CHAPTER

The 6LoWPAN Adaptation Layer

16

16.1 TERMINOLOGY

Before digging into the IP protocols developed for smart object networks, several terms that may be confusing need to be defined. According to [156] a LoWPAN is Low-power Wireless Personal Area Networks (LoWPANs) composed of devices conforming to the IEEE 802.15.4-2003 standard defined by the IEEE [129]. IEEE 802.15.4 devices are characterized by short range, low bit rate, low power, and low cost.

IEEE 80.15.4 networks have the following characteristics:

- Small packet size (the maximum transmission unit or MTU on IEEE 802.15.4 links is 127 bytes), which provides even less room for data when including other headers (as discussed in detail in Section 16.2).
- Support for both 16-bit short or IEEE 64-bit extended media access control (MAC) addresses.
- Low data rates; the IEEE 802.15.4 specification allows various data rates from 20 Kbits/s (868 MHz) to 250 Kbits/s (2.45 GHz).
- Support of star and mesh topologies.
- Constrained devices regarding power (e.g., battery-operated devices), memory, and CPU. Most of the time these devices are low cost.
- Large number of deployed devices in the network requiring scalable technologies.
- IEEE 802.15.4 networks are usually ad hoc networks since their location is usually not predetermined. Furthermore, some locations (e.g., mobile smart objects used for asset tracking, wearable sensors) may be moving devices.
- The nodes within a LoWPAN are interconnected by IEEE 802.15.4 links, which are usually unreliable, especially when compared to wired links such as Ethernet or fiber-optic links. This key aspect of such smart object networks has been discussed in Chapter 12.
- It is very common for nodes to be in sleep mode for long periods of time. Depending on the device, it can be in various sleep mode states that have a different impact on the energy consumption while in sleep mode and the speed at which the node can wake up (see Chapter 11 for more details).

A LoWPAN is a Low-power and Lossy Network (LLN) where the links interconnecting the nodes are IEEE 802.15.4 links. When the Internet Engineering Task Force (IETF) 6LoWPAN Working Group was formed, it was decided to exclusively work on the required IPv6 protocol extensions for

Interconnecting Smart Objects with IP. DOI: 10.1016/B978-0-12-375165-2.00016-8

LoWPAN (such as fragmentation and reassembly, header compression, neighbor discovery adaptation, etc.) where the nodes were exclusively interconnected by IEEE 802.15.4 links.

Then the Routing Over Low-power and Lossy network (ROLL) Working Group was formed to deal with routing issues in networks with similar characteristics at the IP layer thus alleviating the restriction of using IEEE 802.15.4 links, since by definition routing operates at the network layer. This led to the use of the more generic term Low-power and Lossy Network (LLN).

Note that the terms "nodes," "routers" (when discussing a routing-related item), and even "devices" (since most smart objects performing sensing or actuating are usually routers) are used interchangeably.

16.2 THE 6LoWPAN ADAPTATION LAYER

Since IPv6 mandates supporting links with an MTU (Maximum Transmission Unit) of 1280 bytes, it was necessary for IEEE 802.15.4 links that have an MTU of 127 bytes to specify an adaptation layer below IP responsible for handling packet fragmentation and reassembly.

The MTU size of IEEE 802.15.4 links was purposely small to cope with limited buffering capabilities and to limit the packet error rate since the bit error rate (BER) is relatively high. Various header compression techniques have been added to the adaptation layer that are specified in [176] and [124]. The compression header techniques originally specified in [176] were improved in [124] in many ways: individual compression on the traffic class (TC) and flow label field, use of share contexts that is particularly useful when using non-link-local addresses, and optimizations for multicast addresses.

The IEEE 802.15.4 frame MTU is 127 bytes minus a set of protocol fields:

- Maximum MAC frame overhead: Frame control (2 bytes) + sequence number (1 byte) + addressing field (up to 20 bytes with the source and destination PAN ID and the source and destination 64-bit extended addresses) + FCS (2 bytes) = 25 bytes.
- MAC security header: 21 bytes (AES-CCM-128), 13 bytes (AES-CCM-64), and 9 bytes (AES-CCM-32).

In the worst case this only leaves 81 bytes (127 bytes − 25 − 21 = 81) for the data payload (IPv6 packets). After removing the size of the IPv6 header (40 bytes), there are 41 bytes left. Next, we must deduct the transport layer protocol header (8 bytes for UDP and 20 bytes for TCP), thus leading to a very short payload for the application layer.

This shows that an adaptation layer is needed to comply with the IPv6 requirement to support a minimum MTU size of 1280 bytes as well as to support compression techniques to reduce protocol overhead.

The 6LoWPAN adaptation layer provides three main services:

- Packet fragmentation and reassembly
- Header compression
- Link layer (layer 2) forwarding when multi-hop is used by the link layer

In most cases the use of efficient compression techniques allows most applications to send their data within a single IPv6 packet.

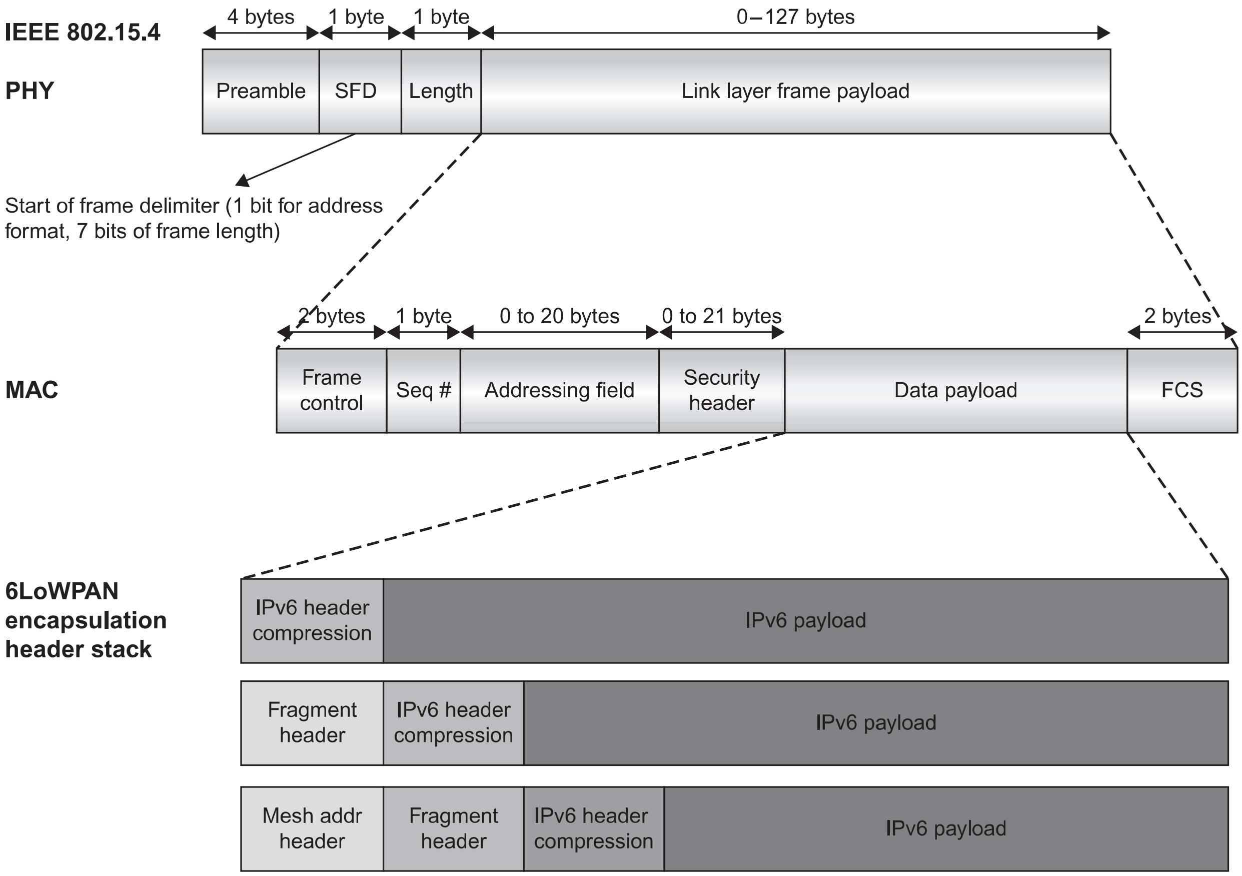


FIGURE 16.1

6LoWPAN encapsulation header stack.

As previously discussed in Chapter 12, IEEE 802.15.4 frames support the use of 16-bit short addresses (temporary addresses allocated by the personal area network or PAN coordinator or 64-bit long addresses (24 bits are used for the organizational unique identifier; OUI + 40 bits assigned by the chipset manufacturer).

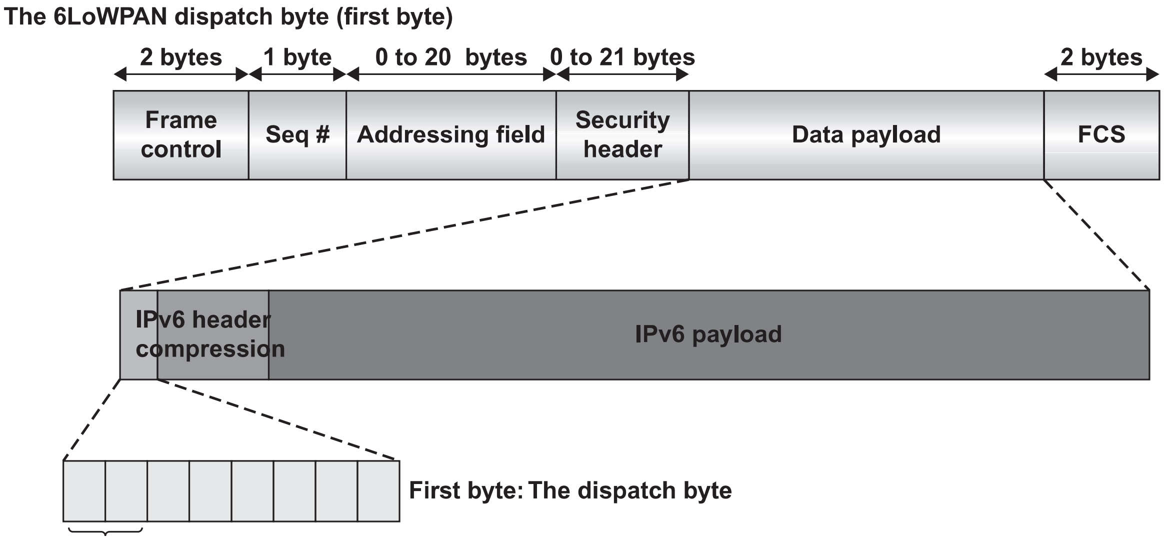
Similar to IPv6, the 6LoWPAN adaptation layer makes use of header stacking (headers are added only when needed).

The 6LoWPAN adaptation currently supports three headers: a mesh addressing header, the fragment header, and the IPv6 header compression header (they must appear in that order when present).

The 6LoWPAN adaptation layer defines what is called the "encapsulation header stack," which precedes each IPv6 datagram. The encapsulation header stack is shown in Figure 16.1.

As shown in Figure 16.2, the first byte of the encapsulation header identifies the next header. For example, if the first 2 bits are equal to 11, the next header is a fragmentation header.

If the first 8 bits are equal to 01000001, what follows is an IPv6 *uncompressed* packet. In contrast, a value of 01000010 indicates that what follows is a header related to a compressed header using HC1 compression (see Section 16.2.3 for details on 6LoWPAN header compression techniques).



00	**Not a 6LoWPAN frame**
01	IPv6 addressing header
10	Mesh header
11	Fragmentation header (6 lower bits are 100xxx)

FIGURE 16.2

Dispatch byte of the IPv6 header compression header.

16.2.1 The Mesh Addressing Header

The mesh addressing header is used in conjunction with a mesh-under "routing" approach where nodes that are not in direct communication make use of multi-hop "routing" at the link layer using link layer addresses. According to IEEE 802.15.4, only full function devices (FFDs) perform mesh-under operation. Reduced function devices (RFDs) systematically send all of their traffic to FFDs.

The source and destination nodes are then referred to as the originator and final destination, respectively.

As shown in Figure 16.2, the first 2 bits of the dispatch byte identify the presence of a mesh-header and are equal to 10.

Figure 16.4 shows the various bits of mesh addressing type and header:

- Bit 2 (V, Very first bit):
  0: The originator address is an IEEE extended 64-bit address (EUI-64).
  1: The originator address is a short 16-bit address.

The 6LoWPAN dispatch byte (first byte)

First byte: The dispatch byte

Pattern	**Header type**
00 xxxxxx	NALP - not a LoWPAN frame
01 000001	IPv6 - uncompressed IPv6 addresses
01 000010	LOWPAN_HC1-LOWPAN_HC1 compressed IPv6
01 000011	reserved - reserved for future use
...	reserved - reserved for future use
01 001111	reserved - reserved for future use
01 010000	LOWPAN_BCO - LOWPAN_BCO broadcast
01 010001	reserved - reserved for future use
...	reserved - reserved for future use
01 111110	reserved - reserved for future use
01 111111	ESC - additional dispatch byte follows
10 xxxxxx	MESH - Mesh header
11 000xxx	FRAG1 - fragmentation header (first)
11 001000	reserved - reserved for future use
...	reserved - reserved for future use
11 011111	reserved - reserved for future use
11 100xxx	FRAGN - fragmentation header (subsequent)
11 101000	reserved - reserved for future use
...	reserved - reserved for future use
11 111111	reserved - reserved for future use

FIGURE 16.3

Value of the 6LoWPAN dispatch byte.

- Bit 3 (F, Final destination):
  0: The final address is an IEEE extended 64-bit address (EUI-64).
  1: The final address is a short 16-bit address.
- Bits 4 through 7 (HopLeft): The HopLeft field value is decremented by each node before sending the packet to its next hop. When the HopLeft field reaches the value of 0, the packet is simply discarded. When equal to 15, an additional byte (called the deep hops left) immediately follows when forwarding along a path with more than 14 hops is needed.
- The originator and final link layer address fields then follow (16 or 64 bits).

It is possible to use short 16-bit addresses for broadcast and 64-bit addresses as a source address since the V and F permit the use of different link layer address formats.

With mesh-under routing it is necessary to provide the originator and final destination as well as the hop-by-hop source and destination addresses.

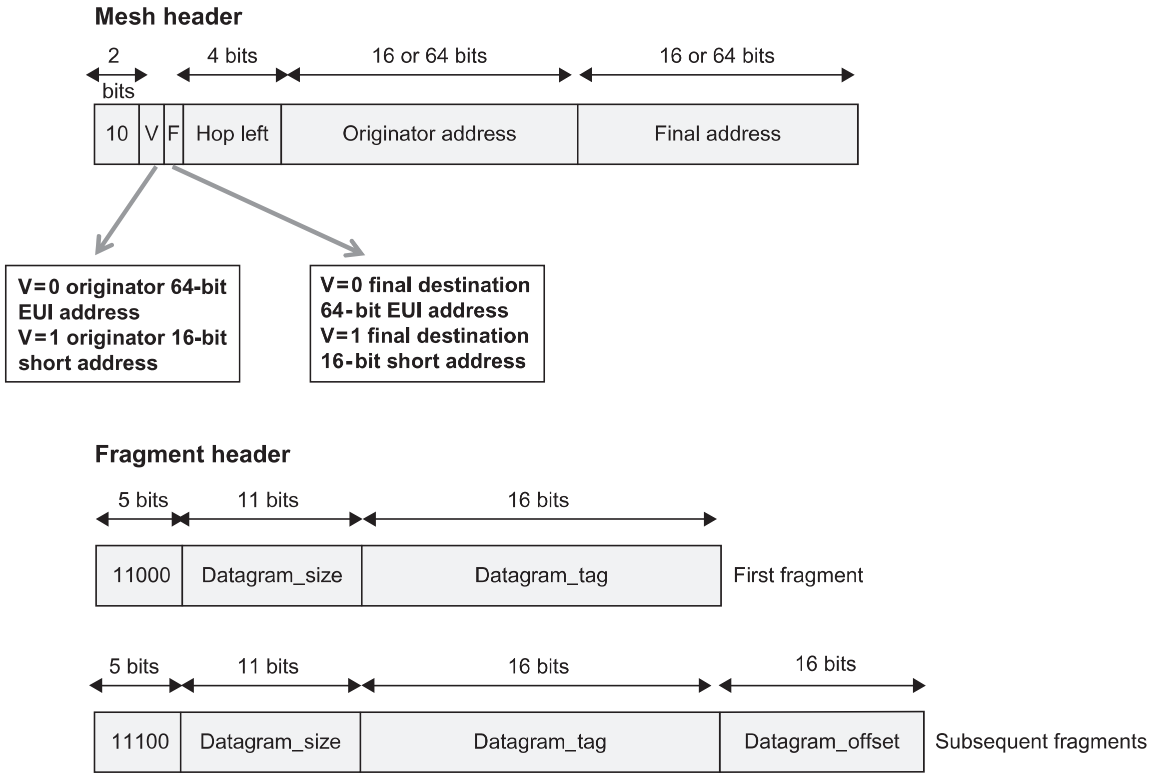


FIGURE 16.4

6LoWPAN mesh and fragmentation headers.

Thus the set of link layer addresses is as follows. When a node A sends a frame to a final destination C via the node B:

- The originator address of the mesh header is set to the link layer address of A.
- The final destination address of the mesh header is set to the link layer address of C.
- The source address of the IEEE 802.15.4 frame is the address of the node sending the frame (A).

The destination address of the IEEE 802.15.4 frame is the link layer address of the next-hop node as determined by the mesh-under routing protocol (B in this example). Upon receiving the frame, B performs the following process:

- The hop left field is decremented.
- If the hop left field is not equal to 0 (if equal to 0, the frame is discarded), then B determines that the next hop is C.
- The originator and final destination address of the mesh header are unchanged.
- The source address of the IEEE 802.15.4 frame is set to the link layer address of B.
- The destination address of the IEEE 802.15.4 frame is set to the link layer address of C.

This is similar to the mode of operation with IP routing over a link layer where the source and destination addresses of the IP packet are never changed, and the source and destination addresses present in the link layer frame correspond to the address of two adjacent nodes (connected by a common link layer).

As previously discussed, there is no mesh-under protocol defined. For further discussion about routing at multiple layers see Chapter 5.

16.2.2 Fragmentation

Fragmentation may be required at the 6LoWPAN adaptation layer when the IPv6 payload cannot be carried within a single IEEE 802.15.4 frame because it exceeds the MTU size. In this case, the link frame is broken into multiple link fragments using the fragment header shown in Figure 16.4. All fragment sizes are expressed in units of 8 bytes. The first fragment does not contain a datagram offset, which makes it slightly different from the subsequent fragment.

Description of the fragment fields (see Figure 16.4):

- datagram_size: This 11-bit field is used to indicate the size in 8-byte units of the original IPv6 packet (or IPv6 fragmentation also taking place at the IP layer). Link layer fragmentation supports a 1280-byte packet as mandated by the IPv6 specification [51]. The datagram_size may only be needed in the first link fragment and then elided in other link fragments. The drawback of this approach is that subsequent link fragments (other than the first link fragment) may arrive first, especially in the presence of multi-hop routing. In this case the receiver would not know how much memory should be allocated for the entire frame.
- datagram_tag: This field is used in conjunction with the IEEE 802.15.4 source address (or originator address if a mesh header is present), the IEEE 802.15.4 destination address (or the final destination address if a mesh header is present), and the datagram_size to uniquely identify the fragmented frame and must be identical for all link fragments. It is recommended to increment the datagram_tag for successive fragmented frames.
- datagram_offset: The 8-bit datagram_offset field is present in all link fragments except the first fragment and indicates the offset in 8-byte units from the beginning of the payload datagram.

[176] specifies the use of a reassembly timer that is started when receiving the first link fragment and upon the expiration of which, if not all link fragments have been received, all fragments must be discarded. The maximum value of the reassembly timer is 60 seconds.

16.2.3 6LoWPAN Header Compression

16.2.3.1 Header Compression Using LOWPAN_HC1 and LOWPAN_HC2

A plethora of IP compression techniques have been designed over the past decade (e.g., ROHC, see [18]). These techniques rely on stateful flow-based compression optimized for long-lived flows. The basis of this principle consists of suppressing common values within a long-lived flow, which is a very efficient approach for long-lived flow between two nodes. Unfortunately these compression techniques are less suited to 6LoWPAN networks with typically short-lived flows (often these devices send a few packets and then go back into sleep mode with the exception of infrequent firmware upgrades that may require large flows to be exchanged). Thus the whole idea of header stateless compression techniques in 6LoWPAN consists of avoiding information redundancy across layers as opposed to between IP packets that belong to the same long-lived flow as ROCH.

The general idea of the 6LoWPAN header compression is to derive the IP address from link layer addresses to avoid needless information duplication and suppression of IPv6 headers that have common values (typically elide the fields that have a value of 0). Furthermore, the use of shared contexts such as the use of a common network prefix for the LoWPAN allows address compression of IPv6 global addresses.

6LoWPAN header compression can either be stateless or stateful and is flow independent. Several of the IPv6 headers have common values and are easily compressed; for example, the IP version (v6), the flow label, TC, and so on. Furthermore, information such as the IPv6 interface ID can be derived from the link layer frame when using extended 64-bit 802.15.4 addresses.

[176] first focused on highly optimizing the compression unicast link-local addresses. A new encoding technique (IPHC) was then introduced in [124] to cope with multicast addresses and non-link-local addresses along with other optimizations such as the individual compression of the IPv6 flow label and TC field.

⚠ More than likely, the header compression part of [176] will be deprecated at some point to only support one header compression technique such as IPHC. There has been no decision to deprecate the header compression defined in [176], so both header compression techniques are described in the following section.

16.2.3.1.1 The HC1 Compression Technique

The HC1 compression technique relies on the following observations:

- IP version is always 6.
- Since HC1 is optimized for link-local addresses, the IPv6 interface ID (bottom 64 bits of the IPv6 address) can be inferred from the link layer MAC address.
- The packet length can be inferred from the frame length field of the IEEE 802.15.4 frame (or from the datagram size field of the fragment header when present).
- Common value for the TC and flow label is 0 (as shown later in Section 16.2.3.3, IPHC allows for individual compression of these fields).
- Next header is UDP, TCP, or ICMP.

These observations allow a considerable reduction of the protocol overhead. The only IPv6 header field that cannot be compressed and must be carried in full is the 1-byte hop limit field. This leads to only 3 bytes instead of the 40-byte IPv6 header: 1 byte for the dispatch byte (equal to 01000010), followed by a 1-byte HC1 byte, and 1 byte for the hop limit field, as shown in Figure 16.5.

When set, bit 7 of the HC1 byte allows for the compression of the next header of the original IPv6 header.

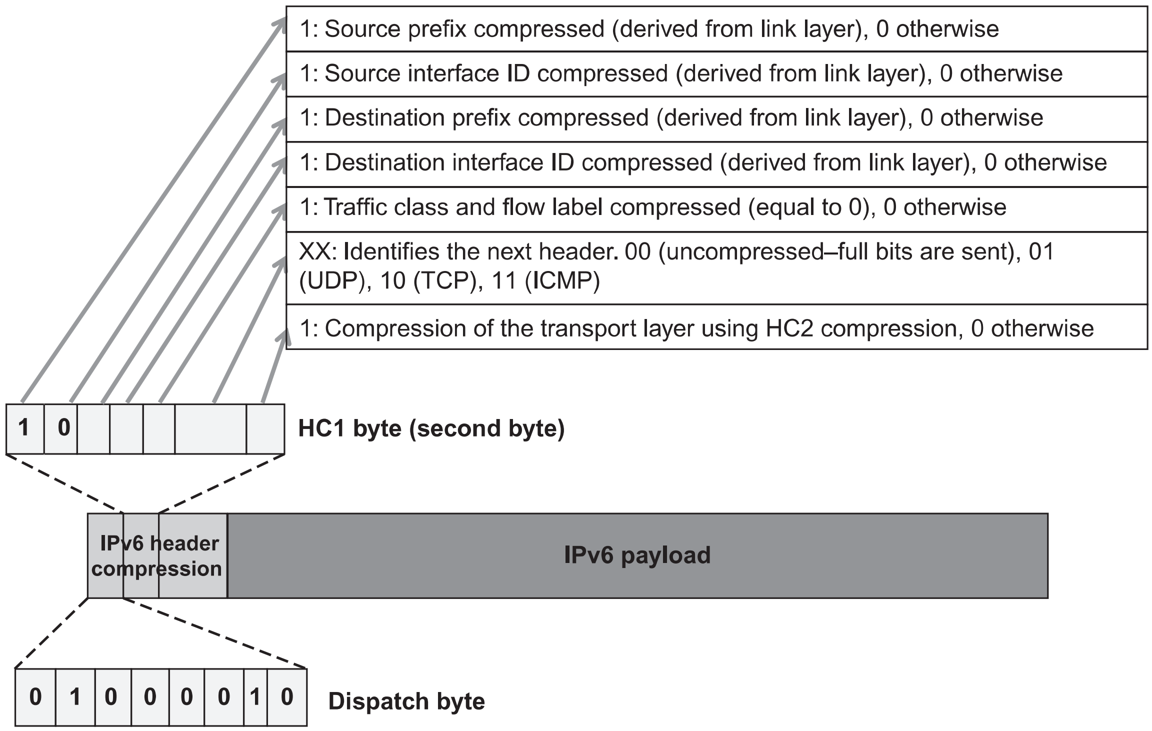


FIGURE 16.5

6LoWPAN HC1 byte.

The non-compressed fields must follow the HC1 byte in this particular order: source address prefix (64 bits) and/or interface ID (64 bits), destination address prefix (64 bits) and/or interface ID (64 bits), TC (8 bits), flow label (20 bits), and next header (8 bits).

16.2.3.2 The HC_UDP Compression Technique (HC2 Byte)

When bit 7 of the HC1 byte is set, it indicates more header compression according to the HC2 encoding format. If bits 5 and 6 of the HC1 byte are equal to 0 and 1, respectively, this indicates the compression of the UDP header (called HC_UDP encoding). In this case, the HC2 byte immediately follows the HC1 byte (thus before the IP hop limit field) and provides information on the UDP header compression scheme. The HC_UDP compression technique allows compression of the UDP header to various degrees. When non-compressed, the UDP fields must appear in the same order as the original UDP header (source port, destination port, length, and checksum).

HC_UDP encoding allows compression of the source and destination UDP ports in addition to the length field. The length field can be inferred from the length field of the IEEE 802.15.4 frame. According to [176], the UDP checksum is never compressed and always carried in full, but improvements have been added to allow UDP checksum compression [124] and are described later in Section 16.2.3.6.

The main idea for compressing the source and destination UDP port uses a short_value 4-bit field instead of the original 16-bit field. The original 16-bit field is simply obtained by the formula short_value+61616 (0xF0B0).