Another difference from the Hands-Free profile is that while Hands-Free supports bi-directional transfer of voice data, A2DP supports audio data streaming in only one direction. This is in line with the use cases meant for these profiles. Hands-Free is meant for transfer of voice data where users may be having a conversation on the mobile phone. A2DP is meant for transfer of audio data where a user may be listening to music on the wireless headset.

This profile is dependent on GAP and GAVDP.

A2DP defines the following two roles. These are shown in Figure 4.22:

- Source (SRC): This is the device that acts as the source of the digital audio stream to the SNK.
- Sink (SNK): This is the device that receives the audio stream from the SRC and processes it.

Typical use cases of this profile are:

- Play stereo music from a laptop to speakers. In this case the laptop acts as the SRC and the speakers act as the SNK.
- Play stereo music from mobile phone to the Bluetooth enabled music system in the car. In this case the mobile phone acts as the SRC and the Bluetooth enabled music system in the car acts as a SNK.

A2DP does not define point-to-multipoint distribution of audio (note that such cases are still supported by the Bluetooth technology. This can be done, for example, by creating two A2DP connections and routing the same audio on both the connections).

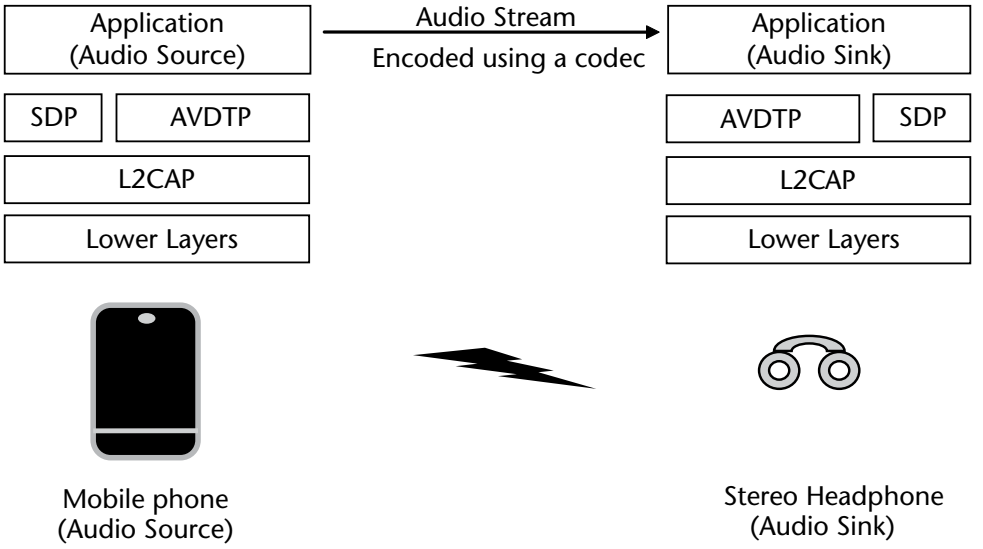

Figure 4.22 Advance audio distribution profile (A2DP).

Since raw streaming of audio data requires lot of bandwidth, A2DP uses on-the-fly encoding and decoding of audio data. There are several codecs that could be used to encode and decode the data.

- Sub-Band Codec (SBC);
- MPEG-1,2 Audio;
- MPEG-2,4 AAC;
- ATRAC family;
- Non-A2DP Codecs: This allows the applications to use their own codecs.

Out of these codecs, supporting SBC is mandatory. All other codecs are optional. SBC is a low complexity codec. It needs less computational power compared to the other codecs and delivers good quality compression of audio samples in real time. It's quite suitable for devices like headphones which may have limited resources like memory and computation power. It supports sampling frequencies from 16 KHz to 48 KHz. 48 KHz sampling frequency is sufficient for CD quality audio.

On the Audio source side, the audio samples coming from the application are encoded with the codec (e.g., SBC) before being given to AVDTP layer for transmission. These encoded samples are sent over the ACL link to the SNK. The AVDTP layer on the SNK side receives those samples. These are decoded and then given to the Audio Sink application. The audio sink application then plays those samples.

4.18 Audio/Video Remote Control Profile (AVRCP)

AVRCP defines the requirements for Bluetooth devices to support use cases related to control of A/V devices. This control can be considered similar to the control provided by a remote control of, let's say, a DVD player.

This profile is dependent on GAP. AVRCP defines the following two roles. These are shown in Figure 4.23:

- Controller (CT): This is the device that initiates a transaction by sending a command to the target.
- Target (TG): This is the device that receives the command, takes the requested action and sends back the response.

Typical use cases of this profile are:

- A headphone sending commands to the mobile phone to pause, play, fast forward, change tracks etc. In this case the headphone acts as the controller and the mobile phone acts as the target.
- A PC sending a command to a DVD player to pause video playback. In this case the PC is the controller and the DVD player is the target.

Typical operations that are carried out by devices that support this profile are:

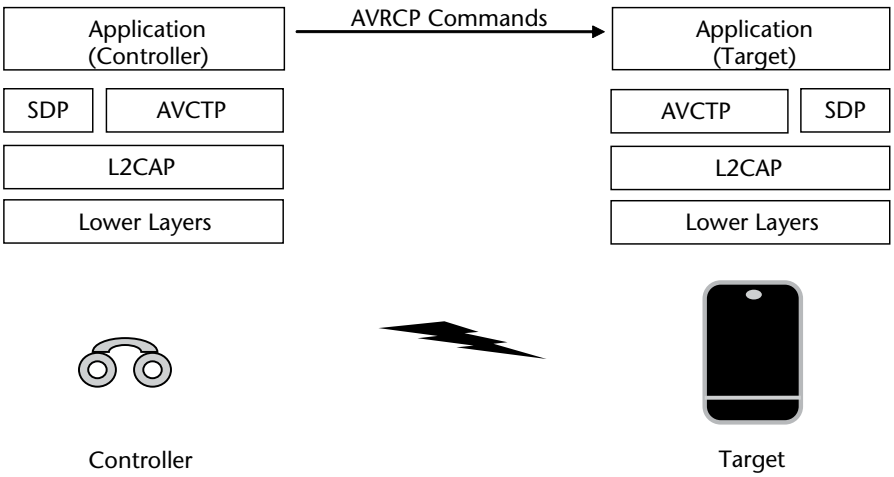

Figure 4.23 Audio/video remote control profile (AVRCP).

- Retrieving information about the type of units and subunits supported by the device (e.g., Player/Recorder, Monitor/Amplifier, Tuner, etc.);
- Volume up;
- Volume down;
- Channel up;
- Channel down;
- Mute;
- Play;
- Stop;
- Pause;
- Rewind;
- Fast forward.

Not all devices support all operations. Rather the operations that are supported by the device depend on the type of the device and the features it supports.

This profile extensively uses the AV/C command set as defined in the 1394 trade association specification (See Bibliography). (Note: This is another good example where Bluetooth borrows from existing specifications instead of writing the specifications from scratch.)

4.19 Summary

This chapter explained the Bluetooth upper layers and profiles. Wherever possible, the Bluetooth protocols and profiles try to reuse implementations that are already available. These are referred to as adopted protocols. The protocols which are defined from scratch by the Bluetooth SIG are referred to as core protocols.

The profiles provide information on how each of the protocol layers comes together to implement a specific usage model. These define how end-to-end

interactions take place between two Bluetooth devices and form the fundamental building block towards ensuring interoperability between devices from various vendors.

The Generic Access Profile (GAP) is a base profile which is mandatory for all devices to implement. A device may implement one or more of the other profiles depending on the end application that the device is intended to support.

Bibliography

Bluetooth Core Specification 4.0 http://www.bluetooth.org.
Bluetooth SIG, Specifications of the Bluetooth System, Profiles http://www.bluetooth.org.
Bluetooth Assigned Numbers, https://www.bluetooth.org/assigned-numbers.
GSM 07.10 version 6.3.0 Release 1997 aka ETSI TS 101 369.
Infrared Data Association, IrDA Object Exchange Protocol (IrOBEX) (http://www.irda.org).
Infrared Data Association, IrMC (Ir Mobile Communications) Specification.
IETF RFC3550 / RFC1889 (obsolete) RTP, A Transport Protocol for Real-Time Applications
3GPP 27.007 v6.8.0. http://www.3gpp.org/ftp/Specs/html-info/27007.htm.
The Internet Mail Consortium, vCard—The Electronic Business Card Exchange Format, Version 2.1, September 1996.
The Internet Mail Consortium, vCalendar—The Electronic Calendaring and Scheduling Exchange Format, Version 1.0, September 1996.
1394 Trade Association, AV/C Digital Interface Command Set—General Specification, Version 4.0, Document No. 1999026 and AV/C Digital Interface Command Set - General Specification, Version 4.1, Document No. 2001012 (http://www.1394ta.org).
1394 Trade Association, AV/C Panel Subunit, Version 1.1, Document No. 2001001 (http://www.1394ta.org).

CHAPTER 5

Getting the Hands Wet

5.1 Introduction

After walking through the concepts of Bluetooth, it's a good time to start doing some practical experiments. This chapter will help you to bring up your own Bluetooth development environment in which you can start writing small scripts and programs. A bit of familiarity with the Linux systems, scripting with the shell and the C programming language is assumed.

This chapter will introduce some of the practical usage of the Bluetooth functionality. This will be extended in further chapters to Bluetooth Low Energy. So it's important that you understand the examples provided here and try out a few of them yourself.

This chapter provides examples based on the BlueZ stack. BlueZ is the "official Linux Bluetooth protocol stack." Support for BlueZ can be found in many Linux distributions. In general it is compatible with any Linux system in the market. It provides support for the core Bluetooth layers and protocols. It is flexible, efficient, and uses a modular implementation. Further details about BlueZ can be found on the BlueZ website (http://www.bluez.org).

5.2 Ingredients

You will need the following:

1. A PC running any flavor of Linux. For the purpose of examples, Ubuntu 12.10 is used here, though any other Linux system will serve the purpose provided it's not too old.
2. A Bluetooth Dongle. If you search for Bluetooth dongle, you will find a lot of vendors offering USB based dongles. Any of those should be good. If your PC (or laptop) has an in-built Bluetooth device, then it should also be fine and you don't need an external dongle.
3. Some off-the-shelf devices that support Bluetooth. For example, mobile phone, mono headset, stereo headset, keyboard, mouse, printer. Depending on the devices that you have, you may be able to run different examples provided later in this chapter.

4. Some off-the-shelf devices that support Bluetooth Low Energy. This is the tricky part since only limited LE devices are available in the market as of writing this book. You will find development kits from some of the vendors which could be useful in developing LE applications. For the purpose of examples, a PTS dongle is used as a Bluetooth Low Energy device. This dongle is available for purchase from the Bluetooth SIG website.

5.3 Basic Bluetooth Operations

Before trying the examples provided in this section, you will need to connect the Bluetooth dongle to the PC and boot up Linux. Some of the examples provided here may also need root privileges.

5.3.1 Enabling and Disabling Bluetooth

The first command to try out is hciconfig. This command is used to configure the Bluetooth devices.

To check whether the Bluetooth dongle is properly connected and initialized, run the hciconfig command. It will give output similar to Figure 5.1. If the dongle is properly connected, then this command should show information like:

- BD_ADDR.
- Class of Device.
- Manufacturer name.
- Packet Types supported.
- Number of ACL buffers and buffer size: The screenshot below shows 10 buffers of 310 bytes each.
- Number of SCO buffers and buffer size: The screenshot below shows 8 buffers of 64 bytes each.

```
# hciconfig -a
hci0: Type: BR/EDR  Bus: USB
      BD Address: 00:1B:DC:05:B5:B3  ACL MTU: 310:10  SCO MTU: 64:8
      UP RUNNING PSCAN
      RX bytes:1127 acl:0 sco:0 events:39 errors:0
      TX bytes:655 acl:0 sco:0 commands:38 errors:0
      Features: 0xff 0xff 0x8f 0x7e 0xd8 0x1f 0x5b 0x87
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH HOLD SNIFF PARK
      Link mode: SLAVE ACCEPT
      Name: 'ubuntu-1'
      Class: 0x6e0100
      Service Classes: Networking, Rendering, Capturing, Audio, Telephony
      Device Class: Computer, Uncategorized
      HCI Version: 4.0 (0x6)  Revision: 0x1d86
      LMP Version: 4.0 (0x6)  Subversion: 0x1d86
      Manufacturer: Cambridge Silicon Radio (10)
```

Figure 5.1 Hciconfig command.

To close the HCI interface, use the command:

```
hciconfig hci0 down
```

To open and initialize the HCI interface, use the command:

```
hciconfig hci0 up
```

Both these commands need root privileges. In these commands, hci0 indicates the hci interface on which the Bluetooth dongle is attached. It's possible that it's attached on hci1 or some other interface instead of hci0. The parameters of these commands will need to be changed accordingly.

5.3.2 Discovering Devices

There are two commands to discover the devices in the vicinity:

```
hcitool scan
hcitool inq
```

hcitool inq performs a Bluetooth inquiry and reports information like BD_ADDR, Clock offset and Class of Device.

hcitool scan performs the Bluetooth inquiry as well as gets the Bluetooth Device names for all the devices that are found during inquiry.

The output of these two commands is shown in Figure 5.2.

5.3.3 Browsing Services

The following command can be used to browse the services of the remote devices:

```
sdptool browse [bdaddr]
```

The command queries the SDP server on the device specified by the BD_ADDR and shows the list of services supported.

A partial output of this command is shown in Figure 5.3.

```
#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


# hcitool scan
Scanning ...
      00:17:83:DC:72:E9   WM_nareshg
      68:ED:43:25:0E:99   BlackBerry 8520
```

Figure 5.2 Discovering devices.