CHAPTER 8

Link Layer

8.1 Introduction

This chapter explains the operation of the link layer. This layer is responsible for controlling, negotiating and establishing the links, selecting frequencies to transmit data, supporting different topologies and supporting various ways for exchanging data. The position of Link Layer in the LE Protocol stack is shown in Figure 8.1. It sits on top of the Physical Layer and provides services to the L2CAP layer.

8.2 Overview of Link Layer States

The operation of the link layer can be described in terms of a very simple state machine with the following five states:

1. Standby State;
2. Advertising State;
3. Scanning State;
4. Initiating State;
5. Connection State.

In general LE devices support one single instance of the state machine, though multiple instances of the state machine are also permitted by the specification. If a device supports multiple instances of the state machine then only one state can be active at a time for each state machine. Besides this, at least one state machine that supports the advertising state or scanning state is mandatory.

The link layer state machine is shown in Figure 8.2. The five states are explained in brief below and will be explained in detail later in this chapter.

8.2.1 Standby State

This is the default state of the link layer. In this state, no packets are received or transmitted. A device can enter into this state from any of the other states.

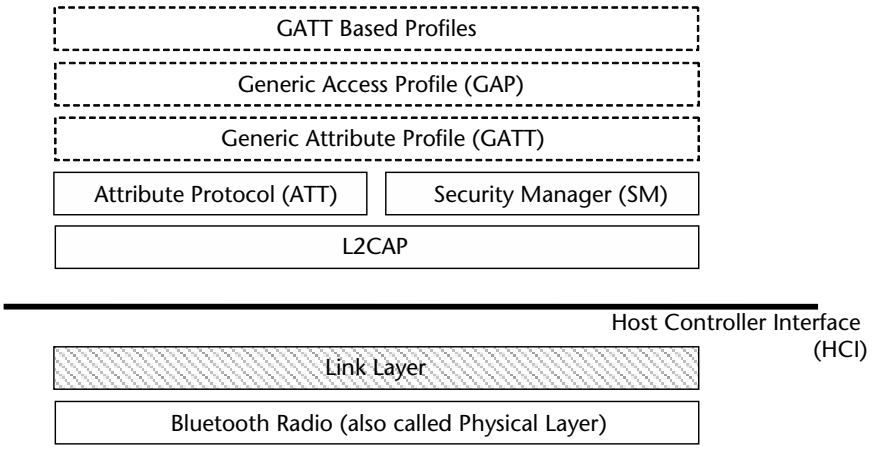

Figure 8.1 Link layer in LE protocol stack.

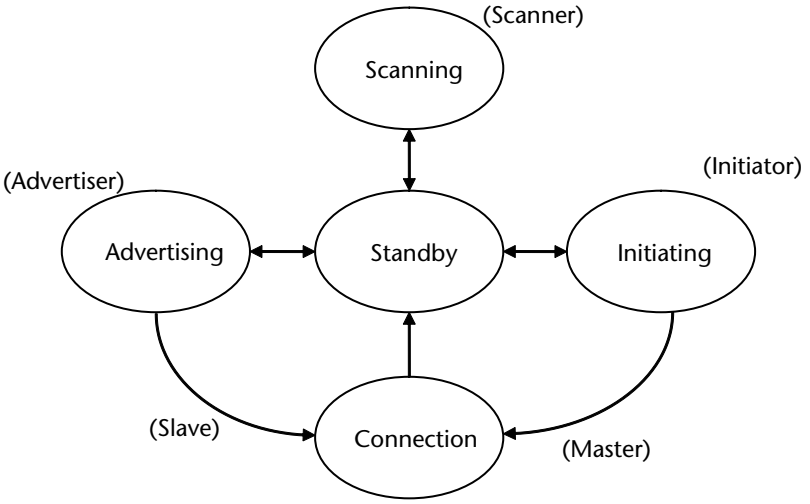

Figure 8.2 Link layer state machine—state diagram and roles.

8.2.2 Advertising State (Advertiser)

In the advertising state the link layer transmits advertising packets. It may also listen to the devices that respond to the advertising packets and then respond to those devices accordingly. This state can be entered from the standby state when the link layer decides to start advertising. A link layer in this state is known as the Advertiser.

An example of an Advertiser could be a thermometer which continuously advertises that "I'm a thermometer" to the devices around it. It may advertise additional information as well. For example it may also advertise that it has data to send. Any device in the vicinity that is in the scanning state may pick up that packet and query the thermometer for additional information or decide to connect to the thermometer.

8.2.3 Scanning State (Scanner)

In the scanning state the link layer listens to the packets from the Advertiser and may request the Advertiser to provide some additional information. This state can be entered from the standby state when the link layer decides to start scanning. A link layer in this state is known as the Scanner.

8.2.4 Initiating State (Initiator)

In the initiating state the link layer listens to the packets from the Advertiser and responds to those packets by initiating a connection. This state can be entered from the standby state when the Scanner decides to initiate a connection with the Advertiser. A link layer in this state is known as the Initiator.

8.2.5 Connection State (Master or Slave)

In the connection, the device is connected to another device. Two roles are defined—Master Role and Slave Role. This state can be entered either from the initiating state or from the advertising state.

- When entered from the initiating state, the device acts as a Master.
- When entered from the advertising state, the device acts as a Slave.

As per specifications 4.0, a link layer in the Slave role could communicate with only one device in the Master role. This means that the link layer did not support scatternet scenarios similar to the ones that are supported in BR/EDR. This helped in a simple design of the link layer.

Specifications 4.1 allowed the link layer to support scatternet scenarios similar to the ones in BR/EDR. The link layer is allowed to be the Master of one piconet and the Slave of another or to be the Slave in different piconets.

8.3 Device Address

In contrast to BR/EDR where a device has only one Bluetooth device address (BD_ADDR), an LE device may either have a Public Address or a Random Address or both. It is mandatory to have at least one of these addresses so that the device can be identified.

8.3.1 Public Device Address

This address is similar to the BD_ADDR for BR/EDR devices. This was explained in Chapter 2. In fact in case of dual mode devices, both the LE controller and the BR/EDR controller should have the same address. The BD_ADDR of BR/EDR is referred to as the public device address by LE. Public Device Address is a globally unique 48-bit address. It is similar to an Ethernet MAC address and is, in fact, administered by the same organization, IEEE.

The public device address (BD_ADDR) consists of two fields:

1. 24-bit company id assigned by IEEE Registration authority. This is called the Organizationally Unique Identifier (OUI) [24 most significant bits] and is different for each company.
2. 24-bit unique number assigned by the company to each controller. [24 least significant bits]. This is different for each controller manufactured by the company.

8.3.2 Random Address

Random Address is a privacy feature of LE where the device can hide its real address and use a random address that can change over time. So the real address is not revealed at any time. This helps to ensure that a device cannot be tracked.

Consider an example where a person is wearing LE enabled shoes which keep on transmitting data about the number of steps that the person has walked. Somebody can listen to those packets and track the person. So wherever the person is moving, he or she can be followed by just listening to the packet transmitted from the shoes if the shoes are using a fixed address.

To prevent tracking, a random address can be used to transmit. A different random address can be used every time the shoes transmit data. This ensures that the person cannot be tracked. LE provides a mechanism to resolve that random address so that only the intended recipient of the data keeps getting the data and knowing that it is coming from the same device even though it may be from different addresses.

The Generic Access Profile defines the random address to be of two types:

1. Static Address: A device may choose to initialize it's static address to a new value after each power cycle but cannot change it while it is still powered. If the device changes its static address, the peer device will not be able to connect to it with the old address that they may have stored.
2. Private Address: The private address may further be of following two types:
    1. Non-resolvable private address: The peer device can never discover the real address.
    2. Resolvable private address: The peer device can derive the real address using the random address and the link key of the connection.

The format of different types of device addresses is illustrated in Figure 8.34 towards the end of this chapter. These addresses will be explained further in Chapter 14.

8.4 Physical Channel

As mentioned in the previous chapter, LE uses 40 RF channels in the 2.4 GHz ISM band. These channels are spaced 2 MHz apart.

The RF channels are divided into two physical channels:

- Advertising Physical Channel: This physical channel uses three RF channels, viz channel 0, 12 and 39 for the following activities:

- Discovering devices.
- Creating a connection.
- Broadcasting and receiving data.

- Data Physical Channel: This physical channel uses the remaining 37 RF channels for communication between devices that are in the connection state.

The advertising channels are carefully chosen by the Bluetooth specification to be spread far apart to ensure that if there is interference from other devices in one of the frequency ranges, then at least other frequency ranges are available for advertising. For example, if there is interference in the frequency band 2400 MHz to 2420 MHz, then only channel 0 experiences interference and channels 12 and 39 can still be used for advertising.

Another reason behind the choice of these channels is to reduce interference with WiFi. Many of the devices (like mobile phones, tablets, and laptops) which support Bluetooth have a WiFi radio as well. So it's important that the interference with WiFi is reduced as much as possible.

When a WiFi device is switched on, it uses WiFi channels 1, 6, and 11 as default. The frequency range of the advertising channels is chosen carefully so as not to overlap with channels 1, 6 and 11 of WiFi.

The 40 RF channels are mapped to either a Data Channel Index or an Advertising Channel Index. This is shown in Figure 8.3. For example:

- RF Channel 1 is mapped to Data Channel 0.
- RF Channel 2 is mapped to Data Channel 1.
- RF Channel 12 is mapped to Data Channel 11.
- RF Channel 0 is mapped to Advertising Channel 37.

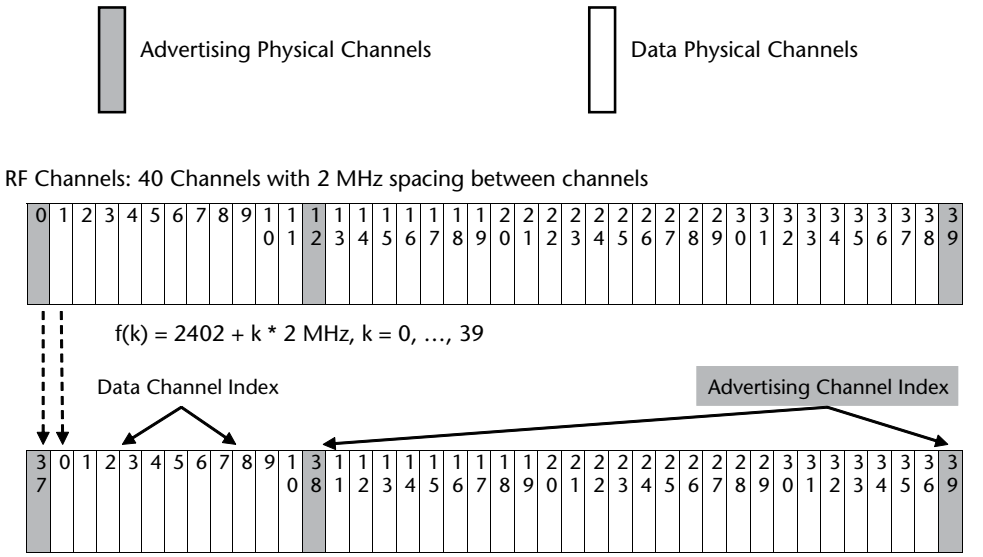

Figure 8.3 Advertising physical channels, data physical channels, and mapping to RF channels.

- RF Channel 12 is mapped to Advertising Channel 38.
- RF Channel 39 is mapped to Advertising Channel 39.

The link layer uses only one physical channel at any given time.

8.5 Channel Map

Similar to BR/EDR, the Master's link layer classifies the data channels as used or unused. If the Master's link layer suspects interference on any of the channels, it can mark the channel as unused. This has two advantages:

- The channels which potentially have interference can be excluded from the frequency hopping pattern. This reduces the number of retransmissions that would have been done if the packets had been transmitted on those channels.
- The channels which have interference may, in fact, be used by other technologies which are also sharing the ISM band. So this reduces the impact of Bluetooth transmissions on those technologies.

The information about used and unused channels is provided as a bitmap by the Master to the Slave so that both the devices can use the same hopping frequencies. This channel bitmap is called a Channel Map.

The Master can keep on marking channels as unused if it suspects interference until it reaches a minimum of 2 used channels. This is the minimum number of channels that the Master has to use.

8.6 Adaptive Frequency Hopping

LE uses adaptive frequency hopping to hop frequencies across the 37 data channels. The algorithm used is very simple:

$$f_{n+1} = (f_n + \text{hopIncrement}) \bmod 37$$

- If f_n is a used channel, then it is used as it is.
- If f_n is an unused channel, then it is remapped to the set of good channels. (The link layer builds a remapping table to map all the unused channels to used channels.)

The Master sets the hop increment value at the time of creation of a connection. It sets it to a random value between 5 and 16. It may be noted that hop increment is a new concept that has been introduced in LE to simplify the calculation of the next hopping frequency. In BR/EDR, the next hop in the frequency hopping pattern was a function of the Master's parameters like clock and BD_ADDR. LE simplifies this by just using a random value between 5 and 16 as a hop increment to calculate the next hooping frequency. A simpler algorithm, of course, leads to lesser number of gates when this is implemented in silicon (thereby reducing cost)

and lesser amount of processing (thereby reducing power consumption). A random value is needed since if, by chance, there is a collision with another Master on one of the frequencies, the subsequent frequencies for the two Masters will be based on different random numbers. So, further collisions can be avoided.

8.7 Events

The physical channel is subdivided into time units which are known as events. There are two types of events:

- Advertising Events.
- Connection Events.

8.7.1 Advertising Events

Advertising events are used for transmissions on the advertising physical channels. At the start of each advertising event, the Advertiser sends an advertising packet. The Scanner receives this packet and depending on the type of the advertising packet, it may send a request back to the Advertiser. The Advertiser responds to that request within the same advertising event. After that the advertising event is closed. The Advertiser uses the next advertising channel for the next advertising packet. An example of three advertising events is shown in Figure 8.4.

Figure 8.5 shows a sniffer capture of six advertising events.

1. Frame #425: First advertising event (ADV_IND) on Channel 38.
2. Frame #426: Second advertising event (ADV_IND) on Channel 39.
3. Frame #427, #428, #429: Third advertising event on Channel 37. This consists of three packets.
   a. ADV_IND packet from Advertiser to Scanner.
   b. SCAN_REQ packet from Scanner to Advertiser.

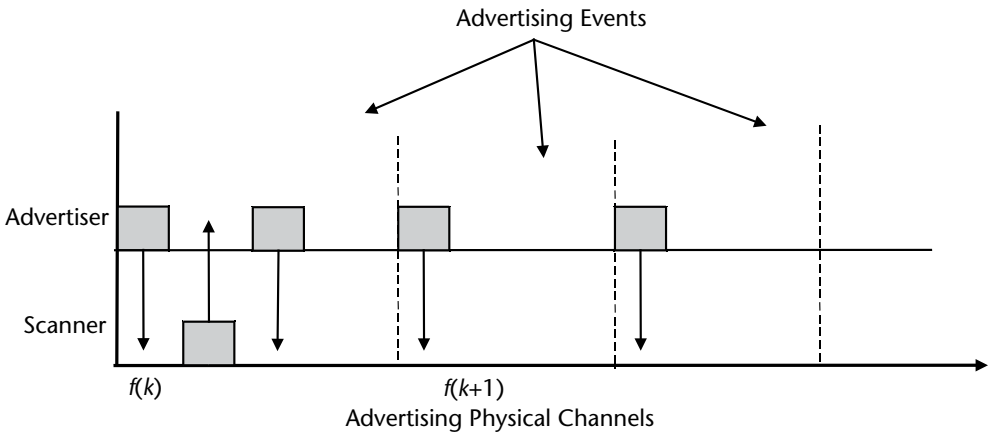

Figure 8.4 Advertising events.

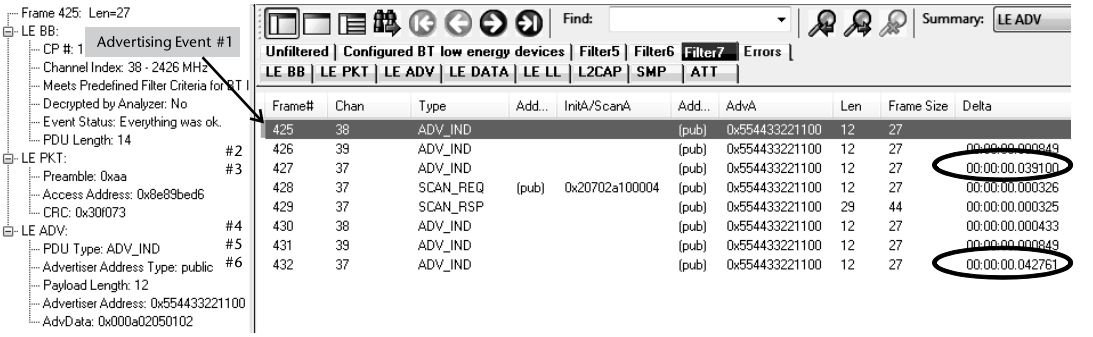

Figure 8.5 Example of advertising events.

c. SCAN_RSP packet from Advertiser to Scanner in response to the SCAN_REQ packet.

d. All these packets comprise one single advertising event and are sent on the same channel.

4. Frame #430: Fourth advertising event (ADV_IND) on Channel #38.
5. Frame #431: Fifth advertising event (ADV_IND) on Channel #39.
6. Frame #432: Fourth advertising event (ADV_IND) on Channel #37.

This indicates that the Advertiser is advertising on all the three advertising channels consecutively in rotation: Channel 37, Channel 38, Channel 39 and then again Channel 37, Channel 38, and so on.

There is one more thing worth noting from the Delta time stamps in the last column. The Advertiser advertises on Channels 37, 38, and 39 in quick succession. The approximate delta time between those is 300 to 400 microseconds. After that the Advertiser waits for about 39 or 42 milliseconds before starting the next cycle of advertisements. This means that there are quite big gaps between the advertisements so that the Advertiser can save battery power during that time.

The timing of advertising events is determined by two parameters:

- Advertising Interval (advInterval): The advertising interval ranges from 20 ms to 10.24 seconds.
- Advertising Delay (advDelay): The advertising delay is a random value that ranges from 0 to 10 ms.

The time between start of two consecutive advertising events, T_advEvent is defined as follows:

$$T\_advEvent = advInterval + advDelay$$

The connection latency and advertising interval are inversely proportional. If the advertising interval is high, it may take longer to establish connections, while a lower advertising interval would lead to the establishment of a faster connection. A lower connection interval would also mean that advertising packets are sent out more frequently before a connection is established, thereby increasing the power

consumption. Hence, a trade-off needs to be made between connection setup time and battery life.

Specifications 4.1 introduced a low duty cycle mode for directed advertising. This is useful in cases where a reconnection is desired but a fast reconnection is not mandatory, or if it is not known whether the device that is supposed to make a connection is in range. With a low duty cycle, less power would be consumed because the advertisement packets would be sent out at a lower rate. SIG recommends scan intervals based on the mode of advertisement used, but it is up to the application to select the desired mode and the interval as per preference (i.e., if the peripheral wants a faster connection just after turning on, it can do a fast advertisement (low advertisement interval), it may then subsequently ask for slower connections depending on the type of application used).

8.7.2 Connection Events

Connection events are used to send data packets between the Master and Slave devices. The start of a connection event is called an Anchor Point. At the Anchor Point, the Master transmits a data channel PDU to the Slave. After that the Master and Slave send packets alternately during the connection event. The Slave always responds to a packet from the Master while the Master may or may not respond to a packet from the Slave. The connection event can be closed by either the Master or the Slave. All packets in a connection event are transmitted on the same frequency. Channel hopping occurs at the start of every connection event. An example of three connection events is shown in Figure 8.6.

The timing of connection events are determined by two parameters:

- Connection Event Interval (connInterval): The connInterval is the interval between two successive starting points of connection event. The start of a connection event is called an anchor point. So the connInterval is the time difference between two successive anchor points. It is a multiple of 1.25 ms and in the range of 7.5 ms to 4.0s.

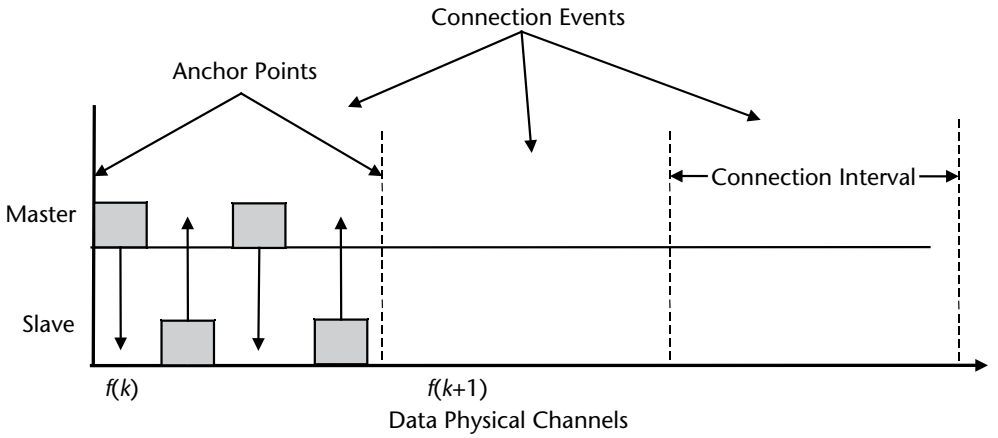

Figure 8.6 Connection events.

- Slave Latency (connSlaveLatency): The connSlaveLatency indicates the number of consecutive connection events that the Slave can skip before listening to the Master. For example, if connSlaveLatency is set to ten then the Slave has to listen to every tenth connection event. If it is set to 0, then the Slave has to listen to every connection event.

Besides these, a Supervision Timeout (connSupervisionTimeout) parameter is used by both the Master and the Slave to detect whether a connection has been lost. If a packet has not been received for a duration of connSupervisionTimeout, then the connection is considered to be lost and no further packets are sent. The host is informed about the loss of connection.

The first connection event is scheduled after the CONNECT_REQ PDU. The master provides two parameters in the CONNECT_REQ PDU to indicate the transmit window:

- transmitWindowOffset: This indicates the time difference between CONNECT_REQ PDU and the transmit window.
- transmitWindowSize: This indicates the size of the transmit window.

The connection interval and advertising interval are two important parameters that impact the battery life of a device. It is important to note that these parameters are not related to each other. While the advertising interval plays a role during connection establishment, connection interval plays a role during data transfer.

As an example, consider an art gallery where paintings are on display. Once the user walks close to a painting, the particulars of the artist and description of the painting may be sent to the user's mobile phone. In this case, it may be acceptable if it takes a few seconds to establish a connection (because the user behavior in this case may be that users walk slowly from one painting to another and may first study the painting before reading the particulars). However, once the connection is established, it would be expected that the data is transferred quickly. The intervals may be fine-tuned accordingly; a higher advertising interval and a lower connection interval would be more power efficient without compromising the user experience.

8.8 Topology

The possible topologies for LE are shown in Figure 8.7. In scenario A, one Advertiser is sending advertising packets on the advertising physical channel. There are two scanners listening to those advertising packets. These scanners may request more information from the Advertiser or send a request to connect on the advertising physical channel.

Scenario B shows a piconet where one Master is connected to three Slaves. The data between the Master and the Slaves is exchanged on the data physical channels. Besides this, the Master is also acting as a Scanner and listening to packets from an Advertiser on the advertising physical channel. In this scenario if the Master/

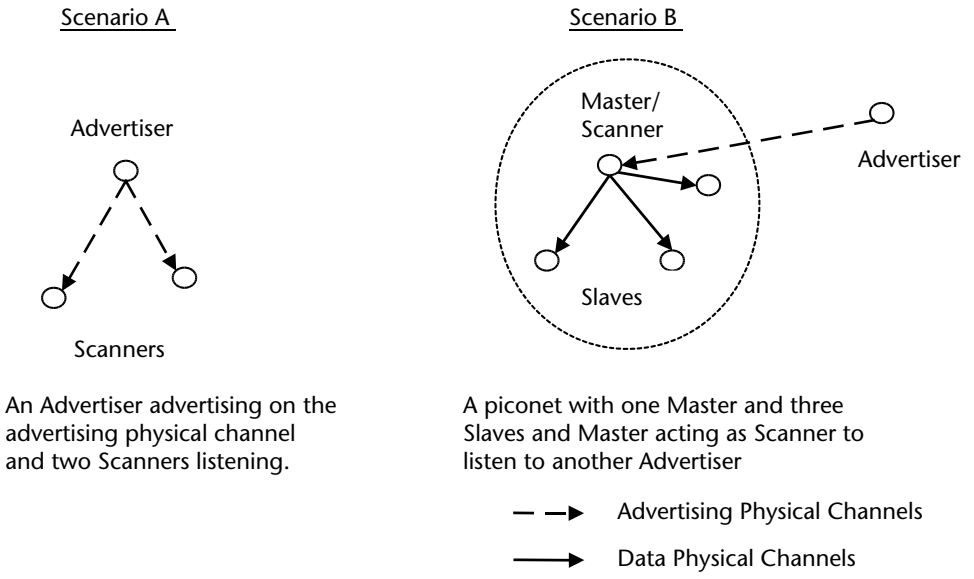

Figure 8.7 LE topology.

Scanner decides to connect to the Advertiser, then it will send a connect request to the Advertiser on the advertising physical channel. If the connection is established successfully, the Advertiser will also join the same piconet and become a Slave of the same Master. Then the Master will have four Slaves.

While the BR/EDR specification put an upper limit of up to seven Slaves connected to a Master, the LE specification does not put any such upper limit. A Master can connect to as many Slaves as it wants. This is only restricted by the amount of resources available with the Master in terms of memory and processing power.

As with BR/EDR only one device can be a piconet Master and all other devices are piconet Slaves. All the communication is between the Master and Slave devices. The Slaves are not allowed to directly talk to each other.

As per specifications 4.1, an LE device could belong to only one piconet at a particular time. Scatternets were not supported. This restriction helped in simplifying the design of the link layer for LE devices.

Specifications 4.1 relaxed this restriction and allowed LE devices to be part of multiple piconets (also known as scatternet). The device could either be a Master in one piconet and Slave in another piconet, or a Slave in two piconets. This opened up support for several new use case scenarios for LE devices. For example, a temperature sensor can connect to two mobile phones and provide alerts to two mobile phones that can be with different users. In that case, either of the users can act on the alert, thereby making it more convenient.

8.9 Packet Format

The link layer has only one packet format that is used for both advertising and data physical channels. This is in contrast to BR/EDR which has several packet formats

(ID, NULL, POLL, FHS, DM). This is another step towards simplification by LE. The packet format is shown in Figure 8.8.

8.9.1 Preamble

The preamble is used by the receiver to perform frequency synchronization, symbol timing estimation, and Automatic Gain Control (AGC) training. The Preamble is an alternate sequence of 1s and 0s. So it can be 10101010b or 01010101b. The receiver uses this sequence to synchronize its radio to the exact frequency and also adjust the gain so that the remaining packet is correctly received.

8.9.2 Access Address

The access address is used as a correlation code by devices tuned to the physical channel. Since LE uses a limited number of data channels, there is a possibility of unrelated LE devices using the same RF channel at the same time. The access address is used as a code to ensure that the transmission is indeed meant for the device that is receiving it. The access address is different for each link layer connection between any two devices.

A link layer is said to be connected to a channel if it is synchronized to the following parameters of the channel:

- Timing.
- Frequency.
- Access Address.

It is not mandatory for the link layer to be actively involved in data exchange to remain connected. This is another enhancement over BR/EDR. For the link layer to remain connected, it doesn't need to continuously send and receive data. In the case of BR/EDR, if the link layers are connected and they have no data to send, then they still exchange POLL/NULL packets continuously.

8.9.3 CRC

The CRC is a 24-bit checksum calculated over the PDU. One of the enhancements done in LE is that the CRC is 24-bit as compared to 16-bit in the case of BR/EDR. A 24-bit CRC helps to check for many more types of bit-errors compared to a 16-bit CRC. This leads to enhanced robustness especially in noise environments where the chance of multiple bit errors is higher.

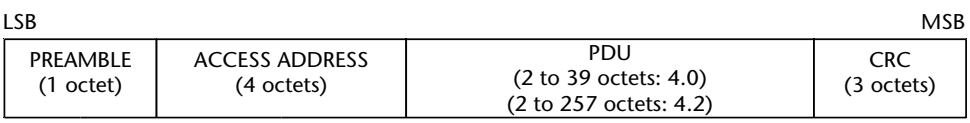

Figure 8.8 Link layer packet format.

8.9.4 PDU

The PDU is of two types:

- Advertising channel PDU: To transmit a packet on the advertising physical channel.
- Data channel PDU: To transmit a packet on the data physical channel.

8.9.4.1 Advertising Channel PDUs

The format of the advertising channel PDUs is shown in Figure 8.9.

The PDU Type field indicates the type of the advertising channel PDU. The TxAdd and RxAdd fields are defined for each PDU type separately and may not be valid for all PDU types. These are used to indicate whether the address contained in the payload is a public address (RxAdd/TxAdd = 0) or random address (RxAdd/TxAdd = 1).

The Length field indicates the length of the payload field in octets.

There are three types of advertising channel PDUs:

- Advertising PDUs: These PDUs are sent by the link layer in advertising state and received by the link layer in the scanning state or initiating state. The different type of advertising PDUs are shown in Table 8.1.
- Scanning PDUs: These PDUs are used by the link layer of the Scanner to request data from the Advertiser and by the Advertiser to respond to the request from the Scanner. The different types of scanning PDUs are shown in Table 8.2.
- Initiating PDUs: These PDUs are used by the link layer to initiate a connection to the Advertiser. There is only one type of PDU and that is shown in Table 8.3.

An example of the advertising and scanning PDUs was shown in Figure 8.5.

- The Advertiser sent ADV_IND PDUs in Frames #425, #426, #427, #430, #431 and #432.

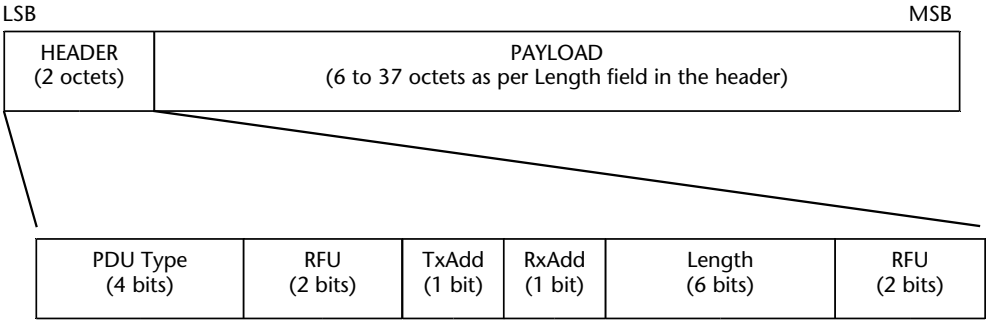

Figure 8.9 Advertising channel PDU.

Table 8.1 Advertising PDUs

PDU Name	*Advertising Event Type*	*Sender Link Layer State*	*Receiver Link Layer State*
ADV_IND	Connectable Undirected	Advertising	Scanning/Initiating
ADV_DIRECT_IND	Connectable Directed	Advertising	Scanning/Initiating
ADV_NONCONN_IND	Non-Connectable Directed	Advertising	Scanning/Initiating
ADV_SCAN_IND	Scannable Undirected	Advertising	Scanning/Initiating

Table 8.2 Scanning PDUs

PDU Name	*Scanning Event Type*	*Sender Link Layer State*	*Receiver Link Layer State*
SCAN_REQ	Scanner requesting data from Advertiser	Scanning	Advertising
SCAN_RSP	Advertiser responding to the request from Scanner	Advertising	Scanning

Table 8.3 Initiating PDUs

PDU Name	*Initiating Event Type*	*Sender Link Layer State*	*Receiver Link Layer State*
CONNECT_REQ	Initiator requesting to connect to Advertiser	Initiating	Advertising

- The Scanner sent SCAN_REQ PDU in Frame #428.
- The Advertiser responded with SCAN_RSP PDU in Frame #429.

8.9.4.2 Data Channel PDUs

The format of the data channel PDUs is shown in Figure 8.10.

The Payload field is 0 to 251 octets in length. The Message Integrity Check (MIC) field is included only in the case of encrypted link layer connection when the payload field has a nonzero size. This is used to authenticate the data PDU.

The LLID indicates the type of the link layer PDU. There are two possible types:

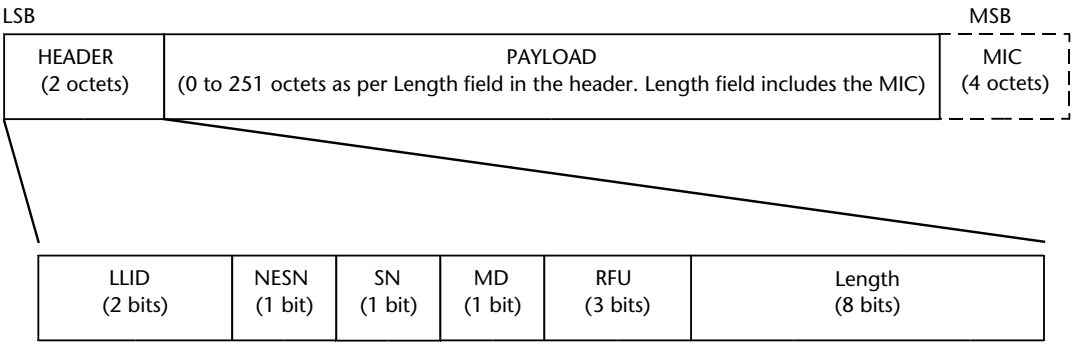

Figure 8.10 Data Channel PDU.

- Link layer data PDU: This type of PDU is used to send L2CAP data.
- Link layer control PDU: This type of PDU is used to control and negotiate the connection between the two link layers. There are various types of link layer PDUs defined for the link layer to exchange control information with the link layer of the other device. The various layer control procedures and corresponding PDUs will be explained later in this chapter.

MD indicates that the device has more data to send. It is used to decide whether the current connection event can be closed or not. Length indicates the size in octets of Payload and MIC fields. The SN field indicates the Sequence Number and the NESN field indicates the Next Expected Sequence Number. These two bits provide a very simple mechanism for acknowledgment and flow control of packets.

The SN bit identifies the current packet while the NESN bit identifies which packet from the peer device is expected next. If a packet is correctly received then the NESN bit is incremented. This serves as an acknowledgment to the sender that the packet has been received. If the packet had an error, then NESN bit is not incremented. This indicates to the sender that it has to resend the previous packet. This is shown in Figure 8.11.

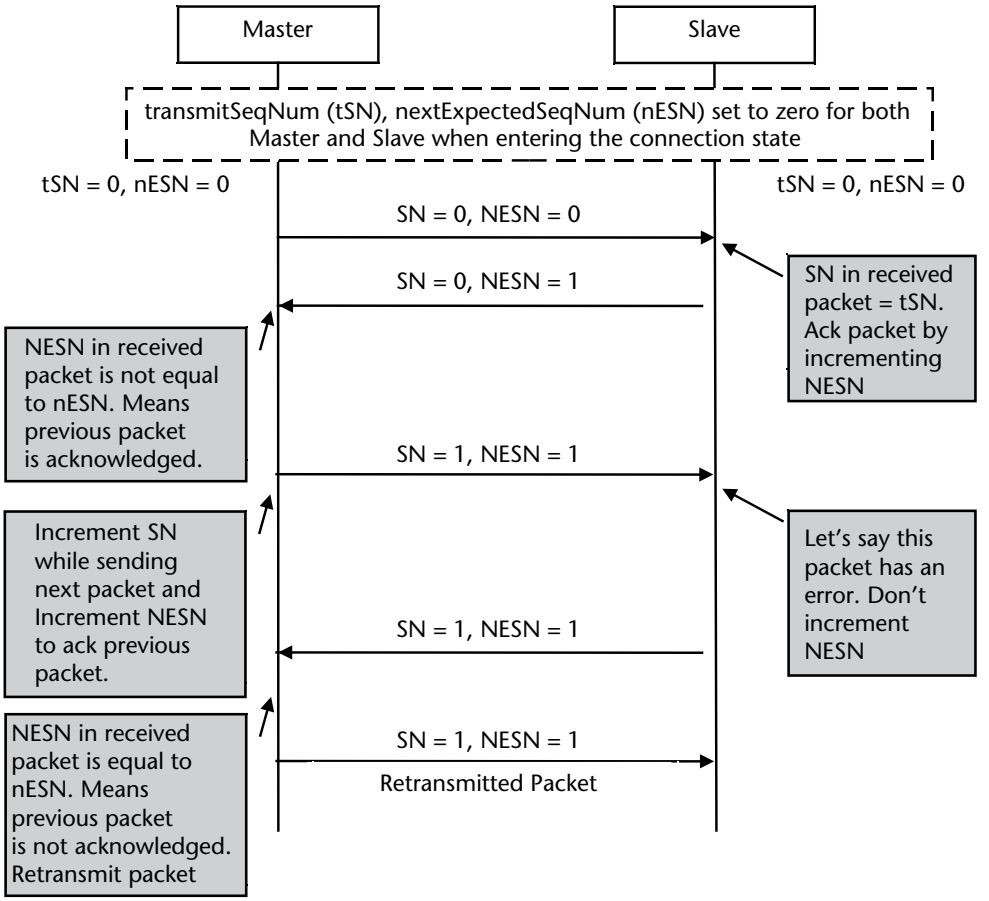

Figure 8.11 Acknowledgment and flow control using SN and NESN.

Low Energy Data Packet Extensions

One of the most important changes in 4.2 specifications is that the length field has been increased from 5 bits to 8 bits. This has led to an increase in the supported packet size for data packets from 27 to 251 bytes. This is an almost ten-fold increase and is useful in the following cases:

1. Over the air (OTA) firmware upgrades: After the device has been deployed in the field, there may be newer versions of the software containing additional features and bug fixes. The most convenient way to update the firmware is through OTA updates, but with a packet size of 27 bytes, an OTA update would take a long time. Depending on the firmware size, this may take several minutes. Besides taking time, this would also lead to significant power consumption because the device would have been active for a long duration, thereby reducing its battery life. With the increase in packet size, the firmware updates would take a fraction of the time and the energy as compared to the time and energy taken by 4.0 compliant devices.
2. Uploading the logs: One of the primary usages of BLE devices is in sensor tags, where the tags keep collecting the data and then upload it to the Internet. Sometimes, this data may be huge (i.e., a person's temperature and heart rate during the day if a reading is taken every minute), and transporting it on 27-byte data packet sizes would take a lot of time. Transporting it on bigger packets would be much faster and more power efficient.

In general, larger packet sizes are more efficient when compared to smaller ones. This is because the fractional overhead of the bytes used for the header is less. For example, if a total of 6 bytes were used for the header (including bytes used in the lower layers of the protocol stack and message integrity check), then the overhead for 27-byte packets would be 6/27 (22%), while that for 251-byte packets would be 6/251 (2%). Also, increased data transfer speeds and packet sizes reduce the window and chances of transmission losses, which would help in getting the important packets in time to or from the sensor device (for example, a hearing aid or a critical medical equipment).

Besides this, there would be less processing power consumed in fragmenting the packets at the transmitter end and reassembling them at the receiver end. Fragmentation also makes it worse because the packets are only sent at connection intervals, and there is a dead time between connection intervals. This translates to a longer transmission time, effectively keeping the device powered on longer.

8.10 Bit Stream Processing

Figure 8.12 shows the sequence of steps that are carried out by the link layer before transmitting data and the link layer on the other side after receiving the data. The data to send is treated as a bit stream with the LSB first. At the time of transmission, the different steps that are performed are encryption of the data follower by CRC

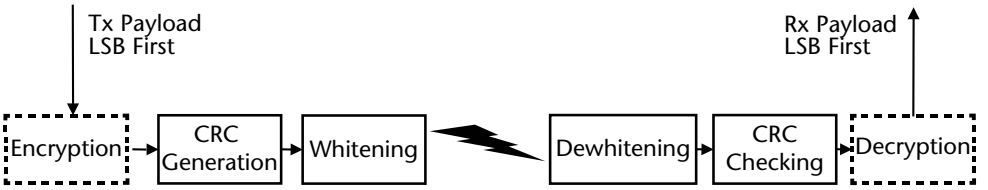

Figure 8.12 Link layer bit stream processing.

generation and then finally whitening. When the data is received on the receiving side, exactly the reverse of these steps are performed.

The Encryption stage on the transmitter side and the decryption stage on the receiver side are optional. Encryption is done only if the host requested an encrypted link.

Data whitening is the process used to avoid long sequences of zeros or ones while transmitting. Before transmitting the header and payload are scrambled with a data whitening word. This randomizes the data to reduce the possibilities of long sequences of zeros or ones. At the receiver end, the data is descrambled using the same data whitening word to get back the original data.

When a packet is received, the first step is to check for errors. This includes the following:

- Check the Access address to ensure that the packet is meant for the channel that the link layer is connected to.
- CRC checking.

One of the optimizations done by LE is that the encryption is done before CRC generation while transmitting data and decryption is done after CRC checking while receiving data. This is opposite to BR/EDR where CRC generation is done before encryption while transmitting and CRC checking is done after decryption while receiving. So in the case of LE, there are the following advantages:

1. CRC checking takes far lesser time as compared to decryption while receiving. So if CRC checking is done before decryption:
   a. The received packet can be acknowledged as soon as the CRC check is complete. So the radio can be immediately switched off instead of waiting for the complete decryption process.
   b. The complete decryption process can then be done offline when the radio is switched off. This helps in reducing the peak power consumption since only decryption is going on and the radio has been switched off.
   c. If the packet got corrupted by the time it was received, then the CRC check will detect it early and the packet can be dropped immediately. For such packets power does not need to be consumed while doing the decryption.

8.11 Link Layer States

The five link layer states were briefly introduced at the beginning of this chapter. This section provides a detailed explanation of each of these states. Broadly the states can be categorized into Non-Connected States and Connection States as shown in Figure 8.13.

To reduce the complexity, LE does not allow scatternet scenarios. (Note that a scatternet is a combination of multiple piconets. It is formed in BR/EDR scenarios when one of the devices acts as Slave in two piconets or as Master in one piconet and Slave in another piconet.)

This imposes the following restrictions on LE devices:

1. The device cannot act as Master and Slave at the same time.
2. The device cannot also act in initiating state if it is already in Slave role. This is because it would lead to a Master connection in addition to the Slave role.
3. The device cannot have more than one Slave connections.
4. If the device is already operating in Connection or Initiating state, it cannot operate in Advertising state with advertising type that will lead to a Slave role connection.

8.11.1 Nonconnected States

In the Nonconnected states the link layer can be in any one of the following four states:

1. Standby State.
2. Advertising State.
3. Scanning State.
4. Initiating State.

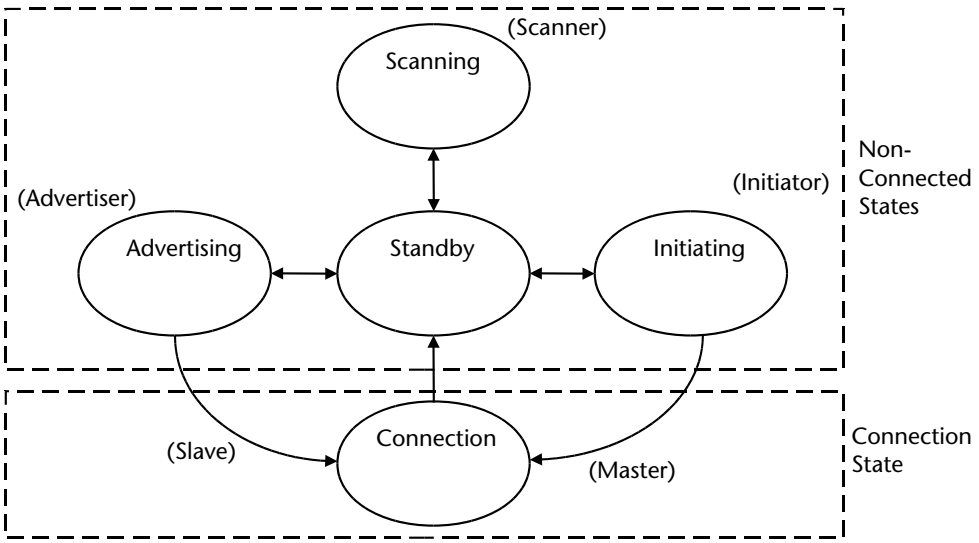

Figure 8.13 Link layer states.

These states are explained in detail below.

8.11.1.1 Standby State

This is the default state of the link layer. No packets are sent or received in this state. From this state, a device can enter advertising state, scanning state or initiating state. It cannot go into connection state directly from the standby state.

A device can enter into this state from any other state. In fact, to aid simplicity of the state machine and reduce the number of possible combinations of transitions from one state to another, the link layer state machine has been designed to only have the minimum needed transitions and this state is used as an intermediate state. For example, to go from scanning state to initiating state the link layer first goes from scanning state to standby and then from standby to initiating state.

8.11.1.2 Advertising State

In this state, the link layer transmits advertising PDUs in advertising events.

During an advertising event, the link layer transmits one or more advertising PDUs on each of the used advertising channels (The host may request the link layer to use either all or a subset of the three advertising channels—37, 38 and 39). The device in this state is known as Advertiser.

The advertising events are of following four types. The advertising PDUs associated with each of these events were shown in Table 8.4.

1. Connectable undirected event.
2. Connectable directed event.
3. Non connectable undirected event.
4. Scannable undirected event.

Connectable Undirected Event

The connectable undirected event is sent by an Advertiser when it wants another device to connect to it. It sends an advertising indication (ADV_IND) PDU on the advertising channel. The other device may also request for additional information before deciding to connect to the Advertiser.

Table 8.4 Advertising Event Types, PDUs Used and Acceptable Responses

		Acceptable Response PDU from Remote Device	
Advertising Event Type	*PDU Used*	*Scanner (SCAN_REQ)*	*Initiator (CONNECT_REQ)*
Connectable Undirected Event	ADV_IND	Yes	Yes (From any Initiator)
Connectable Directed Event	ADV_DIRECT_IND	No	Yes (Only from the addressed Initiator)
Nonconnectable Undirected Event	ADV_NONCONN_IND	No	No
Scannable Undirected Event	ADV_SCAN_IND	Yes	No

The receiver of the connectable undirected event can be either in the scanning state or initiating state.

- If it's in the scanning state, it may request for more information using SCAN_REQ PDU.
- If it's in the initiating state, it may send a connect request using CONNECT_REQ PDU.

The two scenarios are shown in Figure 8.14. A device may use the first one or the second one or first one followed by the second one.

The payload in a connectable undirected event contains the following two fields:

- AdvA (6 octets): Public address or random address of the Advertiser. The type is indicated by TxAdd field (See Figure 8.9).
- AdvData (0-31 octets): This contains the advertising data from the Advertiser's host.

One possible example of this could be a thermometer that is placed in a building.

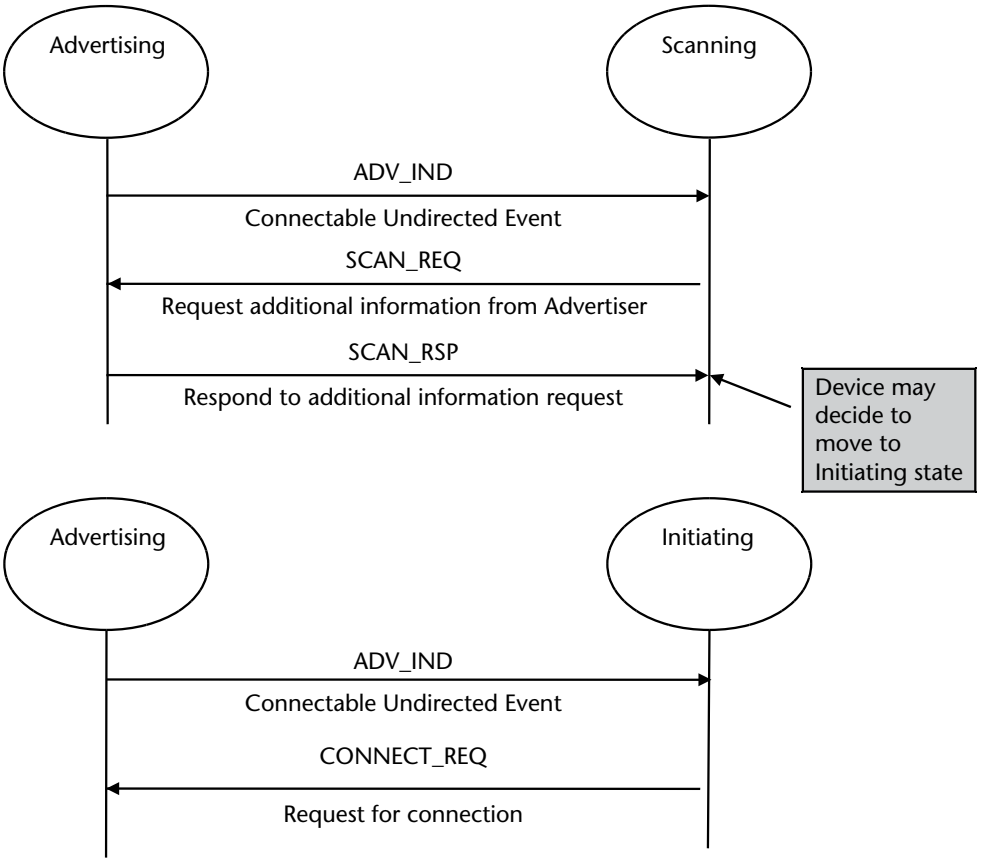

Figure 8.14 Connectable undirected advertising event.

- The thermometer could keep on advertising—'I am a thermometer';
- Any mobile phone in the vicnity could query—'Do you display temperature in Fahrenheit?';
- The thermometer could say—'Yes';
- The mobile phone could then connect to the thermometer and get the temperature.

An example of air logs for ADV_IND was shown in Figure 8.5. The Advertiser sends an advertising PDU in frame #425, #426, etc. In response to the ADV_IND packet sent in Frame #427, the Scanner requests for additional information using the SCAN_REQ PDU.

Connectable Directed Event

The connectable directed event type is used when an Advertiser wants a particular device to connect to it. It sends a directed advertising indication (ADV_DIRECT_IND) PDU on the advertising channel. The other device may request to connect to the Advertiser on receiving this PDU.

The ADV_DIRECT_IND PDU contains the device address of both the Initiator and the Advertiser. So only the Initiator for which the address was contained in the PDU is allowed to make a connection.

The payload in a connectable directed event contains the following two fields:

- AdvA (6 octets): Public address or random address of the Advertiser. The type is indicated by TxAdd field (See Figure 8.9).
- InitA (6 octets): Public address or random address of the Initiator. The type is indicated by RxAdd field (See Figure 8.9).

This is in contrast to connectable undirected event where any device in the scanning or initiating state could request for additional information or connect to the advertiser. Here the request for additional information is not permitted and only a particular device can initiate the connection. The sequence diagram for this is shown in Figure 8.15.

One possible example of this could be a pedometer placed in the jogger's shoe. The pedometer may need to send information to the person's mobile phone.

- The pedometer could advertise: 'I'm a pedometer. I want to send data to mobile phone A.'
- Mobile phone A could receive this request and connect to the pedometer and get the data and display to the person.

Nonconnectable Undirected Event

The nonconnectable undirected event type is used when an Advertiser wants to provide some information to all devices but does not want the devices to connect to it or ask for more information. It sends a nonconnectable advertising indication

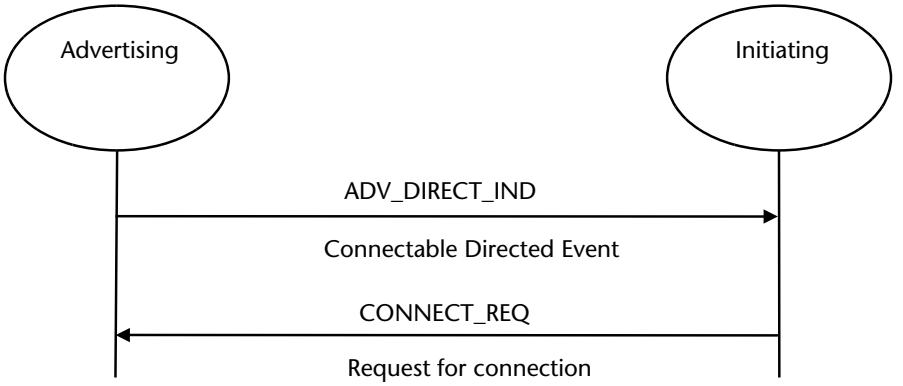

Figure 8.15 Connectable directed advertising event.

(ADV_NONCONN_IND) PDU on the advertising channel. The other device may only listen to this information.

This is in contrast to connectable undirected and connectable directed events because the receiver (which has to be in the scanning state) can just receive this information. It can neither connect nor ask for more information.

The payload in a non-connectable undirected event contains the following two fields:

- AdvA (6 octets): Public address or random address of the Advertiser. The type is indicated by TxAdd field (See Figure 8.9).
- AdvData (0–31 octets): This contains the advertising data from the Advertiser's host.

The sequence diagram for this is shown in Figure 8.16.
One possible example of this could be a microwave oven.

- The microwave could advertise: 'I'm a microwave. The food is cooked. Please take it out.'
- The people in the house could receive this information on their mobile phone, television or set top box and take appropriate action.

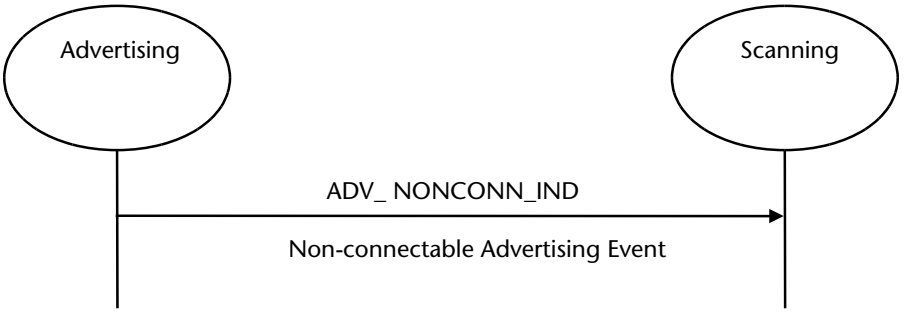

Figure 8.16 Nonconnectable undirected advertising event.

Another example could be an Advertiser at the airport:

- The Advertiser could advertise that flight ABC will take off from gate 123.
- Any person who is interested in the flight information could scan for that information.

Scannable Undirected Event

The scannable undirected event type is used when an Advertiser wants to allow a Scanner to request more information from it. It sends a scannable advertising indication (ADV_SCAN_IND) and the Scanner may request more information using the SCAN_REQ PDU.

This is slightly different from the Nonconnectable Undirected advertising event since in the nonconnectable advertising event, the Scanner cannot send any request back. Here the Scanner may send a scan request to get more information.

The payload in a scannable undirected event contains the following two fields:

- AdvA (6 octets): Public address or random address of the Advertiser. The type is indicated by TxAdd field (See Figure 8.9).
- AdvData (0–31 octets): This contains the advertising data from the Advertiser's host.

The sequence diagram for this is shown in Figure 8.17.
One possible example of this could be a remote control.

- The remote control could advertise: 'I'm a remote. A key has been pressed.'
- The Scanner could send a SCAN_REQ PDU to gather information about which key has been pressed.

A summary of the four advertising event types is provided in Table 8.4.

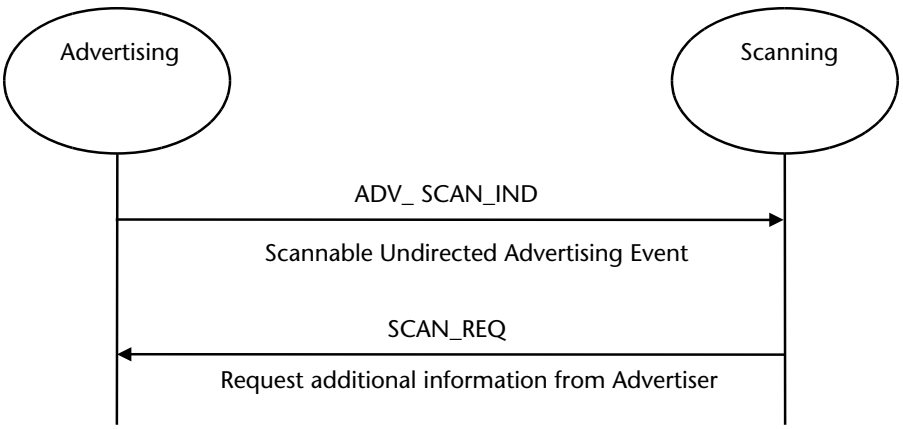

Figure 8.17 Scannable undirected advertising event.

8.11.1.3 Scanning State

In this state, the link layer listens on the advertising channels (Advertising channels 37, 38, 39) for any PDUs from the Advertiser. The device in this state is known as Scanner.

The scanning events are of following two types:

- Passive Scanning.
- Active Scanning.

The advertising PDUs associated with each of these events were shown earlier in Table 8.2

Passive Scanning

In passive scanning, the link layer only receives the packets. It does not send back any packets. Once it receives the packets, it removes the duplicates and then sends the advertising reports to the host. The sequence diagram for this is shown in Figure 8.18.

Active Scanning

In active scanning, the link layer listens to the advertising PDUs and then depending on the advertising PDU type, it may request additional information from the Advertiser using the SCAN_REQ PDU.

The Scanner is permitted to send a SCAN_REQ only if the Advertiser used a connectable undirected event (ADV_IND PDU) or a scannable undirected event (ADV_NONCONN_IND PDU). These are the ones that are marked as Yes in the column for SCAN_REQ in Table 8.4.

The payload in a SCAN_REQ PDU contains the following two fields:

- ScanA (6 octets): Public address or random address of the Scanner. The type is indicated by TxAdd field (See Figure 8.9).
- AdvA (6 octets): Public address or random address of the Advertiser. The type is indicated by RxAdd field (See Figure 8.9).

The payload in a SCAN_RSP PDU contains the following two fields:

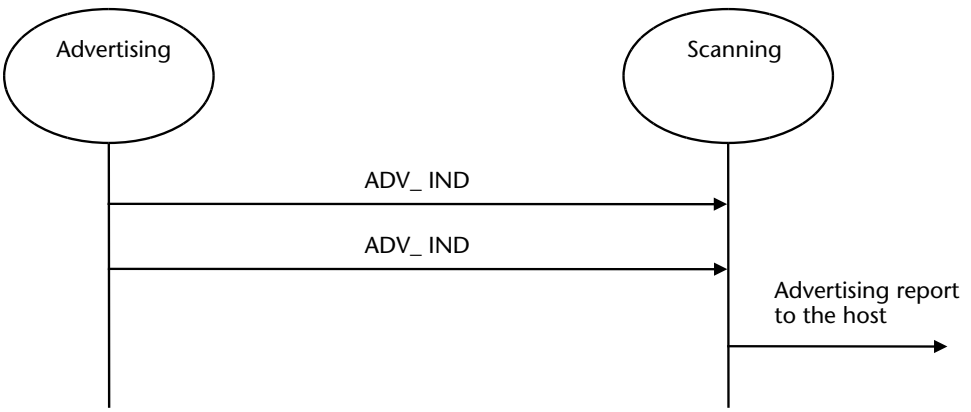

Figure 8.18 Passive scanning.

- AdvA (6 octets): Public address or random address of the Advertiser. The type is indicated by TxAdd field (See Figure 8.9).
- AdvData (0–31 octets): This contains the advertising data from the Advertiser's host.

The sequence diagram for this is shown in Figure 8.19.

A practical example of active scanning was shown in Figure 8.5. At Frame #428, the Scanner sent a SCAN_REQ to get more information from the Advertiser.

8.11.1.4 Initiating State

In the initiating state, the link layer listens on the advertising channels (Advertising channels 37, 38, 39) for any PDUs from the Advertiser and if it's permitted, it sends a connection request to the Advertiser.

The Initiator is permitted to send a CONNECT_REQ only if the Advertiser used a connectable undirected event (ADV_IND PDU) or a connectable directed event (ADV_DIRECT_IND PDU). In the latter case, the address of the Initiator must match the address that was provided in the connectable directed event PDU. These are the scenarios that are marked as Yes in the column for CONNECT_REQ in Table 8.4. The sequence diagram for this is shown in Figure 8.20.

Figure 8.21 shows a sniffer capture of the connection initiation procedure. The following things may be observed:

- The Advertiser sent an advertising event (ADV_IND) in Frames #670, #671, and #672.
- The Initiator responded with a CONNECT_REQ in Frame #673 to create a connection.

8.11.2 Connection State

This state is entered when the Initiator sends a CONNECT_REQ PDU to the Advertiser. As shown in Figure 8.13, this state can be entered in two ways.

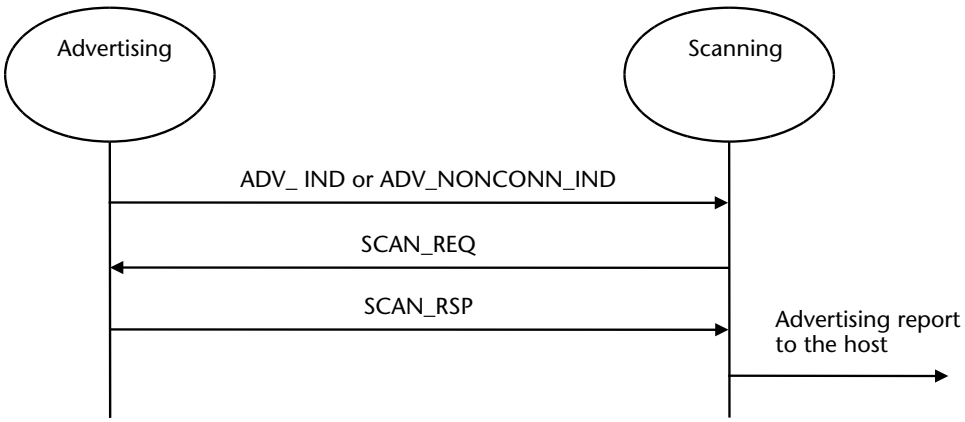

Figure 8.19 Active scanning.

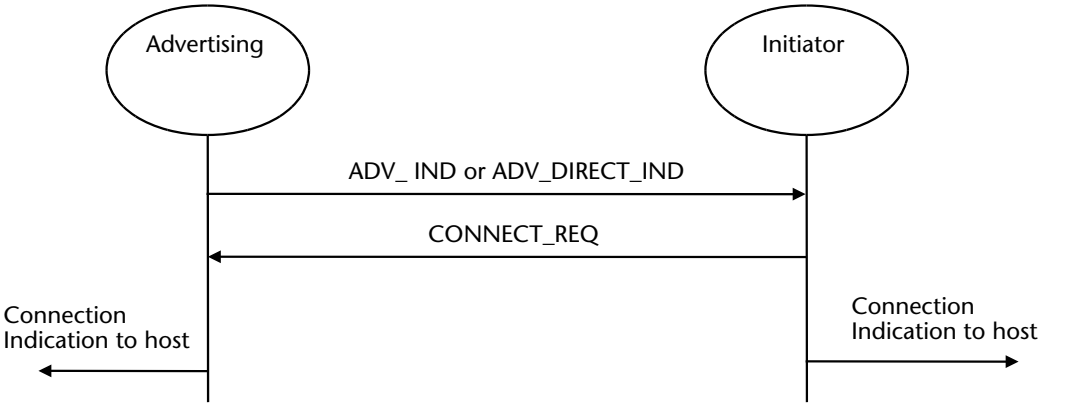

Figure 8.20 Initiating connections.

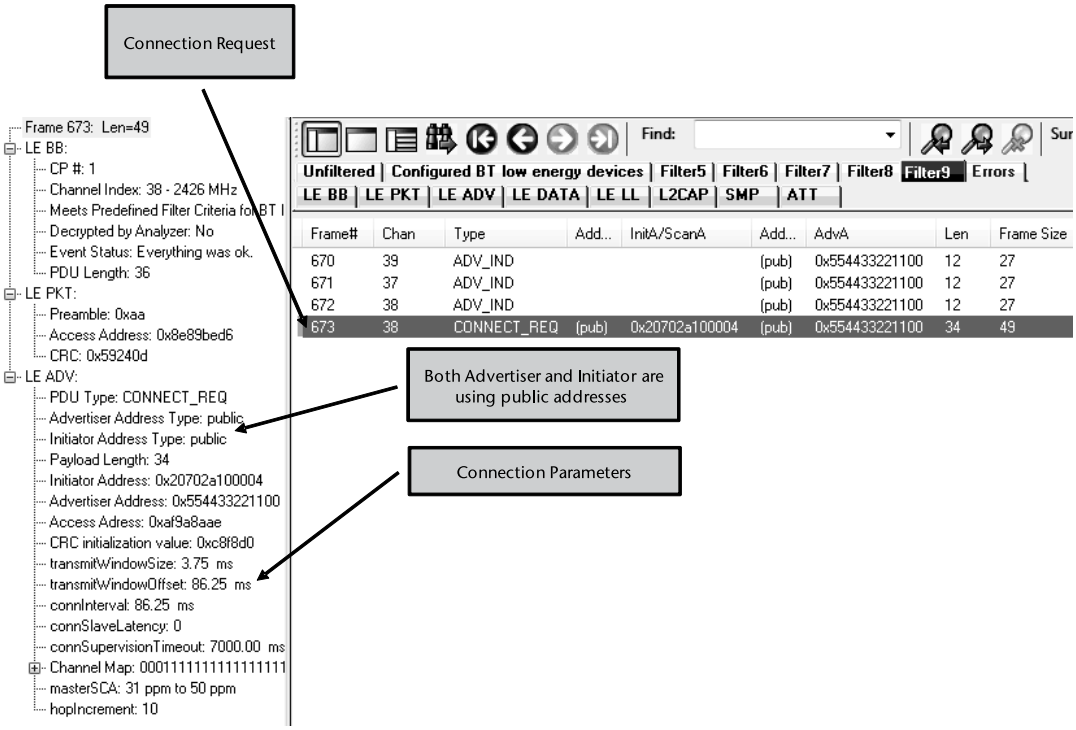

Figure 8.21 Example of initiating a connection.

- From a link layer in the initiating state. The Initiator becomes the Master of the connection.
- From a link layer in the advertising state. The Scanner becomes the Slave of the connection.

After entering the connection state, the connection is considered to be created (but not established). After the connection is created, once a data channel

packet has been received from the peer device, the connection is considered to be established. There can only be one LE connection between two devices.

The payload in a CONNECT_REQ PDU contains the following three fields:

- InitA (6 octets): Public address or random address of the Initiator. The type is indicated by TxAdd field (See Figure 8.9).
- AdvA (6 octets): Public address or random address of the Advertiser. The type is indicated by RxAdd field (See Figure 8.9).
- LLData (22 octets): This contains various parameters related to the connection.
  - AA (4 octets): The Access Address for the Link Layer's connection.
  - CRCInit (3 octets): Initialization value for the CRC calculation. (Random value.)
  - WinSize (1 octet): Indicates transmit window size.
  - WinOffset (2 octets): Indicates transmit window offset.
  - Interval (2 octets): Connection Interval (connInterval).
  - Latency (2 octets): Connection Latency (connSlaveLatency).
  - Timeout (2 octets): Connection Supervision Timeout (connSupervisionTimeout).
  - ChM (5 octets): Channel bitmap showing used and unused channels.
  - Hop (5 bits): Hop increment to be used for frequency hopping algorithm.
  - SCA (3 bits): Indicate worst case Master's sleep clock accuracy.

Some of these parameters were already explained earlier in this chapter in the section related to Connection Events. The sequence diagram for this is shown in Figure 8.22. Once the connection is established, the Master and Slave can exchange data channel PDUs in connection events.

Figure 8.21 showed a sniffer capture of the connection initiation procedure. The parameters of the CONNECT_REQ PDU are shown on the left hand side. These include the following:

- InitA (6 octets): Public address of the Initator.
- AdvA (6 octets): Public address of the Advertiser.
- Access Address of the connection.
- LLData (22 octets) containing the following:
  - CRC Initialization value. Random value provided by Master to be used for CRC calculations.
  - transmitWindowSize = 3.75 ms.
  - transmitWindowsOffset = 86.25 ms.
  - connInterval = 86.25 ms.
  - connSlaveLatency = 0 (This means that the Slave has to listen for each connection event).

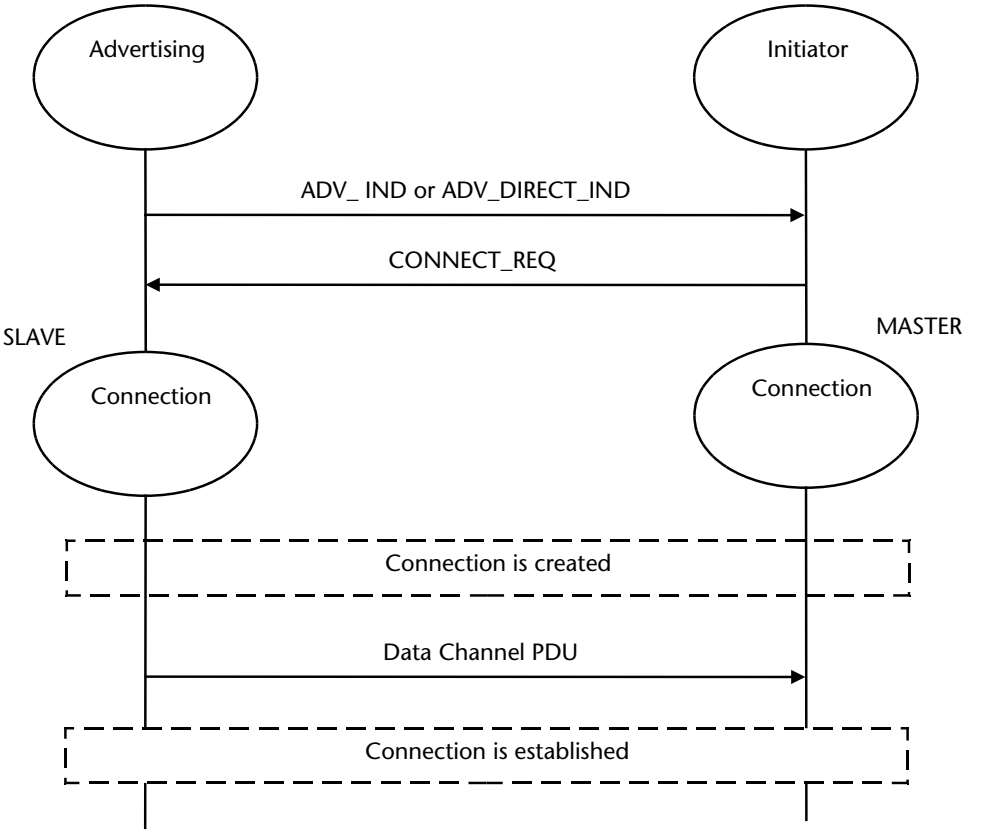

Figure 8.22 Connection state.

- connSupervisionTimeout = 7000 ms. (This means that if no packet is received for 7 seconds then the connection is considered to be lost).
- ChannelMap: Indicating which channels can be used for data transfer.
- HopIncrement: 10. This indicates the Hop Increment to be used in the algorithm to calculate the next hopping frequency.

8.12 Link Layer Control Procedures

The Link Layer Control Protocol (LLCP) is used to control and negotiate the connection between the two link layers. The summary of the link layer control procedures and the associated PDUs is shown in Table 8.5. Specifications 4.1 and 4.2 introduced certain new procedures and certain new PDUs within existing procedures. These are mentioned in Table 8.5. The PDUs are encapsulated within the link layer control PDUs that are part of the data channel PDUs. The format of the data channel PDUs was explained earlier in Figure 8.10.

These procedures can only be invoked sequentially. This means that at any given time only one link layer procedure is initiated. The next link layer procedure can only be initiated after the previous one either completes or has a timeout. The only exception is the termination procedure which can be initiated at any time.

Table 8.5 Link Layer Control Procedures and PDUs

Link Layer Procedure	*Link Layer Control PDU Name*	*Brief Purpose*
Connection Update Procedure	LL_CONNECTION_UPDATE_REQ	This procedure is used by the Master any time after entering the connection state to update the link layer parameters of a connection.
Channel Map Update Procedure	LL_CHANNEL_MAP_REQ	This procedure is used by the Master after entering the connection state to update the channel map to be used for frequency hopping.
Encryption Start Procedure	LL_ENC_REQ LL_ENC_RSP LL_START_ENC_REQ LL_START_ENC_RSP	This procedure is used by the Master to start encryption or to re-start encryption after a pause encryption procedure.
Encryption Pause Procedure	LL_PAUSE_ENC_REQ LL_PAUSE_ENC_RSP	This procedure is used if the Master wants to change the encryption key. A Pause procedure is followed by the Encryption Start procedure to change the link key.
Feature Exchange Procedure	LL_FEATURE_REQ LL_FEATURE_RSP	This procedure is used by the Master after a connection has been established to initiate exchange of feature set information.
Version Exchange Procedure	LL_VERSION_IND	This procedure is used by either the Master or the Slave after entering the connection state to exchange version information.
Termination Procedure	LL_TERMINATE_IND	This procedure is used in the connection state by either the Master or the state to terminate the connection.
Unused/ Unsupported PDU	LL_UNKNOWN_RSP	This PDU is sent as a response if an unused or unsupported PDU is received.
Rejection	LL_REJECT_IND	This PDU is sent if a request from the other side is rejected. For example the Slave sends this response to Master if the Master tries to enable encryption and the Slave does not support it.
Connection Parameters Request Procedure (Enhancement in 4.1)	LL_CONNECTION_PARAM_REQ LL_CONNECTION_PARAM_RSP	This procedure is used by the Master or the Slave to request the remote device to update the connection parameters.
LE Ping Procedure (Enhancement in 4.1)	LL_PING_REQ LL_PING_RSP	This procedure is used by the Master or the Slave to verify the presence of the remote link layer.
Data Length Update Procedure (Enhancement in 4.2)	LL_LENGTH_REQ LL_LENGTH_RSP	This procedure is used by the Master or the Slave to inform the remote link layer about changes in the values of data PDU length and PDU time.
Feature Exchange Procedure (Enhancement in 4.1)	LL_SLAVE_FEATURE_REQ	This is an additional PDU defined by specifications 4.1 to allow the Slave to request the features supported by the Master.
Rejection (Enhancement in 4.1)	LL_REJECT_IND_EXT	This is an additional PDU defined by specifications 4.1 to support extended reject indication to the remote side.

8.12.1 Connection Update Procedure

The connection update procedure is used to update the following link layer parameters of the connection:

- Connection Interval.
- Connection Slave Latency.
- Connection Supervision Timeout.

This procedure can only be initiated by the Master after entering the connection state.The sequence diagram for this procedure is shown in Figure 8.23.

8.12.2 Channel Map Update Procedure

The channel map procedure is used to update the channel map of the connection. The channel map contains two parameters:

- Channel Map: The bitmap indicating which channels are enabled.
- Hop Increment: This indicates the number of channels to hop for each subsequent hop.

This procedure allows the Master to disable frequency hopping on channels which potentially have interference thereby reducing the number of retransmissions that would have been required if the packets were transmitted on those channels. This is the key procedure that provides support for adaptive frequency hopping (AFH).

This procedure can only be initiated by the Master after entering the connection state. The sequence diagram for this procedure is shown in Figure 8.24.

8.12.3 Encryption Procedure

8.12.3.1 Encryption Start Procedure

The encryption start procedure is used by the link layer to enable encryption of packets. It is initiated by the host of the Master by sending a request to the link layer to start encryption. The host of the link layer also provides the Long_Term_Key (LTK). The Long_Term_Key is a 128-bit key used to generate the session key to be

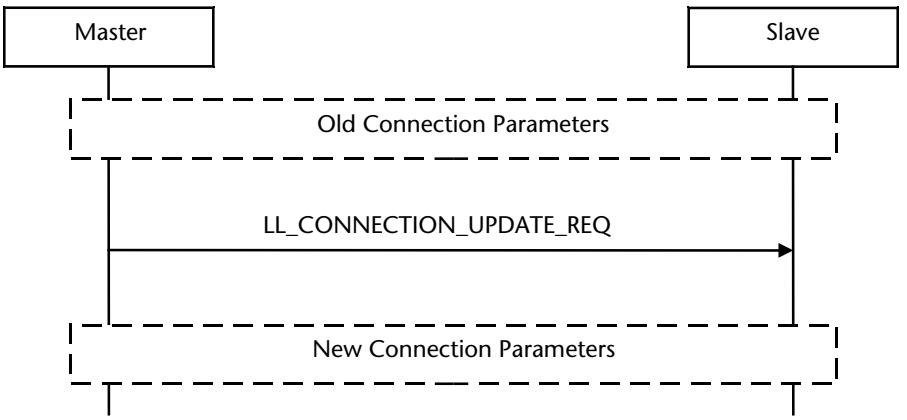

Figure 8.23 Connection update procedure.

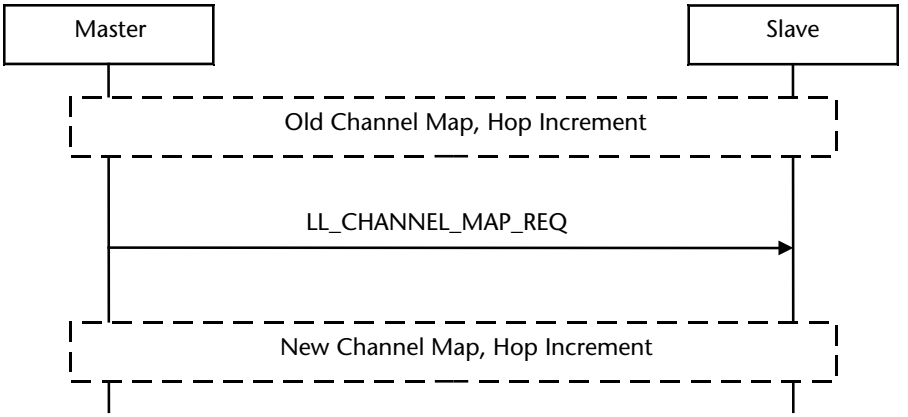

Figure 8.24 Channel map update procedure.

used for the encrypted connection. If the connection is not already encrypted, then the encryption start procedure is used.

If the connection is already encrypted, then the encryption pause procedure followed by encryption start procedure is used. The encryption pause procedure will be explained in the next section. The sequence diagram for this procedure is shown in Figure 8.25.

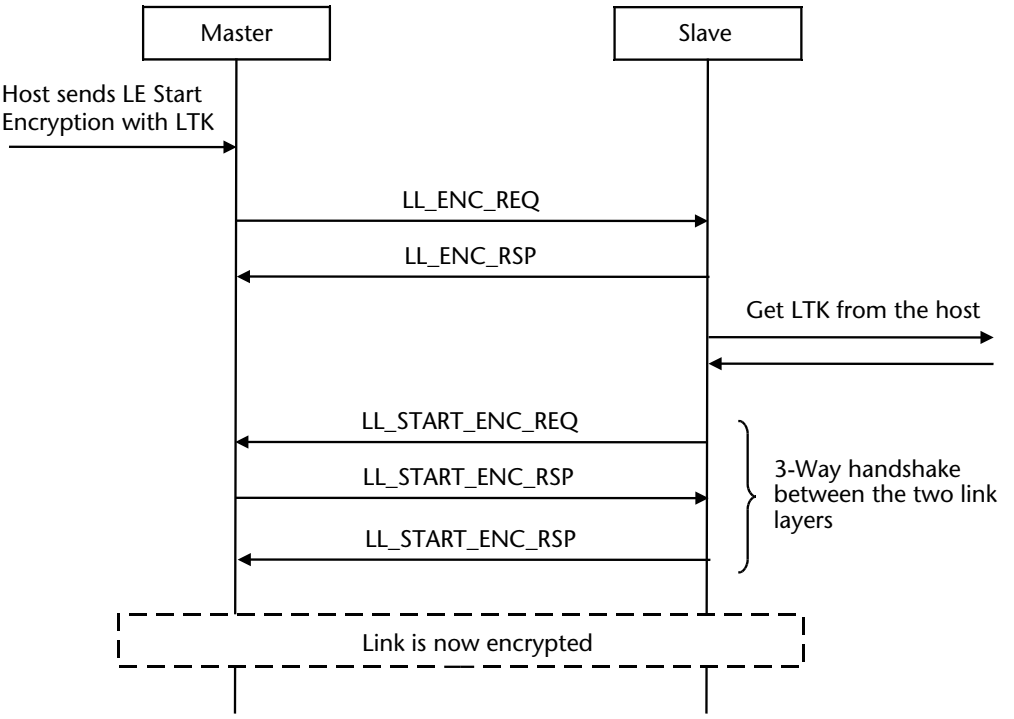

Figure 8.25 Encryption start procedure.

8.12.3.2 Encryption Pause Procedure and Encryption Restart Procedure

This procedure is used by the link layer if the link is already encrypted but a new encryption key is to be used without disconnecting the link. So, the link layer pauses encryption and then follows the same procedure as described in the previous section with the new encryption key.

Encryption restart procedure is very useful when the level of encryption has to be increased or decreased dynamically. So a link can be established with one level of security, and later, if the security needs to be increased it can be done using this procedure.

One example of this could be when a connection was established with a lower level of security but the security level needs to be increased (for example) to transmit some sensitive data. In that case, the encryption pause procedure is used followed by changing the link key and finally the encryption restart procedure.

The sequence diagram for this procedure along with the procedure to restart encryption with the new link key is shown in Figure 8.26.

8.12.4 Feature Exchange Procedure

The feature exchange procedure is used to exchange information about the current supported feature set.

Feature set is a bitmap that provides the feature capabilities of the Master or Slave. As per specifications 4.0, the feature set contained only information on whether encryption was supported on not. Other fields of this bitmap were reserved for future use. This procedure should only be initiated by the Master after

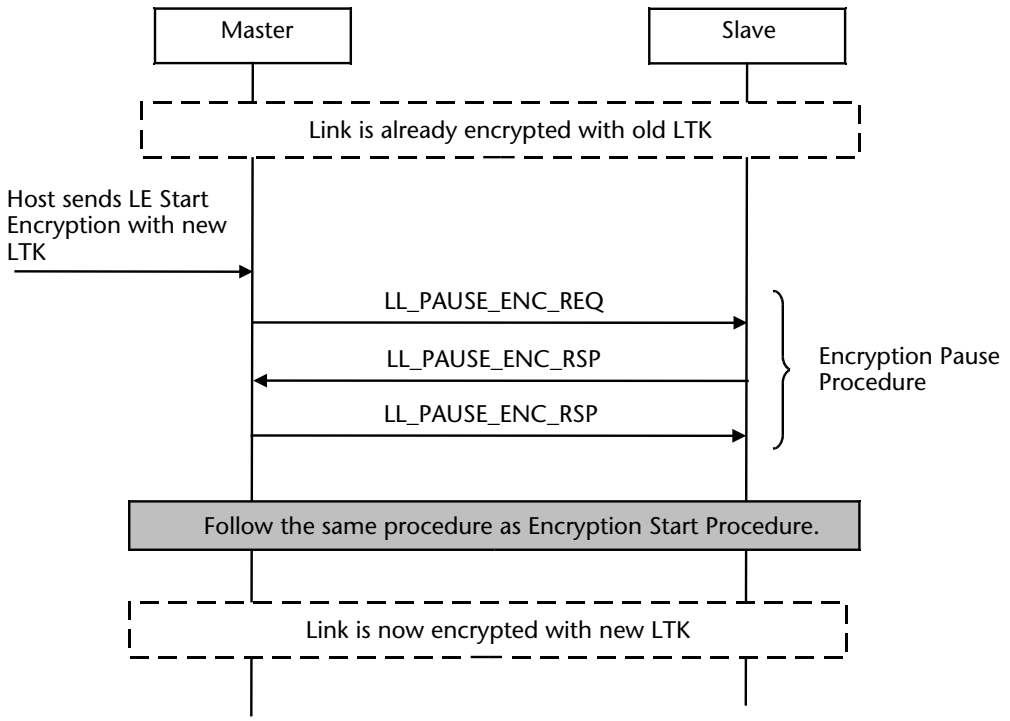

Figure 8.26 Encryption pause procedure and encryption restart procedure.

entering the connection state. The sequence diagram for this procedure is shown in Figure 8.27.

Specifications 4.1 and 4.2 made several enhancements to the feature set and feature exchange procedure.

The information about the supported features can be exchanged at two levels:

- Information sent from a Controller to the Host: In this case, the features that are not supported are indicated by setting the corresponding bit to 0 in the FeatureSet field.
- Information sent from a Controller to the Peer Controller: In this case, if the Controller allows a feature to be used, it sets the corresponding bit to 1 in the FeatureSet field.

The following features were added to the Feature Set in Specifications 4.1:

- Connection parameters request procedure;
- Extended reject indication;
- Slave-initiated features exchange;
- LE ping.

Specifications 4.1 and above allow both the Master and the Slave to initiate a feature exchange procedure. This is shown in Figure 8.27.

The following features were added to the Feature Set in Specifications 4.2 and above:

- LE data packet length extension;

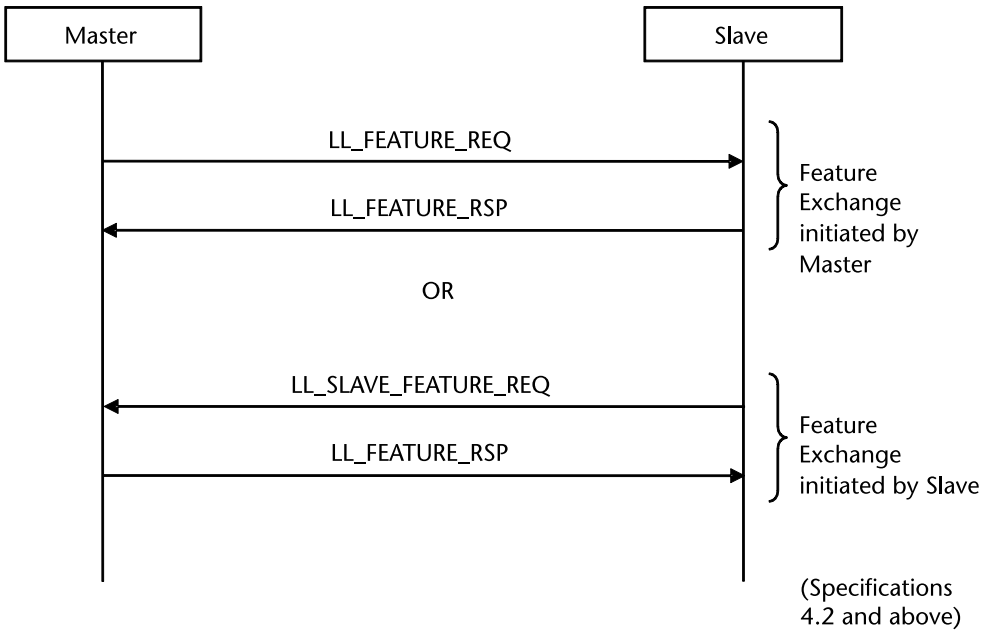

Figure 8.27 Feature exchange procedure.

- LL privacy;
- Extended scanner filter policies.

The details on bit positions, their corresponding features, and whether these bits are valid from Controller to Host or Controller to Controller are show in Table 8.6.

8.12.5 Version Exchange Procedure

The version exchange procedure is used by either the Master or the Slave after entering the connection state to exchange version information. The version information consists of:

- Company ID;
- Link Layer Version;
- Sub Version Number.

The remote side responds with its own version information using the same PDU (LL_VERSION_IND) if it has not sent it in the past for the same connection. This procedure can be initiated by either the Master or the Slave after entering the connection state. The sequence diagram for this procedure is shown in Figure 8.28.

8.12.6 Termination Procedure

The termination procedure is used by either the Master or the Slave to terminate the current connection. This procedure can be used after entering the connection state. The sequence diagram for this procedure is shown in Figure 8.29.

Table 8.6 Bit Mapping for Feature Set

Bit position	*Link Layer Feature*	*Specifications Version in which this was introduced*	*Valid from Controller to Host*	*Valid from Controller to peer Controller*
0	LE Encryption	4.0	Y	Y
1	Connection Parameters Request Procedure	4.1	Y	Y
2	Extended Reject Indication	4.1	Y	Y
3	Slave-initiated Features Exchange	4.1	Y	Y
4	LE Ping	4.1	Y	N
5	LE Data Packet Length Extension	4.2	Y	Y
6	LL Privacy	4.2	Y	N
7	Extended Scanner Filter Policies	4.2	Y	N
8–63	RFU			

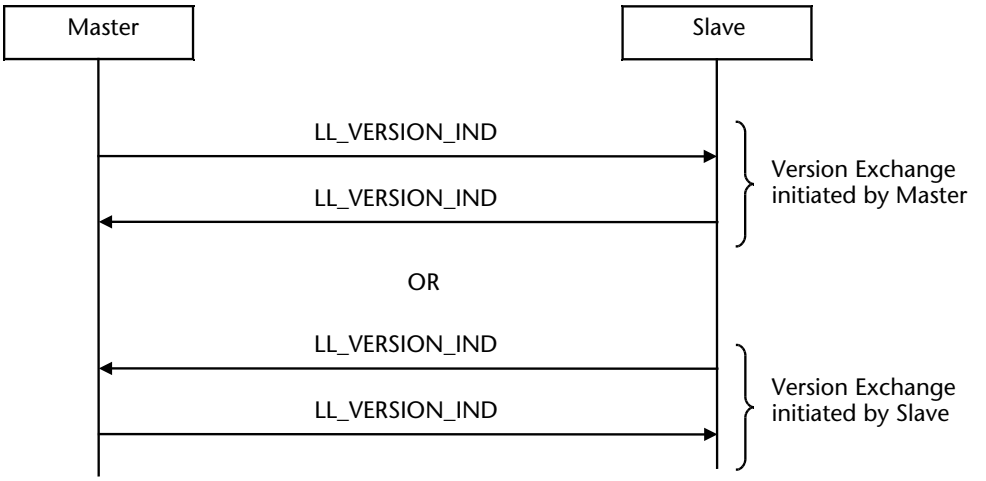

Figure 8.28 Version exchange procedure.

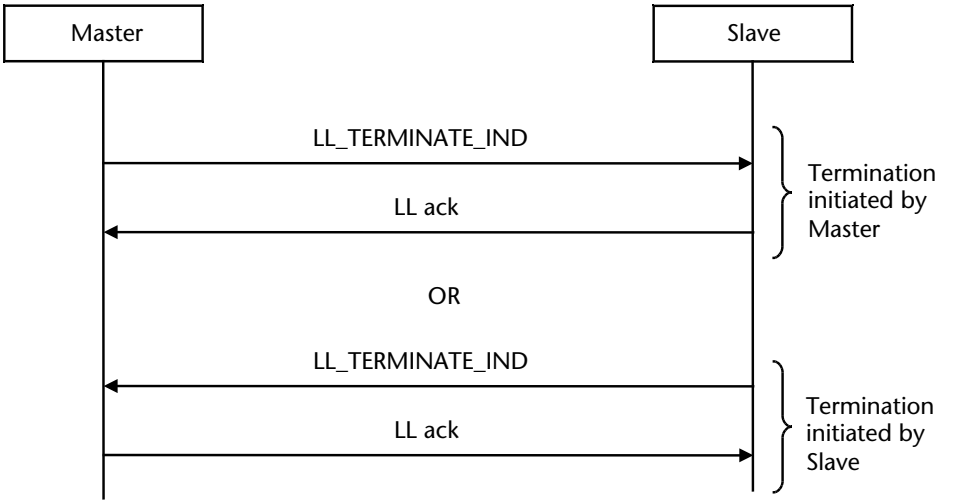

Figure 8.29 Termination procedure.

The procedures mentioned above are supported by specifications 4.0 and above. Besides these, specifications 4.1 introduced two new procedures (the connection parameters request procedure and the LE ping procedure). Specifications 4.2 introduced one additional procedure (the data length update procedure). These procedures are described below.

8.12.7 Connection Parameters Request Procedure

The connection parameters request procedure is used by either the Master or the Slave to request the remote device in order to update the connection parameters during the connection state. The connection parameters include:

- connInterval;

- connSlaveLatency;
- connSupervisionTimeout.

If the parameters are not acceptable to the receiving device, it may respond with either an alternative set of parameters or LL_REJECT_IND_EXT PDU. The sequence diagram for this procedure is shown in Figure 8.30.

8.12.8 LE Ping Procedure

In the networking world, a ping command is used to check the reachability of a remote host and whether or not it can accept and respond to requests. It is a very powerful diagnostic tool and can be used to check connectivity to a remote device and certain other metrics.

The specifications 4.1 introduced the ping procedure in order to verify the presence of a remote link layer. Besides this, it can also be used to check message integrity by requesting the remote ACL layer to send a data packet containing a valid message integrity check (MIC).

This procedure may be initiated by the Master or the Slave by sending the LL_PING_REQ PDU. The remote link layer responds with a LL_PING_RSP PDU. If the remote link layer does not support the ping command, it sends an LL_UNKNOWN_RSP PDU. The sequence diagram for this procedure is shown in Figure 8.31.

8.12.9 Data Length Update Procedure

The data length ppdate procedure is used to inform the remote link layer about the following parameters:

- Maximum receive data channel PDU payload length (connMaxRxOctets);

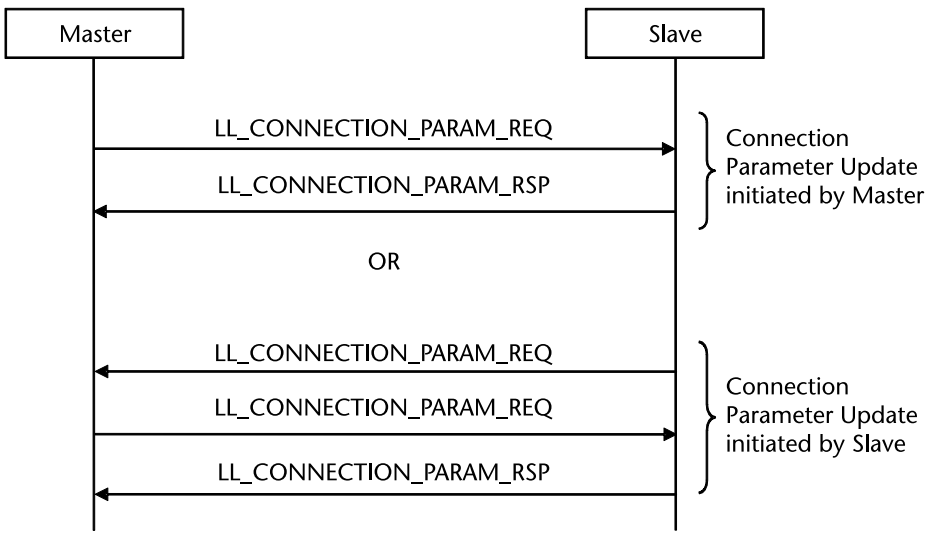

Figure 8.30 Connection parameters request procedure.

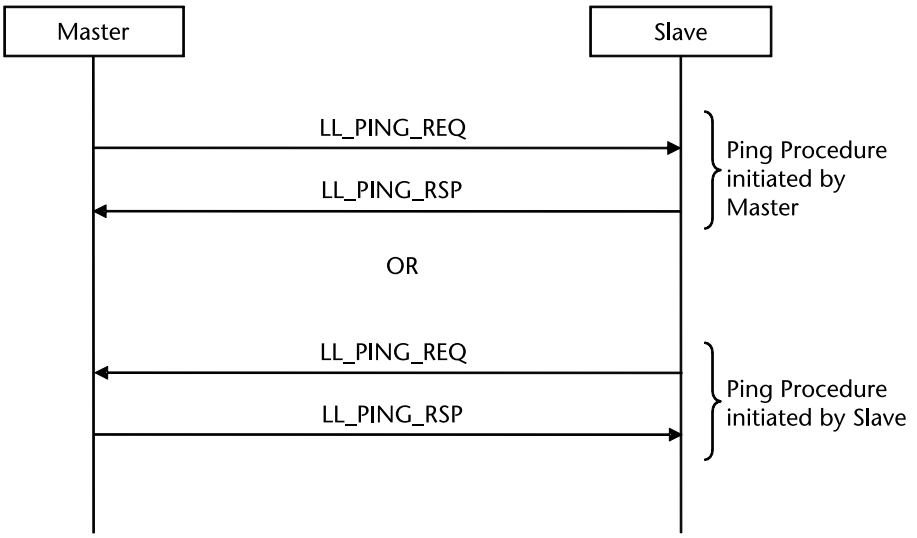

Figure 8.31 LE ping procedure.

- PDU time (connMaxRxTime);
- Maximum transmit data channel PDU payload length (connMaxTxOctets);
- PDU time (connMaxTxTime).

While responding, the remote entity provides its own parameters, which could, for example, have changed because of the request that was just received.

This procedure may be initiated by the Master or Slave and allows the initiating device to inform the remote entity whenever any of these parameters change. For example, if the data buffers are getting full, it may inform the remote entity that the maximum receive data channel PDU payload length has decreased. Once the buffers are freed up, it may increase the maximum receive data channel PDU payload length again. The sequence diagram for this procedure is shown in Figure 8.32.

8.13 Management of Link Layer Procedures

In order to have a smooth interaction between peer link layers, certain rules have been defined regarding timeouts and the handling of collisions. These are explained in the following sections.

8.13.1 Procedure Response Timeout

Since LE is a wireless protocol, it's possible that the peer devices may go out of range at any time and possibly come back in range within a certain time interval. Besides this, it's possible that the remote device may stop responding (i.e., if there is a bug and the device hangs or the battery dies).

The link layer defines a procedure response timeout of 40 seconds. If a response to a link layer PDU is not received within this timeout, the connection is considered to have been lost and the Host is notified of this connection.

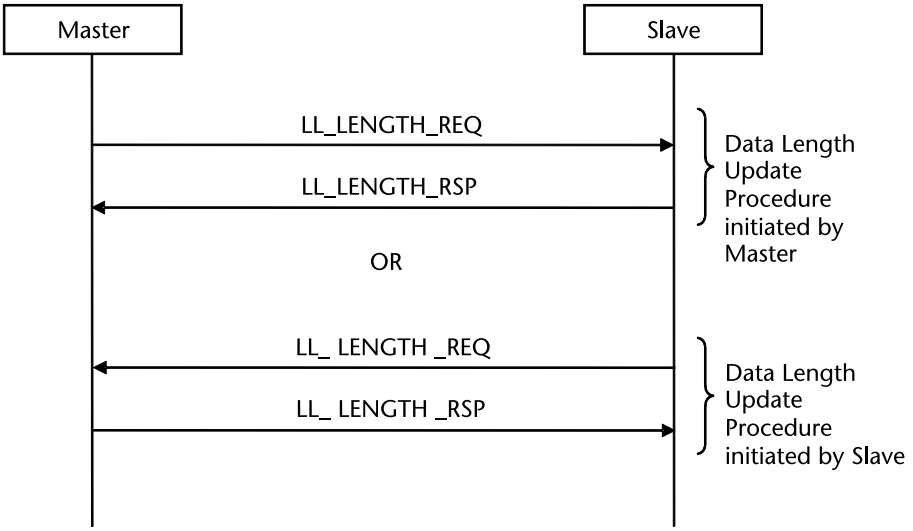

Figure 8.32 Data length update procedure.

Please note that this is quite similar to the Link Supervision Timeout used in BR/EDR to monitor link loss. A Link Supervision Timeout Event is indicated to the Host if no packets are received from the remote side for that duration. The default link supervision timeout is 20 seconds for BR/EDR as compared to 40 seconds timeout used for link layer procedures.

8.13.2 Procedure Collisions

It is possible that the Master and Slave initiate the same procedure or procedures that update the same link layer parameters. In that case, the procedure initiated by the Master gets priority. Therefore, if the Master has initiated or is in the process of initiating a similar procedure as the one initiated by the Slave, the Master will reject the procedure initiated by the Slave.

8.13.3 LE Authenticated Payload Timeout

Specifications 4.1 introduced the support of having a maximum time interval within which an authenticated packet must be received from the peer device. An authenticated packet is a packet containing a valid MIC.

If a packet is not received during this interval and the timeout is about to expire, the link layer can send a ping request (LE_Ping procedure) in order to receive an authenticated packet from the remote side.

The default value of authenticated payload timeout is 30 seconds.

8.14 Link Layer Privacy 1.2

Since many of the LE devices are supposed to be carried or worn by people (like shoes, watch, heart rate sensor, etc.), tracking those devices would allow a person

to be tracked by tracking the transmissions from his or her devices. This could compromise a person's privacy.

Take, for example, a fitness band which is worn on the user's wrist. Whenever it has to transmit data to a mobile phone, it begins advertisement. The mobile phone will receive these advertisement packets, initiate a connection, and transfer the data. These advertisement packets can be picked up by sniffers or other malicious devices to determine the user's location. Thus, tracking the advertisement packets over a period of time could be used as a mechanism with which to track the user.

The privacy feature is used to prevent the tracking of devices over a period of time. This is done by changing the Bluetooth device address frequently. Instead of sending out the real address in advertisement packets, a pseudo-random value that changes over time is inserted. The information on how to resolve this random address to know the real address of the device is known only to the peer device because it is exchanged when the devices are paired. (As explained briefly in Chapter 3, pairing is the process of associating two devices with each other and creating a trust relationship between the two by exchange security keys. It will be covered at length in Chapter 11.) During pairing, identity resolution keys (IRK) are exchanged between the two devices, and only the device that has an IRK can resolve a pseudorandom address to the real address. The pseudorandom addresses change at frequent intervals that are known to both the devices.

This is an optional feature and was introduced in specification 4.0. The processing related to privacy was done by the upper layers of the protocol stack (usually running in the Host).

One of the key enhancements introduced in specifications 4.2 is privacy at the link layer level. This will be described in the following sections.

8.14.1 Address Resolution in the Controller Instead of the Host

Consider the previous example of a user wearing a fitness band. The address of the fitness band keeps on changing in order to maintain privacy. When the mobile phone receives the advertisement packet from the fitness band, the link layer (running in the Controller) can ascertain whether the packet is coming from the fitness band by resolving that address. Therefore, the link layer passes on the information to the Host for only the selected devices instead of all devices. In specifications 4.0, the private addresses were generated and resolved by the Host. In specifications 4.2, the private addresses are generated and resolved by the Controller without involving the Host. The Host provides the Controller with information called device identity information; this allows the Controller to resolve the addresses.

This can lead to a huge power savings in case there are several devices around and the Host is not interested in transmissions from all the other devices.

8.14.1.1 Device Identity and Resolving List

The Host provides the device identity information to the Controller. Thereafter, the Controller can resolve the addresses on its own without any involvement from the Host.

A device identity contains the peer device's identity address and the local and peer's key pair needed to resolve identities. These keys are called identity resolution keys (IRK) and will be discussed in Chapter 11.

The Host and the Controller refer to the peer device by the identity address when communicating with each other. This means that the Host refers to the peer device by providing the device identity address (i.e., while creating an LE connection). If the Controller is able to resolve the peer device, it sends the device identity address in the events (i.e., when providing an LE advertising report event).

A resolving list is a set of device identities for all bonded devices. The Host maintains this list and provides it to the Controller. It may add or remove device identities at any stage using the HCI commands LE_Add_Device_To_Resolving_List and LE_Remove_Device_From_Resolving_List.

There may be cases where the Controller cannot store all the device identities. In such cases, the Host may provide a subset of the resolving list to the controller.

8.14.2 Better Privacy

The private address of the device is used within advertisement packets and is used by the peer device for creating connections. This address changes at periodic intervals. The peer device resolves this address before initiating a connection.

8.14.2.1 Private Address Generation Interval

The link layer uses a timer to generate private addresses at periodic intervals. The private address is generated under the following two conditions: the Link Layer is reset or the timer expires.

Once the new address is generated, the timer is restarted. The specification recommends the timer to be 15 minutes.

8.14.2.2 Privacy in Different States

The link layer uses the private address during advertising, scanning, and initiating state.

Continuing the previous example, when the fitness band wants to create a connection with the mobile phone, it sends its private address in the advertisement packets (ADV_IND or ADV_DIRECT_IND). The mobile phone then resolves this address, and it may send its own private address in the scanning and initiating requests while also using the private address of the fitness band.

8.15 Device Filtering and White List

One of the important power savings mechanisms introduced by LE is device filtering. With the use of device filtering, the link layer can be restricted to respond to only a certain set of devices. This is done through white lists that are maintained by the link layer. This is nothing but the set of devices that the link layer responds to (advertisers, scanners or initiators).

The transmissions from devices that are not in the white list are simply ignored. This reduces the number of transmissions made by the link layer thereby reducing the power consumption of the controller. In addition to this, it also reduces the communication the controller has with the host thereby reducing both the host's and controller's power consumption.

Besides power savings, the white list also eases the host processing in use cases where a host may wish to establish a connection with any one of the devices amongst a list of devices (for example, if those devices provide similar data). At the time of LE connection, the host can specify a list of devices to connect to using white list. The controller connects to one of the available devices. The address to which the connection was established is returned in the Peer_Address parameter of LE_Connection_Complete Event. White List is very useful in such a scenario because the host does not have to repeatedly attempt making connections to each of the devices till the time it is able to successfully make a connection. This eases the host processing and lowers power consumption by reducing the communication between the host and the controller. The white list is initialized to empty by the controller at the time of reset. There are three filter policies that make use of the white lists. These are described below.

8.15.1 Advertising Filter Policy

This policy determines how the link layer of the Advertiser processes scan and connection requests.

There are four possible modes and one of these can be selected by the host by using the HCI command HCI_LE_Set_Advertising_Parameters.

- Process scan and connect requests only from devices in white list.
- Process scan and connect requests from all device (White list not in use). This is the default on reset.
- Process scan request from all devices but connect request from only devices in the white list.
- Process connect request from all devices but scan request from only devices in the white list

This policy is used only when the link layer is not using connectable directed advertising. In the case of connectable directed advertising, the Advertiser accepts the scan or connect request only from the device which it addresses in the advertising events. So this policy is not needed.

8.15.2 Scanner Filter Policy

This policy determines how the Scanner's link layer processes advertising packets. There are two possible modes and one of these can be selected by the host by using the HCI command HCI_LE_Set_Scan_Parameters.

- Process advertising packets only from devices in the white list.

- Process all advertising packets. (White list is not in use). This is the default on reset.

As per specifications version 4.0, a connectable directed advertising packet that did not contain the scanner's address was ignored. This simply meant that all of the packets that were not meant for the scanner were ignored. This is not useful if the scanner is using a resolvable private address because whether the packet is meant for the scanner or not would be known only after the address is resolved.

Specifications version 4.2 provided support for extended scanner filter policies. With this, two new scanner filter policies were added:

- Process advertising packets only from devices in the White List (same as 4.0) and do not ignore a connectable directed advertising packet if the initiator address (InitA field in ADV_DIRECT_IND PDU) is a resolvable private address.
- Process all advertising packets (same as 4.0) and do not ignore a connectable directed advertising packet if the initiator address (InitA field in ADV_DIRECT_IND PDU) is a resolvable private address.

The default on reset is still the same as specifications 4.0—process all advertising packets and the White List is not in use.

This mechanism has strengthened the Link Layer Privacy. Now the scanner can enable the filter policy and both the advertiser and the scanner can use the resolvable private address of the scanner instead of using a public address. The filter policy would enable the resolving of the address before deciding to accept or discard the advertising PDU.

8.15.3 Initiator Filter Policy

This policy determines how the Initiator's link layer processes advertising packets. There are two possible modes and one of these is selected at the time of creation of a connection using the HCI_LE_Create_Connection command.

- Process connectable advertising packets from all devices in the white list.
- Process connectable advertising packets from a specific single device specified by the host. (White list is not in use). This is the default on reset.

8.16 Practical Examples

Figure 8.33 shows an air log capture of the transactions happening between the link layers of two devices to support the Proximity profile. These procedures are carried out after a connection has been established between the two devices.

The sequence of procedures that are carried out is as follows:

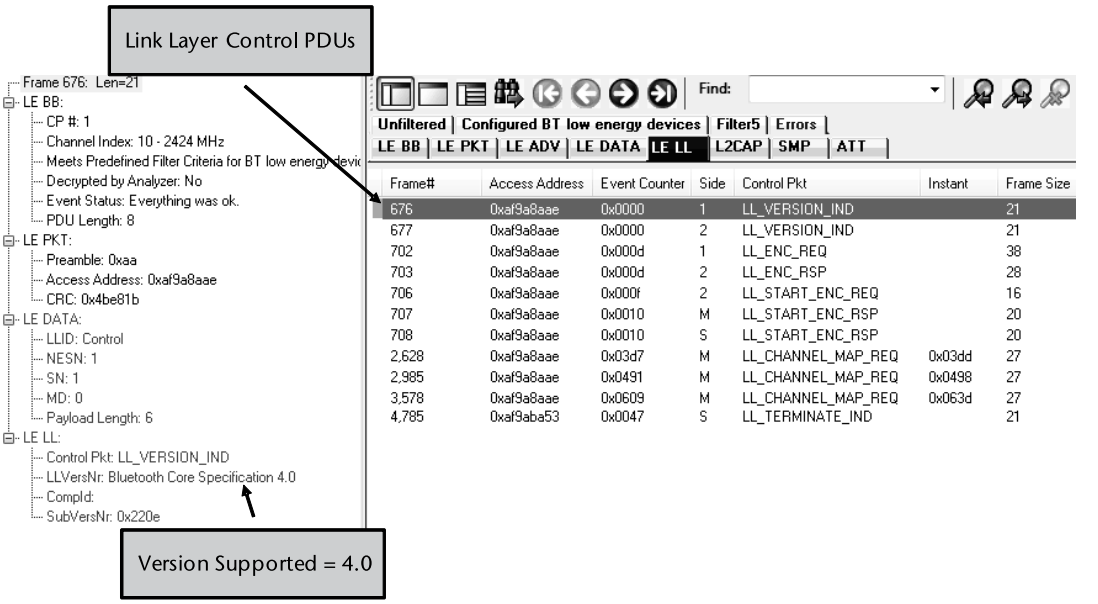

Figure 8.33 Practical example of link layer transactions.

- Frames #676 and #677: The link layer of the Master and Slave exchange information about the versions that they support.
- Frames #702 to #708: These frames are used to start encryption on the link.
- Frames #2628, #2985, #3578: The link layer of the Master provides updated Channel Map to the Slave.
- Frame #4785: The Slave requests the link to be terminated.

8.17 Summary

As the name suggests, the link layer is responsible for the maintenance of the link. This includes establishing the link, selecting the frequencies, supporting different topologies and disconnecting the link. LE uses a very simple architecture for link layer with just five states. It imposes several restrictions in order to make the link layer state machine very simple, thereby reducing both the silicon cost of implementation as well as the power consumption.

LE uses dedicated channels for advertisement and data. The PDUs that can be exchanged between the link layers of the two devices were explained in detail in this chapter.

The next chapter will focus on HCI interface. The HCI interface is used by the upper layers to interface with the link layer.

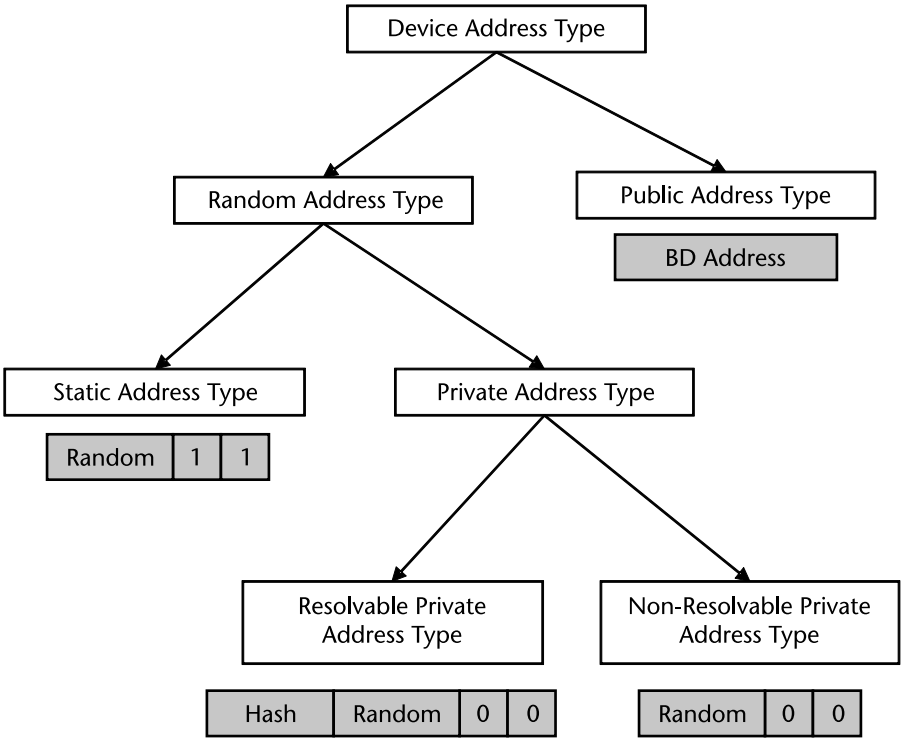

Figure 8.34 Format of various types of device addresses.

Bibliography

Bluetooth Core Specification 4.0 http://www.bluetooth.org.

Bluetooth Low Energy Training and Marketing information from the Bluetooth SIG website. http://www.bluetooth.org.