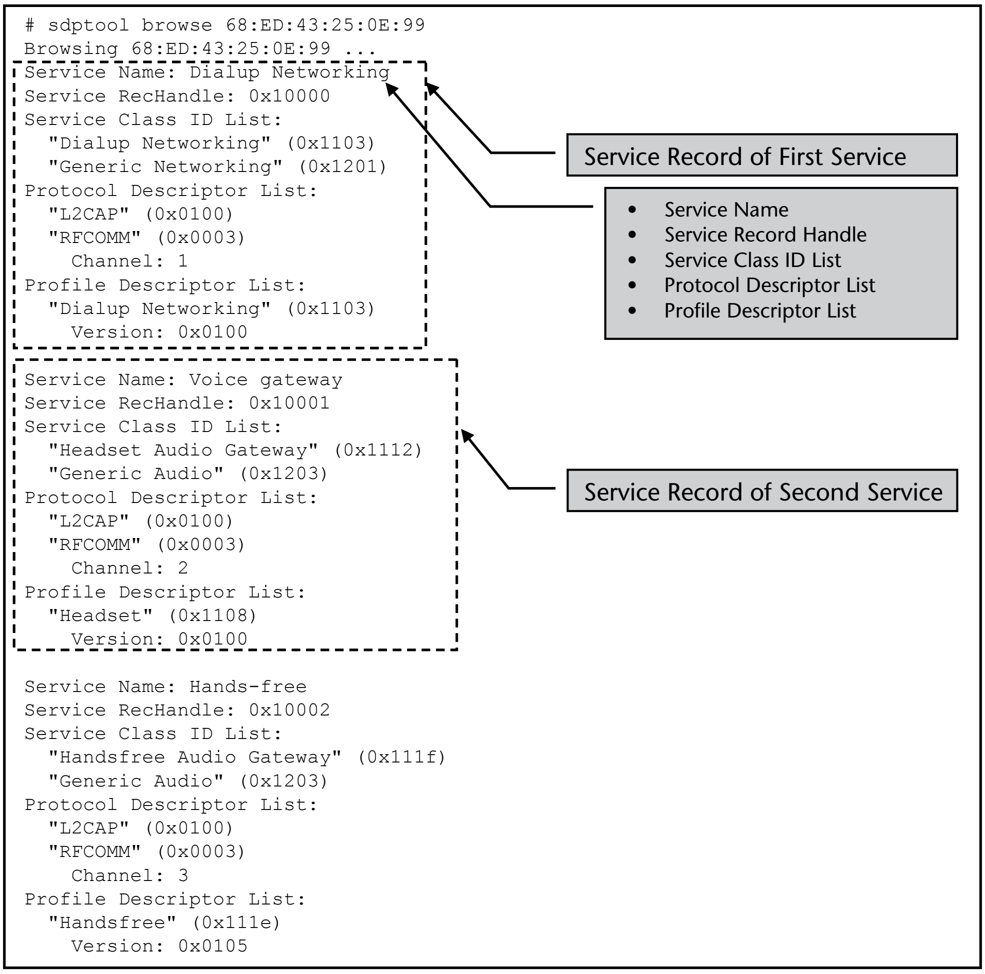

Service Record of First Service
• Service Name
• Service Record Handle
• Service Class ID List
• Protocol Descriptor List
• Profile Descriptor List
Service Record of Second Service

```
# sdptool browse 68:ED:43:25:0E:99
Browsing 68:ED:43:25:0E:99 ...
Service Name: Dialup Networking
Service RecHandle: 0x10000
Service Class ID List:
  "Dialup Networking" (0x1103)
  "Generic Networking" (0x1201)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Profile Descriptor List:
  "Dialup Networking" (0x1103)
    Version: 0x0100

Service Name: Voice gateway
Service RecHandle: 0x10001
Service Class ID List:
  "Headset Audio Gateway" (0x1112)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 2
Profile Descriptor List:
  "Headset" (0x1108)
    Version: 0x0100

Service Name: Hands-free
Service RecHandle: 0x10002
Service Class ID List:
  "Handsfree Audio Gateway" (0x111f)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 3
Profile Descriptor List:
  "Handsfree" (0x111e)
    Version: 0x0105
```

Figure 5.3 SDP browsing.

5.4 Real World Application—Café Bluebite

This section explains how to use the Bluetooth commands to implement a simple real world application. It is assumed that the reader is familiar with writing simple shell scripts using bash shell.

Let's say you have been requested by Café Bluebite (fictitious name) to create a Bluetooth based advertisement application for them. They are located in a shopping mall and they want to send the deal of the day to anyone who visits the mall. The deal is to be sent on the person's mobile phone via Bluetooth assuming that the person has his Bluetooth switched on when he enters the Mall.

5.4.1 Requirements Specification

The requirements specification provided by Café Bluebite is as follows:

- Advertise the deal of the day to any user who enters the shopping mall.

- Send this advertisement to the person’s mobile phone using Bluetooth.

5.4.2 High Level Design

The high level design of the application is provided in the form of a flow chart in Figure 5.4

Some key points to note in the high level design are as follows:

- Step 1c
  - We switch off the discoverable and connectable mode of our own device. This is because we only want to send advertisements messages and we don’t want to receive any messages from the remote devices.
- Step 7
  - We want to send messages to only Mobile phones and Tablets. There is no point in sending messages to headsets, keyboards, etc. because they don’t have display capabilities.
- Step 11
  - This example uses an endless loop. In practice it will be needed to terminate this loop based on certain conditions.

5.4.3 Code

Before we start writing the code, let’s look at the pre-requisites to run this program.

This program will use OBEX to send the advertisements to the remote device. It’s possible that the obexftp software is not installed on your Linux system. It’s possible that the obexftp software is not installed on your Linux system. To check this, you may run the command shown in Figure 5.5. This indicates that the obexftp is not installed on you system. To install it, run the command shown in Figure 5.6.

You will need to be online to download the obexftp package. During installation it will ask for a few confirmations and then install obexftp on your system.

Now let’s start writing the code for each of the steps mentioned above.

We will write the code as bash scripts for simplicity. You may either use a programming language or scripts in another shell depending on what you are conversant with.

Step 1: Initialize Bluetooth. The following steps assume that the Bluetooth controller is attached to hci0 interface. If it’s attached to another interface, then the name of the correct hci device needs to be put here. This step also disables inquiry and page scans. After this other devices will not be able to discover our device or connect to our device.

```
echo “Initializing the Bluetooth Controller on hci0...”
hciconfig hci0 up      # Initialize the Bluetooth controller.

echo “Configuring the name to Cafe Bluebite...”
hciconfig hci0 name “Cafe Bluebite”  # Configure the BT name.
```

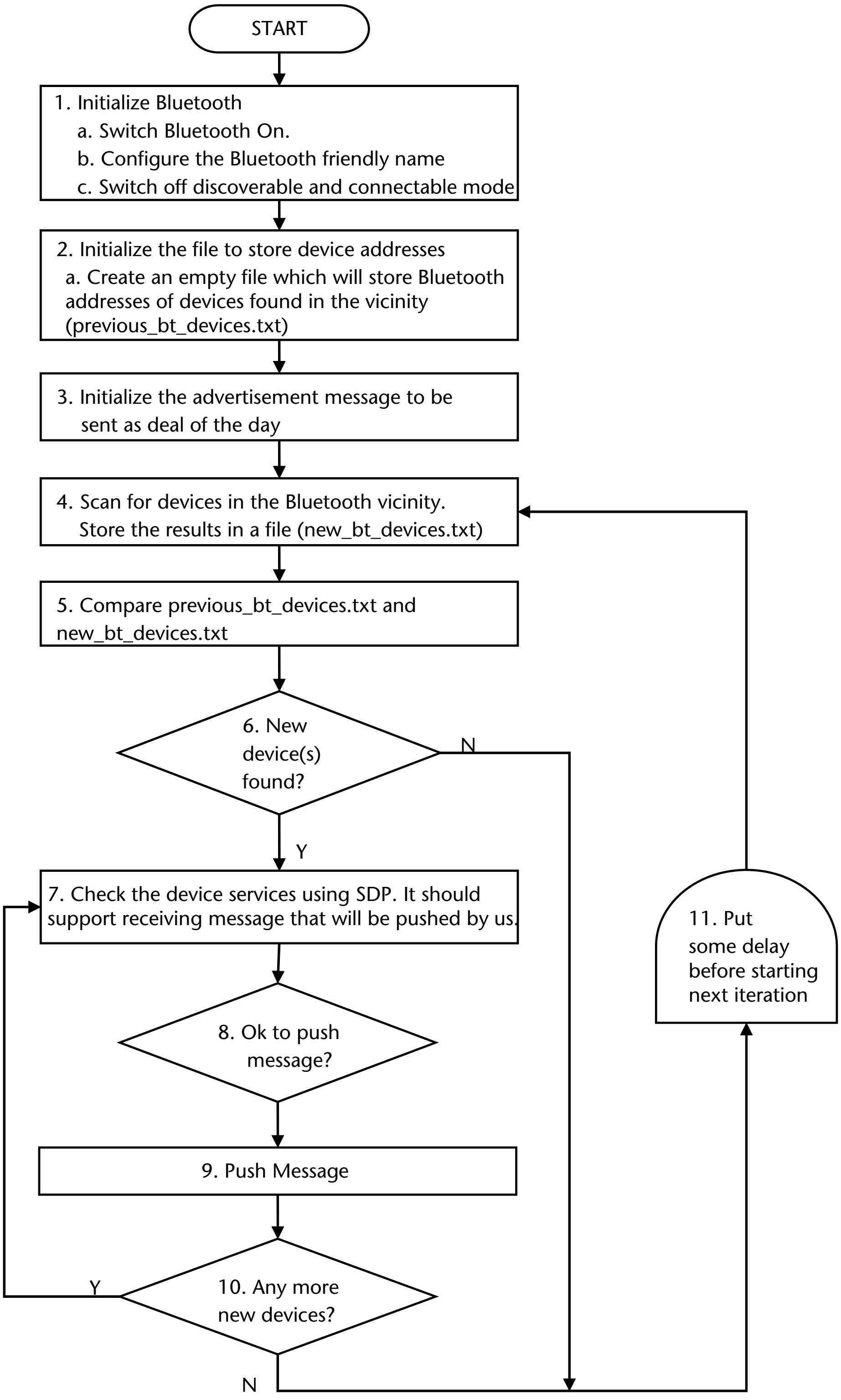

START
1. Initialize Bluetooth
a. Switch Bluetooth On.
b. Configure the Bluetooth friendly name
c. Switch off discoverable and connectable mode
2. Initialize the file to store device addresses
a. Create an empty file which will store Bluetooth addresses of devices found in the vicinity (previous_bt_devices.txt)
3. Initialize the advertisement message to be sent as deal of the day
4. Scan for devices in the Bluetooth vicinity. Store the results in a file (new_bt_devices.txt)
5. Compare previous_bt_devices.txt and new_bt_devices.txt
6. New device(s) found?
N
Y
7. Check the device services using SDP. It should support receiving message that will be pushed by us.
8. Ok to push message?
9. Push Message
10. Any more new devices?
Y
N
11. Put some delay before starting next iteration

Figure 5.4 High level design—flow chart for Bluebite.

```
# obexftp
The program 'obexftp' is currently not installed.  You can install it by typing:
apt-get install obexftp
```

Figure 5.5 OBEX FTP check for installation.

```
# apt-get install obexftp
```

Figure 5.6 OBEX FTP installation.

```
echo “Switching off inquiry and page scans...”
hciconfig hci0 noscan     # Disable page and inquiry scan.
```

Step 2: Create an empty file to store device names. This step is quite easy. We can just do a simple “> previous_bt_devices.txt” to create such a file.

```
# Create an empty file to store the previous list of devices
> previous_bt_devices.txt
```

Step 3: Initialize the advertisement message to be sent as deal of the day.

```
# Initialize the advt message to be sent as deal of the day
echo “Welcome to the shopping mall !!” > advt.txt
echo “Visit Cafe Bluebite for exciting deals” >> advt.txt
echo “Get 15% off if you show this message” >> advt.txt
```

Step 4: Scan for devices in the Bluetooth vicinity. Store the results in a file (new_bt_devices.txt). “hcitool inq” performs a Bluetooth inquiry for remote devices in the vicinity and reports the Bluetooth device address, clock offset and Class of Device for each device that is found.

We redirect the output of this command to the file temp_bt_devices.txt. The file temp_bt_devices.txt will contain the list of devices in the following format:

```
#hcitool inq
Inquiring ...
  68:ED:43:25:0E:99  clock offset: 0x298e  class: 0x7a020c
  00:17:83:DC:72:E9  clock offset: 0x7596  class: 0x1a0114
```

We need only the BD_ADDRs out of this list. So the remaining information can be removed. We will use a combination of the following two commands to remove the remaining information:

tail –n +2: To remove the first line.

cut –c2-12: To keep only columns 2 to 18 which contain BD_ADDR.

```
# Perform an inquiry and store the results in a temporary file.
hcitool inq > temp_bt_devices.txt
# Remove extra info from the temporary file
tail –n +2 temp_bt_devices.txt | cut –c2-18 > new_bt_devices.txt
```

If we use these two commands on the shell prompt, the file new_bt_devices.txt will contain the following.

```
68:ED:43:25:0E:99
00:17:83:DC:72:E9
```

This file will be used in the following steps for comparing with the previous list of Bluetooth devices stored in previous_bt_devices.txt.

Step 5: Compare previous_bt_devices.txt and new_bt_devices.txt to find out the devices that have got added.

There are several methods to do this with varying level of complexity and simplicity. In the interest of simplicity we choose a very rudimentary method here.

We read each line of the file new_bt_devices.txt and check if that line was present in previous_bt_devices.txt. This comparison is done using the grep command. The grep command returns 0 if the line is found and 1 if it is not found.

One interesting thing in the code below is that we redirect the output of grep command to /dev/null. This is because the output of the grep command is the list of matching lines. Besides that it also sets the exit status ($?) to 0 if a match was found and 1 if a match was not found. Since we are only interested in the exit status and not the list of matching lines, we let the list of matching lines go to /dev/null.

```
# Run a loop for all lines in new_bt_devices.txt
for line in $(cat new_bt_devices.txt)
do
    # Check if a match is found in previous_bt_devices.txt
    grep $line previous_bt_devices.txt > /dev/null
    # If a match is not found, then it's a new device
    if [ "$?" -eq "1" ]
    then
        echo "New Device found: $line"

  #### Some more code will be put here in Step 7 ####
    fi
done
```

Step 7: Check the remote device services using SDP. For simplicity, let's create a function check_sdp that will do this.

check_sdp function will search the SDP records of the remote device to see if it supports the OBEX Object Push service. This service is referred to by SDP as OPUSH.

It first creates two temporary files temp_sdp.txt and temp_sdp1.txt. It then searches for the OPUSH service on the remote device using the sdptool command. If the OPUSH service is found, it finds the RFCOMM Channel number on which the service is present.

```
function check_sdp ()
{
    # Create a blank file to store the results of SDP search
    > temp_sdp.txt
    > temp_sdp1.txt
    # Do an SDP search for OPUSH service on the BD_ADDR
    sdptool search --bdaddr $1 OPUSH > temp_sdp.txt

    # The output is in temp_sdp.txt
    # if the device has OBEX Object Push service then we must
    # be able to find the string OBEX Object Push in the file
    grep "OBEX Object Push" temp_sdp.txt > /dev/null
    retval=$?

    if [ $retval -eq 0 ]
    then
        # Check for the Channel number
        grep "Channel: " temp_sdp.txt > temp_sdp1.txt
        retval=$?

        if [ $retval -eq 0 ]
        then

            # Channel number is in the format
            # Channel: 6
            # So extract the field after colon (:)
            channel_num=`cut -d: -f2 temp_sdp1.txt`
            echo OBEX Object Push service found on Channel Num-
ber: $channel_num
        fi
    fi

    # grep returns 0 if found. 1 otherwise.
    # this is the same that our function has to return.
    return $retval
}
```

Step 8: If the check_sdp function returned 0, it means that the device supports OPUSH and we can push the advertisement.

Step 9: Push the advertisement.

```
          # Check if the device supports Object Push
          echo "Checking Services..."
          check_sdp $line

          if [ "$?" -eq "0" ]
          then
              echo "Device supports OPUSH. Pushing advertisement"
              push_advt $line
          else
              echo "Device does not supports OPUSH."
          fi

function push_advt ()
{
    echo executing obexftp --bluetooth $1 -B $channel_num -p
advt.txt
    obexftp --bluetooth $1 -B $channel_num -p advt.txt}
```

Step 10: Repeat this for any more devices found. This is already done by the for loop in Step 5.

Step 11: Put some delay before starting next iteration. Before doing this also copy the temp_bt_devices.txt file to previous_bt_devices.txt so that this file can be used for comparison to find whether any new devices came in the vicinity.

```
cp temp_bt_devices.txt previous_bt_devices.txt
sleep 5s
```

5.4.4 Complete Code

The previous section provided code in pieces. The complete code for Bluebite.sh is provided below.

```
#! /bin/bash

#################################################################
# function check_sdp
# Input: BD_ADDR of the device for which SDP search is to be done
# Output: 0 if OPUSH record is found. 1 otherwise
# Synopsis: This function does an SDP Search for OBEX Object Push
#           service on the remote device
#################################################################
function check_sdp ()
{

    # Create a blank file to store the results of SDP search
    > temp_sdp.txt
    > temp_sdp1.txt

    # Do an SDP search for OPUSH service on the BD_ADDR
    sdptool search --bdaddr $1 OPUSH > temp_sdp.txt

    # The output is in temp_sdp.txt
    # if the device has OBEX Object Push service then we must
    # be able to find the string OBEX Object Push in the file
    grep "OBEX Object Push" temp_sdp.txt > /dev/null
    retval=$?

    if [ $retval -eq 0 ]
    then
        # Check for the Channel number
        grep "Channel: " temp_sdp.txt > temp_sdp1.txt
        retval=$?

        if [ $retval -eq 0 ]
        then
          # Channel number is in the format
          # Channel: 6
          # So extract the field after colon (:)
          channel_num=`cut -d: -f2 temp_sdp1.txt`
          echo OBEX Object Push service found on Channel Number:
$channel_num
        fi
    fi

    # grep returns 0 if found. 1 otherwise.
    # this is the same that our function has to return.
    return $retval
```

```
#################################################################
# function push_advt
# Input: BD_ADDR of the device to which advertisement is to be
#        pushed
# Output: None
# Synopsis: This function pushes the advertisement to the remote
#           device
#################################################################
function push_advt (){

    echo executing obexftp --bluetooth $1 -B $channel_num -p
advt.txt
    obexftp --bluetooth $1 -B $channel_num -p advt.txt
}

echo “Initializing the Bluetooth Controller on hci0”
hciconfig hci0 up          # Initialize the Bluetooth controller.

echo “Configuring the name to Cafe Bluebite”
hciconfig hci0 name “Cafe Bluebite”  # Configure the BT name.

echo “Disabling Simple Secure Pairing”
hciconfig hci0 sspmode 0

echo “Switching off inquiry and page scans.”
hciconfig hci0 noscan             # Disable page and inquiry scan.

# Create an empty file to store the previous list of devices
> previous_bt_devices.txt

# Initialize the advt message to be sent as deal of the day
echo “Welcome to the shopping mall !!” > advt.txt
echo “Visit Cafe Bluebite for exciting deals” >> advt.txt
echo “Get 15% off if you show this message” >> advt.txt

echo “Starting infinite loop. Press Ctrl-C to exit program”

while [ 0 -eq 0 ]
do
    # Perform an inquiry and store the results.
    hcitool inq > temp_bt_devices.txt

    # Remove extra information from the temporary file and
    # store results in new_bt_devices.txt
    tail -n +2 temp_bt_devices.txt | cut -c2-18 > new_bt_devices.
txt

    num_devices=`cat new_bt_devices.txt | wc -l`
    echo Found $num_devices devices in this iteration

    # Run a loop for all lines in new_bt_devices.txt
    for line in $(cat new_bt_devices.txt)
    do
       # Check if a match is found in previous_bt_devices.txt
        grep $line previous_bt_devices.txt > /dev/null
```

```
            # If a match is not found, then it's a new device
            if [ "$?" -eq "1" ]
            then
                echo "New Device found: $line"

                # Check if the device supports Object Push
                echo "Checking Services..."
                check_sdp $line

                if [ "$?" -eq "0" ]
                then
                  echo "Device supports OPUSH. Pushing advertisement"
                  push_advt $line
                else
                  echo "Device does not supports OPUSH."
                fi
            fi
        done
        # Sleep for 5 seconds
        cp temp_bt_devices.txt previous_bt_devices.txt
        sleep 5s
    done # Infinite While loop
```

5.5 Disclaimer

The code provided here is only for educational purposes to illustrate the use of different Bluetooth related commands and features. It has been tested with only a few mobile phones and may not work with all phones or lead to some unknown problems. So you are advised to use it at your own risk. To make the code suitable for commercial use several enhancements, error checks, and exhaustive testing would be needed.

5.6 Summary

This chapter explained the basic requirements of bringing up a setup to try out some Bluetooth operations like enabling and disabling Bluetooth, discovering devices, etc. An example application was also developed to get accustomed to how the various Bluetooth operations can be invoked through simple shell commands.

With this chapter, the background of Bluetooth is completed. The next chapter onwards will focus on Bluetooth Low Energy.

Bibliography

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

Bluetooth SIG, Specifications of the Bluetooth System, Profiles http://www.bluetooth.org.

BlueZ website (http://www.bluez.org).

CHAPTER 6

Bluetooth Low Energy—Fundamentals

6.1 Introduction

As described in Chapter 1, Bluetooth Low Energy is the next major evolution of the Bluetooth technology. It specifies requirements for devices to have ultra low power consumption. This is a radical change from the direction in which the technology was evolving through previous versions. While the focus of previous versions was either feature enhancements or increase in the throughput, LE focused in an entirely new direction—how to cut down the power consumption drastically? LE technology is fully optimized from the ground up to ensure that the power consumption is kept to a minimum. This meant a complete redesign of several key components to ensure that all steps are taken to reduce the power requirements.

In general Bluetooth devices are battery powered. It's expected that the LE devices may have smaller batteries like the coin cell batteries (or even smaller ones). This technology focuses on reducing both the peak current and the average current. A reduced average current ensures that the battery drains down slowly. A reduced peak current means that the devices can continue operating even when the battery has started running down and the maximum current that the battery can provide has been reduced.

Some of the uses for LE were shown in Chapter 1. These included finding devices, alerting devices, proximity detection, sensors, healthcare, sports and fitness equipment, mobile payments, etc. It may be noted that none of those use cases focused on high throughput or transferring big chunks of data. Rather the use cases focused on transferring very short pieces of information that could be transferred only when needed, which may generally not be very frequent.

Typical LE use cases would include creating a connection, transferring a few bytes or kilobytes of data and then disconnecting. The connection time is so low that it's easier to reestablish the connection every time a data transfer is needed without any impact on the user experience. This is in contrast to BR/EDR use cases like a connection that is maintained for a long time to ensure that there is least latency when there is an incoming call, or use cases like the exchange of big file in the case of FTP profile.

There are two broad classifications of Bluetooth systems.

- The first is the classic Bluetooth system that conforms to versions prior to 4.0 of the Bluetooth specification. It is also referred to as BR/EDR. BR stands for Basic Rate indicating that the device can support up to a maximum data rate of 721 kbps. EDR stands for Enhanced Data Rate indicating that the device can support up to a maximum data rate of 2.1 Mbps.
- The second is the LE system which conforms to 4.0 (or higher) version of the Bluetooth specification and supports enhancements for ultra low power. These systems have lower complexity and lower cost compared to BR/EDR systems. The throughput is significantly lower. The maximum throughout is about 305 kbps for 4.0 compliant devices and 800 Kbps for 4.2 compliant devices, though devices generally don't need or use such high rate data transfers.

A device can support only BR/EDR, only LE, or both BR/EDR and LE. A device which supports both BR/EDR and LE is also referred to as BR/EDR/LE or dual mode device. This will be explained in further detail later in this chapter.

The architecture of BR/EDR devices was covered in detail in previous chapters. This chapter and further chapters will focus on the architecture of LE devices.

6.2 Single Mode versus Dual Mode Devices

Depending on the functionality supported, the Bluetooth devices may be categorized into 3 types:

1. *BR/EDR Devices:* These are the classic Basic Rate/Enhanced Data rate devices which do not support the LE functionality.
2. *LE Only Devices or Single Mode LE Devices:* These devices support only LE functionality. Examples of these devices include watches, key fobs, heart rate monitors, thermometers, sports and fitness equipment, sensors, etc. These devices are expected to have ultra low power consumption and last for several months or years on coin cell batteries.
3. *BR/EDR/LE or Dual Mode Devices:* These devices support both BR/EDR and LE functionality. Typically these devices are smartphones, tablets, PCs, etc. These devices are expected to communicate with both the BR/EDR devices and single mode LE devices even at the same time. These devices don't have as stringent requirements on power consumption as the single mode LE devices since these have bigger batteries or are generally recharged frequently.

The use cases of LE only devices were explained in Chapter 1. Some of the possible use cases of Dual Mode devices are as follows:

1. A person is listening to music or taking a call on the Bluetooth headset and the child (or the pet) goes out of range. In this case the user would be

alerted about the child (or the pet) going out of range by the proximity detection feature of LE so that the person can take immediate action.

2. A person walks into the home while on a call. The presence detection function of LE may automatically switch on the lights or switch on the air conditioning.
3. A person is jogging on a treadmill and wants to listen to music on the Bluetooth headset and at the same time monitor his/her heart rate, number of steps he/she has run, etc.
4. A person using a GPS enabled smartphone while cycling wants to see distance covered on the track and parameters like current heart rate, maximum heart rate, average heart rate, calories burnt, etc. In addition to displaying on the smartphone, the same data could be sent to the laptop or fitness center for further analysis using the smartphone as a gateway.

Table 6.1 provides information on compatibility between different categories of devices.

Some examples of communication between the various devices types are provided below:

1. BR/EDR to BR/EDR—Classic Bluetooth communications between:
   a. Mobile phone to headset.
   b. Laptop to laptop.
   c. PC to printer.
2. Single Mode LE to Single Mode LE—Low energy ecosystem
   a. Sports sensor displaying data on a watch.
3. Single Mode LE to Dual Mode
   a. A laptop sending an alert to a key fob.
   b. A heart rate monitor sending data to the hospital's computer using smartphone as the gateway.

6.3 Bluetooth Smart Marks

The Bluetooth smart marks were created to help consumers ensure compatibility among their Bluetooth devices. There are two trademarks from the Bluetooth SIG:

Table 6.1 Compatibility Between Single Mode and Dual Mode Devices

	BR/EDR	*Single Mode LE*	*Dual Mode*
BR/EDR	Yes	No	Yes
Single Mode LE	No	Yes	Yes
Dual Mode	Yes	Yes	Yes

- Bluetooth Smart;
- Bluetooth Smart Ready.

6.3.1 Bluetooth Smart (Sensor-Type Devices)

Bluetooth Smart devices are sensor-type devices that are used to collect a specific piece of information. After collecting this information, these devices send it to the Bluetooth Smart Ready devices. Bluetooth Smart devices include only a single mode LE radio and are expected to consume ultra low power.

Some examples of Bluetooth Smart devices are heart rate monitors, thermometers, sports equipment, etc. These devices collect a specific piece of information like heart rate or temperature and then relay it to the Bluetooth Smart Ready devices.

Bluetooth Smart trademark was developed to brand qualified devices as meeting the following three requirements:

1. Conform to Bluetooth 4.0 or higher with GATT based architecture.
2. Contain Single mode LE radio.
3. Use GATT-based architecture to enable a particular functionality (GATT-based architecture will be explained in detail in subsequent sections).

6.3.2 Bluetooth Smart Ready (Hubs)

Bluetooth Smart Ready devices are the devices that receive data sent from the classic Bluetooth and Bluetooth Smart devices and give it to applications that make use of that data. The applications could be running on these devices themselves or could be running anywhere else on the internet. These devices implement the dual mode radio and can connect to the BR/EDR devices as well as the Bluetooth Smart devices. Such devices have one single Bluetooth device address which is used for both the BR/EDR and LE radios. Some examples are phones, tablets, PCs etc.

Bluetooth Smart Ready mark was developed to brand qualified devices as meeting the following three requirements:

1. Conform to Bluetooth 4.0 or higher with GATT based architecture.
2. Contain Dual mode radio.
3. Provide a means by which the end user can choose to update the functionality for a Bluetooth Smart device on a Bluetooth Smart Ready device. For example if the user buys a new Bluetooth Smart device then new software can be installed on the smart phone to communicate to that device.

Figure 6.1 shows Bluetooth, Bluetooth Smart and Bluetooth Smart Ready devices. In this scenario, the mobile phone is the Smart Ready device and it is communicating to two devices at the same time:

1. Streaming audio data to a Bluetooth headset.
2. Collecting temperature information from a Bluetooth Smart thermometer and acting as a hub to relay that information to a server located in the hospital. The server can then take the appropriate action like informing the doctor or pharmacist.

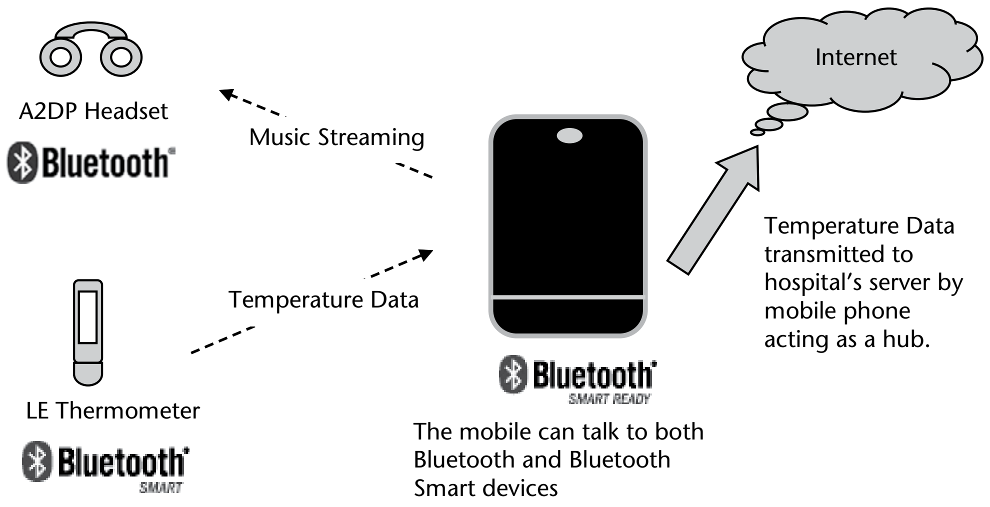

A2DP Headset
Bluetooth
Music Streaming
Internet
Temperature Data transmitted to hospital's server by mobile phone acting as a hub.
Temperature Data
LE Thermometer
Bluetooth SMART
Bluetooth SMART READY
The mobile can talk to both Bluetooth and Bluetooth Smart devices

Figure 6.1 Bluetooth, Bluetooth Smart, and Bluetooth Smart ready devices.

6.4 LE Fundamentals

The devices which are based on BR/EDR require a recharge in a few days or few weeks and are generally using larger batteries than coin cell batteries. Take, for example, a Bluetooth keyboard, mouse, headset, etc. All have relatively large batteries and need a recharge every few days or weeks.

Achieving a power consumption of several months to several years with LE was not as easy as optimizing the power consumption of various layers in the Bluetooth architecture. That would not have led to drastic reduction in the power consumption. So LE has been designed almost from scratch to ensure that all possibilities to achieve ultra low power consumption have been incorporated. It is designed ground up for simplicity, low cost and ultra-low power consumption without compromising robustness, security, global usage, or ease of use. Most importantly the compatibility of dual mode devices with the existing BR/EDR devices has been preserved to ensure that nothing gets broken when manufacturers upgrade their existing devices to BT 4.0-based devices.

There are several enhancements done in BT 4.0 specification to achieve low power. Some of the fundamental concepts related to LE operation are introduced below. These will be explained at length in the following chapters.

6.4.1 Frequency Bands

Similar to the BR/EDR radio, the LE radio operates in the 2.4 GHz ISM band. This band is globally license free and is shared by several other wireless technologies. LE also uses a frequency hopping mechanism to combat interference. (Frequency hopping was explained in Chapter 3).

One important difference between BR/EDR and LE is that while BR/EDR uses 79 channels for frequency hopping, LE uses only 40 channels. Secondly there are dedicated channels for advertising and sending data in the case of LE. This will be explained in detail later in Chapter 8.

6.4.2 Mostly Off Technology

LE can be termed as a "mostly off" technology. This means that the LE devices are expected to be sending data only occassionally and be in a switched-off state for the remaining time. For example, the heart rate monitor may collect all data and then send it across once per hour or once per day, the weighing machine may send the weight only once per day or once per week, or a temperature sensor may send the data only if the temperature crosses a certain threshold limit.

LE technology is designed in such a manner that the LE devices remain off most of the time and switch on only when they need to transmit some data. This ensures that the duty cycle (ratio of device off to device on) is almost close to zero and the device will only switch on when some specific conditions are triggered. Normally the device would just remain off.

6.4.3 Faster Connections

LE takes much less time to create connections as compared to BR/EDR. This is because LE uses only 3 dedicated advertising channels which can be used for creating connections. This is in contrast to BR/EDR where 32 channels are used for inquiry and paging (connection).

Since BR/EDR has more channels, the device takes more time in scanning across all channels before a connection can be created. A typical BR/EDR connection can take up to 20 milliseconds while in the case of LE the connection time is less than 3 milliseconds because the device has to scan on only 3 channels.

A faster connection means that whenever the device needs to send the data, it can quickly connect, transmit data, and then disconnect. The total time for this transaction may be in the range of only 3 to 4 milliseconds. This means that the LE radio needs to be switched on for a very short time. The shorter the time that the radio is switched on the lower the power consumption. The device may switch itself off until it needs to transmit data again.

This is in contrast to BR/EDR. For example, in the case of a headset that is connected to a mobile phone, the headset may remain connected to mobile phone for several days. During this time the headset would be in sniff mode. This would mean that it would be waking up periodically to see if the mobile phone has data to send. This would translate into continuous drain of battery since the headset would wake up periodically and check to see if the mobile phone has any data to send.

6.4.4 Reduced Functionality

LE incorporates some major reductions in the functionality so that it can target a specific market—the one which needs devices to be consuming ultra low power. So it cuts down heavily on the functionality to reduce the memory required to implement that functionality.

Some of the major reductions are:

1. Not mandatory to implement both transmitter and receiver—LE is designed in such a manner that a device can implement only transmitter, only receiver or both transmitter and receiver. For example an LE weighing machine may implement only the transmitter. As soon as it has measured the weight, it just transmits and does not need to receive anything. Such devices would reduce the silicon area to almost half as compared to devices which implement both transmitter and receiver.
2. No support for voice channels—LE is intended for devices that send short amount of data infrequently. It is not intended for devices like headsets which need to transfer continuous stream of voice data. So the SCO/eSCO functionality has been completely removed.
3. No support for Scatternet—LE provides support for only a piconet and the support for scatternet has been removed. This has simplified the state machine of the link layer since a device can only be in one piconet at a particular time. While BR/EDR supports only up to 7 devices in a piconet, an LE piconet can have any number of devices, limited only by the resources available on the piconet master device. So removal of scatternet functionality in LE did not put any restriction on the number of devices that could be connected together while it also simplified the link layer state machine. The support for scatternets was added in specifications 4.1 as optional, so the system designers could choose this flexibility at the cost of adding complexity.
4. No support for Master/Slave role switch—BR/EDR allowed the possibility to switch the roles of Master and Slave. In LE, once a connection is made, the role switch is not permissible. It is also clearly defined right in the beginning which device will end up being a Master and which device will end up being a Slave. This led to a lot of simplification of the link layer state machine.
5. No need for continuous polling of the link. In the case of BR/EDR, even if there is no data exchange, POLL/NULL packets are continuously exchanged to check if the remote device is still present, leading to consumption of power. In the case of LE, there is no need to continuously monitor the link. Devices can just shut down the link and recreate it whenever needed without any impact on the user experience. This is because the link setup time is far less in case of LE as compared to BR/EDR.
6. No support for sniff and park modes. The LE controllers are designed to be very simple and power efficient from ground up. The connection is created for a very short duration—only when the data is transferred, and then disconnected. So there is no need for separate power saving modes. It can be said that the default mode for LE is already power savings mode and so no additional power savings modes are defined.

6.4.5 Shorter Packets

LE uses packets of much shorter size as compared to BR/EDR which means that the time needed to transmit or receive them is lesser. So the radio will be switched on for a lesser time leading to savings in power consumption.

Besides this the buffer space needed to store the packets is much lesser. The maximum size of LE packet is 27 bytes which is much shorter than the maximum size of 1021 bytes supported by BR/EDR (As discussed in Chapter 3, the maximum size of 3DH5 packets is 1021 bytes). The maximum packet size allowed by 4.2 specification is 251 bytes, which is still much shorter than BR/EDR packets.

Besides lesser times, shorter packets also require much lesser power to transmit. This is because of radio characteristics. If a long packet has to be transmitted, the radio needs to be in a high power state for a longer period of time, resulting in the heating up of silicon. This changes the material's physical characteristics and deviation in the transmission frequency which could result in a packet loss or even link loss. To counter this, the radio needs to be constantly recalibrated in order to ensure that the transmission frequency is correct. Recalibration logic makes the radio more complex and also requires power for the recalibration. In comparison shorter packets ensure that the silicon stays cool and the characteristics don't change. Hence no recalibration logic is needed.

6.4.6 Reduced Dynamic Memory Footprint

Another design principle used in LE is to optimize the usage of dynamic memory as much as possible. This is because firstly memory requires silicon area which contributes to the costs and secondly dynamic memory requires constant supply of current to retain the contents that are stored in it. So it adds to the current consumption of the device. (Dynamic memories typically need to be refreshed continuously so that they do not lose the contents. In contrast ROM and FLASH memories do not need to be refreshed).

To reduce the dynamic requirements on memory, LE incorporates the following:

1. Shorter packets—The amount of buffer memory space needed for storing packets is much smaller in case of LE due to shorter packet sizes. This was explained in the previous section.
2. Shorter headers—LE uses only a 32-bit access code to reduce the overall size of the packet. This again leads to less time to transmit the packet and lesser buffer requirement.
3. Simple Protocol—The LE protocol has been designed to be very simple so that there is least state information to be stored. For example the Link Layer has only 5 states and a very limited number of state transitions. Similarly the L2CAP layer has also been simplified a lot to support a very limited number of CIDs (Channel Identifiers) and signaling commands. These will be explained in detail in the subsequent chapters.
4. Uniform Packet Format—LE uses only one packet format for all types of packets. This makes the logic for creating packets on the transmitter side

and parsing packets on the receiver side much simpler thereby reducing the code that has to be implemented.

6.4.7 Optimized Power Consumption of Peripherals

LE is designed in such a manner that the peripherals consume the least amount of power while the central devices could consume a bit more power to compensate. This is because generally the peripherals are resource constrained. They have smaller batteries, memory and limited processing power. In contrast, the central devices may have much higher processing power and much bigger batteries. These may even be powered from mains (For example a TV, PC, or a set-top box) or may be recharged frequently (For example a mobile phone or a tablet).

As an example, if a weighing machine gets connected to a mobile phone to send data, then it's more important to optimize the power consumption of the weighing machine as compared to optimizing the power consumption of the mobile phone. The mobile phone may be charged every day or every alternate day while the weighing machine's battery may be replaced only once in a few months or years.

The LE protocol is designed in such a manner that the peripheral needs to be powered on for as less time as possible while the central device may be continuously powered on to wait for any data coming from the peripheral. So in the case of weighing machine transmitting data to the mobile phone, the mobile phone may always be scanning for packets while the weighing machine may just transmit packets when it has some information to transmit. So the weighing machine will be powered on for far lesser time as compared to the mobile phone.

Another important point is that, in the wireless radio world, transmitting packets consumes much less power than receiving packets. So LE tries to reduce the time for which the peripherals are receiving packets. Instead of this the LE protocol is designed in such a manner that the peripherals mostly transmit packets instead of scanning and receive only for very short durations, if they really have to. So in the case of weighing machine example, the weighing machine would transmit (advertise) to consume less power while the mobile phone would receive (scan). The weighing machine, may in fact never receive anything and may not even have a receiver in it.

6.4.8 No Need for Continuous Polling

In the case of LE, for devices to remain connected, there is no need for continuous exchange of packets as long as the link layer is synchronized to the timing, frequency and access address parameters.

This is in contrast of BR/EDR where, to remain connected, continuous POLL/NULL packets are exchanged even if there is no data to send. BR/EDR does provide the possibility to put the connection into low power mode to reduce the exchange of packets but packets are still exchanged leading to power consumption.

6.4.9 Backward Compatibility with BR/EDR

There are billions of BR/EDR device already in use. Mobile phones, tablets and laptops have an attach rate of almost 100%. LE technology is designed in such a manner that dual mode devices which are based on the BT 4.0 specification are backward compatible with the existing devices. This means that the next generation of mobile phones, laptops and other devices can be upgraded to BT 4.0 devices without breaking existing compatibility. Once these devices get upgraded to BT 4.0, they will also be able to talk to LE devices in addition to BR/EDR. So LE builds on to the ecosystem that has already been established by BR/EDR and paves the way for next millions and billions of devices.

A summary of the key LE features is shown in Table 6.2.

6.5 LE Architecture

LE has a layered architecture just like BR/EDR. The architecture of LE is shown in Figure 6.2. In many ways it's similar to the architecture of classic BR/EDR stack. It also uses the concept of protocols and profiles. (This was explained in Chapter 2).

The design of profiles is quite simplified in the case of LE. LE introduced the concept of GATT based profiles. Most of the common functionality that is needed by all profiles is moved into the ATT protocol and GATT profile. The profiles on top of GATT use the services that are provided by GATT and only implement the bare minimum things that are needed to support that specific use case.

In the next chapters, each of the protocol layers will be covered in details. The mechanisms that are employed by each of the protocol layers to reduce the

Table 6.2 Summary of Key LE Features

Connection Type	*Frequency Hopping Spread Spectrum.*
Spectrum	2.4 GHz ISM Band. Regulatory range: 2400 – 2483.5 MHz.
Frequency Hopping	Across 40 RF channels. The channels are separated by 2 MHz.
Modulation	Gaussian Frequency Shift Keying (GFSK).
Maximum Data Rate	305 kbps (4.0), 800 kbps (4.2)
Maximum Data Packet size	27 bytes (4.0), 251 bytes (4.2)
Typical Range	30 m to 100 m.
Topology	Master Slave architecture. The number of slaves is limited only by the availability of resources on the master.
Connection Time	In the range of 2.5 milliseconds. LE supports a much lower connection time as compared to BR/EDR. So it's easier to just re-establish the connection and transfer data in case of LE instead of keeping the connection alive.
Data Security: Authentication Key	AES 128 bit key.
Data Security: Encryption Key	AES-128 (Stronger than BR/EDR)
Voice Channels	Not supported.
Applicability	Does not require line of sight. Intended to work anywhere in the world since it uses unlicensed ISM band.

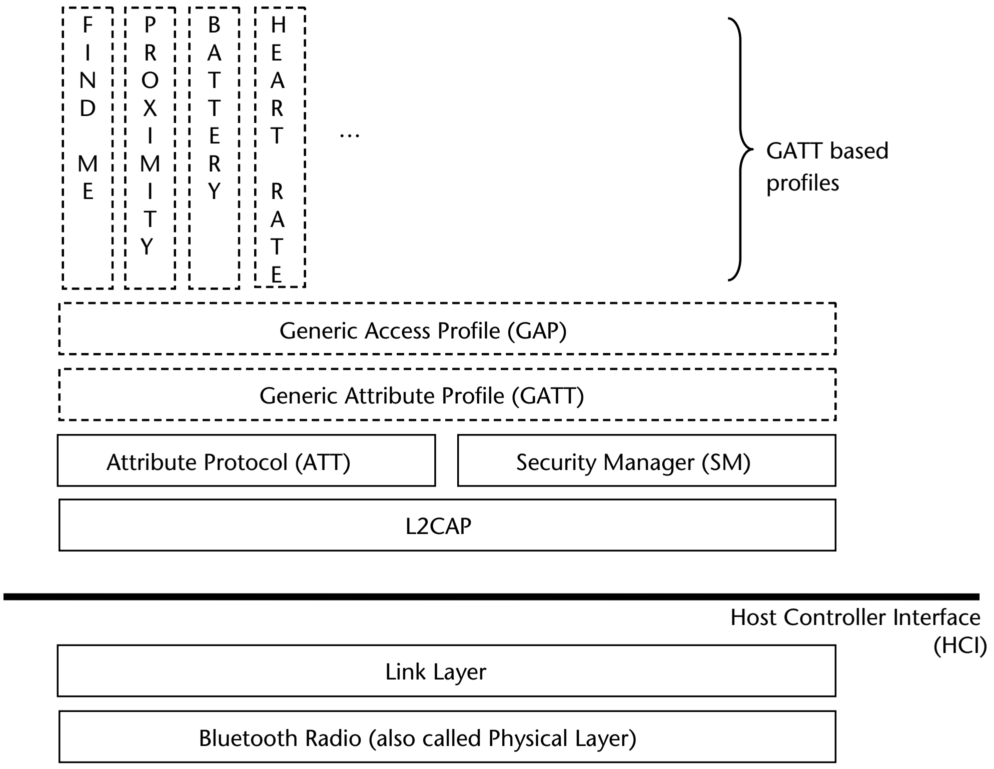

FIND ME
PROXIMITY
BATTERY
HEART RATE
...
GATT based profiles
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Attribute Protocol (ATT)
Security Manager (SM)
L2CAP
Host Controller Interface (HCI)
Link Layer
Bluetooth Radio (also called Physical Layer)

Figure 6.2 LE architecture.

power consumption will also be pointed out specifically so that the reader is able to appreciate the reasons why this technology is designed for ultra-low power operations.

6.6 Comparison between BR/EDR and LE

The architectural comparison of the BR/EDR and LE protocol stack is shown in Figure 6.3. The LE stack modifies some of the existing protocol layers and profiles like the L2CAP layer and the GAP profile. The LE radio has also been modified as explained above. LE also replaces some layers entirely to gain significant power savings. For example the link layer is defined from scratch.

In the case of dual mode devices, the implementation of some of the layers can be shared. For example a combined LE + BR/EDR radio can be used and the implementation of the L2CAP layer, HCI and GAP profile can be shared for the dual mode devices. This helps in maintaining backward compatibility as well as re-using the efforts that were spent already in developing code for these layers.

A broad level comparison between BR/EDR and LE is shown in Table 6.3. There are several other differences besides these and those differences will be highlighted in the next chapters at the appropriate places.

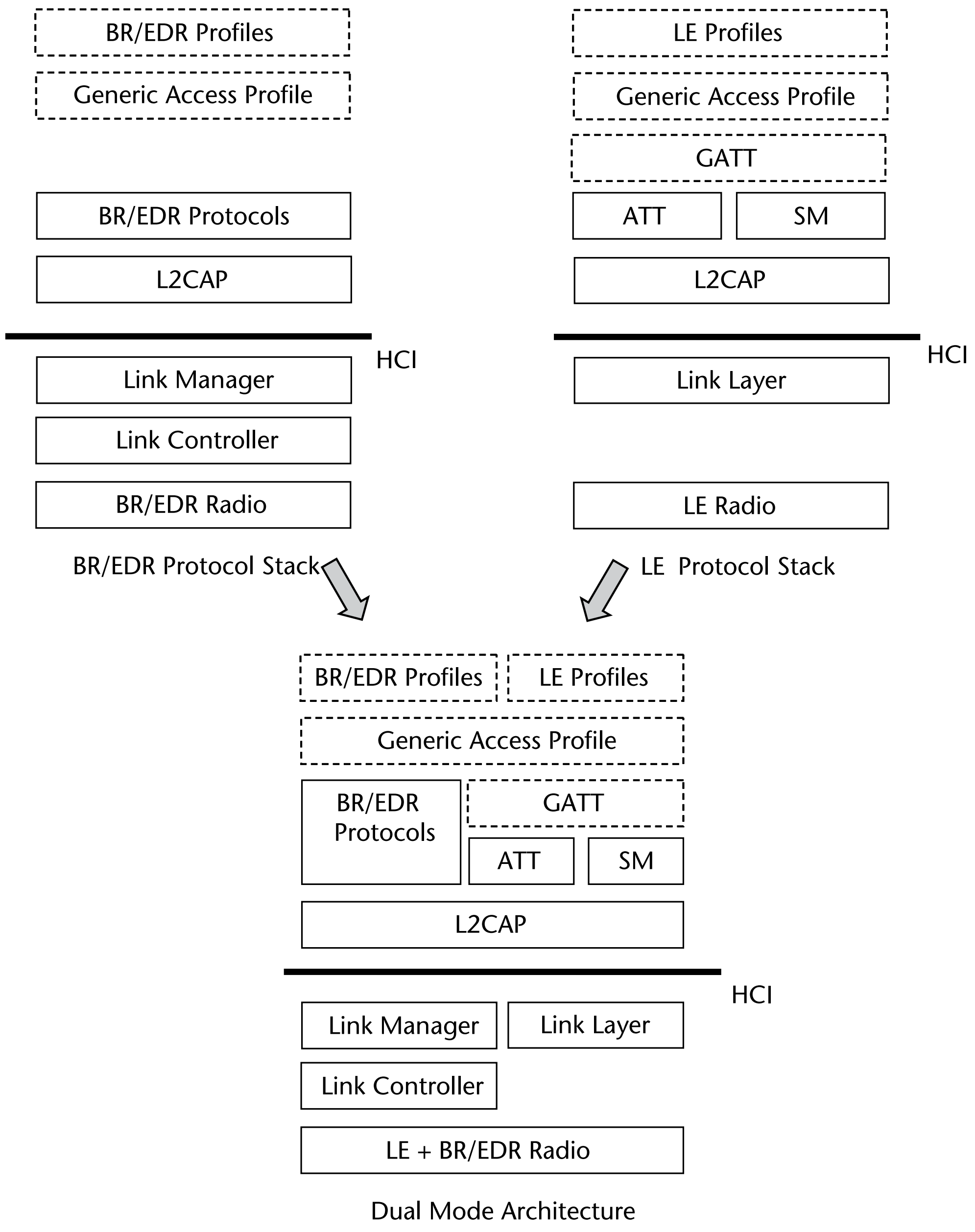


Figure 6.3 BR/EDR protocol stack, LE protocol stack, and dual mode architecture.

6.7 Summary

LE technology offers dual benefits. First, it builds on to the existing ecosystem of Bluetooth devices, and secondly, it offers a drastic reduction in the power consumption for LE only devices. This has opened up several new use cases where this technology can be effectively deployed.

Table 6.3 Comparison of BR/EDR and LE

Feature	*BR/EDR*	*LE*	*Remarks*
RF Channels and spacing	79 channels spaced by 1 MHz each	40 channels spaced by 2 MHz each	
Dedicated RF Channels	No dedicated RF channels. All are used for data.	3 channels dedicated for advertising. 37 channels dedicated for data.	
Range	Typically 10m to 30m. Can go to max of 100 m	Typically 30m to 50m. Can go to a max of 100m.	Typically LE based devices have a longer range
Connection time	In the range of 20 ms	Less than 3 ms	Much less time required to make a connection.
Maximum Packet Size	1021 bytes	27 bytes	LE uses much smaller packets.
Maximum Data Rate	BR: 721 kbps EDR: 2.1 Mbps	305 kb/s	LE supports much lower data rates. Even these data rates are seldom used.
Scatternet Support	Yes	No	LE simplifies implementation complexity by disallowing scatternets.
Master/Slave Role Switch	Supported	Not Supported	Simpler state machine.
Transmitter/Receiver	Both transmitter and receiver are mandatory	Device can have only transmitter, only receiver or both	LE saves a lot of silicon area if the device has to only transmit or only receive.
PDU Format	Several	One	LE has only one PDU format. This makes it simpler to create and parse packets
CRC Strength	16-bit	24-bit	Much more robust, especially in noise environments.
Voice Channels	Yes	No	No support for voice channels in LE.

The architecture of LE was introduced in this chapter. Besides this some of the enhancements done by LE to reduce power consumption were briefly introduced. These will be explained in depth in next chapters.

Bibliography

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

Bluetooth SIG, Specifications of the Bluetooth System, Profiles http://www.bluetooth.org.

SIG Low Energy Training, http://www.bluetooth.org.

Bluetooth Assigned Numbers, https://www.bluetooth.org/assigned-numbers.

CHAPTER 7

Physical Layer

7.1 Introduction

This chapter explains the operation of the physical layer of LE. As shown in Figure 7.1, this is the bottom most layer in the protocol stack. It is responsible for sending and receiving data over the air.

7.2 Frequency Bands

Like BR/EDR, the LE radio also operates in the 2.4 GHz ISM band (ISM stands for Industrial, Scientific and Medical). This frequency band is globally unlicensed and is used by several other devices like remote control toys, cordless telephones, NFC, Wireless LAN, etc. Some microwave ovens also generate interference in this frequency band.

Since this band may be shared by multiple devices, there is a good possibility that there may be interference from the other devices. So, the frequency is continuously changed for subsequent transmissions so that a new frequency is chosen for exchanging the data. The pattern of changing the frequencies is predefined so that the devices that need to exchange data know which frequency to hop to for the next data exchange. This is known as frequency hopping. LE uses frequency hopping technique to combat interference and fading.

The frequency band is divided into 40 channels which are spaced 2 MHz apart. (In contrast BR/EDR uses 79 channels which are spaced 1 MHz apart). The channels are numbered from 0 to 39 starting at 2402 MHz. The frequencies of various channels are derived from the formula:

$$f(k) = 2{,}402 + k * 2 \text{ MHz}, k = 0, \ldots, 39$$

The complete range used by LE is 2.400 to 2.4835 GHz. The frequency bands used by LE are shown in Figure 7.2.

Figure 7.3 shows the sniffer capture of transactions between a mobile phone and an LE device. It shows the transmissions on the 40 channels which include:

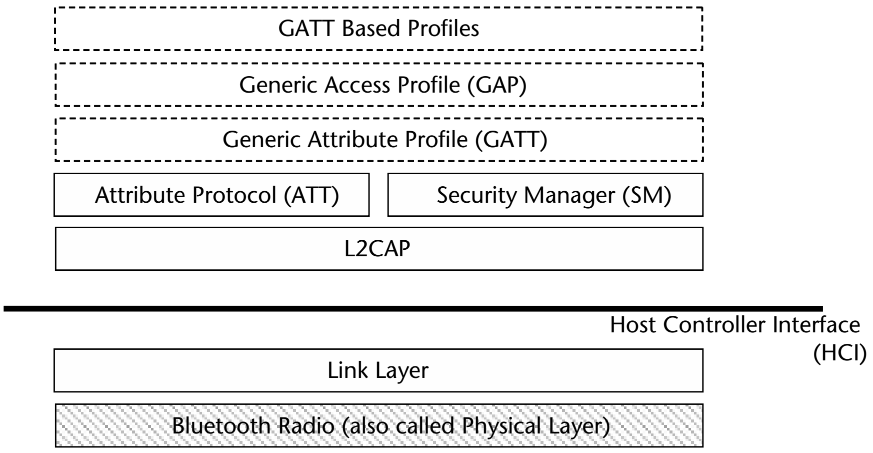

GATT Based Profiles
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Attribute Protocol (ATT)
Security Manager (SM)
L2CAP
Host Controller Interface (HCI)
Link Layer
Bluetooth Radio (also called Physical Layer)

Figure 7.1 Bluetooth Radio in LE protocol stack.

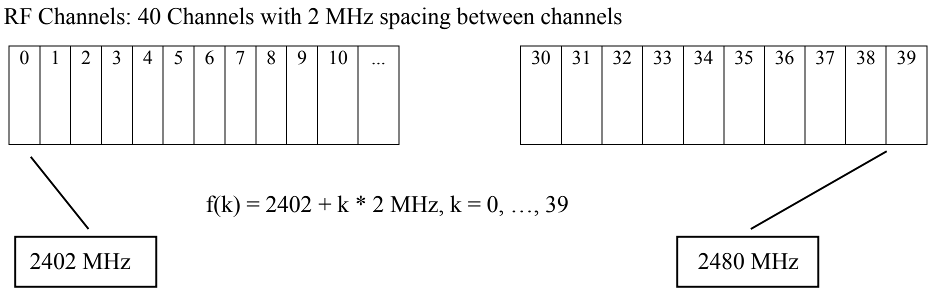

RF Channels: 40 Channels with 2 MHz spacing between channels
0
1
2
3
4
5
6
7
8
9
10
...
30
31
32
33
34
35
36
37
38
39
f(k) = 2402 + k * 2 MHz, k = 0, …, 39
2402 MHz
2480 MHz

Figure 7.2 RF channels used by LE.

1. Data transmissions on channel 0 to 36.
2. Advertisements on channels 37, 38, and 39.

As seen in the figure, the channels are spaced 2 MHz apart.

7.3 Transmitter Only, Receiver Only, or Both

An LE radio may have only a transmitter, only a receiver or both. This is in contrast to BR/EDR specification where it’s mandatory for the BR/EDR radio to have both a transmitter and a receiver. This helps in reducing the cost and simplifying the design of devices which only need to transmit or receive data.

An example of this could be an LE based TV remote control. The remote control may need to only transmit the commands based on which buttons the user presses and not receive anything back from the TV. As per the LE specification, such a device is allowed to have only a transmitter.

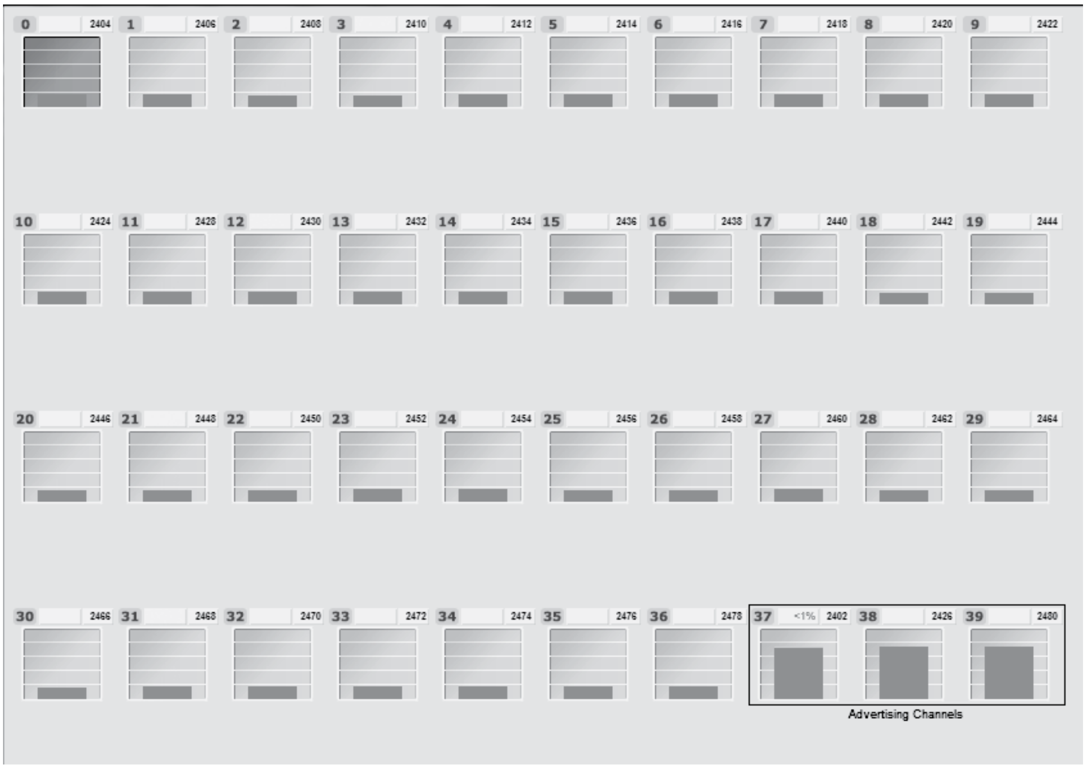


Figure 7.3 Sniffer capture of transmissions on LE channels.

7.4 Output Power

LE defines the transmitter output power to be in the range of 0.01 mW (–20 dBm) to 10 mW (+10 dBm). The device can change the output power dynamically to optimize the power consumption and reduce the interference on other equipment.

7.5 Range

Based on the output power defined in the previous section, LE devices support a range of about 30m to 100m.

7.6 Modulation Characteristics

LE uses Gaussian Frequency Shift Keying (GFSK) mechanism.

The bandwidth-bit period product BT=0.5 and the modulation index is between 0.45 to 0.55. A binary one is represented by a positive frequency deviation and a zero is represented by a negative frequency deviation.

The reference signal is defined in Table 7.1.

Some of these terms are beyond the scope of this book and are provided here just for reference. The details can be looked up in the Bluetooth specifications.

Table 7.1 Reference Signal for LE

Modulation	*GFSK (Gaussian Frequency Shift Keying)*
Modulation Index	0.5 +/– 1%
Bandwidth Bit period Product, BT	0.5 +/– 1%
Bit Rate	1 Mbps +/– 1 ppm
Modulation data for wanted signal	PRBS9
Modulation data for interference signal	PRBS15
Frequency accuracy better than	+/– 1 ppm

What is GFSK modulation?

Modulation is the process of mixing one signal with another signal. The signal that contains the information to be transmitted is called the *modulating signal*. It is mixed with a high-frequency signal called the *carrier signal*. Generally the frequency of the carrier signal is much higher than the modulating signals. For example the carrier signal would be in the range of GHz (2.4 GHz for Bluetooth) and the modulating signal would be in the range of MHz.

The three key properties of the carrier signal are: amplitude, phase, and frequency. Any of these can be modified during the process of mixing with the carrier signal. The resulting modulated signal is then transmitted to the remote side where it is demodulated to reconstruct the original signal.

For example, when a binary number (string of 0s and 1s) is to be transmitted over the air, it is mixed with a carrier signal, and then transmitted over the air. If the 0s and 1s were mixed to the amplitude then one level of amplitude would represent a 0 and another level of amplitude would represent a 1 after modulation. This modulated signal is now in a state that can be transmitted over the air. At the receiver side, the receiver would interpret the different amplitude levels as 0 and 1 and reconstruct the binary number.

Frequency Shift Keying (FSK) conveys information by varying the carrier signal frequency to represent a 0 or a 1. So a binary 1 can be represented by increasing the frequency of the carrier signal and 0 can be represented by decreasing the frequency of the carrier signal. One of the reasons for using frequency modulation to encode data compared to amplitude modulation is that generally the noise signals change the amplitude of a signal. So, modulation signals which ignore the amplitude of the signal are relatively more immune to noise.

Gaussian Frequency Shift Keying (GFSK) applies Gaussian filter to the modulating signal before it is mixed with the carrier signal. The Gaussian filter smoothens the shape of the frequency pulse so that high frequencies at the time of switching can be avoided. This helps to reduce the spectral width of the signal and is also called *pulse shaping*.

7.7 LE Timeline

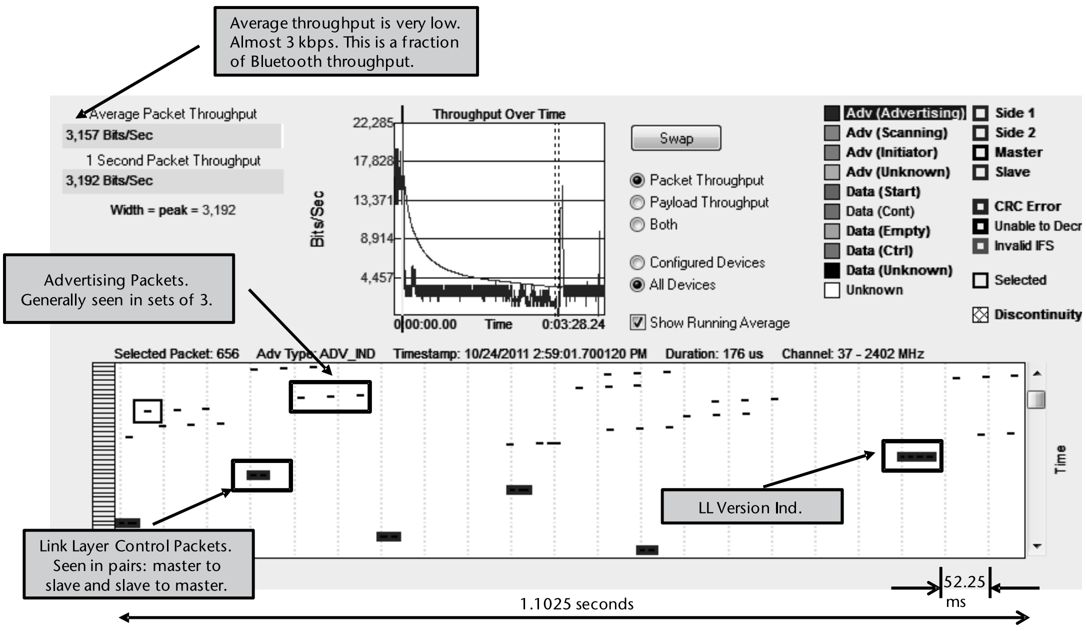
Figure 7.4 shows a typical timeline of transactions happening on the air for LE. The air captures have been taken for one of the GATT profiles where initially the LE device is advertising. Then later on a dual mode device tries to discover the services and creates a connection to this device.

Some of the interesting points to note are:

1. The advertising packets are generally seen in sets of 3 packets. This is because the devices generally advertise consecutively on the three advertisement channels and then wait for some time to restart the advertisement.
2. Most of the space is empty space. This means that for most of the time, there are no transmissions happening. This confirms the statement in the previous chapter that LE is a "mostly off" technology.
3. The average throughput is very low. For this particular capture, this is in the range of 3 kbps. This is much lower than the average throughput seen in classic Bluetooth systems.
4. There could be some instances where lot of data is exchanged. During that time the throughput may go up, but even then it is peaking to a maximum of 17 kbps in this particular example. This is still quite low.

7.8 Summary

This chapter explained the physical layer which is the bottom layer of the LE protocol stack and is responsible for sending and receiving data over the air. Like the BR/


Average Packet Throughput
3,157 Bits/Sec
1 Second Packet Throughput
3,192 Bits/Sec
Width = peak = 3,192

[✓ Packet Throughput] Payload Throughput Both
Configured Devices [✓ All Devices]
[✓ Show Running Average]

Selected Packet: 656 Adv Type: ADV_IND Timestamp: 10/24/2011 2:59:01.700120 PM Duration: 176 us Channel: 37 - 2402 MHz

Figure 7.4 LE timeline.

EDR radio, it operates in the 2.4 GHz ISM band though it uses only 40 channels as compared to 79 channels in the case of Bluetooth.

The next chapter will focus on the link layer which is responsible for controlling, negotiating, and establishing the links. The link layer uses the services of the physical layer to send and receive packets. It provides the packets to the physical layer for transmission on the sender side and processes the received packets from the physical layer on the receiver side.

Bibliography

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

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.

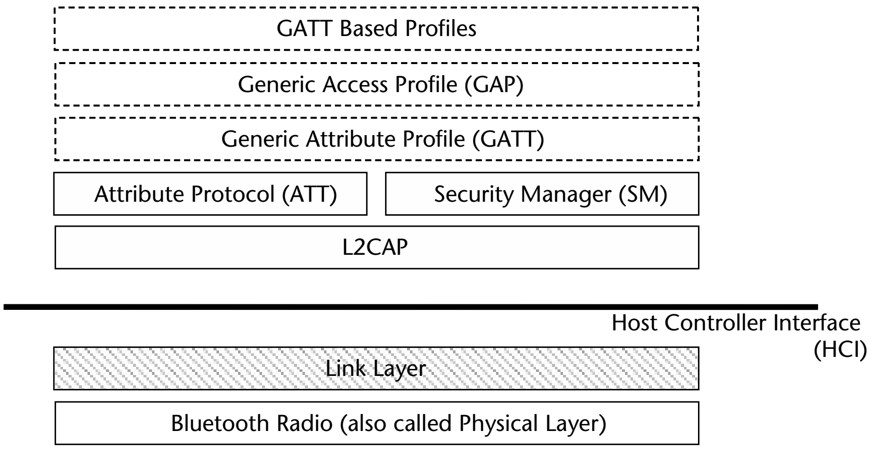

GATT Based Profiles
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Attribute Protocol (ATT)
Security Manager (SM)
L2CAP
Host Controller Interface (HCI)
Link Layer
Bluetooth Radio (also called Physical Layer)

Figure 8.1 Link layer in LE protocol stack.

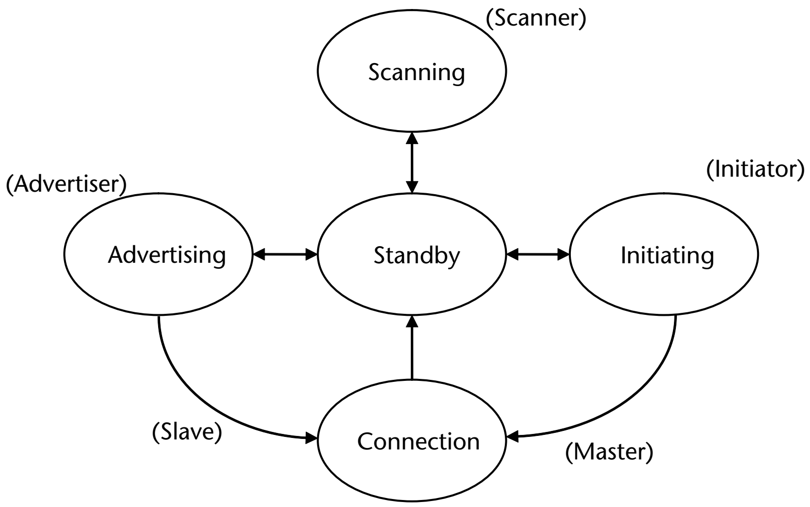


Figure 8.2 Link layer state machine—state diagram and roles.

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.

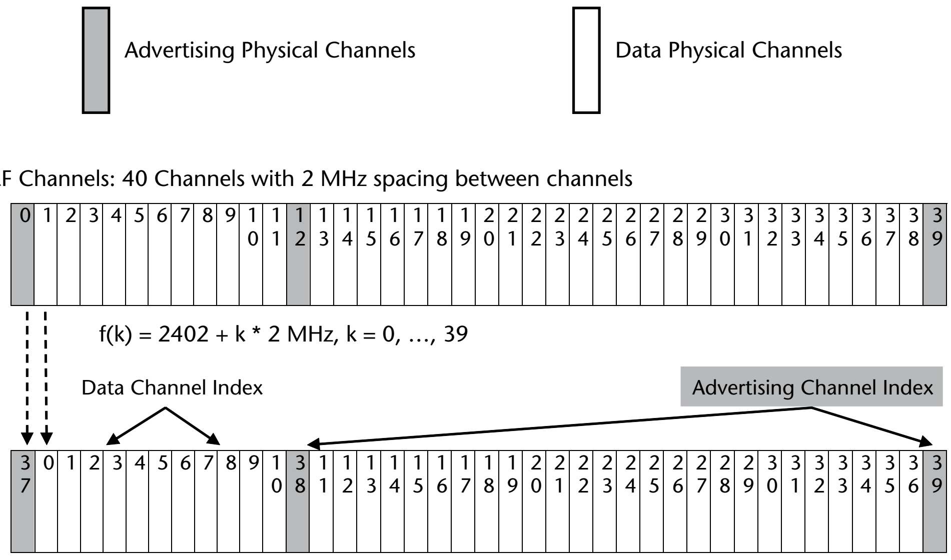


Figure 8.3 Advertising physical channels, data physical channels, and mapping to RF channels.

- 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.

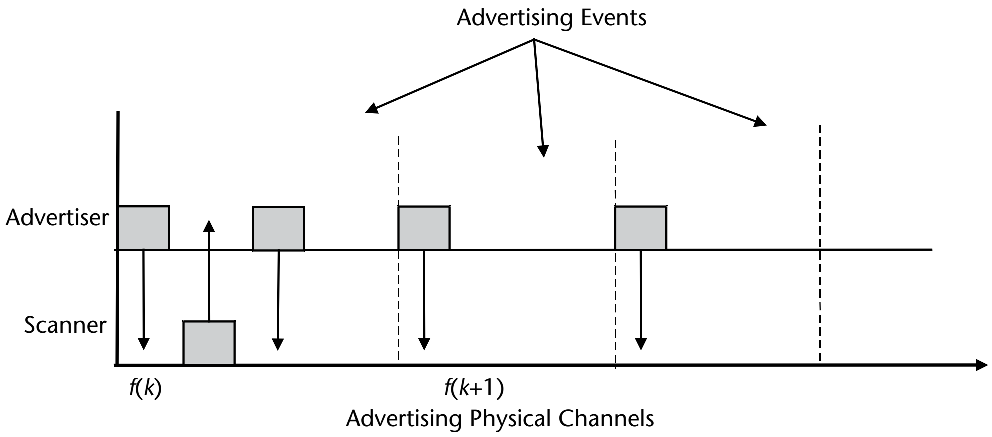
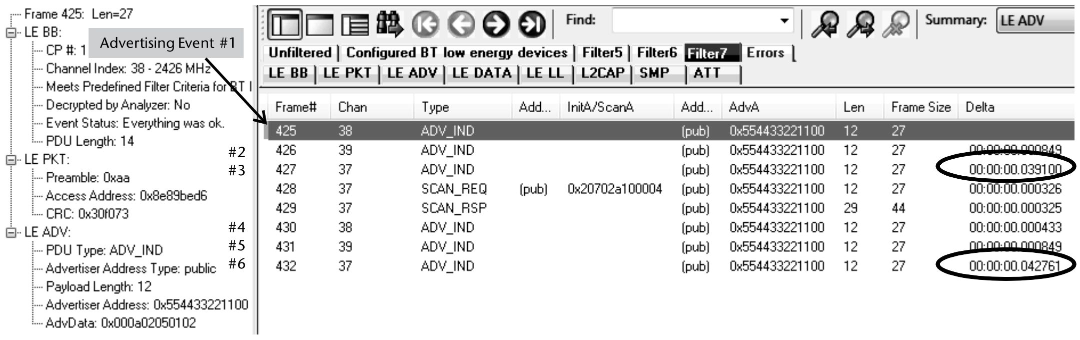


Figure 8.4 Advertising events.


Advertising Event #1
#2
#3
#4
#5
#6
Find:
Summary: LE ADV
Unfiltered | Configured BT low energy devices | Filter5 | Filter6 | Filter7 | Errors
LE BB | LE PKT | LE ADV | LE DATA | LE LL | L2CAP | SMP | ATT

```
Frame 425:  Len=27
LE BB:
    CP #: 1
    Channel Index: 38 - 2426 MHz
    Meets Predefined Filter Criteria for BT l
    Decrypted by Analyzer: No
    Event Status: Everything was ok.
    PDU Length: 14
LE PKT:
    Preamble: 0xaa
    Access Address: 0x8e89bed6
    CRC: 0x30f073
LE ADV:
    PDU Type: ADV_IND
    Advertiser Address Type: public
    Payload Length: 12
    Advertiser Address: 0x554433221100
    AdvData: 0x000a02050102
```

Frame#	Chan	Type	Add...	InitA/ScanA	Add...	AdvA	Len	Frame Size	Delta
425	38	ADV_IND			(pub)	0x554433221100	12	27	
426	39	ADV_IND			(pub)	0x554433221100	12	27	00:00:00.000849
427	37	ADV_IND			(pub)	0x554433221100	12	27	00:00:00.039100
428	37	SCAN_REQ	(pub)	0x20702a100004	(pub)	0x554433221100	12	27	00:00:00.000326
429	37	SCAN_RSP			(pub)	0x554433221100	29	44	00:00:00.000325
430	38	ADV_IND			(pub)	0x554433221100	12	27	00:00:00.000433
431	39	ADV_IND			(pub)	0x554433221100	12	27	00:00:00.000849
432	37	ADV_IND			(pub)	0x554433221100	12	27	00:00:00.042761

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.

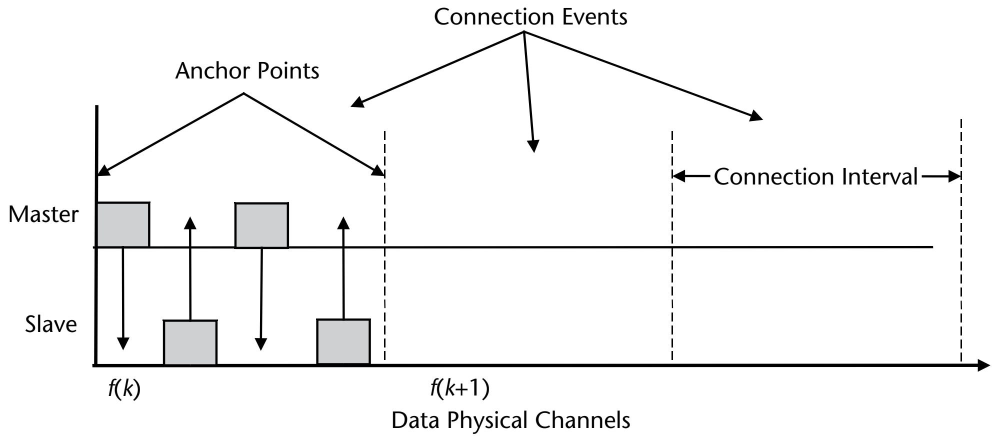


Figure 8.6 Connection events.

- 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/

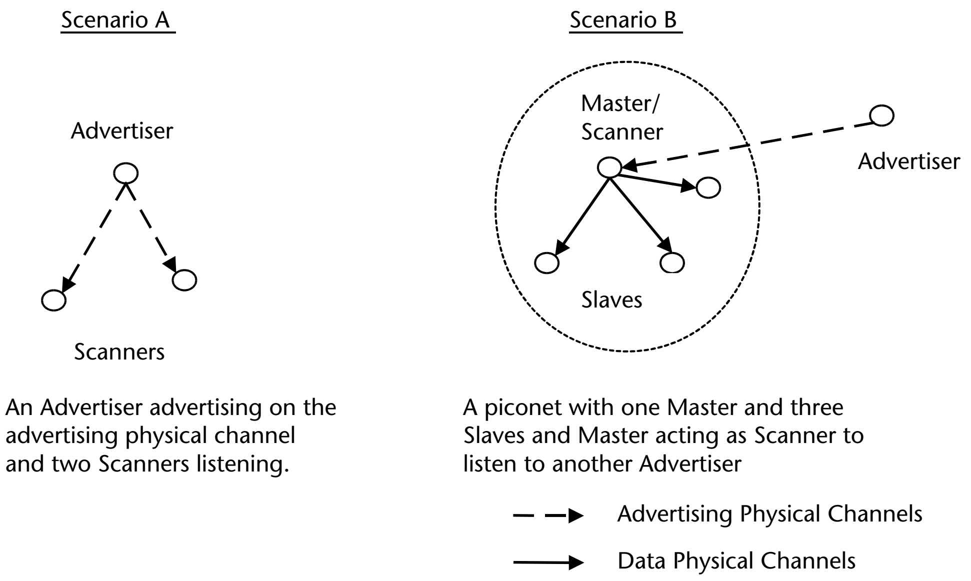

Scenario A
Advertiser
Scanners
An Advertiser advertising on the advertising physical channel and two Scanners listening.
Scenario B
Master/ Scanner
Advertiser
Slaves
A piconet with one Master and three Slaves and Master acting as Scanner to listen to another Advertiser
Advertising Physical Channels
Data Physical Channels

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.

LSB MSB

PREAMBLE (1 octet)	ACCESS ADDRESS (4 octets)	PDU (2 to 39 octets: 4.0) (2 to 257 octets: 4.2)	CRC (3 octets)

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.

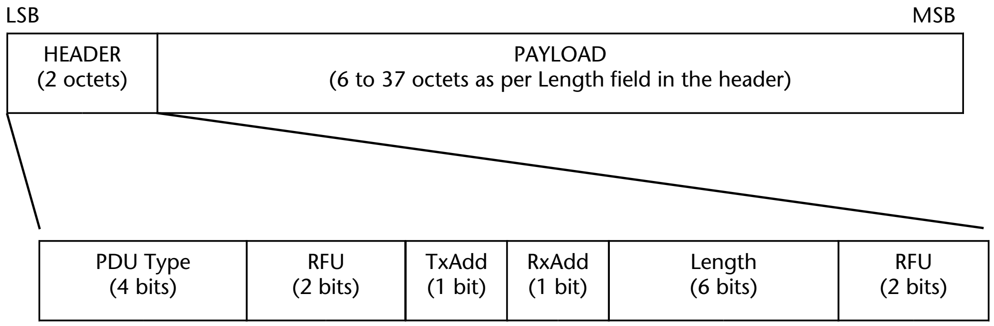


Figure 8.9 Advertising channel PDU.

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:

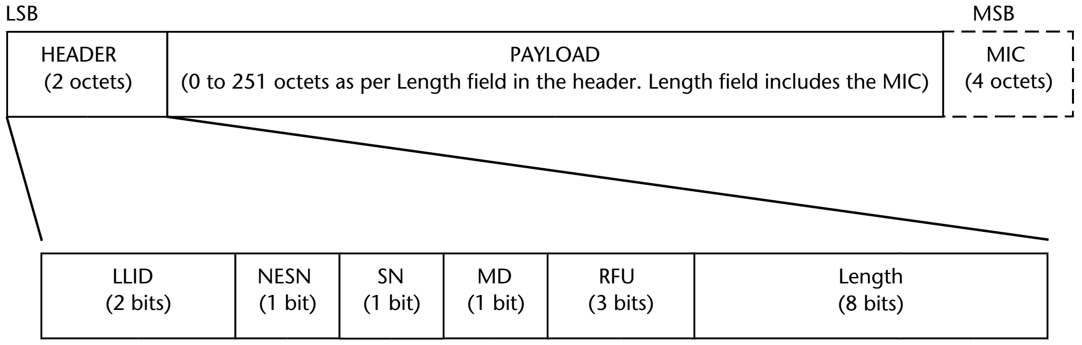


Figure 8.10 Data Channel PDU.

- 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.

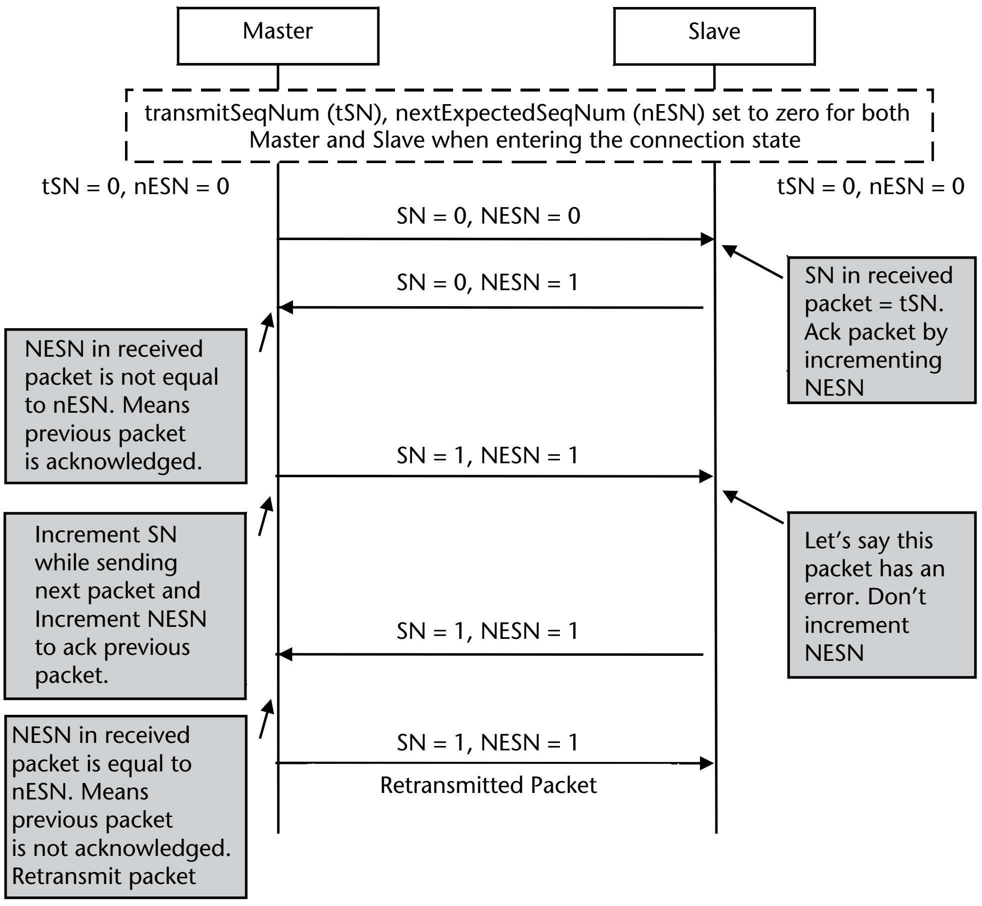

Master
Slave
transmitSeqNum (tSN), nextExpectedSeqNum (nESN) set to zero for both Master and Slave when entering the connection state
tSN = 0, nESN = 0
tSN = 0, nESN = 0
SN = 0, NESN = 0
SN in received packet = tSN. Ack packet by incrementing NESN
SN = 0, NESN = 1
NESN in received packet is not equal to nESN. Means previous packet is acknowledged.
SN = 1, NESN = 1
Let's say this packet has an error. Don't increment NESN
Increment SN while sending next packet and Increment NESN to ack previous packet.
SN = 1, NESN = 1
NESN in received packet is equal to nESN. Means previous packet is not acknowledged. Retransmit packet
SN = 1, NESN = 1
Retransmitted Packet

Figure 8.11 Acknowledgment and flow control using SN and NESN.