CHAPTER 3

Bluetooth Lower Layers

3.1 Introduction

The layered architecture of the Bluetooth protocol stack was introduced in the previous chapter. This chapter covers the lower layers of the Bluetooth protocol stack including the Bluetooth Radio, Baseband Controller, Link Manager and Host Controller Interface. These are shown in the bottom half of Figure 3.1.

The Host Controller Interface specification is common for BR/EDR and LE. So the Host Controller Interface will be explained in detail in this chapter for both BR/EDR and LE. Only a few LE specific parts will be explained in Chapter 9.

Some scenarios on how these layers come together to implement certain practical uses are shown toward the end of this chapter.

3.2 Bluetooth Radio

Bluetooth Radio operates in the 2.4 GHz ISM band. It uses a frequency hopping mechanism with 79 channels to combat interference. A Time Division Duplex (TDD) scheme is used for full duplex transmission.

This layer is responsible for the following primary tasks:

1. Transmission and Reception of packets: This includes modulation and demodulation of the packets. Two modulations modes are defined:
   a. Basic Rate: This mode uses a shaped binary FM modulation mechanism and is designed to minimize complexity of the transceiver. It provides a gross air data rate of 1 Mbps.
   b. Enhanced Data Rate: This mode uses Phase Shift Keying (PSK) Modulation and supports higher data rates. The gross air data rate supported is 2 Mbps or 3 Mbps.
2. Support appropriate power class: Three power classes are defined by the Bluetooth specification based on the maximum output power. A higher value of maximum output power leads to a longer range. The device can support the power class most appropriate to its intended use:
   a. Power Class 1: Maximum output power of 100 mW.

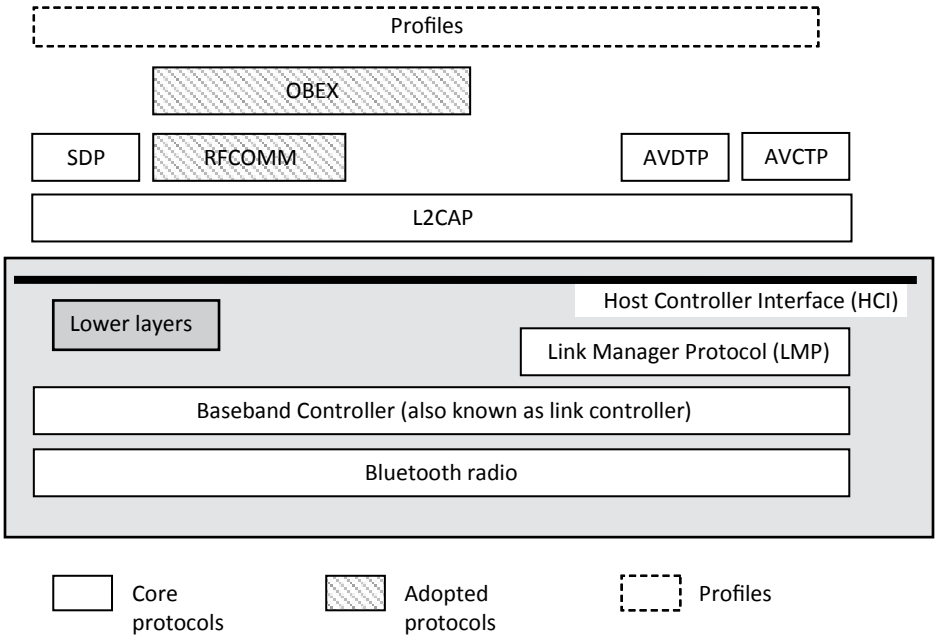


Figure 3.1 Lower layers in Bluetooth protocol stack.

b. Power Class 2: Maximum output power of 2.4 mW.
c. Power Class 3: Maximum output power of 1 mW.

3.2.1 Frequency Band and Hopping

Bluetooth uses the globally unlicensed 2.4 GHz ISM band for communication.

Bluetooth divides the frequency band into 79 channels. Each channel is 1 MHz wide. To combat interference, the Bluetooth devices change channels up to 1600 times per second. So even if there is noise on one channel, the next transmission is on a different channel which may be noise free.

What is ISM?

ISM stands for Industrial, Scientific and Medical radio band. Besides 2.4 GHz, these bands include frequencies in the range of 13.560 MHz, 27.120 MHz, 5.8 GHz, 24.125 GHz and several more. These bands are reserved internationally and do not require any special license to operate.

ISM bands are shared by several devices including Remote control toys, cordless phones, near field communication, wireless LAN, etc. Some microwave ovens also generate interference in these bands. Since these bands may be shared by many devices, different wireless technologies employ different mechanisms to combat interference.

The pattern of changing the channels is pseudo-random so that all devices which are communicating with each other know which frequency to hop to next.

The frequencies of various channels are derived from the formula:

$$f(k) = 2402 + k \text{ MHz}, k = 0, \ldots, 78$$

This is called *frequency hopping*. The set of devices that communicate with each other follow the same hopping pattern so that they can listen to data sent by the other devices. This set of devices is referred to as a piconet which is the fundamental unit of communication in Bluetooth. Piconet will be explained in detail in the next section.

3.3 Baseband Controller

The Baseband controller (also referred to as the Link Controller) performs the following major functions:

1. Management of physical channels and links for single or multiple links.
2. Selection of the next hopping frequency for transmitting and receiving packets.
3. Formation of piconet and scatternet.
4. Formation of packets and then giving them to the Bluetooth radio for transmission.
5. Inquiry and Inquiry Scan.
6. Connection and Page Scan.
7. Security (including data encryption).
8. Power management (including low power modes).

3.3.1 Topology—Piconet and Scatternet

Bluetooth supports both point-to-point connections and point-to-multi-point connections. In point-to-point connection, the physical channel is shared by two devices while in point-to-multi-point connection it is shared by multiple devices.

The piconet is the smallest unit of communication in Bluetooth. Two or more devices sharing the physical channel are said to form a piconet. It is characterized by one Master and up to seven active Slaves. All the devices are synchronized to each other in terms of clock and frequency hopping pattern. This common piconet clock is the same as Master's clock and frequency hopping pattern is determined by Bluetooth device address (BD_ADDR) and clock of the Master.

Several piconets may co-exist in proximity with each other without interfering with each other. This is because each piconet will have its own Master and thus its own frequency hopping pattern. The chances of two pseudo random frequency hopping patterns which are hopping on 79 different frequencies to select the same frequency for next transmission are quite remote.

A scatternet is formed when two piconets share a device. The shared device participates in the two piconets in a Time Division Multiplexing manner. So it

participates in the first piconet for some time and then participates in the second piconet for the remaining time. Before moving to the second piconet, the device puts itself in low power mode in the first piconet so that the other devices of the first piconet are aware of its temporary absence. This can be extended further to any number of piconets.

Figure 3.2 shows three piconets joined together to form a scatternet. Two scenarios are shown:

- Device M2 is Master in one piconet and Slave in the second piconet.
- Device S7 is Slave in two piconets.

It's not possible for a device to be in a Master in two piconets. Why?

This is because the piconet is defined by the Master's BD_ADDR and clock. So, in effect, all devices that are synchronized with that Master form one single piconet and not two different piconets.

3.3.2 Time Division Duplex

The physical channel is divided into slots in the time domain. Each slot is 625 microseconds and a packet may be sent in 1, 3, or 5 slots depending on the length of the packet. The Master and Slave send the packet alternately in slot pairs. A slot pair starts with a Master transmitting a packet to one of the Slaves. The packet is 1, 3, or 5 slots in length. The response to that packet is received from the Slave (to which the Master had sent the packet) in the next slot. The response packet may also be 1, 3 or 5 slots in length.

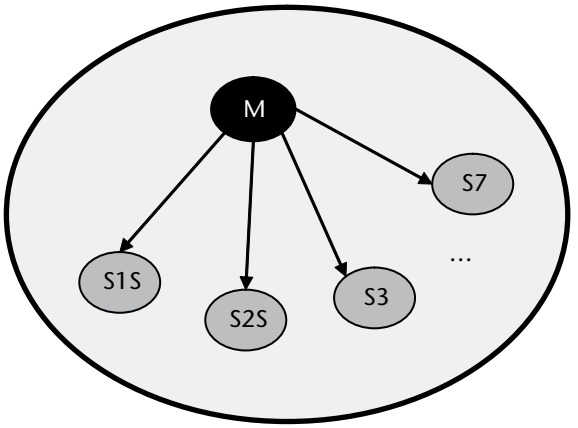
The Master can send packets to a Slave in only EVEN Slots. The Slave can send packets to Master only in ODD slots. This is illustrated in Figure 3.3.

3.3.3 Adaptive Frequency Hopping (AFH)

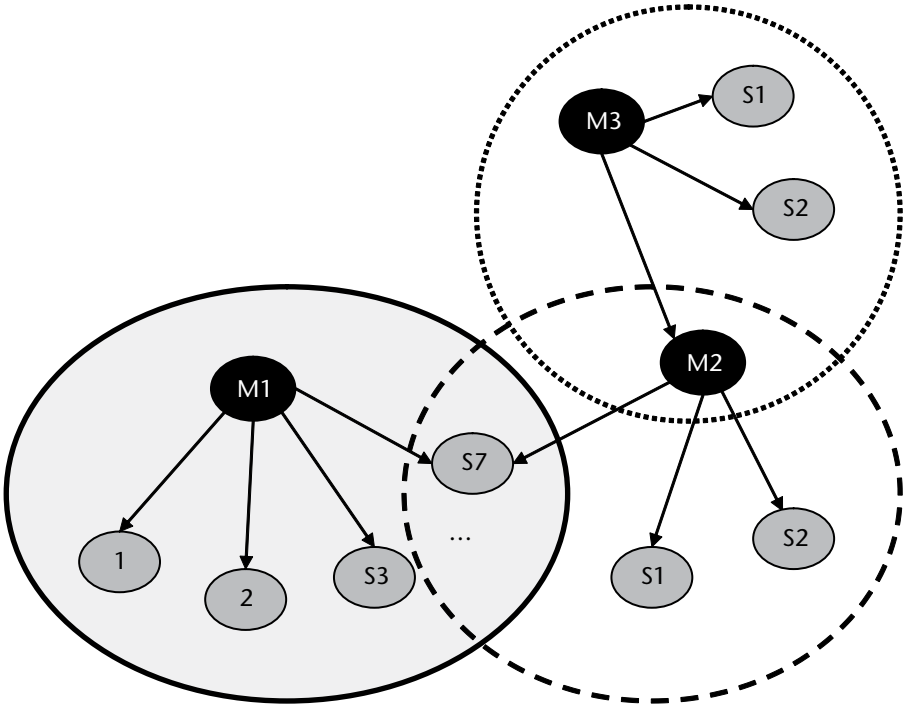
Normal frequency hopping provides some level of protection to interference from other devices in the sense that if another device (whether Bluetooth or not) is using the same RF channel and the data is corrupted due to interference, then the data would be retransmitted next time on the next pseudo random channel. It is possible that the next pseudo random channel is noise free and the packet gets received successfully. This will still lead to loss of throughput since there are retransmissions in case of interference and retransmissions need bandwidth.

The Bluetooth 1.2 specification introduced AFH which helps to increase the immunity of Bluetooth devices against interference generated by other systems in the ISM band.

Also, AFH serves another purpose of reducing the interference caused by Bluetooth on other devices in the ISM band.



Piconet with one Master and up to seven Slaves



Scatternet operation with 3 piconets
M2 is Master in 1 piconet and slave in another
S7 is slave in 2 piconets

Figure 3.2 Bluetooth topology.

When AFH is enabled, the channels which have interference are marked as unused. The Master informs the unused channels to the Slaves and these channels are excluded from the frequency hopping pattern. So, even though the number of channels on which the frequency hopping occurs decreases, there is no decrease in the throughput. If AFH were not enabled, the frequency hopping would have occurred

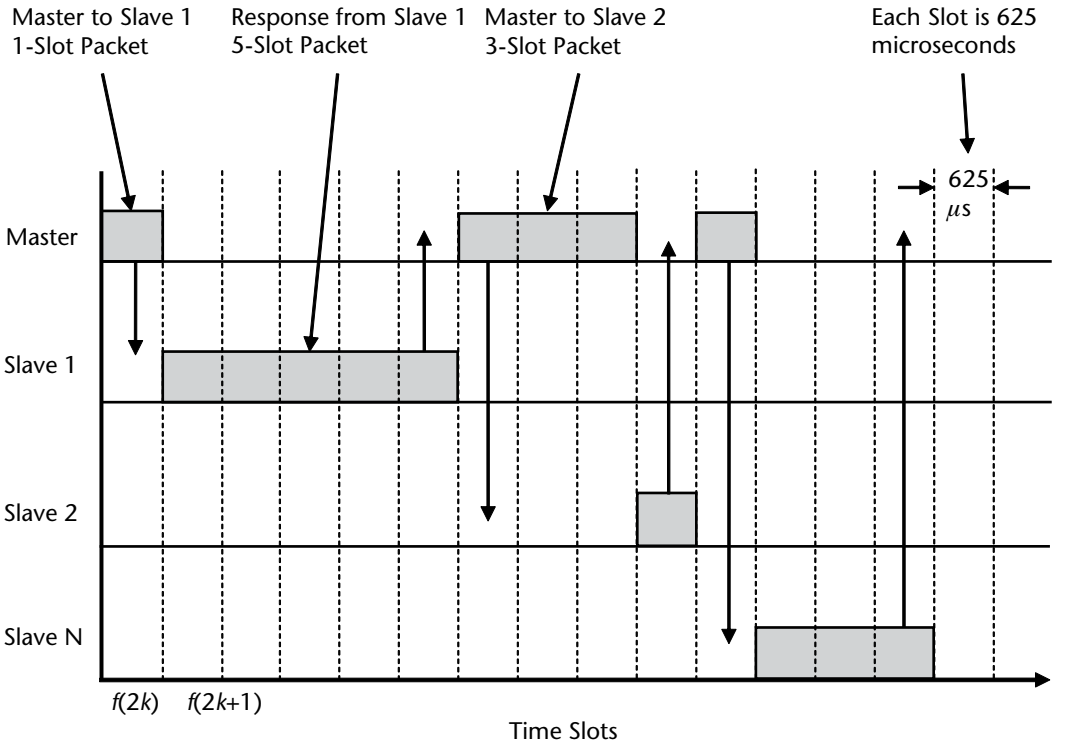


Figure 3.3 Time Division Duplex.

on some of the bad channels as well and it would have resulted in retransmissions. The retransmissions would have led to decrease in the throughput.

With AFH, the number of channels in use can be reduced until it reaches 20. Remember that without AFH the number of channels used by Bluetooth is 79.

To use AFH, the Master maintains a channel map in which it classifies a channel as used or unused. It keeps on updating the channel map based on information that it gathers about whether the packets could be transmitted successfully or not on a particular frequency. It can gather this information based on whether it received an acknowledgement of the packet that it sent or not.

The Slaves also help the Master by providing information on whether the channels are good or bad. Based on this information, the Master can mark the channel as used or unused.

AFH is enabled by the Master after a connection is made.

The Master may also request Channel Classification information from the Slave using LMP_channel_classification_req PDU. (The LMP PDUs are exchanged between the link manager of the Master and the link manager of the Slave. These will be explained in detail later). A Slave that supports this feature periodically provides the information on whether the channels are Good, Bad or Unknown using the LMP_channel_classification PDU.

The Master can use this information to update its channel map. It sends the updated channel map to the Slave using the LMP_set_AFH PDU.

The updated channel map is sent to Slaves periodically so that only the channels marked as *used* are used during frequency hopping.

In a piconet it's possible that a Master enables AFH with connection to some Slaves but disables it for some other Slaves. This will especially be the case if some Slaves support Bluetooth 1.2 or newer specification while others support an older version.

3.3.4 Master, Slave Roles and Role Switch

As mentioned earlier, a piconet consists of one Master and up to seven Slaves. The devices in the piconet are synchronized to a common clock and frequency hopping pattern. This synchronization reference is provided by the Master.

When a device decides to connect to another device (maybe after doing an inquiry), it initiates the connection procedure. By default, the device which initiates the connection becomes the Master. It is also possible for the other device (which is acting as Slave) to become the Master by doing a role switch. It can do so either during the connection establishment itself or any time after the connection is established.

The procedure to swap the roles of two devices connected in a piconet is called role switch. It is initiated using the HCI_Switch_Role command. (HCI commands are sent by the host to the controller on the HCI interface to request the controller to take specific action. These will be explained in detail later).

Either of the two devices (Master or the Slave) can initiate a role switch.

> One interesting point to note at this stage is that LE does not allow Role Switch in order to keep things simple. We'll come to that later when we dig deeper into LE.

3.3.5 Channel, Transport and Links

The Bluetooth data transport system follows a layered architecture. This is shown in Figure 3.4. The lowest layer comprises the physical channel that provides the communication mechanism between the devices on a Time Division Duplex channel. At the top, L2CAP channels provide communication channels for upper layers of the protocol stack to exchange data. Each of the layers is explained in the following sections going from bottom to top.

3.3.5.1 Physical Channel

The physical channel is the lowest architectural layer for communication in Bluetooth. It is characterized by the pseudo-random frequency hopping sequence, the specific slot timing of the transmissions, the access code and the packet header encoding. This means that for two or more devices to communicate with each other, their radios must be tuned to the same RF frequency at the same time. Besides this, the radios must be in range to listen to each other's transmissions.

When a device is synchronized to the timing, frequency and access code of a physical channel it is said to be connected to that channel.

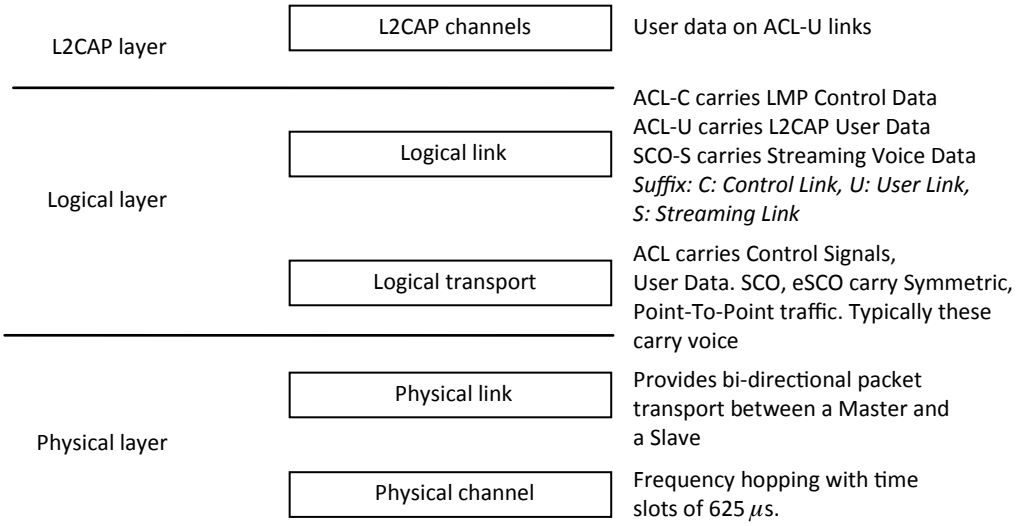


Figure 3.4 Data transport architecture.

The physical channel is subdivided into time units called slots. Each slot is 625 μs and full duplex transmission is achieved using Time Division Duplex (TDD) scheme. A packet may occupy 1, 3, or 5 slots depending on the packet type. As mentioned earlier a frequency hopping scheme is used and each packet is transmitted on the next pseudo-random frequency. The frequency hop occurs once per packet and not once per time slot.

Four physical channels are defined:

- *Basic Piconet Channel:* This channel is used for communication between devices in a piconet.
- *Adapted Piconet Channel:* This is similar to basic piconet channel and is used with devices that have AFH enabled.
- *Inquiry Scan Channel:* This channel is used for discovering the devices.
- *Page Scan Channel:* This channel is used for creating connections with the devices.

A device can use only one physical channel at a time and has to use time division multiplexing between the channels to support other channels if it has to support multiple concurrent operations.

For example, if a device is already connected to another device but still needs to be in discoverable mode so that other devices can discover it. It already has a physical channel established with another device which is either the basic piconet channel or adapted piconet channel depending on whether AFH is enabled on that channel. To support discoverable mode along with the physical channel, it has to use time division multiplexing to support the inquiry scan channel. This feature would be useful when forming a scatternet or adding more devices to the piconet.

3.3.5.2 Physical Link

A physical link defines the baseband to baseband connection at the lowest level. The physical link is what actually carries the data physically over the channel. There can be only one physical link between two Bluetooth devices. This means that if we consider two devices—one Master and the other Slave—there will be only one physical link between these two devices although at a logical level there could be two or more links on top of it. (Logical links are explained in the next section.)

Within a physical channel, a physical link is established between any two devices that transmit packets. In a piconet, a physical link is formed between the Master and each Slave. It is not permitted to form a physical link between two Slaves.

It is worthwhile to note that Bluetooth does not support direct communication between two Slaves. If the two Slaves need to talk, then they should disconnect from their respective piconets and form a separate piconet with one of them acting as the Master and the other acting as a Slave.

A physical link is always associated with exactly one physical channel.The physical link supports link supervision so that a connection loss can be detected. This could happen, for example, when the device moves out of range or one of the devices is powered off. The physical link also supports encryption of the packets that are transmitted.

3.3.5.3 Logical Transport

A logical transport is formed on a physical link to transfer control signals and synchronous or asynchronous user data. The following three Logical Transports are most commonly used.

1. Asynchronous Connection Oriented (ACL) Logical Transport.
2. Synchronous Connection Oriented (SCO) Logical Transport.
3. Extended Synchronous Connection Oriented (eSCO) Logical Transport.

Asynchronous Connection Oriented (ACL) Logical Transport

The Asynchronous Connection Oriented (ACL) transport is used to carry control signals, user data and broadcast traffic. It provides a packet-switched connection between the Master and all Slaves that are active in the piconet. The default ACL connection is created when a device joins a piconet. This transport is used to send data in bursts whenever data is ready and whenever a slot is available and not reserved for the synchronous connection oriented transport.

Between a Master and a Slave only one single ACL logical transport is created. The higher layer protocols multiplex their data on top of this logical transport. Packet retransmissions are supported on the ACL transport to ensure data integrity when the packets are delivered to the application on the receiver side.

If a SCO or eSCO transport exists between the Master and the Slave, then the time slots are first allocated to the SCO or eSCO transport. (This is because SCO and eSCO slots carry time bounded data like voice which need guaranteed bandwidth.) The remaining slots are reserved for the ACL transport. (The SCO and eSCO transports are covered in the next section.)

You might be wondering why the acronym is ACL and not ACO?

This is for historical reasons. In previous versions of the Bluetooth specification (Prior to Core Spec 1.2) this link was called Asynchronous Connection Less and abbreviated as ACL. That abbreviation still continues…

Synchronous Connection Oriented (SCO) Logical Transport

The Synchronous Connection Oriented (SCO) transport provides support for continuous transfer of data. It reserves time slots on the physical channel at the time of establishment of the connection so it can be considered as a circuit-switched connection. It is a symmetric point-to-point link between a Master and a particular Slave. SCO links carry 64 kb/s of information in both directions. Typically this is useful in transporting voice streams where timing of the data is as important as the data content.

If a data packet is corrupted during transmission it is not retransmitted. This is suitable for voice data since if there are retransmissions, these would introduce delays. Hence the retransmitted packets may arrive too late to be played. On the other hand if the average number of dropped packets is not too high, it may not lead to noticeable degradation in the user experience.

Unlike ACL logical transport, the SCO (and eSCO) logical transports do not support multiplexed logical links on top of them. So there is no further layering of any stack components above SCO (and eSCO) logical transport.

A Master may support up to three SCO links to the same Slave or to different Slaves. A Slave may support up to three SCO links from the same Master or two SCO links if the links originate from different Masters. (SCO links can originate from different Masters in the scatternet topology where a Slave is connected to two Masters in two different piconets.)

A Master can support an ACL transport and a SCO transport simultaneously with the Slave. One practical example of this is when connecting to a mono headset from a mobile phone. The mobile phone establishes both the ACL transport as well as the SCO transport with the headset. The ACL transport is used to carry commands to create a connection to the headset, increase/decrease volume, disconnect, etc. while the SCO transport is used to carry the voice traffic.

In theory the Master can support up to 3 SCO transports simultaneously with an ACL transport to the Slave. In practice this is not used much. Most of the practical implementations use only one SCO link to transfer voice data along with ACL link to transfer control data.

Extended Synchronous Connection Oriented (eSCO) Logical Transport

The extended synchronous connection oriented (eSCO) link is a symmetric or asymmetric point-to-point link between a Master and a particular Slave. This is in contrast to SCO link which supports only symmetric traffic. The eSCO logical transport also reserves slots similar to SCO logical transport and can be considered as a circuit-switched connection.

The support for eSCO Logical Transport was added from version 1.2 of the Bluetooth specification as an enhancement to the functionality of the SCO Logical Transport.

It offers the following extensions over the SCO link:

- The transmission may be symmetric or asymmetric in the case of eSCO while it's only symmetric in the case of SCO. This means that in the case of eSCO it's possible to use 3-slot packets in one direction and 1-slot packet in the reverse direction.
- eSCO logical transport provides better reliability (and voice quality) by providing a limited number of retransmissions of packets that get corrupted. The retransmission slots are allocated right after the reserved slots and are used only if needed. This offers better voice quality as compared to SCO transport. The number of retransmission slots is limited to ensure that the timing constraints of voice data are still met in case of retransmissions and that packets don't arrive too late to be played.
- eSCO provides higher data rates as compared to SCO links. This is achieved by introducing additional eSCO packet types related to Enhanced Data Rate.
- eSCO is used to transfer not only 64 kb/s voice packets, but also any other types of packets which require constant traffic. For example the eSCO Logical Transport is used to transfer *Wide Band Speech* data.

What is Wide Band Speech (WBS)?

The bandwidth of the sound signals used in telephony is limited to about 200–3400 Hz. This is referred to as Narrow Band Speech. As per Nyquist sampling theorem, it is sampled at the rate of 8 KHz. (The Nyquist sampling theorem is beyond the scope of this book. In brief it states that if a signal is band limited at B Hz, then it can be perfectly reconstructed from a sequence of samples if the sampling frequency is greater than 2 * B samples per second).

The 200–3400 Hz limitation on the bandwidth imposes a limit on the communication quality in Narrow Band Speech.

Most of the frequencies in speech signals are present below 7000 Hz. So increasing this bandwidth to 50–7000 Hz increases the naturalness of speech and gives the feeling of face-to-face communication. This is referred to as Wide Band Speech and is quite frequently used in cellular systems, voice over IP, etc. The signal is sampled at the rate of 16 KHz in the case of Wide Band Speech.

The 3G systems support Wide Band Speech and since the audio may be routed to a Bluetooth headset, the Bluetooth headsets also need to support Wide Band Speech to ensure that the voice quality is not degraded when routing the call over a Bluetooth link.

3.3.5.4 Logical Links

The logical links are supported on top of the logical transport. Five types of logical links are defined. These are described in Table 3.1.

The LC (link control) Logical link is carried in the packet headers. It contains low level information like acknowledgment/repeat request (ARQ), flow control and payload characterization. All other links are carried in the packet payloads.

The control ACL (ACL-C) logical link carries the control information between the link managers and the Master and Slave(s). It has a higher priority than ACL-U logical link.

The user ACL link (ACL-U) is used to carry user data. This link is used by the L2CAP layer. For example this link is used for transferring a file during a file transfer using FTP profile. (We will discuss L2CAP layer and FTP profile in the next chapter.)

The SCO-S link is supported on top of the SCO logical transport. Each SCO-S link is supported by a single SCO logical transport. Same is the case for eSCO-S logical link which is supported on top of the eSCO logical transport.

Most of the time the C, U, or S suffix is dropped and the links are just referred to as ACL, SCO, and eSCO, links. This does not lead to any ambiguity since the type of links being used is clear from the context. For example, if link manager data is being transferred then ACL would mean ACL-C logical link. Similarly if user data (L2CAP) is being transferred, ACL would mean ACL-U logical link. Some practical scenarios are explained in the Figure 3.5 for a better understanding of these links.

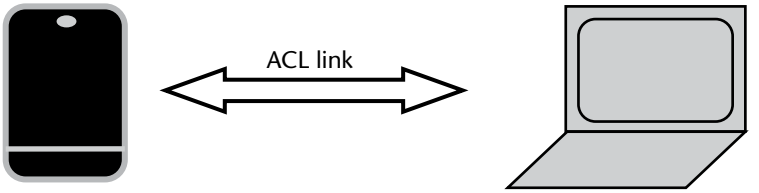
Scenario 1 shows a file transfer between a mobile phone and laptop. In this scenario, an ACL link is established between these two devices and the file transfer happens on this link.

Scenario 2 is a bit more complex. It shows a connection between a mobile phone and mono headset and routing of an audio call (from the cellular network) on the Bluetooth link to the headset. In this scenario, the initial connection establishment happens on the ACL link. Once the two devices are ready to exchange audio (For example, when the user presses the button on the headset to accept incoming call), the SCO or eSCO link is established.

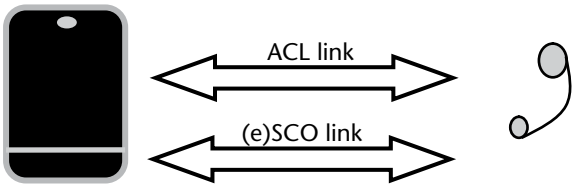
Scenario 3 shows the case where a stereo headset is connected to a mobile phone and a music file (MP3) is streamed over the Bluetooth link. The MP3 file is

Table 3.1 Types of Logical Links

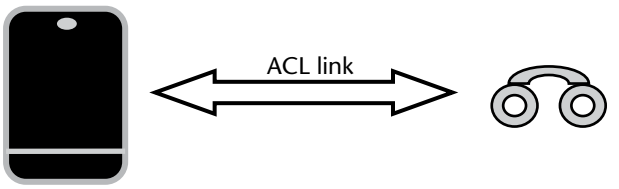
Link Type	*Traffic Type*	*Carrier*
Link Control (LC)	Low level link control information.	Mapped in the packet header and carried in every packet. (Except identity packet since it does not have a packet header).
ACL Control (ACL-C)	Control information between the link manager layers of Master and Slave(s).	Mostly ACL logical transport. May also be carried in data part of DV packets on SCO. (DV will be discussed later).
User Asynchronous/ Isochronous (ACL-U)	Asynchronous or isochronous user information between L2CAP layers	
User Synchronous (SCO-S)	Synchronous user information	SCO logical transport
User Extended Synchronous (eSCO-S)	Synchronous user information	eSCO logical transport



Scenario 1: File transfer between a mobile phone and a laptop



Scenario 2: Routing a voice call from mobile phone to Bluetooth mono headset



Scenario 3: Listening to music on a stereo Bluetooth headset

Figure 3.5 Usage of different link types.

generally encoded in a different format by the mobile phone (by default Bluetooth specification specifies the usage of an SBC codec) and then streamed on the ACL link.

The link to transfer *music* should not be confused with the link to transfer *voice*. Music links require much higher bandwidth as compared to voice links. This is because voice is sampled at 8 KHz (or 16 KHz for Wideband speech) while music is sampled at (up to) 48 KHz.

The SCO link has bandwidth to carry only 8 KHz voice. This can be enhanced to 16 KHz by using eSCO links. This is still not sufficient for music links.

So music is transferred over ACL links that provide a much higher bandwidth. In fact even in the case of ACL links, the bandwidth is not sufficient to transfer raw music. So the music file is first encoded and then transmitted. It is decoded after reception and then played back.

If this is not clear at this stage, just hang on and we will revisit the subject when we discuss Bluetooth profiles and in particular the A2DP profile in the next chapter.

3.3.6 Packet Format

The format of BR and EDR packets is shown in Figure 3.6.

All transmissions begin with an access code. This is the only mandatory part of a packet; all other parts are optional. It is used for synchronization, DC Offset compensation, and identification. All packets sent on the same physical channel are preceded by the same access code. It indicates the arrival of a packet on the receiver side.

Three different access codes are defined:

- *Device Access Code (DAC):* This is used during pre-connection phase when the devices are trying to connect to each other.
- *Channel Access Code (CAC):* This is used when the devices are connected and is prefixed to all packets that are exchanged between the devices in a piconet. The receiver uses this code to check if the packet belongs to the piconet that it is in or not.
- *Inquiry Access Code (IAC):* This is used during the inquiry phase when the devices try to find out other devices in the Bluetooth vicinity.

The packet header contains the following information:

- *Logical Transport Address (LT_ADDR):* Indicates the logical transport address of the destination device.
- *Packet Type (TYPE):* Various packet types are defined for ACL, SCO and eSCO logical transports. These will be explained in detail in the next section.
- *Flow control (FLOW):* To start and stop the flow of packets depending on whether the receiver is capable of receiving more packets or not. (For example if the buffers in the receiver are full, then this bit will be used to indicate to the transmitter to stop sending further packets)
- *Acknowledgment (ARQN):* Positive or negative acknowledgment to indicate the transmitter of a successful transfer of payload data after checking the CRC.
- *Sequence Number (SEQN):* Sequential numbering of packets to ensure that packets are received in the correct order.
- *Header Error Check (HEC):* To check the header integrity on reception. If the HEC is incorrect on the receiver side, the entire packet is discarded.

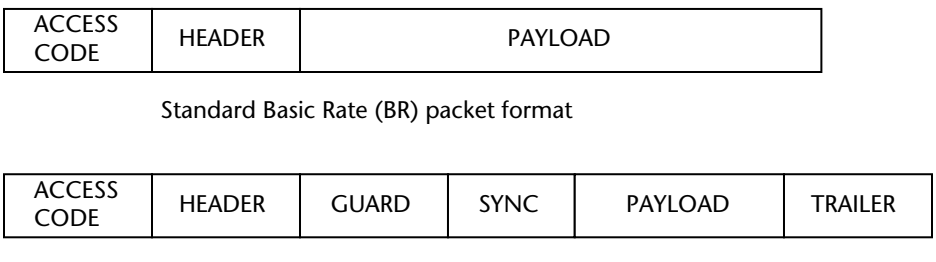


Figure 3.6 BR and EDR Packet Formats.

The payload contains the data that is to be transferred along with the CRC. The data can have asynchronous data field, synchronous data field or both. The ACL data packets have the asynchronous data field and the SCO/eSCO packets have the synchronous data field. There are a special DV packet types which are defined as a part of SCO packet types. These have both asynchronous and synchronous data field.

For EDR packets, GUARD and SYNC fields are placed before the payload and a TRAILER field is placed after the payload. This is because EDR packets use a different modulation scheme for the payload to achieve higher data rates. These fields are used to facilitate the change in the modulation scheme just before payload data is transmitted.

3.3.7 Packet Types

Different logical transports use different packet types. As mentioned in the previous section, the TYPE field within the packet header is used to indicate the various packet types. Before getting into the details of the packet types, it's important to understand the nomenclature. Figure 3.7 shows the significance of each of the alphabets in the packet types 3DH5 and 2EV3.

The Packet types can be classified into 3 broad categories:

1. Link Control Packets.

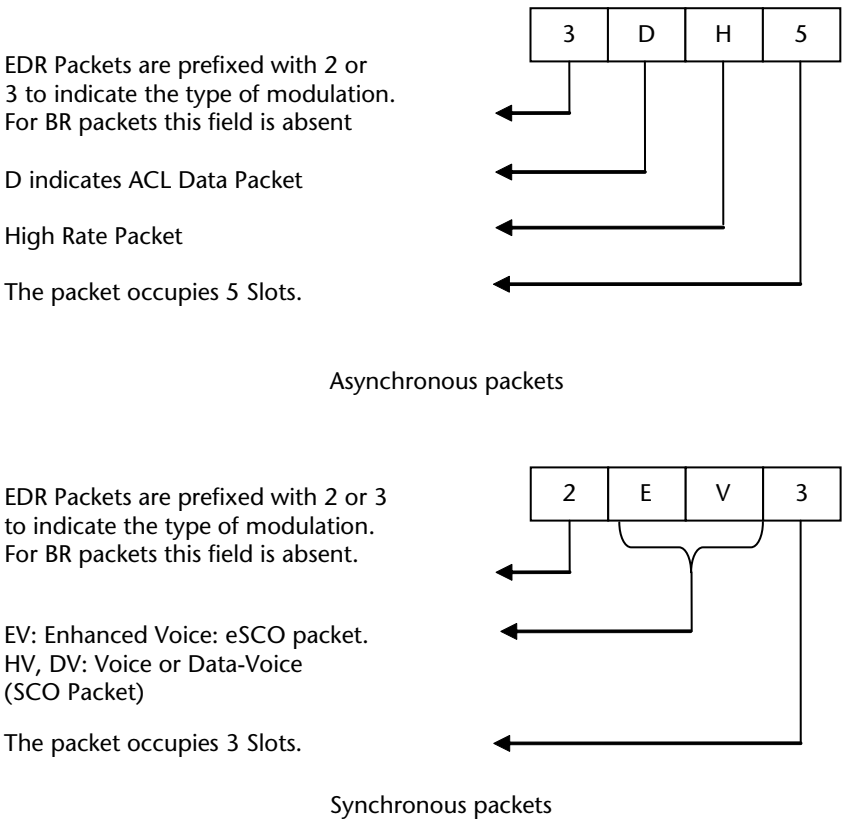


Figure 3.7 Packet Nomenclature.

2. ACL Packets.
3. Synchronous Packets.

These are briefly described in the following sections.

3.3.7.1 Link Control Packet Types

There are 5 link control packet types:

- ID;
- NULL;
- POLL;
- FHS;
- DM1.

The ID (Identity) packet is used before connection establishment. It's a very robust packet and contains the device access code (DAC) or inquiry access code (IAC). It does not have a packet header of payload.

The NULL packet has no payload. It is generally used to indicate success of the previous transmission or status of receive buffer. Typically this is used when a Slave receives a packet from the Master and it has to acknowledge that packet but it has no data to send back to the Master. So the Slave uses the NULL packet to acknowledge. The Slave also uses the NULL packet to indicate to the Master if its receive buffers are full. The Master will then halt transmissions till the time the Slave empties its buffers to receive further packets. The default packet type for data is NULL. This is used if there is no data to be sent. The NULL packet itself is not required to be acknowledged by the Master.

The POLL packet is quite similar to the NULL packet. It also does not have a payload. It is sent only by the Master. The Master generally uses this packet to poll the Slaves to first detect whether they are still present and secondly to check if they have any data to send. Unlike the NULL packet, the POLL packet must be

Table 3.2 Summary of Link Control Packets

Type	*Remarks*
ID	Identity (ID) packet. This packet is used before connection establishment to pass on an address.
NULL	The NULL packet has no payload. It is generally used to indicate success of previous transmission or status of Rx buffer.
POLL	The POLL packet has no payload. It is generally used by the Master to poll the Slaves. The Slave responds to this with a packet. If it doesn't have any information to send, it responds with a NULL packet.
FHS	The Frequency Hop Synchronization (FHS) contains real time clock information. It is used for frequency hop synchronization before the piconet channel has been established or when existing piconet changes to a new piconet (by means of a role switch).
DM1	The DM1 (Data Medium Rate 1-slot) packet type is used to carry control packets and data packets.

acknowledged by the Slave. If the Slave does not have any information to send, it responds to the Master with a NULL packet.

The FHS (Frequency Hop Synchronization) is a special control packet containing real time information. It contains the BD_ADDR and clock information of the sender. The clock information is updated before each retransmission (since it is real time and the clock would have changed since the last transmission) for this particular packet type. This packet is used for frequency hop synchronization before the piconet channel has been established or when existing piconet changes to a new piconet (by means of a role switch).

The DM1 (Data Medium Rate 1-slot) packet is used by the Link Manger and Link Controller to exchange control packets. Besides control packets, this packet can also be used to carry regular data. The details for this packet type are provided in the ACL Packet Types section.

The scenarios in which these packet types are used are illustrated later in Figure 3.10.

3.3.7.2 ACL Packet Types

The different types of ACL packets are explained in Table 3.3. Different packets types are suitable for different scenarios. For example when high throughput is desired in only one direction, 3-DH5 packets are the most suitable. This is the case when transferring files, downloading data from the internet etc. In such cases the majority of the data is transmitted in one direction and there are only a few bytes that need to be transmitted in the reverse direction.

The information about supported ACL packet types is provided at the time of ACL connection creation. The link managers running on the two devices negotiate the packet types that will be used subsequently based on this information. Only the packet types that are successfully negotiated are used for subsequent transmissions.

Table 3.3 ACL Packet Types

Type	*User Payload (bytes)*	*Symmetric Max Rate (kb/s)*	*Asymmetric Max Rate (kb/s)*		*Remarks*
			Forward	*Reverse*	
DM1	0-17	108.8	108.8	108.8	Data – Medium Rate, 1 Slot
DH1	0-27	172.8	172.8	172.8	Data – High Rate, 1 Slot
DM3	0-121	258.1	387.2	54.4	
DH3	0-183	390.4	585.6	86.4	
DM5	0-224	286.7	477.8	36.3	
DH5	0-339	433.9	723.2	57.6	This packet is used for maximum throughput in one direction for Basic Rate (BR) links.
AUX1	0-29	185.6	185.6	185.6	
2-DH1	0-54	345.6	345.6	345.6	
2-DH3	0-367	782.9	1174.4	172.8	
2-DH5	0-679	869.1	1448.5	115.2	
3-DH1	0-83	531.2	531.2	531.2	
3-DH3	0-552	1177.6	1766.4	235.6	
3-DH5	0-1021	1306.9	2178.1	177.1	This packet is used for maximum throughput in one direction for EDR links.

As shown in Table 3.3, the maximum throughput is achieved when using asymmetric data transfer using DH5 packets for BR and 3-DH5 packets for EDR.

3.3.7.3 Synchronous Packet Types

HV (High Quality Voice) and DV (Data-Voice) packets are used for SCO transmissions. The HV packets do not include a CRC and thus these are not retransmitted. The DV is a combined data-voice packet and it has separate sections for voice and data. These packets do not include a CRC on the voice section, but include a CRC on the data section. So in case the data section is not received properly at the remote side, it is retransmitted. The voice section however is not retransmitted and contains the next speech information.

eSCO uses EV packets. These packets include a CRC and retransmission is done if an acknowledgment is not received from the remote side within the retransmission window. EV3, EV4 and EV5 packets are used for Basic Rate (BR) operations. 2-EV3, 2-EV5, 3-EV3, 3-EV5 packets are used for Enhanced Data Rate (EDR) transmissions. As mentioned previously eSCO packets can be used for both 64kb/s speech as well as data at the rates mentioned below.

3.3.8 Link Controller States

There are three major states used in the link controller:

- *STANDBY:* This is the default state of the device. In this state, the device may also enter low power mode to save power. The link controller may leave this state to scan for inquiry or page messages from other devices or to itself inquire or page.
- *CONNECTION:* Once a device knows the address of the device to connect to (using the inquiry procedure), it can create a connection to it. This procedure is known as paging. The device that initiates paging becomes the Master

Table 3.4 Synchronous Packet Types

Type	*User Payload (bytes)*	*Symmetric Max Rate (kb/s)*	*Remarks*
HV1	10	64	
HV2	20	64	
HV3	30	64	
DV	10 + (0–9) D	64.0+57.6 D	Data Voice. This packet type can support both Data and Voice simultaneously. D indicates the Data Payload.
EV3	1-30	96	
EV4	1-120	192	
EV5	1-180	288	
2-EV3	1-60	192	
2-EV5	1-360	576	
3-EV3	1-90	288	
3-EV5	1-540	864	

after the connection is established. In this state, packets can be exchanged between the Master and the Slave(s). This state is left through a detach or reset command.

- *PARK:* A Slave can enter the park state if it does not need to participate in the piconet but still needs to remain synchronized with it. This mode helps conserve power. The Slave wakes up periodically to resynchronize and listen to the channel to decide whether it should come back to active state or not.

Besides this, there are seven substates. These are interim states that are used for discovering devices in the vicinity and establishing a connection. The seven substates are divided into two categories:

Device Discovery Substates

1. *Inquiry scan:* In this substate, the device listens for incoming inquiry requests.
2. *Inquiry:* This substate is used to discover devices in the vicinity.
3. *Inquiry response:* After receiving the inquiry message, the Slave enters this state and transmits an inquiry response message.

Connection Establishment Substates

4. *Page scan:* In this substate, the device listens for incoming connection requests.
5. *Page:* This substate is used by a device to connect to another device. The device that initiates the paging ends up being a Master after the connection is established and the device that was in page scan mode ends up being Slave.
6. *Slave response:* After receiving the page message, the Slave enters this state and transmits a Slave response message.
7. *Master response:* The Master enters this state after it receives the Slave response message.

The different states along with the permitted transitions from one state to another are illustrated in Figure 3.8. As an example, the transition of states and substates during inquiry, connection and disconnection is illustrated in Figure 3.9 and Figure 3.10.

3.3.8.1 Connection State

During the connection state, the packets can be exchanged between the Master and the Slave. The first few packets contain control messages that are exchanged between the link managers of the devices to setup the link and negotiate link parameters. If the connection is no longer needed, the connection state may be left by either a detach or reset command.

During the connection state, the device may be in one of the following three modes:

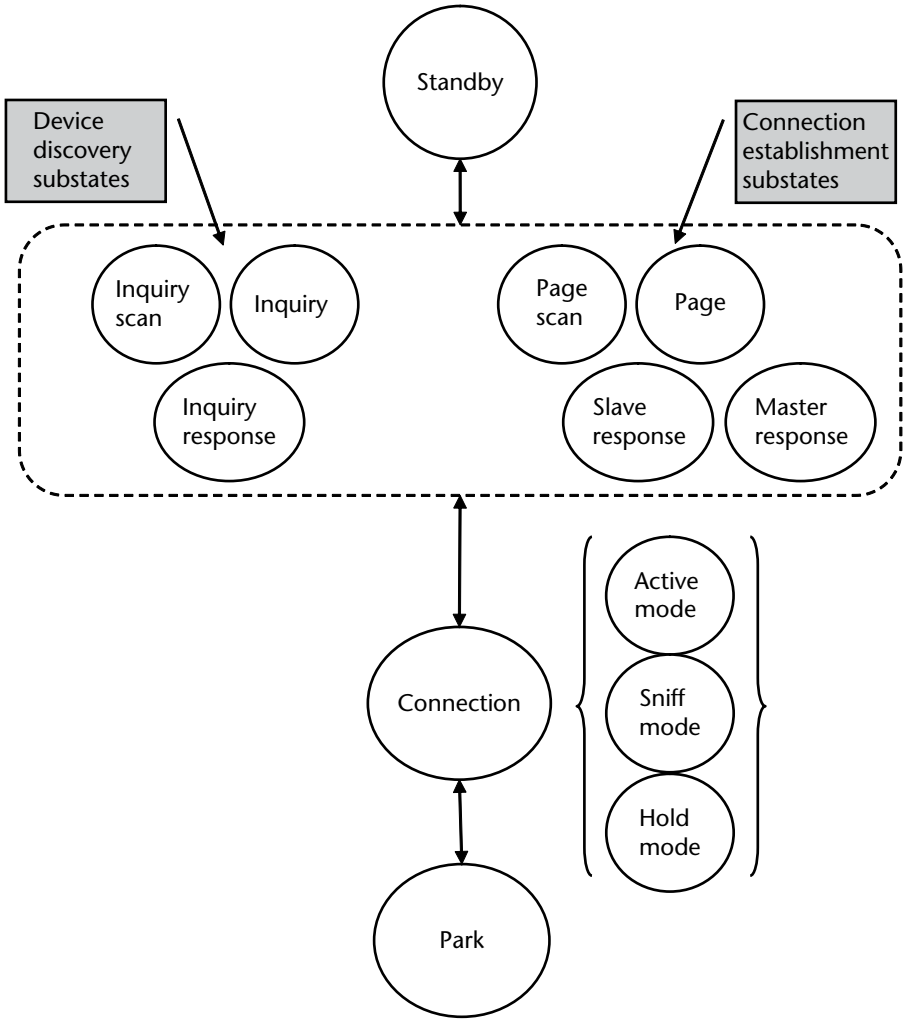


Figure 3.8 Link Controller States.

- Active Mode.
- Hold Mode.
- Sniff Mode.

Active Mode

In the active mode, one Master and up to seven Slaves can be active at any time in the piconet. The Master schedules the transmissions in different slots based on the SCO/eSCO connections that are established and the traffic requirements from different Slaves. Even if the Master does not have any data to send, it keeps on regularly transmitting POLL packets to the Slaves to keep them synchronized to the channel. If the Slave has data to send, it sends the appropriate data packet. Else it responds with a NULL packet to acknowledge the POLL packet.

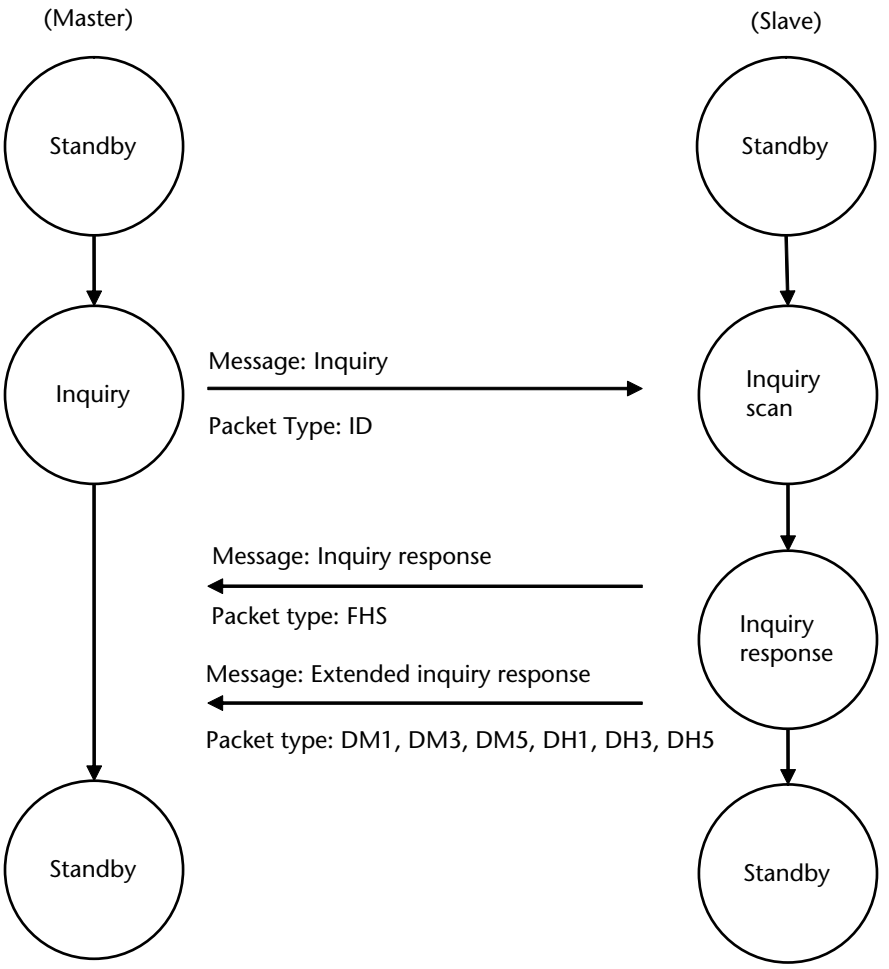


Figure 3.9 Link Controller Messages and States during inquiry.

Each Slave is assigned a 3-bit address called Active Member Address (AM_ADDR). The Master uses this address in the packets to identify the Slave for which those packets are meant.

Hold Mode

During the connection state, the ACL logical transport to the Slave can be set to hold mode. In this mode, the ACL packets are not exchanged. The voice links (SCO, eSCO) are still supported.

Before entering the hold mode, the Master and the Slave agree on the time duration for which the Slave will remain in hold mode. During this time duration, the Slave can either go to low power mode or do other activities like scanning, paging, inquiring or participating in another piconet.

On expiration of the time duration of the hold mode, the Slave wakes up and listens for transmissions from the Master.

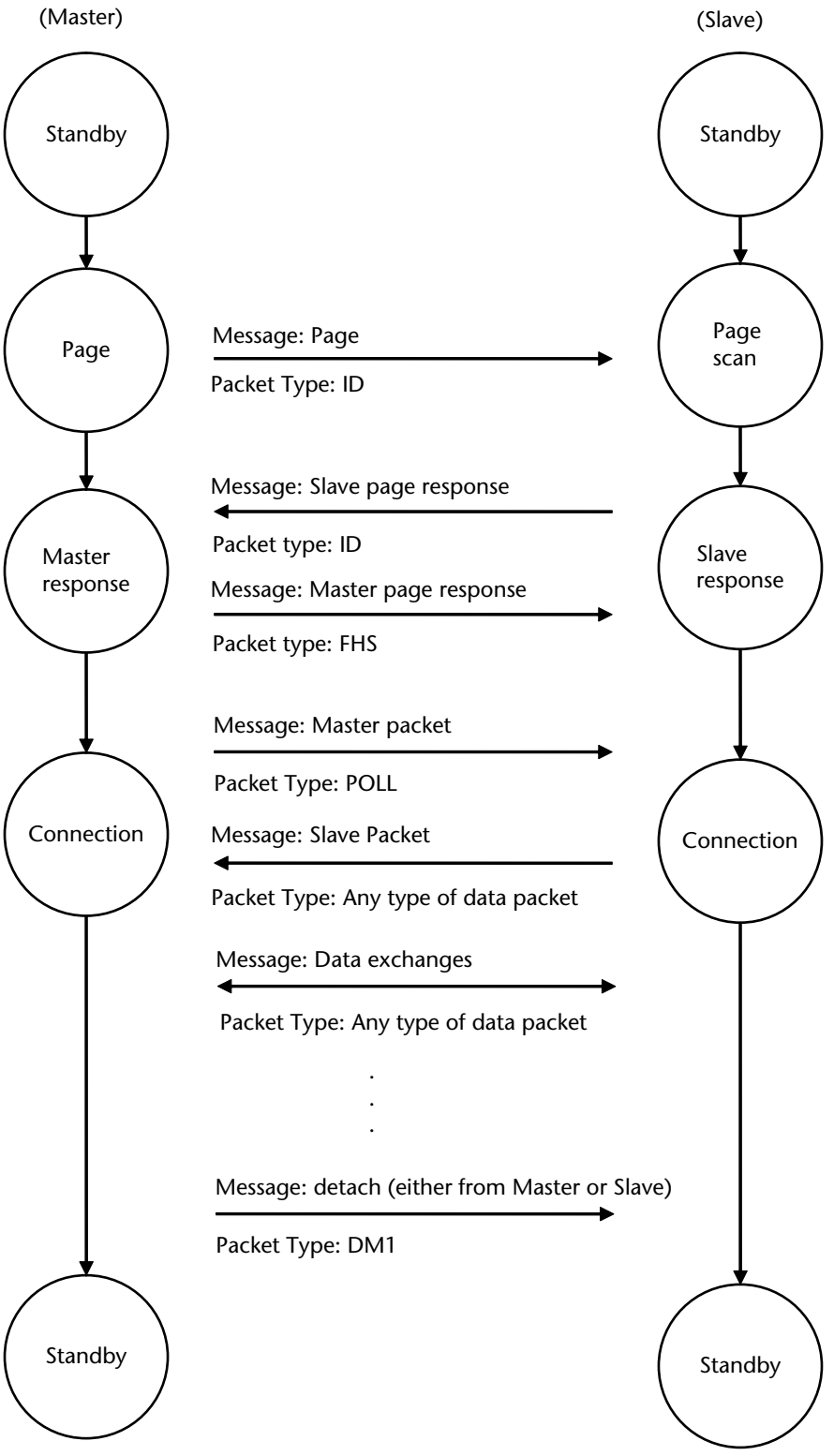


Figure 3.10 Link Controller Messages and States during connection and disconnection.

Sniff Mode

The sniff mode is the most commonly used low power mode. It affects only the ACL logical transport and does not affect the SCO or eSCO logical transports. This means that when this mode is activated, the transmissions on ACL logical transport are reduced, while the transmissions on SCO or eSCO logical transport continue as usual.

In the sniff mode, the device can become absent from the piconet for a certain period of time. So a duty cycle is defined consisting of the duration for which the device will be present and active in the piconet versus the time it will be absent.

A practical use of the sniff mode is when a connection is made between a mobile phone and a mono headset. The two devices can continue to be connected for several days even when there is no active voice call going on. This ensures that as soon as there is an incoming call, time is not wasted in first creating an ACL connection and then creating a SCO connection to route the call. So, to conserve power, the ACL connection between the mobile phone and the headset is put in sniff mode. Most Bluetooth headsets implement the sniff mode.

Another use of the sniff mode is when a Bluetooth keyboard is attached to a PC or laptop. Since the user may not be actively typing all the time, the ACL link is put into sniff mode after an inactivity timer. As soon as the user types a key, the link is brought back into the active mode.

Besides saving power, the sniff mode may also be useful when the device needs to enter into a scatternet scenario. In this scenario, the device may put the ACL link of the current piconet into sniff mode and then become active on another ACL link in the second piconet. So by tuning the sniff parameters it may have periods in which it is absent in first piconet and present in the second piconet and vice versa.

This mode requires a good bit of attention at the link level, especially when there is a connection in sniff mode for a long duration. Any clock drifts on either side while in sniff mode for a long time, could lead to the loss of packets or even loss of the link. Generally controllers wake up time to time in between to synchronize with the Master and go back to sniff mode again to avoid such problems.

3.3.8.2 Park State

A Slave can enter park state if it does not need to participate in the piconet channel but still needs to be synchronized with it. It's a very low power mode with very little activity. In this mode both ACL and SCO or eSCO traffic is stopped. The Slave still remains synchronized to the channel.

The parked Slave wakes up at regular intervals to listen to the channel to check whether there are any messages for it. The Master informs the parked Slaves about any messages for them through broadcast messages.

When a device is put into park state, a different address known as Parked Member Address (PM_ADDR) is assigned to it by the Master during the parking procedure. The Active Member Address (AM_ADDR) that was earlier used by the Slave in active mode is freed up so that it can be re-used by the Master to make a connection with some other device.

Park state is also used when more than seven Slaves need to be connected to a single Master. At a time only seven Slaves can be active. The remaining Slaves are

put into park state. If a parked Slave needs to be unparked then it can be swapped with another active Slave that will be put to park state.

3.4 Link Manager (LM)

The Link Manager performs the functions of link setup and control. The Link Manager Protocol (LMP) is used for communication between the Link Managers of the two devices. The communication messages between the devices are carried over the ACL-C logical link. (As mentioned in the previous section, ACL-C logical link carries control data while the ACL-U logical link carries the user data).

The LMP messages are transmitted using DM1 packets. If HV1 SCO link has been established between the two devices and the length of the payload is less than 9 bytes, then DV packets may also be used. In practice, DV packets are seldom used.

The packets transmitted by LM are referred to as PDUs. Each PDU contains the following fields:

- Transaction ID (TID);
- OpCode;
- Payload.

A transaction is a set of messages that are transmitted to achieve a particular purpose. It may have more than one PDU and all PDUs contain the same TID. The OpCode is used to uniquely identify different type of PDUs. The LMP messages are denoted as LMP_*message_name*. For example, the message to establish a connection is denoted as LMP_host_connection_req. A response PDU to this could be LMP_accepted or LMP_not_accepted.

The functionality supported by LMP includes the following:

- *Connection Control:* This includes procedures for creation and removal of a connection as well as controlling all aspects of a connection.
- *Security:* This includes procedures for authentication, pairing, and encryption.
- *Informational Requests:* This includes various PDUs that the LMs exchange with each other to get the characteristics of the remote device. For example, the LM may find out the version of the remote device using the LMP_version_req PDU. Similarly it may find out the list of features supported by the remote device by sending the LMP_features_req PDU. The remote side would respond using an LMP_features_res PDU.
- *Role switch:* If the Slave device wants to reverse the roles and become the Master, then these procedures are used. A practical example of this would be if a Bluetooth mouse establishes a connection with a laptop. Since the connection establishment is initiated by the mouse, it becomes the Master of the connection. After the connection is established the laptop may prefer to become the Master of the connection so that it can control other peripherals as well. So it can invoke the role switch procedures to become the Master.

- *Modes of operation:* In the baseband section, hold mode, sniff mode, and park state were explained. These different modes of operations are invoked by the link manager. Besides this the link managers of the two devices negotiate the parameters for these modes.
- *Logical transports*: The LM invokes the procedures to create and remove the SCO and eSCO logical transports. The parameters of these transports are also negotiated by the LMs of the two devices. These can also be renegotiated at any time during the connection. For example an eSCO link is established by sending an LMP_eSCO_link_req and removed by sending the LMP_remove_eSCO_link_req.
- *Test Mode:* This includes various PDUs for certification and compliance testing of the Bluetooth radio and baseband.

The connection control and security functionality of link manager is explained in further detail below.

3.4.1 Connection Control

The connection functionality includes all procedures related to creation and removal of a connection and controlling all aspects of the operation of that connection.

Some of the major functions include the following:

- *Connection establishment:* This includes the procedures for establishing a connection with a remote device. If security on the connection is required, then security procedures are also invoked. At the end of this procedure, the device which initiated the connection establishment becomes the Master and the other device becomes the Slave. It's also possible to switch the roles by invoking the role switch procedure during connection establishment. In that scenario, the device which initiated the connection becomes the Slave and the other device becomes the Master.
- *Detaching a connection:* This includes the procedures for detaching a connection. It may be started by either the Master or the Slave.
- *Link supervision:* This feature is used to detect loss of a physical link, for example, when the devices move away from range or one of the devices loses battery power. The LMs of both devices use a supervision timer to detect if the link has been lost. If the link is lost, then the LM reports the link loss to the host. One of the sections earlier explained the POLL and NULL packets. These packets are like a heartbeat between the pair of devices in a piconet. These ensure that the link is still active and that the other device has not gone out of range or has been reset. Hence there is a timer (timeout is negotiated right at the link establishment time) which is fired at both sides for detecting an inactivity on these heartbeat signals. Once there is an inactivity detected for the stipulated timeslots, the link managers on both sides assume that there is something wrong on the link, or the devices have moved out of range. Hence they indicate a link supervision timeout to the upper layers.

- *Enable, disable AFH and update of channel map:* Adaptive Frequency Hopping (AFH) was explained in previous sections. This feature helps in improving the performance in case of interference by removing the channels which have interference from the frequency hopping pattern. The LM of the Master enables or disables this feature and provides the updated channel map to each Slave. The LM of the Slave receives these PDUs and updates it's channel map accordingly. The LM of the Master may also request the Slave for information about the quality of channels. This is an optional feature and if the Slave supports it then it provides a channel map to the Master indicating whether the channels are good, bad, or unknown.
- *Control of the transmit power level:* If the receiver observes that the characteristics of the received signal differ too much from preferred values, it may ask the transmitter to increase or decrease the power level of the transmitting device. This feature is optional and it allows the devices to use the most optimal power levels for communication. As an example, as soon as the devices move away from each other, the receiver may observe degradation of the received signal. So it may request the transmitter to increase the transmit power level. If the devices move closer, the receiver may again request the transmitter to reduce the transmit power level. This will allow more efficient use of battery power.
- *Quality of Service:* This feature is used for bandwidth allocation on ACL logical transport so that the requested amount of bandwidth can be reserved on the ACL logical transport.

An example of air logs of different LMP transactions is shown in Figure 3.11. Some of the points worth observing are:

1. There are two different columns: *Role* and *Initiated By*
   a. Role indicates which device sent the current packet.
   b. Initiated By indicates which device originally initiated this transaction. For example if the transaction was initiated by the Master and the Slave is now responding, then Role will indicate Slave and Initiated By will indicate Master.
2. Frame #1: Connection request initiated by the Master.
3. Frame #9: Connection completed.
4. Frame #2 to #8: Master and Slave exchange information about supported features and version.
5. Frame #14: SET_AFH command sent by Master to Slave to enable Adaptive Frequency Hopping.
6. Frame #292: detach command sent by Master to disconnect the connection.

3.4.2 Security

The security procedures include the following:

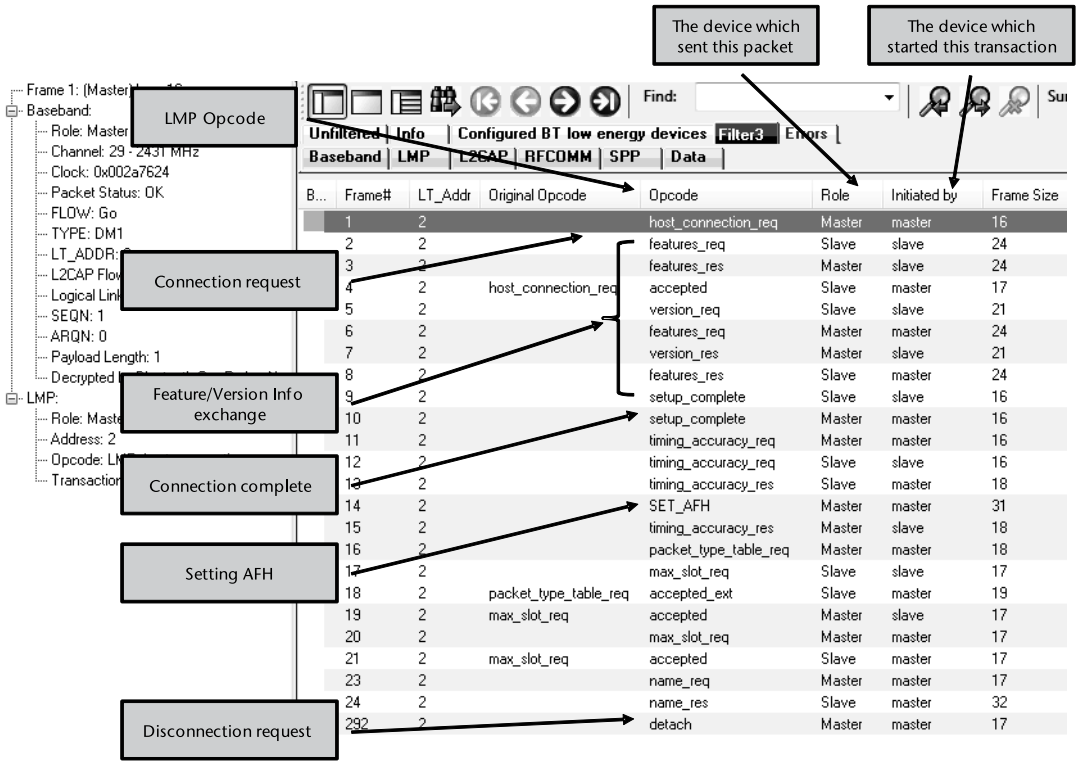


Figure 3.11 Example of LMP transactions.

- Pairing;
- Authentication;
- Encryption;
- Secure Simple Pairing.

3.4.2.1 Pairing

Pairing is the process of associating two devices with each other. When the two devices agree to communicate with each other, they exchange a passkey. This passkey can be considered to be similar to a password that is shared between the two devices. The passkey is also referred to as the Bluetooth PIN and is generally entered on the UI. For example when connecting a mobile phone to a laptop, the user will have to enter identical PIN codes on both the laptop and the mobile phone.

Pairing can be done by using older legacy pairing procedures or by *secure simple pairing* procedures. (Secure simple pairing was introduced in version 2.1 + EDR of the Bluetooth specification).

The Bluetooth PIN that is entered on the UI is used along with a random number and BD_ADDR to create a link key. This link key is used for authentication between the two devices for all subsequent connections. This key is stored in the devices so that when the next time the devices are connected, the user does not need to enter the PIN again.

3.4.2.2 Authentication

Authentication is the process of verifying *who* is at the other end of the link. The authentication process starts when the two devices initiate a connection establishment. It is based on a challenge-response scheme. The verifier sends a challenge to the other device that contains a random number (the challenge). The other device calculates a response that is the function of the challenge, its own BD_ADDR and a secret key. The response is sent back to the verifier that checks whether it is correct or not.

The success of the authentication procedure requires that the two devices share a secret key. This key was generated during the pairing process. If they do not share a secret key, then the pairing procedure is initiated.

3.4.2.3 Encryption

This is an optional procedure and the Master and Slave must agree whether to use encryption or not. To use encryption, it's mandatory to perform authentication.

If encryption is enabled, then all data exchanged on the link is encrypted using an encryption key. The encryption key can be from 8-bits to 128-bits.

3.4.2.4 Secure Simple Pairing

Secure simple pairing was introduced in version 2.1 + EDR of the Bluetooth specification to both simplify the pairing mechanism and improve security. This is explained in further detail later in this chapter.

3.5 Host Controller Interface (HCI)

One of the strengths of Bluetooth specification is that it provides a uniform method for accessing the controller's capabilities. This has several advantages. First the development of the software for controller and host can go on independently. Secondly a host from one vendor can work easily with a controller from another vendor. For example a Bluetooth dongle from any vendor can be plugged into a PC to use Bluetooth functionality.

The Host Controller Interface (HCI) provides this uniform method for accessing the controller's capabilities. The host interfaces with the controller using this layer. It is optional and is required only in implementations where the software for controller and host run on different processors. It may be skipped in implementations where the software for controller and host run on the same processor.

The communication over the HCI interface happens in form of packets. The host sends HCI command packets to the controller and is asynchronously notified by the controller using HCI events.

The commands and events for LE controllers are also exchanged over the HCI interface. The LE controllers use a reduced set of HCI commands. Some commands and events are re-used for multiple controller types.

There are four possible types of controllers:

1. *BR/EDR Controller*: Supports only BR/EDR functionality.
2. *BR/EDR/LE Controller:* Supports both BR/EDR and LE functionality.
3. *LE Controller:* Supports only LE functionality.
4. *AMP Controller:* Supports functionality of AMP Protocol Adaptation Layer (PAL) (BT 3.0 + HS).

3.5.1 HCI Packet Types

There are four types of packets that can be sent on the HCI Interface.

1. HCI Command Packet.
2. HCI Asynchronous (ACL) Data Packet.
3. HCI Synchronous (SCO/eSCO) Data Packet.
4. HCI Event Packet.

These are shown in Figure 3.12.

3.5.1.1 HCI Command Packet

The HCI Command Packet is used by the host to send commands to the controller. Each command is assigned a 2 byte unique OpCode. The OpCode is further divided into two fields:

1. *OpCode Group Field (OGF)* (upper 6 bits): This field is used to group similar OpCodes together.
2. *OpCode Command Field (OCF)* (lower 10 bits): This field is used to identify a particular command in the OpCode group.

The HCI command packets are denoted as HCI_*xxx* where *xxx* is the command that is given by the host to the controller. For example the command to reset the controller is denoted as HCI_Reset.

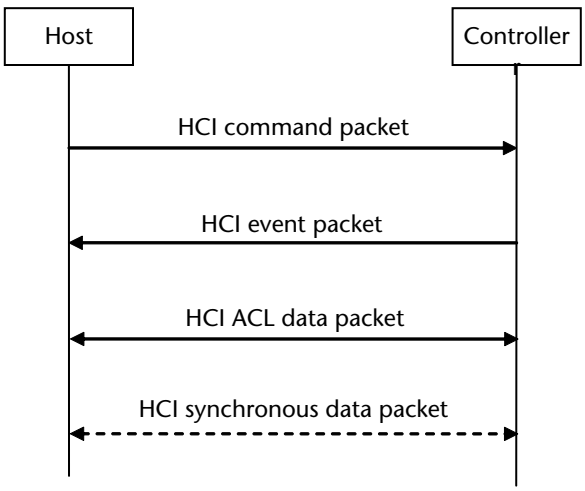


Figure 3.12 HCI Packet Types.

The format of HCI command packet is shown in Figure 3.13. It consists of a 16-bit OpCode followed by an 8-bit *parameter total length* field. The parameter total length field specifies the total length of all parameters that are contained in the remaining packet measured in octets. (An octet denotes an 8-bit value). This is followed by the command parameters.

3.5.1.2 HCI Event Packet

The HCI Event Packet is used by the controller to notify the host when an event occurs. This may be in response to an HCI command that was sent earlier (For example a command to create a connection), or due to any other event (For example loss of connection or incoming connection request). Errors are also indicated using HCI Event Packets.

The format of HCI event packet is shown in Figure 3.14. It consists of an 8-bit Event Code followed by an 8-bit parameter total length field. The parameter total length field specifies the total length of all parameters that are contained in the remaining packet (measured in octets). This is followed by the event parameters.

3.5.1.3 HCI ACL Data Packet

The HCI ACL Data Packet is used to exchange data between the host and the controller. ACL Data packets can only be exchanged after a connection has been established.

The format of HCI ACL data packet is shows in Figure 3.15. It consists of a 12-bit Handle followed by 2-bit Packet Boundary (PB) and 2-bit Broadcast (BC) flags. This is followed by 16-bit Data Total Length field which specifies the length of the following data in octets. This is followed by the data.

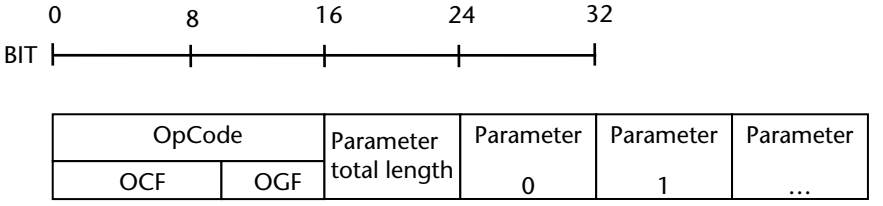


Figure 3.13 HCI Command Packet Format.

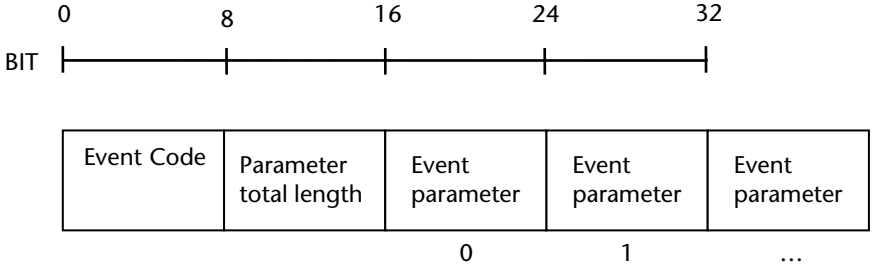


Figure 3.14 HCI event packet format.

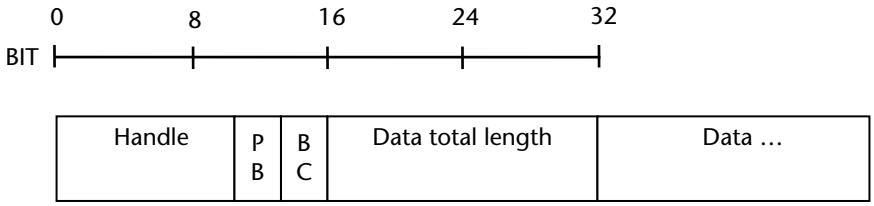


Figure 3.15 HCI ACL data packet format.

The Handle specifies the Connection Handle on which data is to be sent. Each connection is specified by a unique connection handle.

The *packet boundary flag* is useful when a higher layer protocol packet is fragmented into multiple ACL data packets. It indicates whether this packet is the first packet in the fragment or a continuing packet. This field also indicates the flushable characteristics of the packet—whether it is automatically flushable or not. Automatically flushable packets are automatically flushed based on a timer if they cannot be transmitted by the time the timer expires.

The *broadcast flag* indicates whether this is a point-to-point packet or a broadcast packet.

3.5.1.4 HCI Synchronous Data Packet

The HCI Synchronous Data Packet is used to exchange synchronous data between the host and the controller.

The format of HCI synchronous data packet is shown in Figure 3.16. It consists of a 12-bit Connection Handle followed by 2-bit Packet Status (PS) flag and 2-bit reserved (RES) field. This is followed by 16-bit Data Total Length field which specifies the length of the following data in octets. This is followed by the data.

The Connection Handle specifies the Connection Handle on which data is to be sent. Each SCO or eSCO connection is specified by a unique connection handle.

The Packet Status flag gives an indication whether the packet was correctly received or not.

3.5.2 HCI Commands and Events

The HCI commands are grouped into several categories. Some examples of the HCI commands in various groups are provided below.

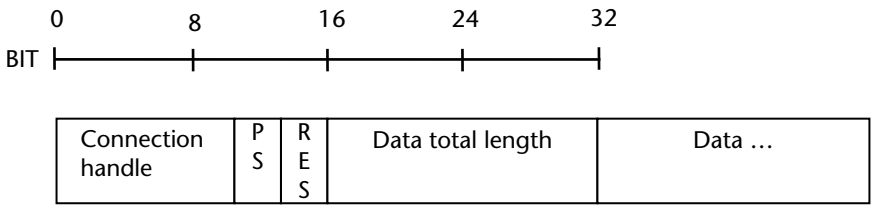


Figure 3.16 HCI synchronous data packet format.

- *Link Control Commands:* These allow a controller to control connection to other controllers.
  - HCI_Inquiry—Discover devices in the vicinity.
  - HCI_Inquiry_Cancel—Cancel Inquiry.
  - HCI_Create_Connection—Create a connection to remote device.
  - HCI_Disconnect—Terminate an existing connection.
  - HCI_Remote_Name_Request—Obtain user friendly name of another controller.
- *Link Policy Commands:* These are used to control how Bluetooth piconets and scatternets are established and maintained.
  - HCI_Hold_Mode—Put the connection in hold mode.
  - HCI_Sniff_Mode—Put the connection in sniff mode.
  - HCI_Switch_Role—Switch the role from Master to Slave and vice versa.
- *Controller and Baseband Commands:* These provide access to various capabilities of Bluetooth hardware.
  - HCI_Set_Event_Mask—Control which events can be generated by the controller to be sent to the host.
  - HCI_Reset—Reset the controller.
  - HCI_Write_Local_Name—Modify the user friendly name of the local device.
  - HCI_Write_Class_of_Device—Write the Class of Device parameter.
- *Informational Parameters:* These provide information about the capabilities of the controller.
  - HCI_Read_Local_Version_Information—Version, Manufacturer name.
  - HCI_Read_BD_ADDR—Reads the BD_ADDR of the controller.
- *Status Parameters:* These provide information about the current state of the controller.
  - HCI_Read_Link_Quality_Information about the link quality.
- *LE Controller Commands:* These are used for LE controllers.
  - HCI_LE_Set_Event_Mask—Controls which LE Events are generated by the controller to be sent to host.
  - HCI_LE_Read_Buffer_Size—Read the maximum size of data packets that can be sent to controller and the total number of buffers.
  - HCI_LE_Create_Connection—Create an LE connection.

Some examples of HCI events are provided below.

- *Inquiry Complete:* Is used to indicate completion of inquiry.
- *Inquiry Result:* Is used to provide information about remote devices found during inquiry.
- *Connection Complete:* Indicates that a connection has been established.
- *Disconnection Complete*: Indicates that a connection has been terminated.

- *Hardware Error:* Indicates some hardware failure.
- *LE Meta Event:* Is used to encapsulate all LE events. A sub-event code indicates the exact event that has occurred. It could be one of the following:
  - LE Connection Complete—Indicates that an LE connection has been established.
  - LE Advertising Report—Indicates that an advertising report has been received.
  - LE Connection Update Complete Event—Indicates that the process to update the connection parameters has been completed.

The following two HCI events are used quite frequently:

- *Command Complete:* This event is used by the controller for most commands to transmit the return status of the command as well as the return parameters associated with the command.
- *Command Status:* This event is used by the controller to indicate that it has received the command and has started performing the task for the command though the task may not have completed yet. This event is needed to provide a mechanism of asynchronous operation so that the host can continue doing other operations while the controller performs the tasks requested by the host. The host is notified by another event when the task is completed by the controller.

An example of the HCI commands and events exchanged during inquiry is shown in Figure 3.17.

3.5.3 Buffers

The controller has buffers to receive the commands and ACL/Synchronous data sent by the host.

- *Command Buffers:* The BR/EDR/LE controller uses shared buffers for the commands.
- *Data Buffers:* The controller may either have shared or separate data buffers for BR/EDR and LE.

Whether the controller has separate or shared data buffers for LE can be determined using the HCI_Read_Buffer_Size command. This will be explained in a later section.

3.5.4 HCI Flow Control

In general the controller has much less buffers and processing power as compared to the host. This means that if the host sends a few packets to the controller to process in fast succession, then it is possible that the buffers of the controller get full.

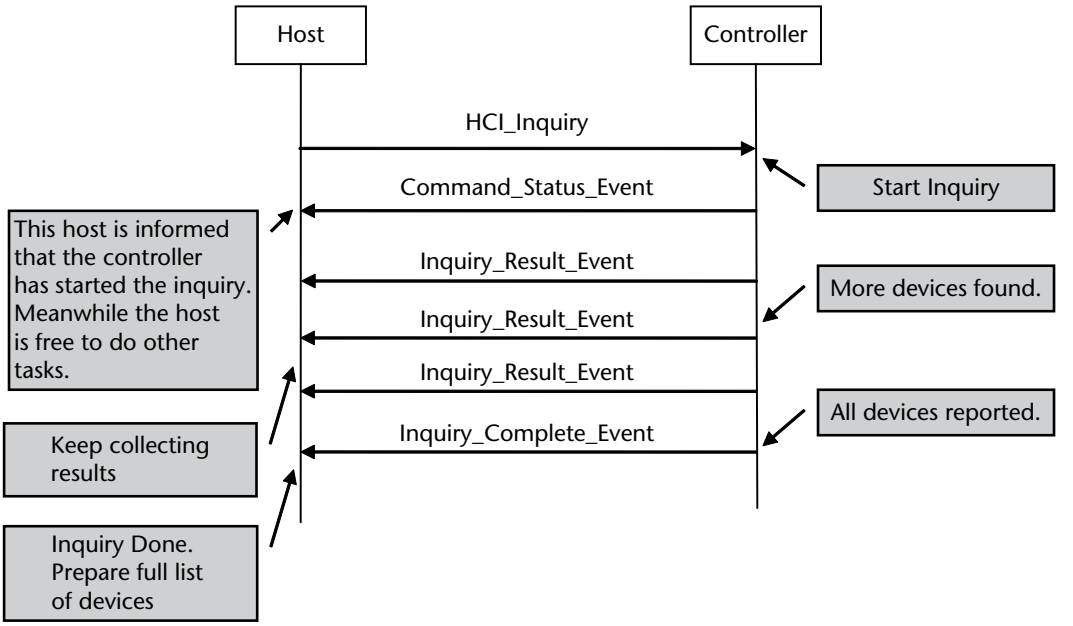


Figure 3.17 HCI commands and events for inquiry.

Flow control mechanism is needed in such a scenario to ensure that the host does not send any packets further till the time the previous ones are processed.

The HCI interface provides separate flow control mechanisms for the following:

1. *Host to Controller Data Flow Control:* ACL Data Packets from host to controller.
2. *Controller to Host Data Flow Control:* ACL Data Packets from controller to host.
3. *Command Flow Control:* HCI Command Packets from host to controller.

Out of these, Controller to Host Data Flow Control is not much widely used since typically the host has more resources than the controller. So most of the time, the controller can send data to the host without any flow control.

3.5.4.1 Host to Controller Data Flow Control

The Host to Controller Data flow control is used to regulate the flow of data from the host to the controller. This is needed because the controller may have much lesser buffers as compared to the host. Besides this the controller also needs time to process (transmit/re-transmit) those packets. During that time, the flow control ensures that the host sends only as many packets as the number of available buffers in the controller.

There are two mechanisms:

1. Packet Based Flow Control.
2. Data-Block-Based Flow Control.

The Packet Based Flow Control is used quite widely and this is explained in detail in this section. It is shown in Figure 3.18.

During initialization the host sends the Read Buffer Size command to get information about the buffers in the controller. The response to the Read Buffer Size contains the following:

1. Number of ACL buffers.
2. Size of each ACL buffer.
3. Number of SCO buffers.
4. Size of each SCO buffer.

If the controller supports LE, then it is possible for the controller to either have separate LE buffers or share the same buffers. The host can send the LE Read Buffer Size command to find this out. If the response to this indicates zero as the length of the buffers, then it means that the same buffers are shared between BR/EDR and LE. Otherwise the response would indicate the number of LE buffers present in the controller and the size of those buffers.

Once the host has information on the number of ACL buffers, it knows that at any given time, it can have a maximum of that many packets outstanding (to be processed) on the controller side. For example if the Number of ACL buffers was four, then the host can have a maximum of four packets outstanding on the controller side at any given time.

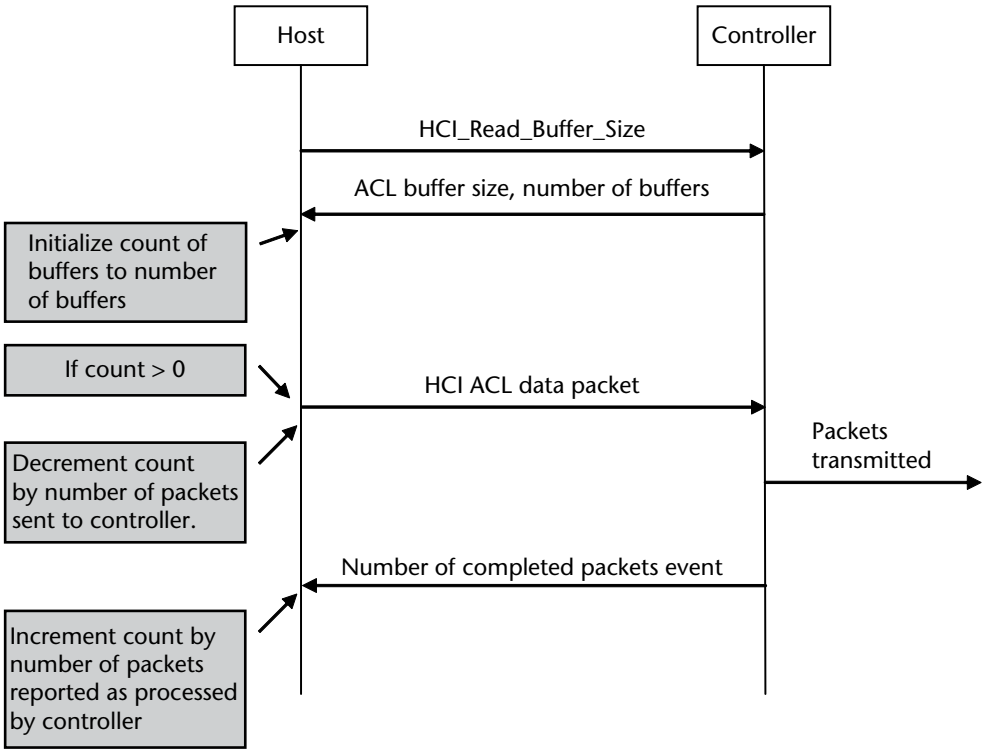


Figure 3.18 Packet based Host to Controller Data Flow Control.

The host maintains a count of the number of ACL buffers available in the controller. It initializes this value to the number of ACL buffers it received in the response to Read Buffer Size command. After that every time it sends a packet to the controller, it decreases this count by one.

Once the controller has completed processing one or more packets, it sends that count of processed packets in the Number of Completed Packets event. The host knows that some additional buffers have been freed up on the controller side and it increments its count by the number of packets reported in the Number of Completed Packets event.

3.5.4.2 Command Flow Control

The flow control used for commands is very simple. The command complete and command status events that are sent by the controller to the host have a field which indicates how many more commands can be sent to the controller. The host can send up to that many commands to the controller before waiting for the next command status or command complete event.

In general, if the controller is not able to complete a command right away, it sends a command status event. Later on when the command is completed it either sends a command complete event or another relevant event depending on the command that was sent. For example if the host requests the controller to start an inquiry, the controller starts the inquiry and returns a command status event. This indicates to the host that the controller has started processing of the inquiry command. Once the inquiry is complete, the controller sends an inquiry complete event to indicate that the inquiry command has been completed. This was shown in Figure 3.17.

3.5.5 Connection Handle

The HCI interface assigns a Connection Handle when a new logical link is created. The value of this connection handle is returned in the Connection_Complete_Event. The host uses this connection handle to manage this connection, send data and finally disconnect the connection.

3.5.6 HCI Transport Layer

The Host Controller Interface defines 4 transport layers that can be used:

1. UART Transport Layer.
2. USB Transport Layer.
3. Secure Digital (SD) Transport Layer.
4. Three-Wire UART Transport Layer.

The UART Transport Layer in described in detail here. This interface is commonly used in systems where the Host and Controller are on the same PCB (Printed Circuit Board). Some examples are Smartphones, Tablets and Printers.

Each packet that is exchanged between the Host and the Controller is prefixed with a packet indicator immediately before the HCI Packet. This allows the differentiation of various packets.

The following Packet Indicators are used on the HCI UART Interface:

1. HCI Command Packet: Packet Indicator 0x01.
2. HCI Asynchronous (ACL) Data Packet: Packet Indicator 0x02.
3. HCI Synchronous (SCO/eSCO) Data Packet: Packet Indicator 0x03.
4. HCI Event Packet: Packet Indicator 0x04.

Figure 3.19 indicates the direction in which these packets are sent along with the packet indicators. (The arrows indicate the direction in which packets are sent.)

In some implementations (e.g., some Smartphones), the HCI Synchronous Data packets are not sent on the UART Transport Layer. Rather these packets are directly sent by the host's application processor or modem to the Bluetooth chip on a different interface. For example a PCM/I2S interface may be used to send voice packets between the Application Processor and Bluetooth Chip. This is shown in Figure 3.20.

3.5.6.1 Decoding HCI Packets

Figure 3.21 shows an example of how to decode the packets on the UART transport layer. This is quite useful when debugging the HCI transactions happening on the UART. Similar method can be used to also debug packets on other transport layers.

In general the UART Transport Layer assumes that the line is free from line errors. (Why? - Since they are closely located on the same PCB so it's assumed that there will not be any errors). In case there is a synchronization loss in the communication from the Host to the Controller, the Controller sends a Hardware Error Event to the Host. Once the Host is aware of the synchronization loss, it has to terminate any pending Bluetooth activity and then reset the Controller to regain

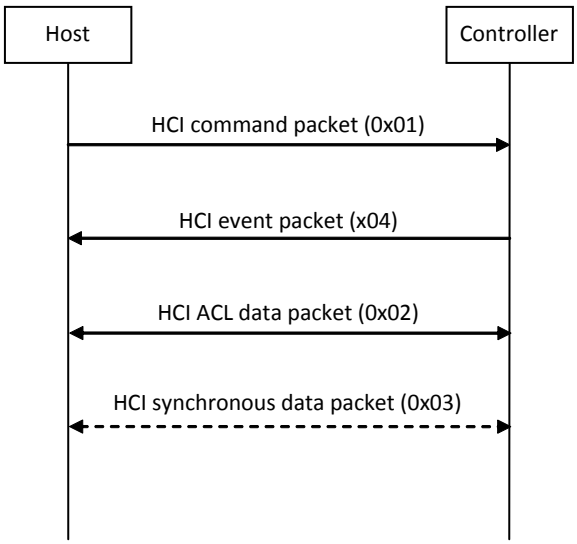


Figure 3.19 HCI UART Transport Layer.

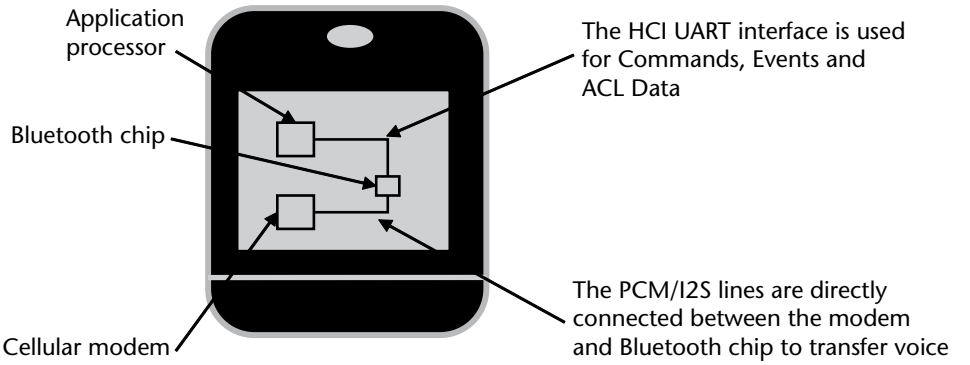


Figure 3.20 Typical HCI UART connections in a smartphone.

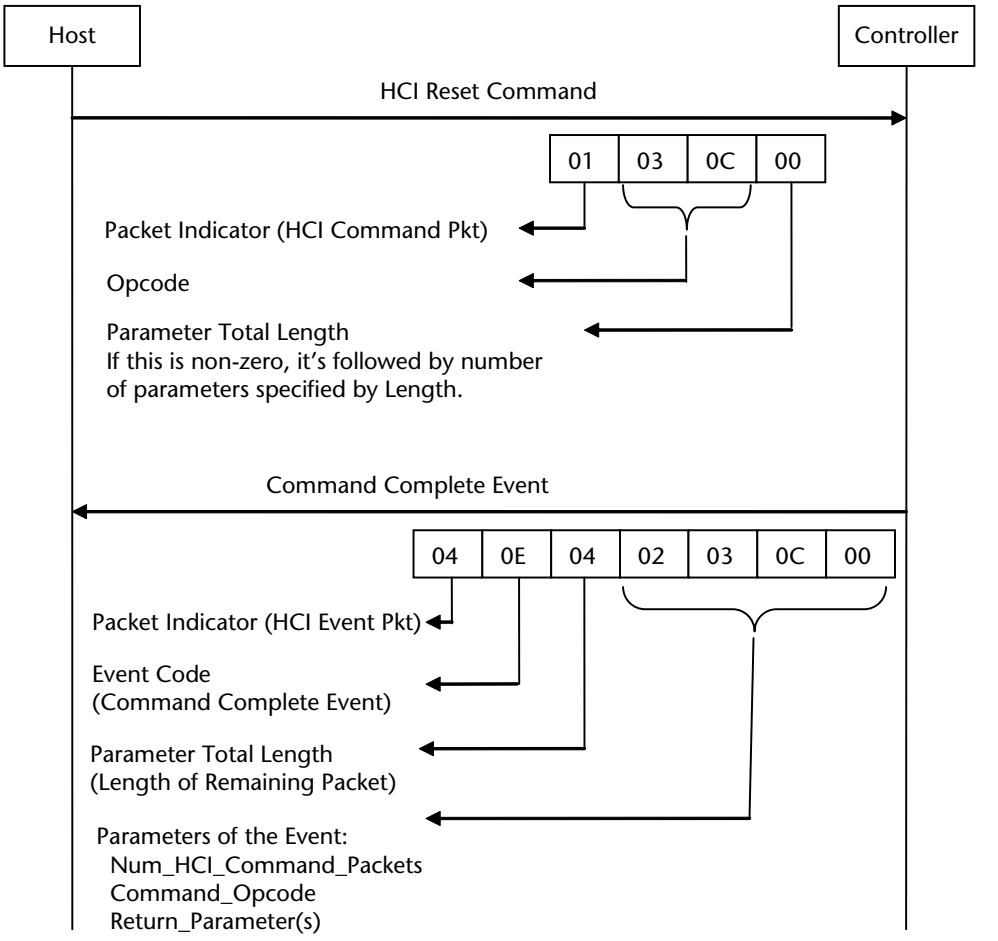


Figure 3.21 How to decode packets sent on HCI UART Transport Layer.

communications. This is done by the host by sending the HCI_Reset command to the controller.

In order to avoid synchronization losses, the Host and Controller use Hardware Flow Control. For details on this as well as the details on the settings of the UART you may refer to the Bluetooth Core specification (Ref [1]).

3.6 Security—Secure Simple Pairing (SSP)

Secure Simple Pairing was introduced in Bluetooth spec 2.1 + EDR. The main goal was to simplify the pairing procedure from the user perspective. It also introduced improved level of security as compared to previous versions.

SSP increased the security level as compared to previous version of the Bluetooth specification. Till Bluetooth spec 2.0 + EDR, only 4-digit numeric pin keys were supported. On top of that several devices used frequently used pin keys like 0000, 8888, or 1234. So hacking those codes was comparatively easier. SSP introduced 16 alphanumeric pin which makes it more difficult to hack the codes.

SSP has two security goals:

- Passive Eavesdropping Protection.
- Man-In-The-Middle (MITM) Attack Protection.

3.6.1 Passive Eavesdropping Protection

Passive Eavesdropping protection is provided by two mechanisms:

- *Strong Link Key:* The strength of a link key is dependent on the amount of randomness (entropy) in its generation. In previous versions of Bluetooth, the only source of this entropy was the 4-digit numeric PIN key. This was relatively easy to break in short time. In comparison SSP uses 16-bit alphanumeric PIN key which provides for much higher entropy.
- *Strong Encryption Algorithm:* SSP uses Elliptic Curve Diffie Hellman (ECDH) public key cryptography. This provides a high degree of strength against passive eavesdropping attacks. It's also computationally less complex than standard Diffie Hellman (DH76) cryptography and suitable for Bluetooth controllers which may have limited computational power.

3.6.2 Man-in-the-Middle (MITM) Attack Protection

An MITM attack occurs when a rouge device attacks by sitting in the middle of two devices that want to connect and relays messages between them. The two devices believe that they are directly talking to each other without knowing that all their messages are being intercepted and relayed by a third device which is setting between them. This is also known as active eavesdropping.

Let's say devices A and B want to make a connection and M is an attacking device (as shown in Figure 3.22).

M receives all information from A and relays it to B and vice versa. So, A and B have an illusion that they are directly connected. They are not aware of the existence of M between them. Since M is relaying the information between the two

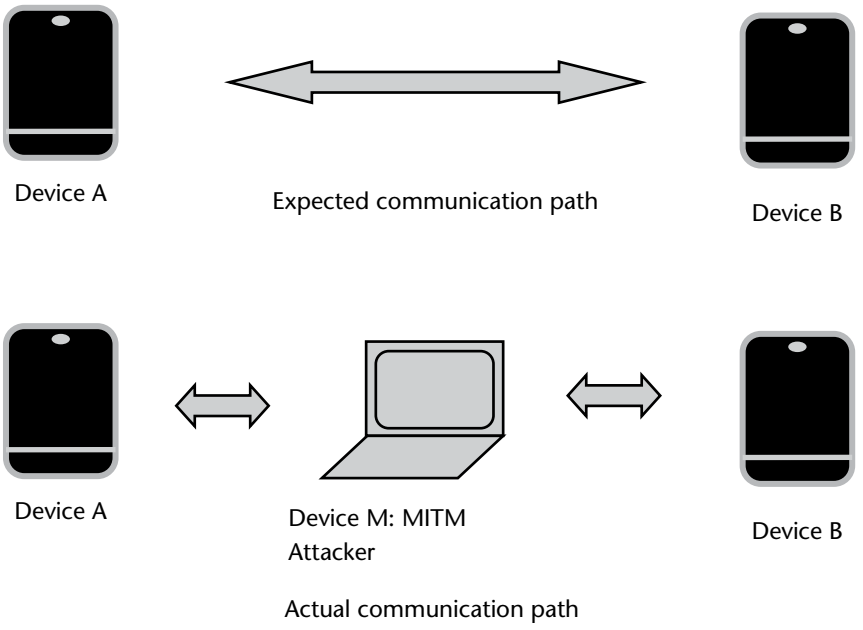


Figure 3.22 Example of MITM attack.

devices, it can interpret all this information and misuse it. Besides this, M can also attack by inducing rogue information between the two devices.

Since A and B can only communicate via M, this type of attack can get detected only if the connection to M is lost. In that case, A and B will not be able to communicate any further and user can detect an MITM attack.

To prevent MITM attacks, SSP provides two mechanisms: Numeric Comparison and Passkey entry. These are described in further detail below.

3.6.3 Association Models

SSP provides four association models based on the I/O capabilities of the two devices. These are as follows:

- Numeric Comparison;
- Just Works;
- Out of Band;
- Passkey Entry.

3.6.3.1 Numeric Comparison

This is used in scenarios where both devices are capable of displaying a six digit number and both are capable of having the user enter a binary "yes" or "no" response. This method displays a 6-bit numeric code on both the devices. The user is then asked whether the number is the same on both the devices. If the user enters yes on both devices, the pairing is successful.

This method has the following advantages:

1. It provides additional confirmation that the correct devices are being paired. This is especially true in cases where the device names are not unique and it's quite hard to remember and identify the device using BD_ADDR
2. It helps to provide protection against MITM attacks.
3. It is also applicable in scenarios where a device may not have a full-fledged keyboard (for entering the PIN) but has only a display. For this method a binary Yes/No input is sufficient.

3.6.3.2 Just Works

As the name implies, this method just works without any user intervention. It's designed for scenarios where the device does not have capability to enter 6 decimal digits nor it has capability to display 6 decimal digits.

It's quite commonly used in pairing mobile phones with mono headsets since mono headsets do not have display and pin entry capabilities. Internally this method still uses the Numeric Comparison though the numbers are not displayed to the user.

It provides protection against passive eavesdropping but does not provide MITM protection. So the security level is still higher than older 2.0 devices but not as good as Numeric Comparison.

3.6.3.3 Out-of-Band (OOB)

Out Of Band Pairing uses an external means for discovering the devices and exchanging pairing information. It's expected that the Out Of Band channel provides protection against MITM attacks to ensure that the security is not compromised.

Typically this could be NFC (Near Field Communication) where the user may touch the two devices. An option would be given to pair the two devices and if the user confirms, the pairing would be successful.

3.6.3.4 Passkey Entry

The passkey mechanism is used when one device has input capability but does not have display capabilities and the second device has display capabilities. One example of such scenario is pairing between keyboard and PC. The user is shown a six digit number on the device which has display capabilities and is then asked to enter this number from the device which has input capabilities. Pairing is successful if the value entered by the user is correct.

3.7 Practical Scenarios

This section describes how the HCI, Baseband Controller and Link Manager interact to implement the following practical scenarios:

- Inquiry;
- Connection Establishment;
- Disconnection.

3.7.1 Inquiry

Inquiry is the procedure to discover other Bluetooth devices in the vicinity. It is also known as scanning or discovering.

Prior to performing an inquiry, it is important that the device that is to be discovered be set in discoverable mode. This includes sending the HCI_Write_Scan_Enable command with *Inquiry Scan* parameter set to *Enabled*. Besides this, HCI_Write_Inquiry_Scan_Activity can also be used to configure additional parameters. Once this is done the device is said to be in inquiry scan mode (or discoverable mode).

The inquiry procedure is started by the host of the first device (which is supposed to discover other devices) by sending an HCI_Inquiry command to the controller. On receipt of this command, the controller initiates an inquiry. The devices in the vicinity respond with inquiry responses.

The controller collects the inquiry responses and provides the results of inquiry to the host in Inquiry_Result events. These events contain the BD_ADDR, Class_Of_Device and Clock_Offset. Multiple devices may be reported in a single Inquiry_Result event. Once the inquiry is complete, an Inquiry_Complete event is sent by the controller to the host (on the device that initiated the inquiry).

This is illustrated in Figure 3.23. (Some of the events are omitted to aid simplicity).

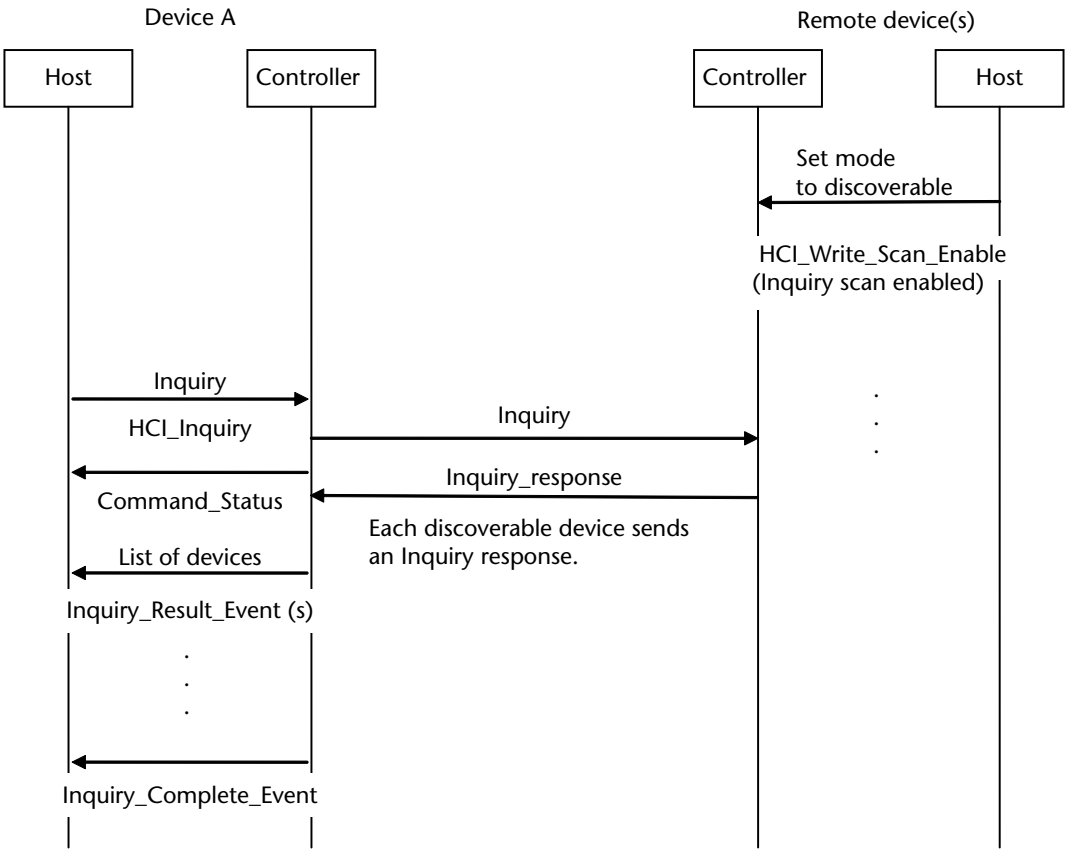


Figure 3.23 Inquiry procedure.

3.7.1.1 Periodic Inquiry

If the inquiry procedure is to be repeated periodically, then HCI_Periodic_Inquiry_Mode command can be used. This command takes the maximum and minimum period between consecutive inquiries as parameters. Once this command is given by the host, the controller performs an automatic inquiry periodically. This can be stopped by the host by sending the HCI_Exit_Periodic_Inquiry_Mode command.

A practical use of this is when a device wants to be regularly alerted of any devices which are coming in or going out of the vicinity. To achieve this periodic inquiry can be used. The time range between two consecutive inquiries is provided as a parameter. The new device list can be compared with the previous device list. If a device was not present in the previous list but is present in the new list then it has entered the Bluetooth vicinity recently. Any specific action like finding details about the device, connecting to it, sending a message or file to it can then be initiated.

3.7.1.2 Extended Inquiry Response (EIR)

The EIR enhancement was added in Bluetooth 2.1 + EDR version of the specification. If a device supports EIR it can provide some additional data while responding to the inquiry. This extended data may contain the name of the device, supported services, Received Signal Strength Indicator (RSSI) etc.

This is more efficient than a normal inquiry response since the inquiring device gets all the information in one go instead of first doing an inquiry, then a get name and finally a service search. This leads to an overall faster connection setup procedure.

3.7.2 Connection Establishment

The procedure to connect to a remote device is known as paging or connecting procedure. In the text below, the device that initiates the connection is referred to as the initiator and the device to which it connects is referred to as the target.

Prior to connecting, the target device (that is to be connected to) is set to Connectable mode. This is done by sending the HCI_Write_Scan_Enable command. This command takes Page_Scan as one of the parameters. The Page_Scan parameter is set to Enabled to make the device connectable. Besides this, HCI_Write_Page_Scan_Activity can be used to configure additional parameters. Once this is done the device is in page scan mode (or Connectable Mode).

Connection procedure is started by the host of the initiator by sending an HCI_Create_Connection command to the controller. The parameters of this command include BD_ADDR, Packet Types, Page_Scan_Repetition_Mode, Clock_Offset and Allow_Role_Switch.

On receipt of this command the controller initiates a connection request using the link manager command LMP_host_connection_req.

Figure 3.24 depicts a very simplified view of the connection establishment procedure. The optional parts like Feature Exchange, Authentication, Encryption and some of the HCI events are omitted to aid simplicity.

The initiator device becomes the Master after successful connection establishment.

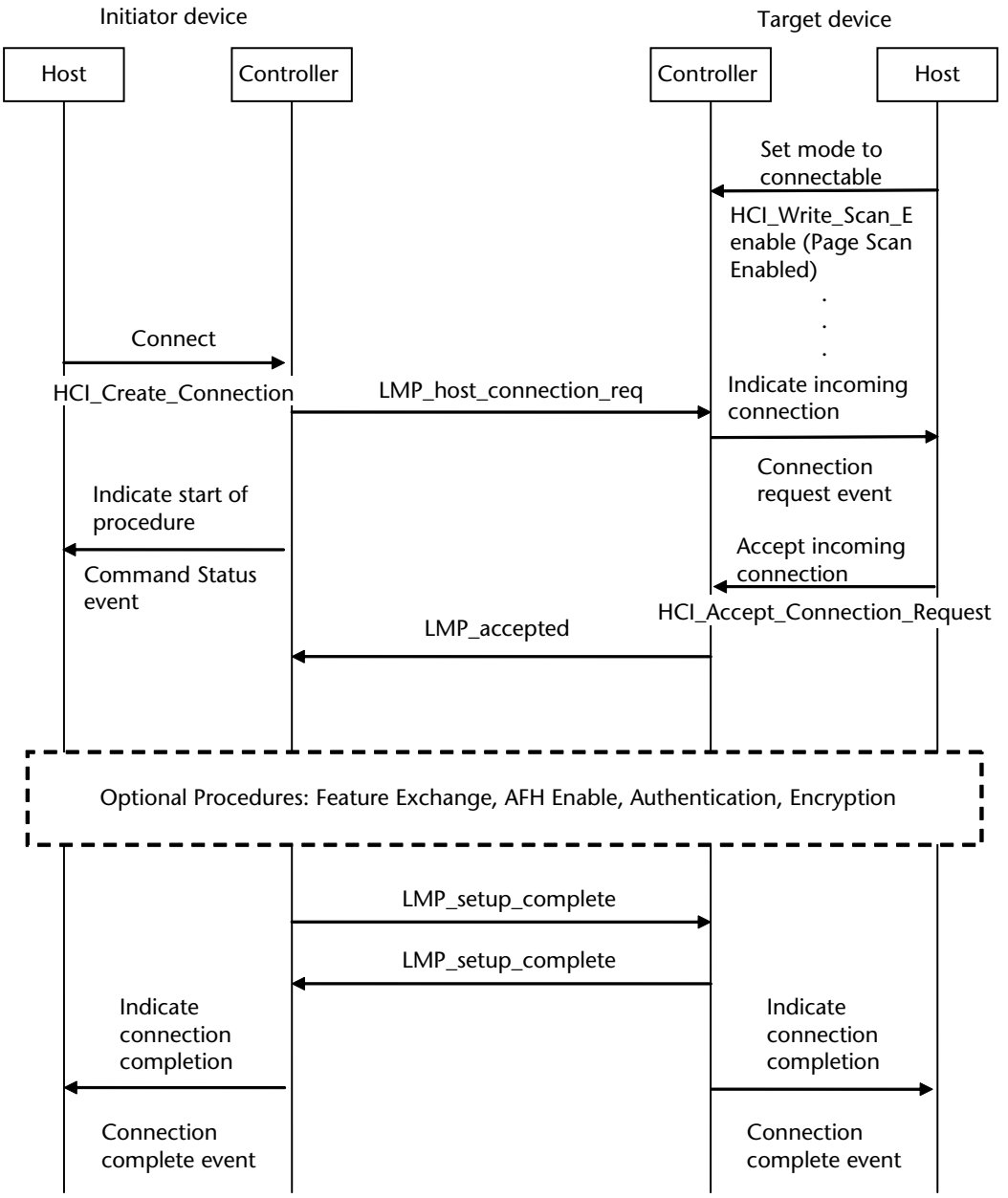


Figure 3.24 Simplified connection establishment procedure.

Figure 3.25 illustrates the disconnection procedure. Either of the devices may decide that the connection is no longer needed and initiate this procedure.

3.8 Summary

This chapter explained the lower layers of the Bluetooth Protocol stack. These include the stack layers up to the HCI interface: Bluetooth Radio, Baseband Controller, and Link Manager. These layers are typically implemented in a controller.

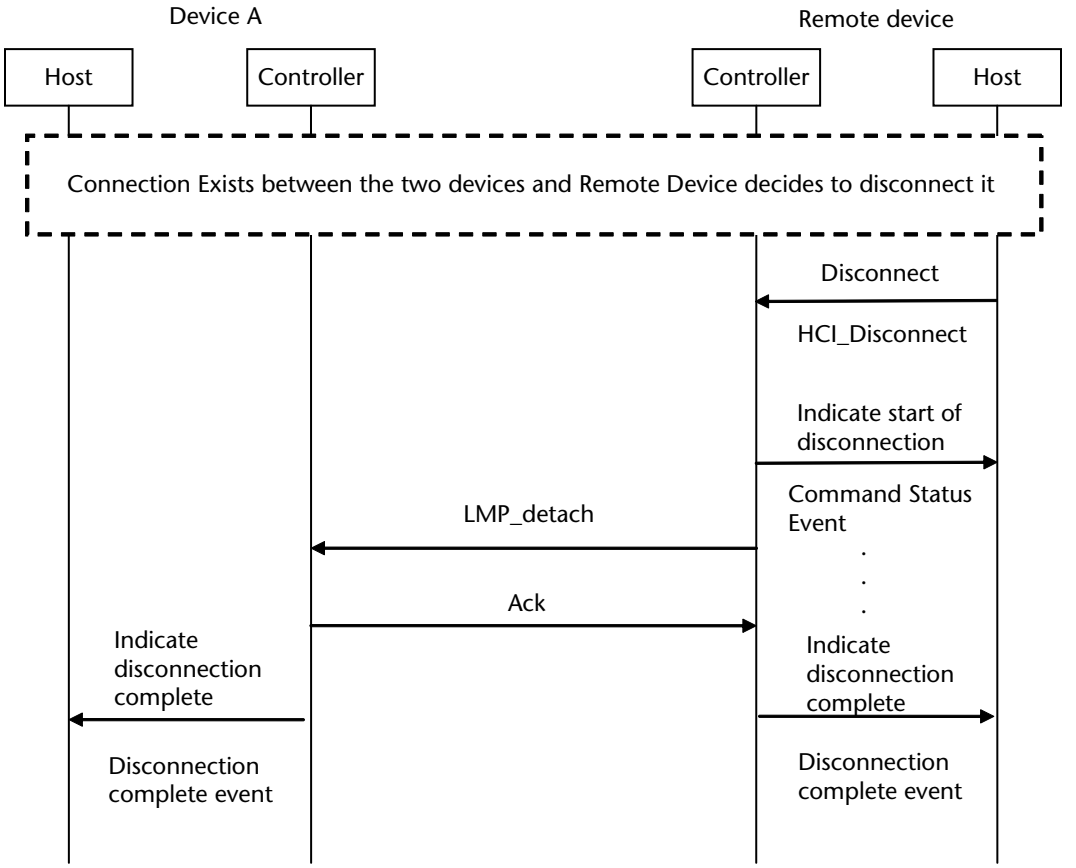


Figure 3.25 Disconnection procedure.

The Bluetooth Radio is responsible for transmitting and receiving the packets to and from the air interface. It uses frequency hopping across 79 channels in the ISM band to combat interference. The Baseband Controller is responsible for carrying out procedures like inquiry, connection, formation of piconet and scatternet, connection states, and low power modes. The Link Manager provides the functionality of link setup and control, security, Master-Slave role switch, etc.

The HCI interface provides a standard mechanism for interfacing Bluetooth upper layers with the controller.

The next chapter will be the last in the series of chapters explaining the Bluetooth architecture. It will cover the Bluetooth upper layers.

Reference

[1] Bluetooth Core Specification 4.0 http://www.bluetooth.org.