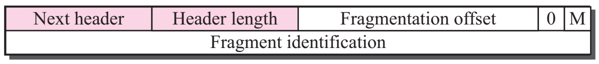


Fragmentation

The concept of **fragmentation** in IPv6 is the same as that in IPv4. However, the place where fragmentation occurs differs. In IPv4, the source or a router is required to frag- ment if the size of the datagram is larger than the MTU of the network over which the datagram travels. In IPv6, only the original source can fragment. A source must use a **Path MTU Discovery technique** to find the smallest MTU supported by any network on the path. The source then fragments using this knowledge.

If the source does not use a Path MTU Discovery technique, it fragments the data- gram to a size of 1, 280 bytes or smaller. This is the minimum size of MTU required for each network connected to the Internet. Figure 4-21 shows the format of the fragmen- tation extension header.

Figure 4-21 *Fragmentation*



Authentication

The **authentication** extension header has a dual purpose: it validates the message sender and ensures the integrity of data. The former is needed so the receiver can be sure that a message is from the genuine sender and not from an imposter. The latter is needed to check that the data is not altered in transition by some hacker.

The format of the authentication extension header is shown in Figure 4-22. The security parameter index field defines the algorithm used for authentication . The authentication data field contains the actual data generated by the algorithm.

Many different algorithms can be used for authentication. Figure 4-23 outlines the method for calculating the authentication data field.

Figure 4-22 *Authentication*

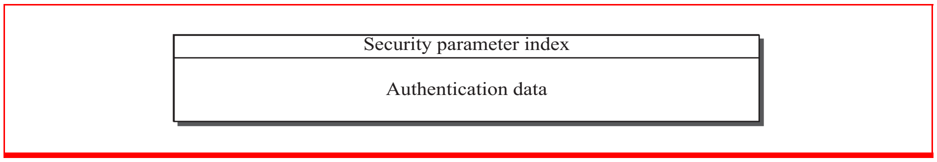
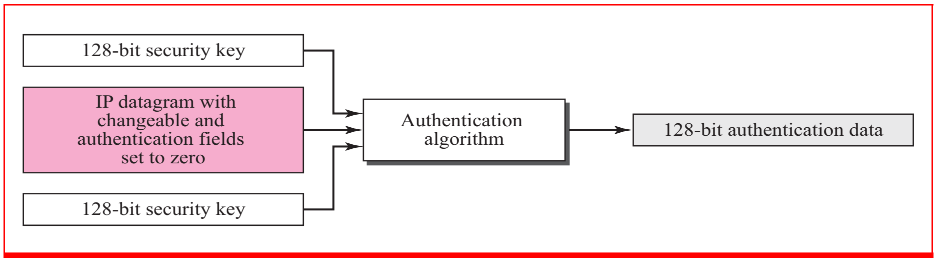


Figure 4-23 *Calculation of authentication data*


128-bit security key
IP datagram with changeable and authentication fields set to zero
128-bit security key
Authentication algorithm
128-bit authentication data

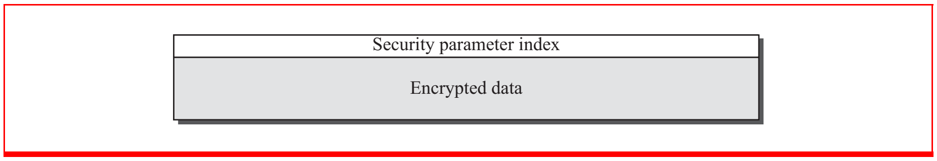
The sender passes a 128-bit security key, the entire IP datagram, and the 128-bit secu- rity key again to the algorithm. Those fields in the datagram with values that change during transmission (for example, hop count) are set to zero. The datagram passed to the algorithm includes the authentication header extension, with the authentication data field set to zero. The algorithm creates authentication data which is inserted into the extension header prior to datagram transmission.

The receiver functions in a similar manner. It takes the secret key and the received datagram (again, with changeable fields set to zero) and passes them to the authentication algorithm. If the result matches that in the authentication data field, the IP datagram is authentic; otherwise, the datagram is discarded.

Encrypted Security Payload

The **encrypted security payload (ESP)** is an extension that provides confidentiality and guards against eavesdropping. Figure 4-24 shows the format. The security parameter index field is a 32-bit word that defines the type of encryption/decryption used. The other field contains the encrypted data along with any extra parameters needed by the algorithm. **Encryption** can be implemented in two ways: transport mode or tunnel mode, which we discussed in the textbook when we discuss IPSec.

Figure 4-24 *Encrypted security payload*



4.3 Comparison of Options between IPv4 and IPv6

The following shows a quick comparison between the options used in IPv4 and the options used in IPv6 (as extension headers).

- The no-operation and end-of-option options in IPv4 are replaced by Pad1 and PadN options in IPv6.
- The record route option is not implemented in IPv6 because it was not used.
- The timestamp option is not implemented because it was not used.
- The source route option is called the source route extension header in IPv6.
- The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6.
- The authentication extension header is new in IPv6.
- The encrypted security payload extension header is new in IPv6.