CHAPTER 4

Bluetooth Upper Layers and Profiles

4.1 Introduction

The previous chapter described the lower layers of the Bluetooth protocol stack. This chapter continues explanation of the Bluetooth protocol stack and covers the upper layers of the Bluetooth protocol stack and the profiles.

The detailed Bluetooth architecture was presented in Chapter 2. It is shown again in Figure 4.1 for ease of reference. The upper layers make use of the functionality provided by lower layers to provide more complex functionality like serial port emulation, transferring big chunks of data, streaming music, and synchronizing information. These help the applications to conveniently implement end user scenarios.

One of the design principles of the Bluetooth protocol stack was the reuse of existing protocols wherever possible instead of rewriting everything from scratch. On one hand this helped to easily and quickly build further on existing and proven technologies, on the other hand it also helped in reusing existing applications that were already implemented and available. Protocols such as Object Exchange (OBEX) and RFCOMM were adopted from other standard bodies and are referred to as *adopted protocols*. These are shown by shaded rectangles in Figure 4.1. OBEX was adopted from the IrOBEX protocol which was defined by the Infrared Data Association. So applications that were designed to run on Infrared transport can generally be reused to run on Bluetooth as well. Similarly RFCOMM was adopted from ETSI standard 07.10. It allows legacy serial port applications to be used on top of Bluetooth without much change.

Some protocols were defined from scratch. These were the protocols that provided the core Bluetooth functionality and are called *core protocols*. These are shown by plain rectangles in Figure 4.1. For example the Service Discovery Protocol (SDP) is one of the core protocols that allows for support of discovering the services of the devices in the vicinity. Since Bluetooth is an ad hoc peer to peer protocol, this layer was essential because there is no prebuilt infrastructure to provide such information and devices can come into vicinity at any time. So this protocol was defined to query the services from the device itself instead of the need of a central server to store information about all devices.

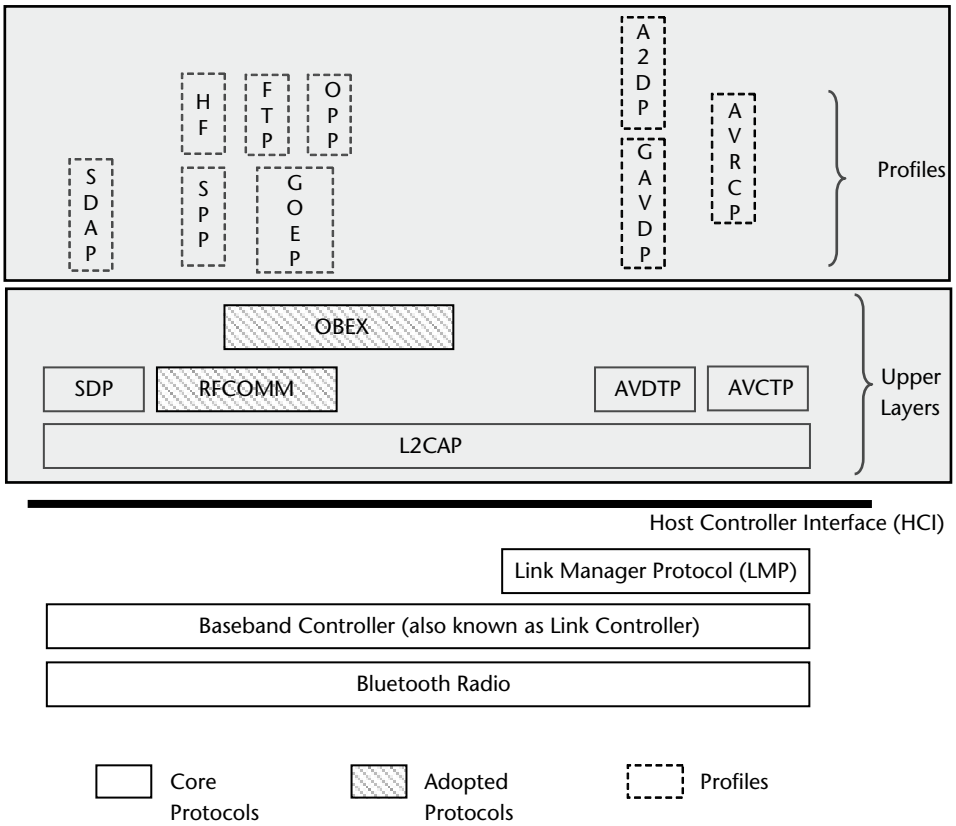

Figure 4.1 Detailed Bluetooth architecture.

Bluetooth profiles provide a usage model of how the different layers of the protocol stack come together to implement a particular usage model. Profiles define the protocols and the features of each of the protocol that are needed to support a particular usage model. For example the File Transfer Profile (FTP) profile defines how and what features of the underlying protocols like OBEX, RFCOMM and L2CAP are needed in order to provide support for transferring files. Profiles are shown by dotted rectangles in Figure 4.1.

4.2 Logical Link Control and Adaptation Protocol (L2CAP)

The L2CAP protocol sits above the Baseband layer and provides data services to the upper layer protocols. It uses the ACL links to transfer packets and allows the protocols above it to send data packets up to 64 KB in length.

L2CAP is based on the concept of Channels. A channel represents a data flow path between L2CAP entities in remote devices. Channels may be connection-oriented or connectionless.

Connection-oriented L2CAP channels are used to transport point-to-point data between two devices. These channels provide a context within which specific properties may be applied to data transported on these channels. For example QoS

parameters may be applied to the connection-oriented channels. These channels are setup using the L2CAP_CONNECTION_REQUEST command before any data can be transferred. Once data transfer is completed, these channels are disconnected using the L2CAP_DISCONNECT_REQUEST command.

Connectionless L2CAP channels are generally used for broadcasting data though they may also be used for transporting unicast data. The Master uses these channels to broadcast the data to all Slaves in the piconet. These channels do not need separate procedures to setup and disconnect the channel. So latency incurred during the channel setup is removed.

Each endpoint of an L2CAP Channel is referred to as a Channel Identifier (CID). So CID is the local name representing a logical channel end point on the device. CIDs are assigned from 0x0001 to 0xFFFF. Out of these, CIDs from 0x0001 to 0x003F are reserved. These are known as fixed channels. The CID 0x0001 is reserved as the L2CAP signaling channel and 0x0005 is reserved as the L2CAP LE signaling channel. CID assignment is specific to the device and is done independent of the other devices. So it's possible, for example, to have one CID number assigned on one device and a different CID number assigned on the other device by the L2CAP entities executing on the respective devices.

The fixed channels (0x0001 for BR/EDR and additionally 0x0005 for LE) are available as soon as the ACL-U logical link is established with the remote device. The L2CAP Signaling Channel is used for negotiating configuration parameters and setting up the other channels.

The allocation of CID numbers is shown in Table 4.1.

Table 4.1 CID Name Space

CID	*Description*
0x0000	Null Identifier. Usage is not allowed.
0x0001	L2CAP Signaling channel. Used to send the signaling commands in the form of requests and responses. Some examples of signaling commands are: • Connection Request • Connection Response • Configuration Request • Configuration Response • Disconnection Request • Disconnection Response
0x0002	Connectionless channel. Used for the following: • Broadcast from Master to all Slaves in the piconet. There is no acknowledgement or retransmission of this data. • Unicast transmission from either a Master or Slave to a single remote device.
0x0003	AMP Manager Protocol. This is used for BT 3.0 + HS operations.
0x0004	Attribute Protocol. Attribute Protocol will be covered in detail in later chapters.
0x0005	LE L2CAP Signaling Channel.
0x0006	Security Manager Protocol (SMP): SMP will be covered in detail in later chapters.
0x0007 – 0x003E	Reserved for future use
0x0040 – 0xFFFF	Dynamically allocated by the L2CAP layer.

4.2.1 Modes of Operation

There are six modes of operation for L2CAP. The mode of operation is selected independently for each channel.

1. *Basic L2CAP Mode:* This is the default mode. The header contains only minimal information including the length of the packet and the channel ID.
2. *Flow Control Mode:* In flow control mode no retransmissions are done. The missing PDUs are detected and reported as lost.
3. *Retransmission Mode:* In retransmission mode a timer is used to ensure that all PDUs are delivered to the peer. The PDUs are retransmitted if they are not ack'ed by the remote side.
4. *Enhanced Retransmission Mode:* The enhanced retransmission mode is similar to the retransmission mode and adds some enhancements to it. For example it adds a POLL bit to request a response from the remote L2CAP layer.
5. *Streaming Mode:* The streaming mode can be used for all streaming applications. In this mode, the PDUs from the transmitting side are numbered but are not acknowledged by the receiver. The numbering of PDUs ensures that they are processed in the correct sequence on the receiving side. A flush timeout is used so that if the PDUs are not sent within that timeout, they are flushed.
6. *LE Credit-Based Flow Control Mode:* This mode was introduced in specifications 4.1. Support for connection-oriented channels for LE was introduced in this version. At the time of connection establishment, each side provides the number of credits that are available. The number of credits indicates the number of LE-frames that the device is capable of accepting. The remote side can send as many LE-frames as the number of credits it has received; if the credits become zero, it stops sending packets. As and when the packets are processed on the receiver side and buffer space becomes available, more credits are given to the transmitter side so that it can send more packets.

4.2.2 L2CAP PDUs

The L2CAP Protocol Data Unit (PDU) is the term used to refer to the packets that are sent and received the L2CAP layer. The PDUs contain control information or data.

L2CAP defines 5 types of PDUs:

1. *B-frame (Basic Frame):* A B-frame is a PDU used in basic L2CAP mode for L2CAP data packets.
2. *I-frames (Information Frame):* An I-frame is a PDU used in enhanced retransmission mode, streaming mode, retransmission mode and flow control mode. It contains additional information encapsulated in the L2CAP header.

3. *S-frame (Supervisory Frame):* An S-frame is a PDU used in Enhanced retransmission mode, retransmission mode and flow control mode. It contains protocol information only and no data.
4. *C-frame (Control Frame):* A C-frame is a PDU that contains L2CAP signaling messages that are exchanged between peer L2CAP entities. This frame is exchanged on the L2CAP signaling channel.
5. *G-frame (Group Frame):* A G-frame is used on the connectionless L2CAP channel. It may be used to broadcast data to multiple Slaves or unicast data to a single remote device.

4.2.3 L2CAP Features

L2CAP performs the following major functions:

- Higher layer Protocol Multiplexing and Channel Multiplexing;
- Segmentation and Reassembly (SAR);
- Per Channel Flow Control;
- Error Control and Retransmissions;
- Streaming Channels;
- Quality of Service;
- Group Management.

L2CAP uses BR/EDR Controller, LE Controller or a Dual mode controller for transporting data packets. It can also use AMP controllers if they exist.

4.2.3.1 Higher layer Protocol Multiplexing and Channel Multiplexing

L2CAP permits several higher layer protocols to share the same ACL links. Each higher layer protocol is assigned a separate CID to transfer data. For example once an L2CAP channel is established with the remote device it may be shared by SDP, RFCOMM and other protocols. Each of these protocols will use a different CID to talk to the respect peer entity on the remote device. This is shown in Figure 4.2.

The Protocol/Service Multiplexer (PSM) field is used during L2CAP connection establishment to identify the higher level protocol that is making a connection on that particular channel. The PSM values are predefined for many of the protocols by the Bluetooth SIG and can be referred to on the Assigned Numbers page on the Bluetooth SIG website. Some of the PSM values for the protocols are show in Table 4.2.

L2CAP also allows the channel to be operated over different controllers though the channel can be active on only one controller at a time. This is useful in scenarios where the channel is initially established over BR/EDR controller and then it's moved to an AMP controller to achieve higher data throughput.

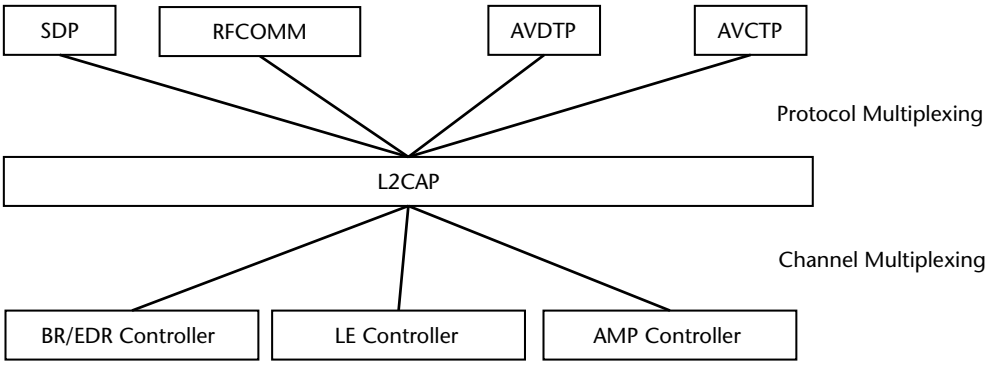

Figure 4.2 L2CAP: protocol/channel multiplexing.

Table 4.2 Assigned Protocol/Service Multiplexer Values (PSM)

Protocol	*PSM*	*Remarks*
SDP	0x0001	Service Discovery Protocol
RFCOMM	0x0003	RFCOMM Protocol
TCS-BIN	0x0005	Telephony Control Specification, TCS Binary Protocol
TCS-BIN-CORDLESS	0x0007	Telephony Control Specification, TCS Binary Protocol
BNEP	0x000F	Bluetooth Network Encapsulation Protocol
HID_Control	0x0011	Human Interface Device Profile
HID_Interrupt	0x0013	Human Interface Device Profile
UPnP	0x0015	
AVCTP	0x0017	Audio/Video Control Transport Protocol
AVDTP	0x0019	Audio/Video Distribution Transport Protocol
UDI_C-Plane	0x001D	Unrestricted Digital Information Profile.

4.2.3.2 Segmentation and Reassembly (SAR)

L2CAP allows higher layer protocols to transmit and receive data packets up to 64 kilobytes. While transmitting, L2CAP breaks the packets into smaller packets depending on the packet size supported by the controller. On the receiving end, L2CAP receives all these segments and reassembles them to form the complete packet. This complete packet is then sent to the higher layers.

As an example, let's say RFCOMM sends a packet of 60 KB to L2CAP. L2CAP will break this packet into smaller chunks of 1021 bytes which can then be transferred over a BR/EDR controller (Note that this is the maximum packet size that can be transmitted over ACL using 3-DH5 packets. A BR/EDR controller may support this as the maximum size or a smaller size). This will result in 60 chunks of 1021 byte packets and one chunk of 180 bytes.

$$60\ \text{KB} = 61440\ \text{Bytes} = 60 * 1021 + 1 * 180$$

All these 61 chunks will be reassembled by the L2CAP on the receiving side and the original 60 KB RFCOMM packet will be recreated. This packet will then be given to the RFCOMM of the remote side.

(Note that the above example is a bit simplified. In practice, L2CAP will also prefix its own 4 byte header to the RFCOMM packet. So it will actually be segmenting and reassembling a packet of 61444 bytes. (61440+ 4 byte L2CAP header).)

Segmentation and Reassembly of packets provides the following advantages:

1. The higher layer protocol doesn't need to care about the size of packets that can be transmitted over the ACL link. The L2CAP layers of the peer devices negotiate a mutually suitable MTU size and use it for the data transfer. This makes the design of the higher layer protocol simpler.
2. Since the packet is split into smaller chunks, if there is an error in transmission of one of the chunks then only that chunk will be retransmitted. The whole packet doesn't need to be retransmitted.
3. L2CAP can interleave packets of different higher layer protocols. This ensures that if one of the higher layer protocols is sending a big packet, the other protocols don't get starved for bandwidth.
4. In general, the controllers have limited buffer space for keeping transmit and receive packets while the host may have much larger buffer space. So L2CAP allows the upper layer protocols to send bigger data packets even though the controller may support much smaller packet sizes.

The MTU (Maximum Transmission Unit) is specified by each device independently and is not negotiated between the two L2CAP entities. This means, for example, if a mobile phone is connected to the headset then the mobile phone may specify a higher MTU size while the headset may specify a smaller MTU size. The mobile phone will always send packets which are smaller than or equal to in length to the MTU size of the headset.

The MTU parameter is very significant in cases where speed of data transfer is important. Lower MTU values will result in the need of segmenting a big chunk into more number of smaller packets. This will decrease the overall speed of data transfer.

4.2.3.3 Per Channel Flow Control

Flow control mechanisms are already in place for the data that is transferred on the HCI interface but that is at an aggregate level to control the data that is flowing from the host to the controller and on the air from one controller to another. L2CAP extends this mechanism to provide individual flow control to each of the channels that are multiplexed on top of it. This means that the data for the RFCOMM layer may be stopped while the data for the SDP layer may still be allowed to flow.

4.2.3.4 Error Control and Retransmissions

The first level of error control is done by the baseband layer. If the received packet has an error, then it is not passed on to the host. L2CAP provides an additional level of error control that detects any erroneous packets that are not detected by

the baseband. L2CAP retransmits packets if they are not received correctly on the remote side. Besides this, it's possible that some of the packets are dropped by the time they reach the host entity on the remote side. This could be, for example, if the packet had an error and was dropped by the baseband layer. The L2CAP layer retransmits these packets so that the layers above it receive all the packets in the correct sequence without any errors.

L2CAP uses a Transmit/Acknowledge mechanism. Each packet that is transmitted is acknowledged by the remote device. If the acknowledgement is not received, then L2CAP may decide to either retransmit that packet or to drop it. Whether packets on a certain channel are to be retransmitted or not is specified using the Flush Timeout option. There are three possible values:

1. No retransmissions at the baseband level.
2. Use a specified flush timeout and continue retransmitting till the time the timeout expires.
3. An infinite amount of retransmissions. In this case, the baseband continues retransmissions until the physical link is lost.

4.2.3.5 Streaming Channels

Streaming channels are useful in scenarios where data with a specific data rate (like audio) is being transferred. The audio applications can setup an L2CAP channel with the specific data rate. A flush timeout is used if L2CAP is not able to transfer packets within the correct time period to comply with that data rate. This ensures that packets don't get queued up indefinitely if, for example, the link quality deteriorates. Since audio packets are real time in nature, it's better to drop a delayed packet and transmit the next packet than to continue trying to retransmit the delayed packet.

4.2.3.6 Quality of Service

Isochronous data has time constraints associated with it. The information has a time bound relation with the previous and successive entities. Audio is a good example of isochronous data. For such a data, the lifetime is limited after which the data becomes invalid and there is no point in delivering that data anymore.

L2CAP can support both isochronous (Guaranteed bandwidth) and asynchronous (Best Effort) data flows over the same ACL logical link. This is done by marking the isochronous packets as automatically flushable and asynchronous packets as nonflushable in the Packet_Boundary_Flag in HCI ACL data packets. This flag was explained in detail in the HCI section in previous chapter. The automatically flushable packets will be automatically flushed by the controller if they are not transmitted within the time window of the flush timeout set for the ACL link.

4.3.3 L2CAP Signaling

The Signaling commands are used between L2CAP entities on peer devices for operations like setting up the connection, configuring the connection, and disconnection. The fixed channels (0x0001 and additionally 0x0005 for LE) are available as soon

as the ACL-U logical link is established with the remote device. The L2CAP Signaling Channel is used for negotiating configuration parameters and setting up the other channels. The L2CAP commands are encapsulated within C-Frames (control frames).

The format of C-Frames is shown in Figure 4.3. The Channel ID is 0x0001 for BR/EDR signaling and 0x0005 for LE signaling. The Code field identifies the type of command. The Identifier field is used to match responses with the requests.

The various L2CAP signaling packets are shown in Table 4.3.

Out of the commands mentioned in Table 4.3, Connection Parameter Update Request and Response are used only for LE. Command Reject can be used for both LE and BR/EDR. These will be explained in detail in Chapter 10.

An example of various L2CAP Signaling PDUs is shown in Figure 4.4. This figure shows an air sniffer capture of L2CAP signaling packets when RFCOMM uses the services of L2CAP.

Some of the points worth noting are:

1. The Signaling packets are being exchanged on Signaling channel (CID 0x0001). See Frames #22–#29.
2. The RFCOMM data packets are exchanged on CID 0x0040. This is the CID allocated to RFCOMM.
3. During the Connection Request, the master provided the following information:
   a. Source Channel ID = 0x0040: The CID that the master will be using.
   b. PSM = RFCOMM: To indicate that L2CAP is creating a channel for RFCOMM to use.
4. The configure request and configure response packets are exchanged between the Master and Slave in both directions to negotiate the connection parameters.
5. Once the data transfer is done, the Master sends a disconnection request on the Signaling channel.

4.4 Service Discovery Protocol (SDP)

Bluetooth provides support for ad hoc connections. This means devices can dynamically discover each other and then decide to connect to each other. Before

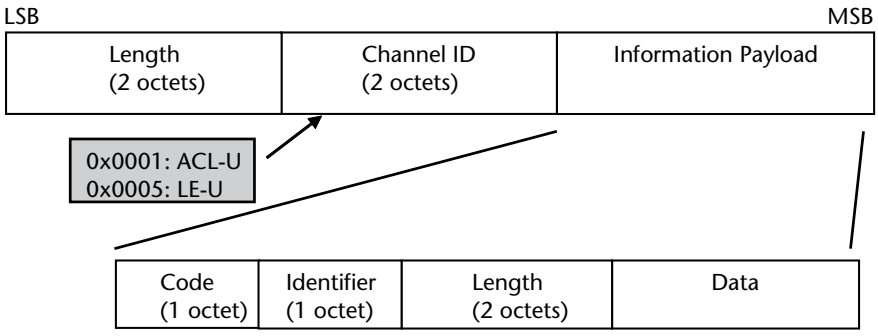

Figure 4.3 Format of L2CAP signaling PDUs (C-Frames).

Table 4.3 L2CAP Signaling Packets

Code	*Signaling Packet*	*Purpose*
0x00	Reserved	
0x01	Command Reject	This is sent in response to a command packet that contains an unknown command or when sending the corresponding response is inappropriate.
0x02	Connection Request	This is used to create an L2CAP channel between two devices.
0x03	Connection Response	This is sent in response to a Connection Request.
0x04	Configure Request	This is used to negotiate configuration parameters of the connection. These include parameters like MTU, Flush Timeout, QoS, etc.
0x05	Configure Response	This is sent in response to Configure Request.
0x06	Disconnection Request	This is used to terminate an L2CAP channel.
0x07	Disconnection Response	This is sent in response to Disconnection Request.
0x08	Echo Request	This is used to request a response from the remote L2CAP entity. This request is generally used to check the status of the link.
0x09	Echo Response	This is sent in response to Echo Request.
0x0A	Information Request	This is used to request implementation specific information from the remote L2CAP entity.
0x0B	Information Response	This is sent in response to Information Request.
0x0C	Create Channel Request	This is used to create an L2CAP channel between two devices over the specific controller. This is used when BT 3.0 + HS is supported and an alternate controller is used.
0x0D	Create Channel Response	This is sent in response to Create Channel Request.
0x0E	Move Channel Request	These are used to move an existing L2CAP channel from one controller to another. These are used when BT 3.0 + HS is supported.
0x0F	Move Channel Response	
0x010	Move Channel Confirmation	
0x11	Move Channel Confirmation Response	
0x12	Connection Parameter Update Request	This command is sent from an LE Slave to an LE Master to update the connection parameters. This will be described in further detail in Chapter 10.
0x13	Connection Parameter Update Response	This is sent in response to Connection Parameter Update Request.

making the decision to connect to each other, one device may need to "discover" details about the other device. These details include which services are available and what are the characteristics of those services. Service discovery protocol provides a mechanism to discover the services provided by the remote devices and the characteristics of those services.

As an example, let's say a laptop needs to play an audio file on Bluetooth wireless speakers. It will first do an inquiry to find out the devices in the Bluetooth vicinity. Once this is done, it will connect to these devices and search for the services provided by those devices. In order to play an audio file, it will search for a device that has the services registered for A2DP. Once it finds such a device, it may create an A2DP connection with that device to play the file.

(Note: In practice the laptop need not connect to all the devices to discover the A2DP service. It can already narrow down its search based on the Class of Device (CoD) information provided by that device during inquiry. It needs to only discover services on the devices of the Audio/Video Major class.)

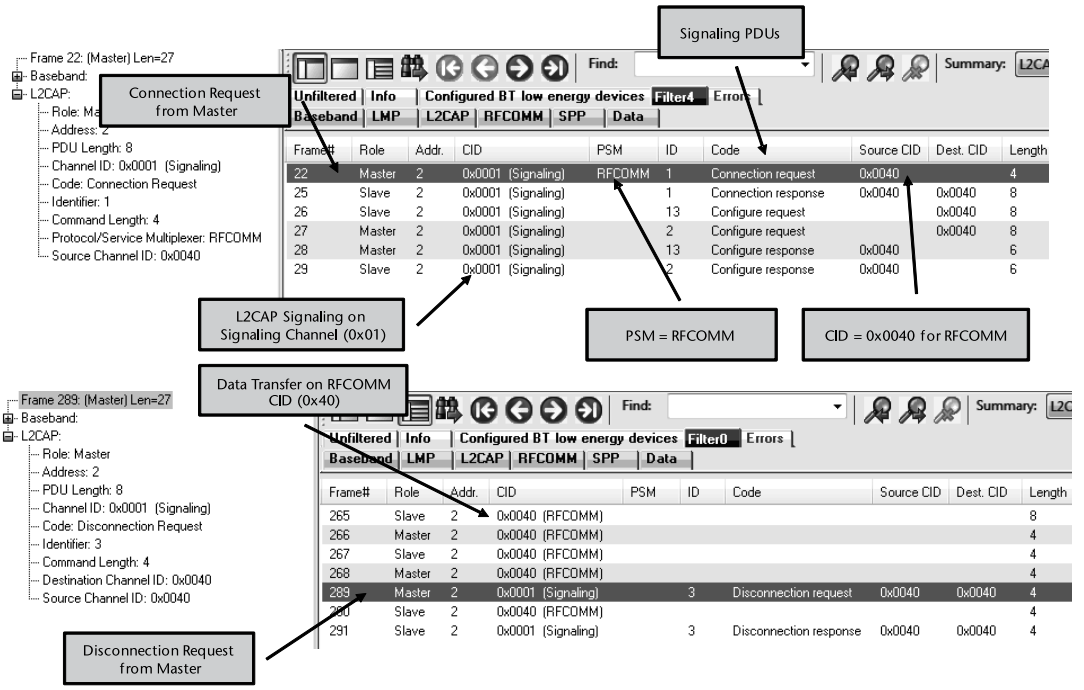

Figure 4.4 Air sniffer capture of L2CAP signal packets.

SDP focuses primarily on providing a uniform method for discovering services available on the Bluetooth devices. It does not define the method to access those services once they are discovered. This is done by the other layers depending on the type of service. For example if a device provides a printing service, then the printing related profiles will make use of that service to print documents.

SDP follows a client server model. The services supported by a device are registered with an SDP server. The server maintains a list of these services in the form of service records. The SDP client queries for these services.

This is illustrated in Figure 4.5.

Only one SDP server is permitted per Bluetooth device. If the device does not need to provide any services to other devices, it can act as a client only device. In such cases it does not need to implement an SDP server.

4.4.1 Service Record, Service Attributes and Service Class

All the server applications register their services with the SDP server in the form of Service Records. A service is any entity that can provide information, perform an action, or control a resource on behalf of another entity. It may be implemented as software, hardware or a combination of both. For example a printer could provide the service of color printing; a smartphone could provide the service of a dial-up-networking gateway.

A service record contains all information that an SDP server wants to provide about the service to the SDP clients. This is in the form of a list of Service Attributes. Each Service Attribute describes a single characteristic of the service and is the form of an Attribute ID and Attribute Value. An Attribute ID is a 16-bit

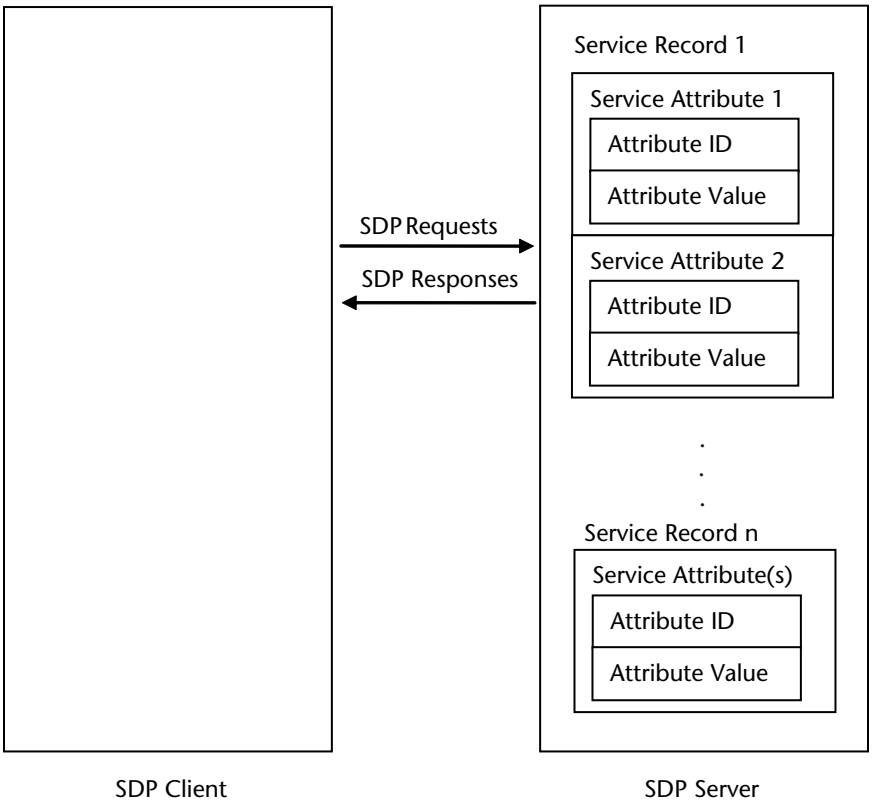

Figure 4.5 SDP Client and Server.

unsigned integer that distinguishes each service attribute from other service attributes within a service record. An Attribute Value is a variable length field whose meaning is determined by the attribute ID associated with it and by the service class of the service record in which the attribute is contained.

Each service is an instance of a service class. The service class definition provides the definitions of all service records that can be present in the service. Each service class is assigned a unique identifier called the service identifier. It is represented as a Universally Unique Identifier (UUID).

What is UUID?

A UUID is a universally unique identifier that is guaranteed to be unique across all space and time. UUIDs can be independently created in a distributed fashion. No central registry of assigning UUIDs is required.

A UUID is a 128-bit value. To reduce the burden of storing and transferring 128-bit values, a range of UUIDs has been pre-defined along with 16-bit or 32-bit aliases. A 16-bit or 32-bit UUID may be converted into 128-bit UUID by pre-defined formulas.

Table 4.4 provides a list of commonly used service attributes supported by SDP. Out of these, the first two attributes viz ServiceRecordHandle and ServiceClassIDList are mandatory to exist in every service record instance. The remaining service attributes are optional. A sample of some of these service attributes is shown in Figure 4.6.

4.4.2 Searching and Browsing Services

There are two ways for an SDP client to get the services from the SDP server:

- Search for Services.
- Browse for Services.

Using the Service Search transaction, the client can search for service records for some desired characteristics. These characteristics are in the form of Attribute Values. The search is in the form of a list of UUIDs to be searched. A service search pattern is formed by the SDP client which is a list of all UUIDs to search. This service search pattern is sent to the SDP server in the service search request. This pattern is matched on the SDP server side if all the UUIDs in the list are contained in any specific service record. A handle to this service record is returned if a match is found.

Another technique to fetch the list of the service records on the server is by Browsing for Services. In this case the client discovers all the services of the server instead of any specific ones. This is useful when the client does not have previous knowledge of the type of services which the servers supports, so it's difficult to form a UUID pattern or if the client is interested in getting the complete list of services supported. An example of browsing the services of a smartphone using the BlueZ stack on Linux is shown in Figure 4.6.

Table 4.4 Commonly Used Service Attributes

Attribute Name	*Description*
ServiceRecordHandle	It uniquely identifies each service record within an SDP server. It is used to reference the service by the clients.
ServiceClassIDList	A list of UUIDs representing the service classes that this service record conforms to. For example Headset Audio Gateway, Dial Up Networking.
ServiceRecordState	This is used for caching of service attributes. Its value is changed when any other attribute value is added, deleted, or changed in the service record. The clients can read this value to check if any values have changed in the service record since the last time they read it.
ServiceID	A UUID that uniquely identifies the service instance described by the service record.
ProtocolDescriptorList	A list of UUIDs for protocol layers that can be used to access the service. For example: L2CAP, RFCOMM.
BluetoothProfileDescriptorList	A list of elements to provide information about the Bluetooth profiles to which this service conforms to.
ServiceName	A string containing name of the service.
ServiceDescription	A string containing a brief description of the service
ProviderName	A string containing the name of the person or organization providing this service.

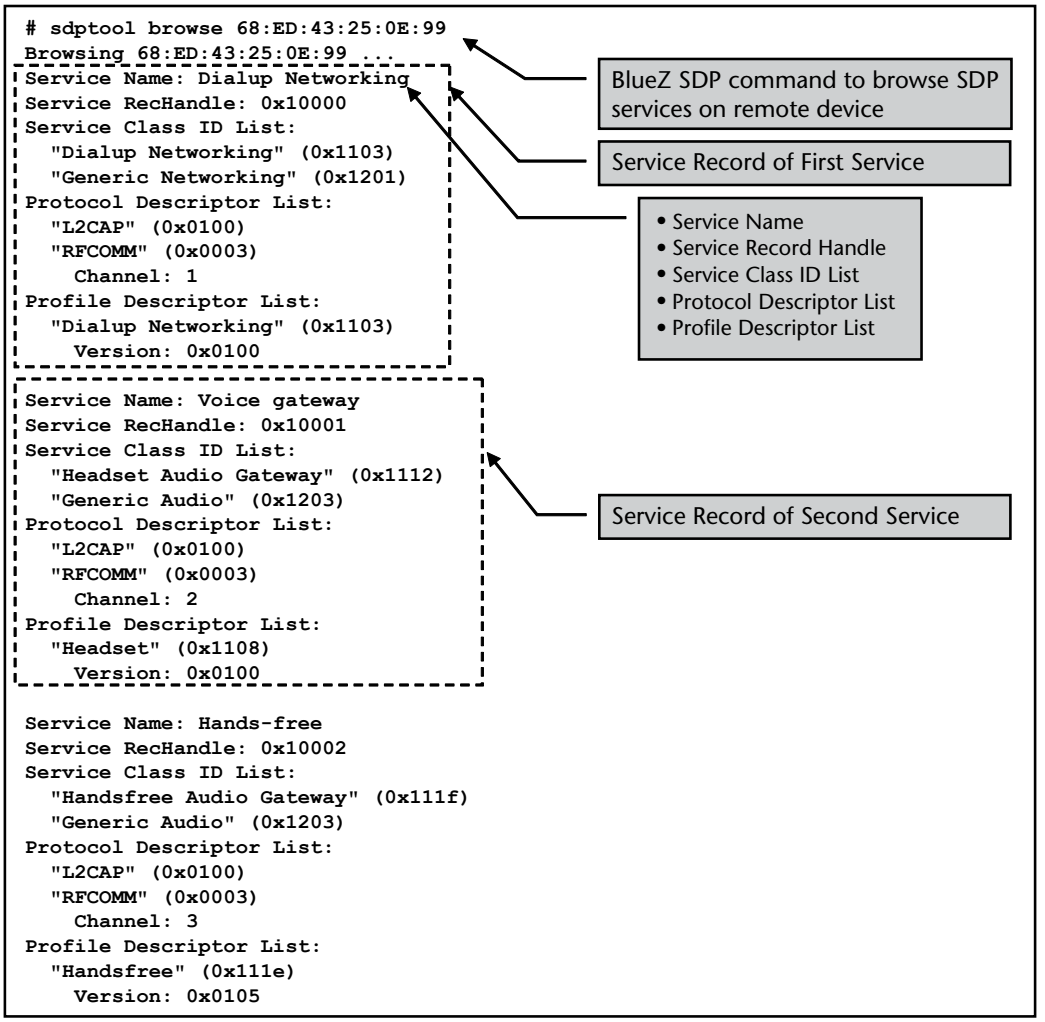

Figure 4.6 Example of browsing SDP services of a smartphone from BlueZ stack running on Linux.

4.4.3 SDP Transactions

SDP is a simple protocol with minimal requirements on the underlying transport. It uses a request/response model. Each transaction comprises one request PDU and one response PDU. The client sends one request and then waits for a response before sending the next request. So, only one request can be pending at any time. This makes the design of the client and server quite simple.

The different transactions that are supported by SDP are described in Table 4.5.

An example of the SDP transactions between a mobile phone and an A2DP headset is shown in Figure 4.7 and Figure 4.8.

Figure 4.7 shows the SDP_ServiceSearchAttributeRequest where the mobile phone queries the A2DP headset to check if the following services exist:

- Audio Sink;

Table 4.5 SDP Transactions

PDU ID	*Transaction*	*Description*
0x00	Reserved	—
0x01	SDP_ErrorResponse	The SDP server sends this PDU if an error occurred and it cannot send the correct response PDU. This could be the case, for example, if the request had incorrect parameters.
0x02	SDP_ServiceSearchRequest	This PDU is sent by the SDP client to locate service records that match a service search pattern.
0x03	SDP_ServiceSearchResponse	Upon receipt of SDP_ServiceSearchRequest, the SDP server searches it's service record data base and returns the handles of the service records that match the pattern using this PDU.
0x04	SDP_ServiceAttributeRequest	This PDU is sent by the SDP client to retrieve specified attribute values from a specific service record. (The service record handle would have been fetched already by the client using SDP_ServiceSearchRequest transaction).
0x05	SDP_ServiceAttribute_Response	The SDP server uses this PDU to provide the list of attributes (Attribute ID, Attribute Value) from the requested service record that was provided in the SDP_ServiceAttributeRequest.
0x06	SDP_ServiceSearchAttributeRequest	This PDU combines the capabilities of SDP_ServiceSearchRequest and SD_ServiceAttributeRequest. The SDP client provides both the service search pattern and list of attributes to retrieve.
0x07	SDP_ServiceSearchAttrbuteResponse	The SDP server uses this PDU to provide a list of attributes (Attribute ID, Attribute Value) from the service records that match the requested service search pattern.

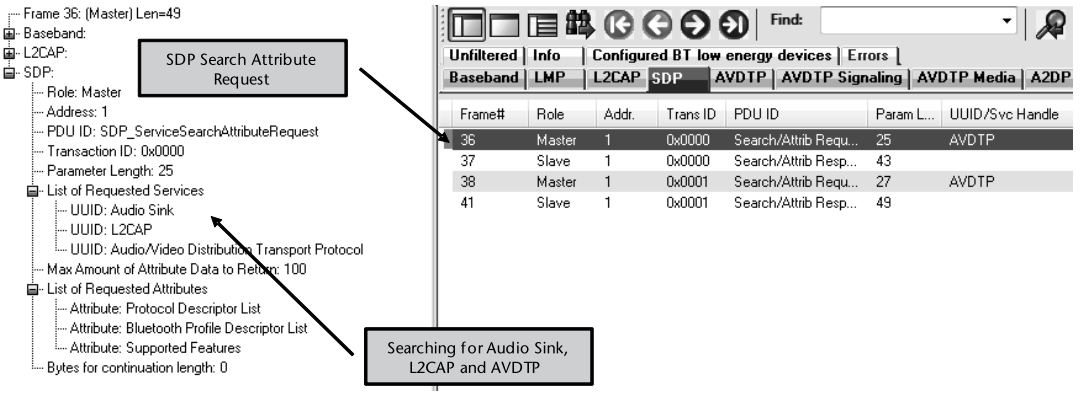

Figure 4.7 Example of SDP_ServiceSearchAttributeRequest.

- L2CAP;
- Audio/Video Distribution Transport Protocol (AVDTP).

Figure 4.8 shows the SDP_ServiceSearchAttributeResponse from the A2DP headset to the mobile phone. The A2DP headset responds to inform the support for the following:

- Audio/Video Distribution Transport Protocol (AVDTP) Version 1.0
- AVDTP is using PSM 0x0019 of L2CAP.
- Advanced Audio Distribution (A2DP) Version 1.0.

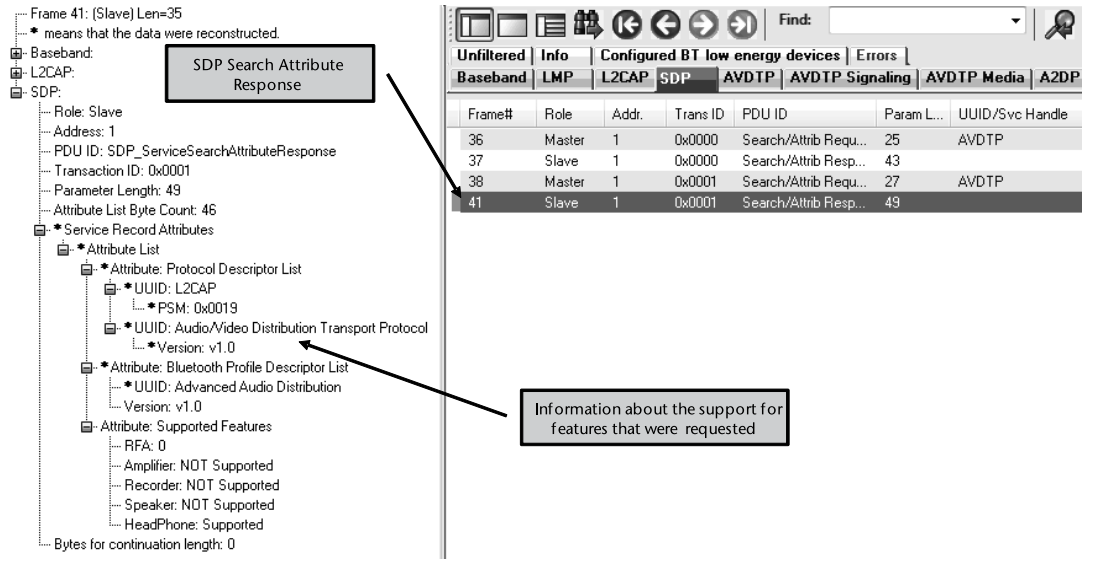

Figure 4.8 Example of SDP_ServiceSearchAttributeResponse.

- Headphone is supported.
- Amplifier, Recorder and Speaker are NOT supported.

4.5 RFCOMM

The RFCOMM protocol is based on the ETSI (European Telecommunications Standards Institute) standard TS 07.10. It is referred to as an adopted protocol because it uses a subset of TS 07.10 protocol and makes some adaptations and extensions to that protocol.

The TS 07.10 is essentially a multiplexer protocol that allows a number of simultaneous sessions over a normal serial interface. Each session could be used for transferring various kinds of data, for example voice, SMS, data, GPRS etc. For details see the bibliography.

RFCOMM provides the emulation of RS-232 serial ports on top of the L2CAP protocol. One of the first intended uses of the Bluetooth technology was as a cable replacement protocol. RFCOMM is the key component to enable this cable replacement. Broadly it may be compared to an RS-232 serial cable where the end points of the cable get replaced with RFCOMM endpoints and the cable is replaced with a Bluetooth connection. This may be used anywhere where serial cables are used to replace those cables with a wireless connection. Some examples are:

- Communication between PCs;
- Connection of mobile phone to headset;
- Connection of mobile phone to laptop.

RFCOMM supports up to 60 simultaneous connections between two Bluetooth devices. As an analogy this can be considered to be similar to two PCs connected

to each other using up to 60 serial cables. The user can run separate applications on each of the serial ports.

There are broadly two types of communications devices:

- Type 1 devices are communication end points. As the name suggests, these devices are at the end of the communication path and are either the producer or consumer of data. For example a laptop that is used to browse the internet is a communication end point.
- Type 2 devices are part of the communication segment. These devices allow data to be relayed from one segment to another. For example a mobile phone may relay the data to the cellular network or a Bluetooth modem may relay the data to the telephony network.

RFCOMM supports both these types of devices.

RS-232 serial interface has following nine circuits which are used for data transfer and signaling.

- TD (Transmit Data) and RD (Receive Data) to carry the data.
- RTS (Request To Send) and CTS (Clear To Send) for Hardware Handshaking or Flow Control.
- DSR (Data Set Ready) and DTR (Data Terminal Ready) for Hardware Flow Control.
- CD (Data Carrier Detect) to indicate a connection to the telephone line.
- RI (Ring Indicator) to indicate an incoming ring signal on the telephone line.
- Signal Common to connect to common ground.

RFCOMM emulates these nine circuits. One of the advantages of this is that it provides backward compatibility with Terminal Emulation programs (like Hyperterminal, TeraTerm etc). These terminal emulation programs can be run on virtual serial ports provided with an underlying RFCOMM connection and provide the same user experience as the wired serial ports connected through RS-232 cables.

As shown in Figure 4.9, up to 60 emulated ports can be active simultaneously though in practice a much lesser number of ports may be supported by the

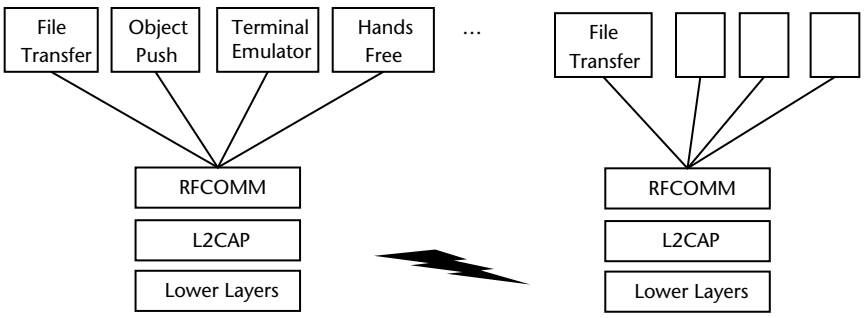

Figure 4.9 RFCOMM multiplexing.

implementation. Each connection is identified by a DLCI (Data Link Connection Identifier).

DLCI 0 is used as a control channel. This is used to exchange Multiplexer Control commands before setting up the other DLCIs.

The DLCI value space is divided between the two communicating devices using the concept of RFCOMM server channels and direction bit. The server applications registering with RFCOMM are assigned a Server Channel Number in the range 1 to 30. The device that initiates the RFCOMM session is assigned the direction bit 1. (This is the lowest significant bit in DLCI).

So the server applications on the non-initiating side are accessible on DLCI 2, 4, 6, …, 60 and the server application on the initiating side are accessible on DLCI 3,5, 7, …, 61.

One of the enhancements that RFCOMM made to TS 07.10 is the credit based flow control. It was introduced after Bluetooth spec 1.0b to provide a flow control mechanism between the two devices. At the time of DLCI establishment, the receiving entity provides a number of credits to the transmitter. The transmitter can send as many frames as it has credits and decrement its credit count accordingly. Once the credit count reaches zero, it stops further transmission and waits for further credits from the receiver. Once the receiver is ready to receive more packets (For example after it has processed the previous packets) it provides further credits to the transmitter. This provides a simple and effective mechanism to emulate the flow control circuits of RS-232.

4.6 Object Exchange Protocol (OBEX)

Bluetooth technology has adopted the IrOBEX protocol from Infrared Data Association (IrDA). OBEX provides the same features for applications as IrDA protocol. So applications can work on both the Bluetooth stack and IrDA stack. That is why OBEX is also referred to as an adopted specification.

Version 1.1 of the OBEX specification provided for support of OBEX over RFCOMM. (It also defined optional OBEX over TCP/IP though it was not used by most of the protocol stacks). Version 2.0 of the specification provided for support of OBEX directly over L2CAP bypassing the RFCOMM layer. This helped in reducing the overheads of RFCOMM and providing higher throughputs, especially in the case of BT 3.0 + HS.

The OBEX protocol follows the client/server model. The purpose of this protocol is to exchange data objects. These data objects could be business cards, notes, images, files, calendars etc.

Some examples of usage of OBEX are provided below:

- Synchronization: OBEX could be used for the synchronization of data between two devices. For example to keep the contacts information and calendar in sync between the laptop and mobile phone.
- File Transfer: OBEX can be used for sending and receiving files, browsing folders, deleting files, etc.

- Object Push: OBEX can be used for sending (pushing) and fetching (pulling) objects like business cards, calendars, notes, etc. Standard formats for these objects are used to ensure interoperability. These formats are referred to as vCard, vCalendar, vMessage and vNotes (electronic business card, electronic calendar, electronic message, and electronic notes).

A device may implement the client role only, server role only or both. For example, a printer may support only the server role. Other devices may connect to the printer and push objects that are to be printed. On the other hand a mobile phone may support both a client and server role so that it can either push or pull objects from other devices or allow other devices to push and pull objects from it.

4.6.1 OBEX Operations

OBEX follows a client/server request-response mechanism as shown in Figure 4.10. The client is the initiator of the OBEX connection. Requests are issued by the client and the server responds to these requests. The request/response pair is referred to as an operation.

The requests/responses are sequential in nature. After sending a request, the client waits for a response from the server before issuing another request.

The various operations supported by OBEX are shown in Table 4.6.

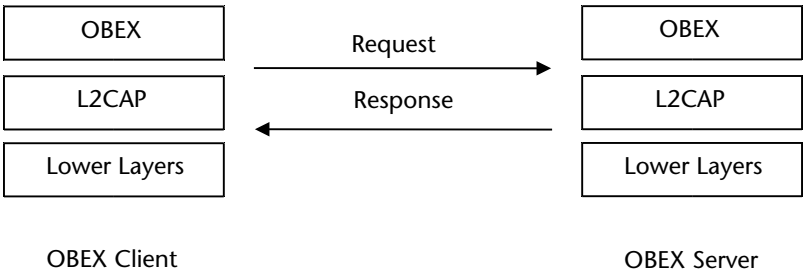

Figure 4.10 OBEX Client and Server (with OBEX over L2CAP).

Table 4.6 OBEX Operations

Operation	*Meaning*
Connect	This operation is used by the client to initiate a connection to the server and negotiate the parameters to be used for further operations (For example OBEX version number, maximum packet length)
Disconnect	This signals the end of OBEX connection.
Put	The Put operation is used by the client to push one object from the client to the server.
Get	The Get operation is used by the client to request to server to return an object to the client.
SetPath	The SetPath operation is used to set the "current directory" on the server side. All further operations are carried on from that directory after this command is sent.
Abort	The Abort request is used when the client decides to terminate an operation that was spread over multiple packets between the client and server.

4.7 Audio/Video Control Transport Protocol (AVCTP)

AVCTP defines the transport mechanisms used to exchange messages for controlling Audio or Video devices. It uses point-to-point signaling over connection oriented L2CAP channels.

Two roles are defined:

- Controller (CT): This is the device that initiates an AVCTP transaction by sending a command message. The device that supports the controller functionality is also responsible for initiating the L2CAP channel connection on request of the application.
- Target (TG): This is the remote device that receives the command message and returns zero or more responses to the controller.

A complete AVCTP transaction consists of one message containing a command addressed to the target and zero or more responses returned by the target to the controller. A device may support both CT and TG roles at the same time.

AVCTP may support multiple profiles on top. It uses the concept of a Profile Identifier to allow applications to distinguish messages from different profiles.

4.8 Audio/Video Distribution Transport Protocol (AVDTP)

AVDTP defines the protocol for audio/video distribution connection establishment, negotiation and streaming of audio/video media over the Bluetooth interface. The transport mechanism and message formats are based on the RTP protocol which consists of two major protocols: RTP Data Transfer Protocol (RTP) and RTP Control Protocol (RTCP). AVDTP uses the L2CAP connection oriented channels for setting up the A/V streams and then streaming the data. A stream (or Bluetooth A/V stream) represents the logical end-to-end connection of streaming media between two A/V devices.

Two roles are defined:

- Source (SRC): This is the device where the streaming data originates.
- Sink (SNK): This is the device which receives the audio data.

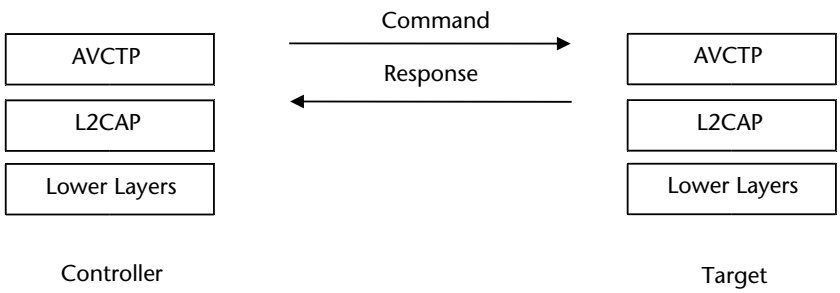

Figure 4.11 AVCTP Controller and Target.

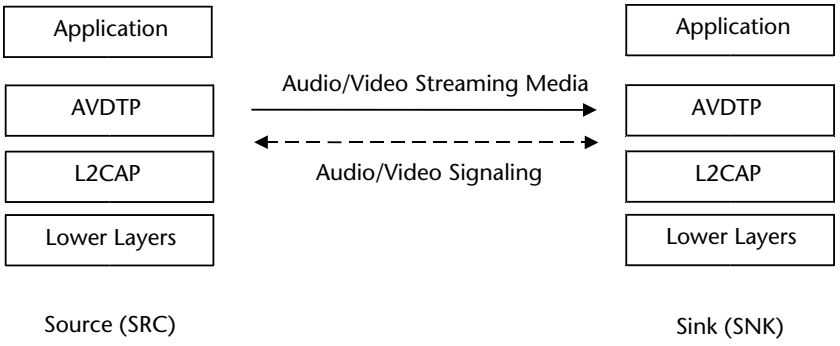

Figure 4.12 AVDTP Source and Sink.

As an example in the scenario of streaming data from a laptop to a Bluetooth stereo headset, the stream corresponds to the audio stream between the laptop and the Bluetooth headset. The laptop acts as a SRC device and the Bluetooth stereo headset acts as the SNK device.

A Stream End Point (SEP) is a concept to expose the available transport services and AV capabilities of the application in order to negotiate a stream. An application registers its SEPs in AVDTP to allow other devices to discover and connect to them.

AVDTP defines procedures for the following:

- Discover: To discover the Stream End Points supported in the device.
- Get Capabilities: To get the capabilities of the Stream End Point.
- Set Configuration: To configure the Stream End Point.
- Get Configuration: To get the configuration of the Stream End Point.
- Reconfigure: To reconfigure the Stream End Point.
- Open: Open a stream.
- Start: To start streaming.
- Close: To request closure of a Stream End Point.
- Suspend: To request that a Stream End Point be suspended.
- Security Control: To exchange content protection control data.
- Abort: To recover from error conditions.

A typical sequence of operations of AVDTP transactions is shown in Figure 4.13. In this scenario a mobile phone (acting as CT) creates a connection to an A2DP headset (acting as TG), streams a music file and then disconnects. The sequence of transactions that happen is as follows:

1. Frame #65: The mobile phone sends a command to discover the stream end points in the headset.
2. Frame #66: The headset responds with information about the stream end points. (In this example, the slave actually provides two stream end points.
   a. One that supports MP3 codec.
   b. Second that supports SBC codec.

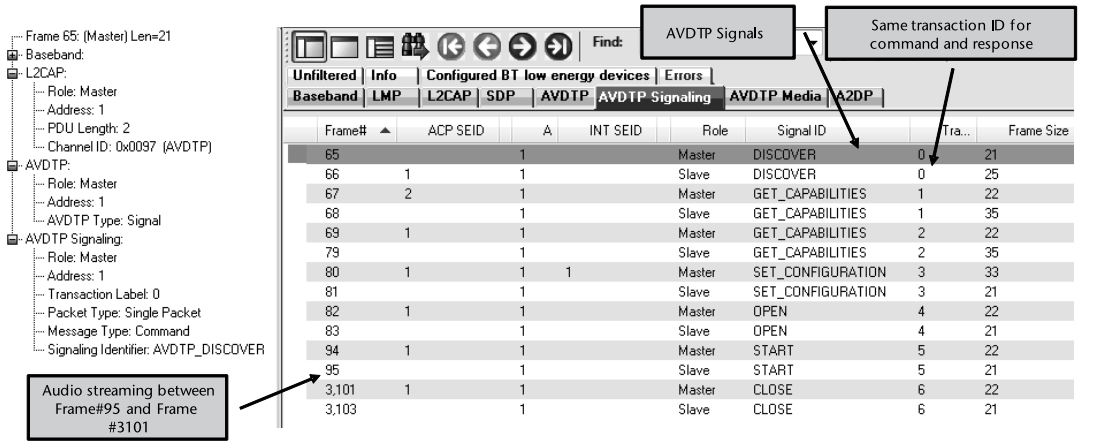

Figure 4.13 Example of AVDTP transactions between mobile phone and stereo headset.

3. Frame #67, #68: The mobile phone gets the capabilities of the first stream end point and the headset responds.
4. Frame #69 and #79: The mobile phone gets the capabilities of the second stream end point and the headset responds.
5. Frame #80: The mobile phone configures the stream end point on the headset with the parameters needed to stream the audio.
6. Frame #82: The mobile phone opens the stream.
7. Frame #94: The mobile phone starts streaming the audio data.
8. Frame #95 to Frame #3100: The audio data is streamed from the mobile phone to the headset.
9. Frame #3101: The mobile phone decides to close the stream.

4.9 Profiles

As explained at the beginning of this chapter, profiles can be considered to be vertical slices through the protocol stack. They provide information on how each of the protocol layers comes together to implement a specific usage model. They define the features and functions required from each layer of the protocol stack from Bluetooth Radio up to L2CAP, RFCOMM, OBEX, and any other protocols like AVCTP, AVDTP, etc. Both the vertical interactions between the layers as well as peer-to-peer interactions with the corresponding layers of the other device are defined.

Profiles help to guarantee that an implementation from one vendor will work properly with an implementation from another vendor. So they form the basis for interoperability and logo requirements. The profiles need to be tested and certified before a device can be sold in the market. A device can support one or more profiles at the same time.

Generic Access Profile (GAP) is mandatory to be implemented for all devices that support Bluetooth. Devices may implement more profiles depending on the requirements of the application. The dependencies amongst profiles are depicted in

Figure 4.14. A profile is dependent on another profile if it uses parts of that profile. A dependent profile is shown in an inner box and the outer box indicates the profiles on which it is directly or indirectly dependent. GAP is shown in the outermost box since all other profiles are dependent on it.

For example Hands-Free Profile is dependent on Serial Port Profile which is in turn dependent on Generic Access Profile. So the box for Hands-Free Profile is shown within the box for Serial Port Profile. The Box for Serial Port Profile is in turn located inside the box for Generic Access Profile.

4.10 Generic Access Profile (GAP)

Generic Access Profile is a base profile which is mandatory for all devices to implement. It defines the basic requirements of a Bluetooth device. For BR/EDR it defines a Bluetooth device to include at least the following functionality:

- Bluetooth Radio;
- Baseband;
- Link Manager;
- L2CAP;
- SDP.

GAP defines how these layers come together to provide the Bluetooth functionality. It also defines procedures for the following:

- Device discovery;
- Connection establishment;

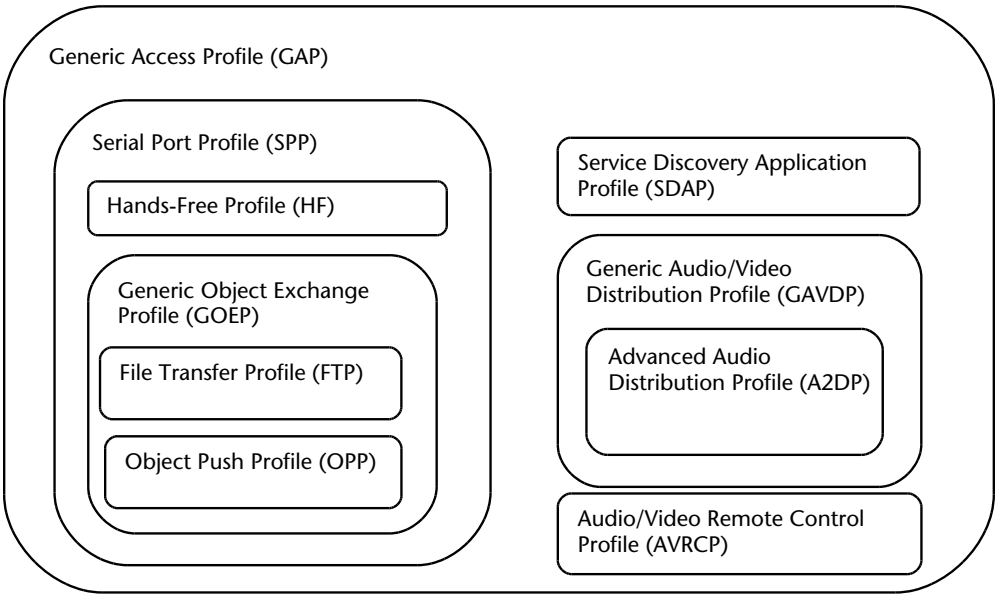

Figure 4.14 Profile dependencies.

- Security;
- Authentication;
- Service discovery.

The purpose of GAP is:

- To introduce definitions, recommendations, and common requirements related to modes and access procedures that are used by transport and application profiles.
- To describe how the devices behave in various states like standby and connecting. Special focus is put on discovery, connection establishment and security procedures.
- To state requirements on user interface aspects, names of procedures, and parameters, etc. This ensures a uniform user experience across all devices.

GAP defines the procedures for both BR/EDR and LE device types. It defines three device types:

- BR/EDR: Devices that support Basic Rate and Enhanced Data Rate.
- LE Only: Devices that support Low Energy configuration.
- BR/EDR/LE: Dual mode devices that support both BR/EDR and LE.

The BR/EDR procedures will be covered in this section and LE procedures will be covered in Chapter 14.

4.10.1 Bluetooth Parameters Representation

GAP states requirements about the generic terms that should be used on the user interface (UI) level. These are useful not only when designing user interfaces but also in user manuals, documentation, advertisements, etc. This helps to ensure a uniform user experience irrespective of the vendor who makes the device or the application.

GAP defines the requirements for:

- Bluetooth Device Address (BD_ADDR)
  - On the user interface level, the address should be referred to as "Bluetooth Device Address" and represented as 12 hex characters possibly separated by ":" symbol. An example of representation of the BD Address is 00:AB:CD:EF:12:34.
- Bluetooth Device Name
  - This is the user friendly name that can be used to refer to a device. This is in the form of a character string which can be retrieved by remote devices using the remote name request. The maximum length of the character string is 248 bytes.
- Bluetooth Passkey (Bluetooth PIN)

  - The pairing process was explained in the previous chapter. The Bluetooth passkey may be used to authenticate two devices via the pairing procedure. The PIN may either be entered on the UI level (For example on the user interface on mobile phone or laptop) or stored in the device (For example the predefined PIN key stored in a headset).
- Class of Device (CoD)
  - Class of device provides information on the type of device and the type of services it supports. The Class of Device is retrieved during the inquiry procedure. GAP defines that the Class of Device parameters should be referred to as "Bluetooth Device Class" and "Bluetooth Service Type" on the UI level. These are obtained from various fields of the CoD.

4.10.2 Modes

GAP defines different modes in which the device can be in with respect to inquiry and connection. These modes are explained in this section.

4.10.2.1 Discoverability Modes

Inquiry is the procedure to discover devices in the Bluetooth vicinity. With respect to inquiry, a device can be in one of the following two discoverability modes:

- Non-Discoverable mode: In this mode a device does not respond to inquiry. So it cannot be discovered by other devices.
- Discoverable mode: In this mode the device is set to discoverable mode and it may respond to inquiry from remote devices. There are two discoverable modes:
  - Limited Discoverable mode: The limited discoverable mode is used by devices that are discoverable only for a limited amount of time, during temporary conditions, or for a specific event. The device responds to a device that makes a limited inquiry.
  - General Discoverable mode: This mode is used by devices that need to be discoverable continuously or for no specific condition. The device responds to any device that make a general inquiry.

4.10.2.2 Connectability Modes

Paging is the procedure used to connect to remote devices. With respect to paging, a device can be in one of the following two connectable modes:

- Non-connectable mode: In this mode, the device does not enter the PAGE_SCAN state. So it's not possible to connect to this device. (PAGE_SCAN state was explained in the previous chapter).
- Connectable mode: In this mode, the device periodically enters the PAGE_SCAN state to check for incoming connection requests. So other devices can connect to this device.

4.10.2.3 Bondable Modes (for Pairing)

Bonding is the process of pairing when the passkey is entered on the device for the purpose of creating a "bond" between two Bluetooth devices. It may or may not be followed later on by creation of a connection.

With respect to bonding, a device can be in one of the following two bondable modes:

- Non-bondable mode: In this mode, the device does not accept a pairing request from other devices. It may still accept an incoming connection from devices that do not require bonding.
- Bondable mode: In this mode, the device accepts a pairing request from other devices. Pairing can be either in the form of legacy pairing or secure simple pairing (SSP).

4.10.3 Idle Mode Procedures

The idle mode procedures include procedures for inquiry, name discovery, bonding etc. These are called idle mode procedures because generally these are initiated when the device is in the idle mode though these can also be done when the device is already connected (for example, to create a scatternet).

4.10.3.1 Inquiry

Inquiry is the procedure to discover information about remote devices. Devices can dynamically enter and move out of the Bluetooth vicinity. This procedure is useful to find out the list of devices that are currently in the vicinity along with some basic information about those devices. Using this information, a decision may be taken to further move on to stages like discovering name, creation of connection, etc.

Inquiry provides the following information about the remote device:

- BD_ADDR;
- Clock information;
- Class of Device;
- Information about Page scan mode;
- Extended Inquiry Response Information if it is supported by the device.

Bluetooth specification defines two types of inquiry:

- General Inquiry: This is used to discover devices which are set to discoverable mode and are set to do an inquiry scan with General Inquiry Access Code. (GIAC). This is used for devices that are made discoverable continuously or for no specific condition.
- Limited Inquiry: This is used to discover devices which are scanning with Limited Inquiry Access Code (LIAC). This is used for devices which are made discoverable only for a limited amount of time, during temporary conditions for a specific event.

Devices that are set to Limited discoverable mode are also discovered in General Inquiry. The term used on User Interface level is "Bluetooth Device Inquiry."

4.10.3.2 Name Discovery

This procedure is used to get the Bluetooth name of the remote device. The term used on User Interface level is "Bluetooth Device Name Discovery."

4.10.3.3 Device Discovery

This procedure is similar to the Inquiry procedure. It is used to find some additional information besides the one provided in inquiry.

Discovery provides the following information about the remote device:

- BD_ADDR;
- Clock information;
- Class of Device;
- Information about Page scan mode;
- Extended Inquiry Response Information;
- Bluetooth Device Name—This is the additional information that is not reported during inquiry.

The term used on User Interface level is "Bluetooth Device Discovery."

4.10.3.4 Bonding

The bonding procedure is used to create a mutual trust relationship between two Bluetooth devices based on a common link key. The link key is created and exchanged during this procedure and is expected to be stored by both the Bluetooth devices. This link key is used for authentication when a connection is created.

The term used on the User Interface level is "Bluetooth Bonding."

4.10.4 Establishment Procedures

These procedures refer to the link establishment, channel establishment and connection establishment. A Device discovery or inquiry procedure is done before establishment procedures to get information about the device to which the connection is to be established. These are shown in Figure 4.15.

4.10.4.1 Link Establishment

This procedure is used to create an ACL link between two devices. The term used on the User Interface level is "Bluetooth link establishment."

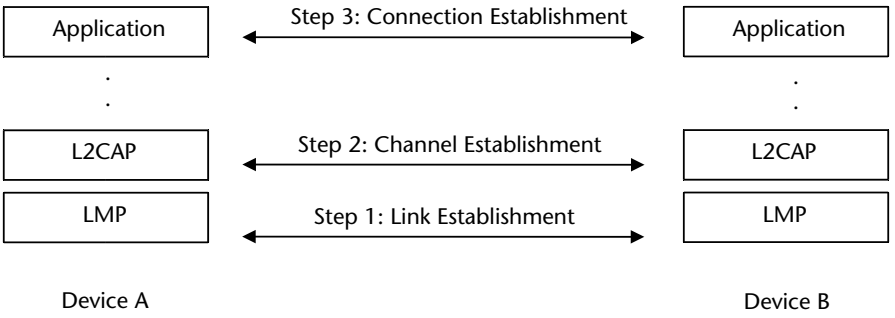

Figure 4.15 Establishment procedures.

4.10.4.2 Channel Establishment

This procedure is used to create an L2CAP channel between two devices. The term used on the User Interface level is "Bluetooth channel establishment."

4.10.4.3 Connection Establishment

This procedure is used to establish a connection between applications on two Bluetooth devices. The term used on the User Interface level is "Bluetooth connection establishment."

4.10.5 Authentication

Authentication was explained in the previous chapter. It is the process of verifying "who" is at the other end of the link. The authentication process starts when the two devices initiate a connection establishment. If the link is not already authenticated and the link key is not available, then the pairing procedure is started.

4.10.6 Security

There are two broad types of security modes: Legacy security mode and Simple Secure Pairing mode. Legacy mode is used by devices that do not support SSP.

Within legacy security mode, there are three types of security modes:

- Security mode 1 (non-secured): This mode means that no security is desired.
- Security mode 2 (service level enforced security): In this mode, the security is initiated after the connection establishment if the channel or service requires security. For example if security is required for a particular profile, then this security mode may be used.
- Security mode 3 (link level enforced security): In this mode, the security is initiated during the connection establishment.

Version 2.1 + EDR of the Bluetooth specification added Simple Secure Pairing and Security mode 4:

- Security mode 4 (service level enforced security): The security can be specified with the following attributes:
  - Authenticated link key required: This is the link key generated using numeric comparison, out-of-band, or passkey entry. It has protection against MITM attacks as explained in Chapter 3.
  - Unauthenticated link key required: This is the link key generated using just works method. It does not provide protection against MITM attacks.
  - Security optional: This is only used for SDP transactions.

It is possible for a device to support two security modes at the same time. This could be the case when it wants to connect to legacy devices using Security Mode 2 and devices that support Simple Secure Pairing with Security mode 4.

4.11 Serial Port Profile (SPP)

The serial port profile (SPP) defines the requirements for setting up emulated serial connections between Bluetooth devices. This provides similar user experience as compared to an RS232 cable connection between the two devices with the only difference that a physical wire is replaced by the Bluetooth connection between the two devices.

Bluetooth was originally designed as a cable replacement technology and this was amongst the first profiles that were used since it supports the cable replacement use case.

One of the strongest points about this profile is that it allows legacy applications (which were designed for RS232 serial ports) to use Bluetooth wireless connection instead of a wired connection. Generally another application (for example, a Bluetooth connection management application) is used to initially discover devices in the vicinity and create an SPP connection with the remote device. Once this is done, the legacy application can transparently use the Bluetooth connection as if it were a wired RS232 connection. As shown in Figure 4.14, SPP depends on GAP and re-uses the terms and procedures defined in the GAP profile. SPP is in turn used by other profiles like Hands-Free and GOEP.

SPP defines two roles:

- Dev A: This is the device that initiates a Bluetooth connection.
- Dev B: This is the device that waits for a device to make a connection and then accepts the incoming connection.

Figure 4.16 shows a typical usage scenario of Serial Port Profile. It uses the RFCOMM, L2CAP and lower layers of the Bluetooth protocol stack. A port emulation layer is used to emulate the serial port. In general this layer is dependent on the operating system. For example on Linux, this layer may expose a virtual serial

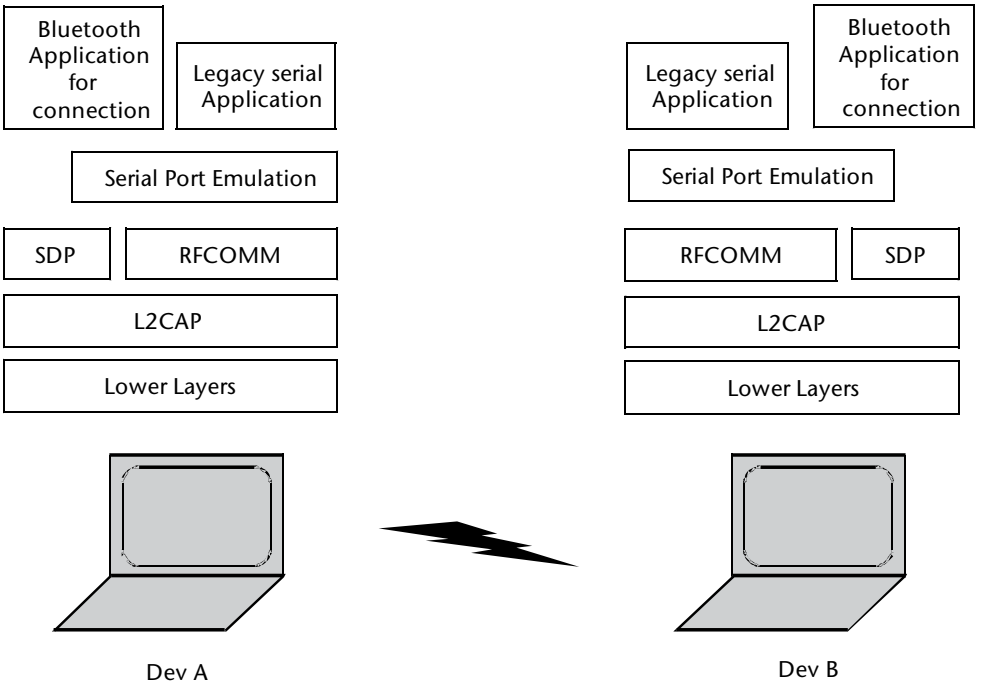

Figure 4.16 Typical usage scenario of serial port profile.

port device driver. A Legacy serial port application could be used on both devices to connect to the emulated serial port.

Besides this a separate Bluetooth application may be used for initial device discovery, services discovery, connection establishment, etc. Once a Bluetooth connection is established, the legacy application can start using the emulated serial port just like any other RS232 serial port.

4.12 Headset Profile, Hands-Free Profile

The Headset and Hands-Free profiles define the set of functions to be used for a Bluetooth connection between a mobile phone and a handsfree device, for example a Bluetooth mono headset, Carkit installed in a car, etc. The Headset profile was one of the first profiles defined by the Bluetooth specification. Later on this profile was superseded by the Hands-Free profile. The Hands-Free profile provides a superset of the Headset profile functionality.

Some of the functionality defined by this profile includes:

- Connection related functionality.
  - Connection to a Hands-Free device so that the audio can be routed from the mobile phone to the Hands-Free device.
  - Accept an incoming call.
  - Reject an incoming call.
  - Terminate a call.

- Display of phone status like signal strength, roaming, battery level, etc, on the Hands-Free device.
- Transfer an audio connection from phone to Hands-Free or vice versa.
- Different options for placing a call from the Hands-Free device.
  - From a number supplied by the Hands-Free device.
  - By memory dialing.
  - Redial last number.
- Activation of voice recognition.
- Call waiting notification.
- Three-way calling.
- Caller line identification (CLI).
- Echo cancellation and Noise reduction.
- Remote audio volume control.

Hands-Free profile defines the following two roles. These are shown in Figure 4.17.

- Audio Gateway (AG): This is the device that acts as a gateway for audio. Typically this is the mobile phone which acts as a gateway of the audio from the cellular network to the Hands-Free device.
- Hands-Free unit (HF): This is the device that acts as the audio input and output mechanism. This may also provide some means to control some of the functionality of the AG. Typically this is a Carkit or a handset.

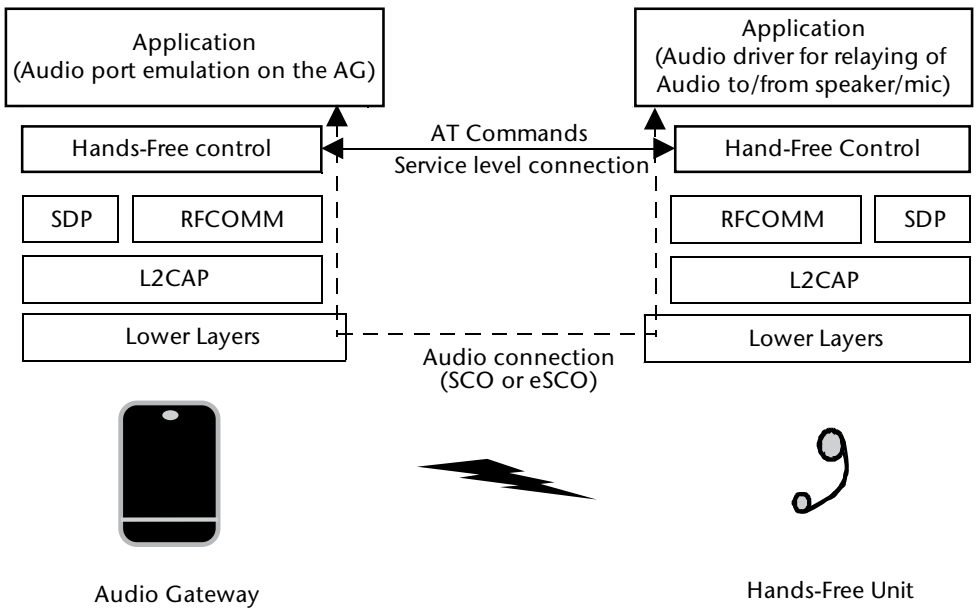

Figure 4.17 Hands-Free profile.

The Hands-Free profile defines two types of connections:

- Service level connection: This is the RFCOMM connection between the AG and HF which is used for transfer of control information.
- Audio Connection: This is the SCO or eSCO connection along with the complete audio path to route the audio (voice data) from the cellular network to the Hands-Free unit.

The Hands-Free profile uses AT commands extensively on the Service level connection to perform various tasks. The AT commands used by this profile are a subset of the 3GPP 27.0.0.2 specification.

4.13 Generic Object Exchange Profile (GOEP)

GOEP defines the requirements for devices like laptops, PDAs and smartphones to support capabilities to exchange objects. As shown in Figure 4.14, this profile is dependent on GAP and SPP. In turn it provides features to support profiles like FTP and OPP.

GOEP defines the following two roles. These are shown in Figure 4.18.

- Server: This is the device that acts as the object exchange server to and from which objects can be pushed and pulled.
- Client: This is the device that can push or pull objects to and from the server.

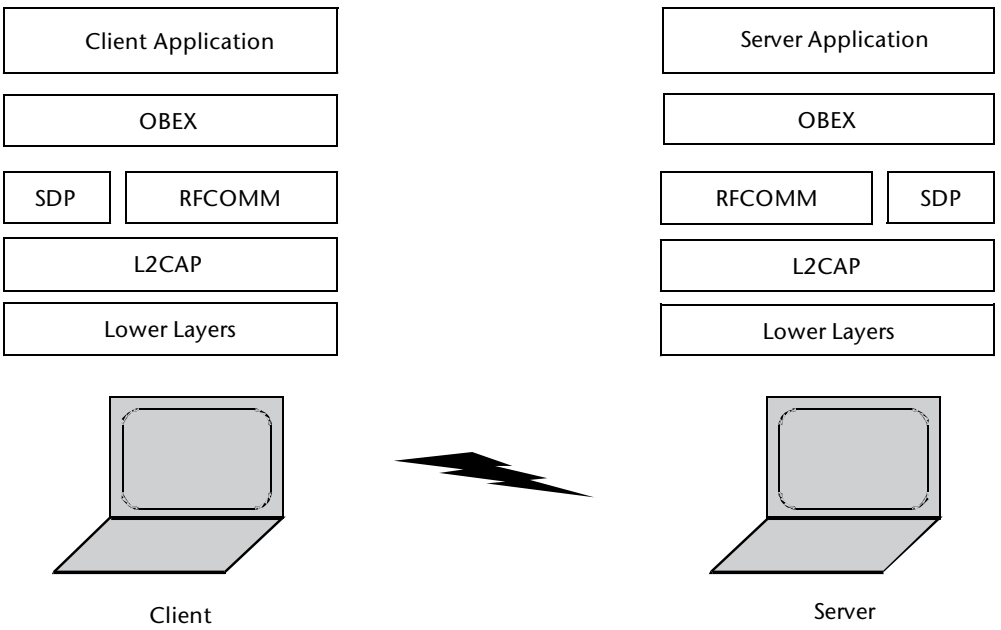

Figure 4.18 Generic object exchange profile.

As shown in Figure 4.18, this profile makes use of the OBEX, SDP, RFCOMM, L2CAP and lower layers of the protocol stack. It provides the following major features:

- Establishment of an object exchange session: This feature is used in the beginning to establish a connection between the client and the server. The remaining features can only be used once this procedure is successfully completed.
- Pushing a data object: This feature is used to transfer an object from the client to the server.
- Pulling a data object: This feature is used to retrieve an object from the server to the client.

4.14 Object Push Profile (OPP)

This profile defines the requirements needed to support the object push usage model between two Bluetooth devices. It is dependent on GOEP, SPP and GAP.

OPP defines the following two roles. These are shown in Figure 4.19.

- Push Server: This is the device that acts as the object exchange server to and from which objects can be pushed and pulled.
- Push Client: This is the device that can push or pull objects to and from the push server.

As shown in Figure 4.19, this profile makes use of the OBEX, SDP, RFCOMM, L2CAP and lower layers of the protocol stack. It provides the following major features:

- Object Push: This feature is used to push an object to the inbox of another device. For example to push a business card or an appointment to a mobile phone.
- Business Card Pull: This feature is used to pull an object from the server. For example to pull a business card from a mobile phone.
- Business Card Exchange: This feature is used to exchange objects. For example to push a business card followed by a pull of the business card.

The different objects that can be pushed by this profile are as follows:

- vCard: This is a format used for transferring contacts.
- vCalendar: This is a format used for transferring appointments.
- vMessage: This is the format used for messaging applications.
- vNote: This is the format used by notes applications.

References to the details of these formats are provided in the Bibliography section.

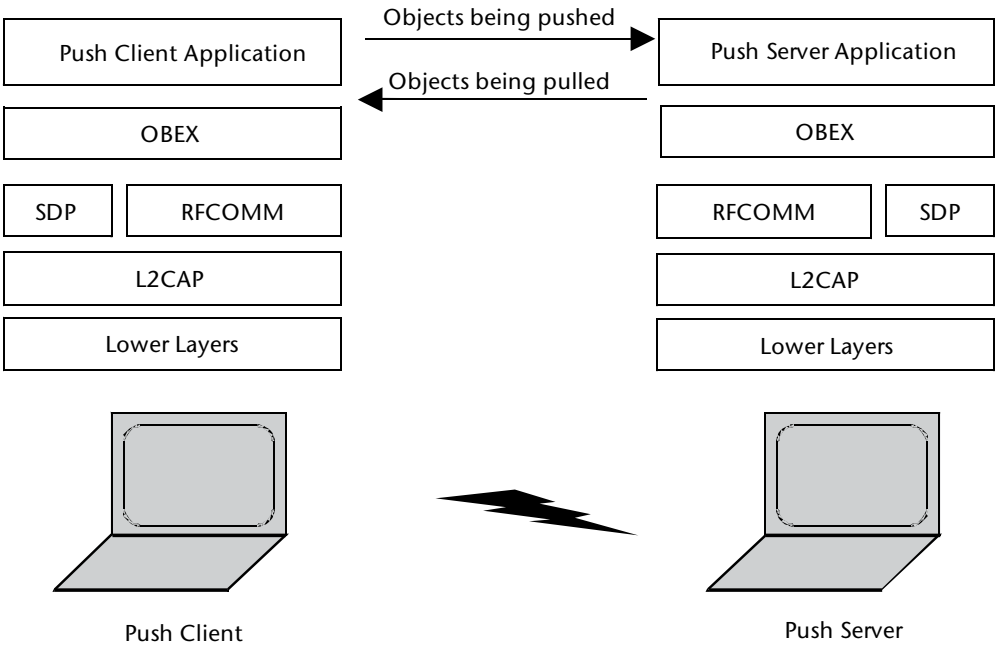

Figure 4.19 Object push profile.

4.15 File Transfer Profile (FTP)

This profile defines the requirements needed to support the file transfer usage model between two Bluetooth devices. It is dependent on GOEP, SPP and GAP.

FTP defines the following two roles. These are shown in Figure 4.20.

- Server: This is the device that acts as the file transfer server to and from which files can be pushed and pulled. It also provides the folder browsing capabilities.
- Client: This is the device that can push or pull files to and from the server.

As shown in Figure 4.20, this profile makes use of the OBEX, SDP, RFCOMM, L2CAP, and lower layers of the protocol stack. It provides the following major features:

- Browse the file system: This feature allows support for browsing the file system of the server. This includes viewing the files and folders and navigating the folder hierarchy of the other device.
- File Transfer: This feature provides support for transferring files and folders from one device to another.
- Object manipulation: This feature allows manipulating the objects on the other device. This may include deleting files, creating folders, deleting folders, etc.

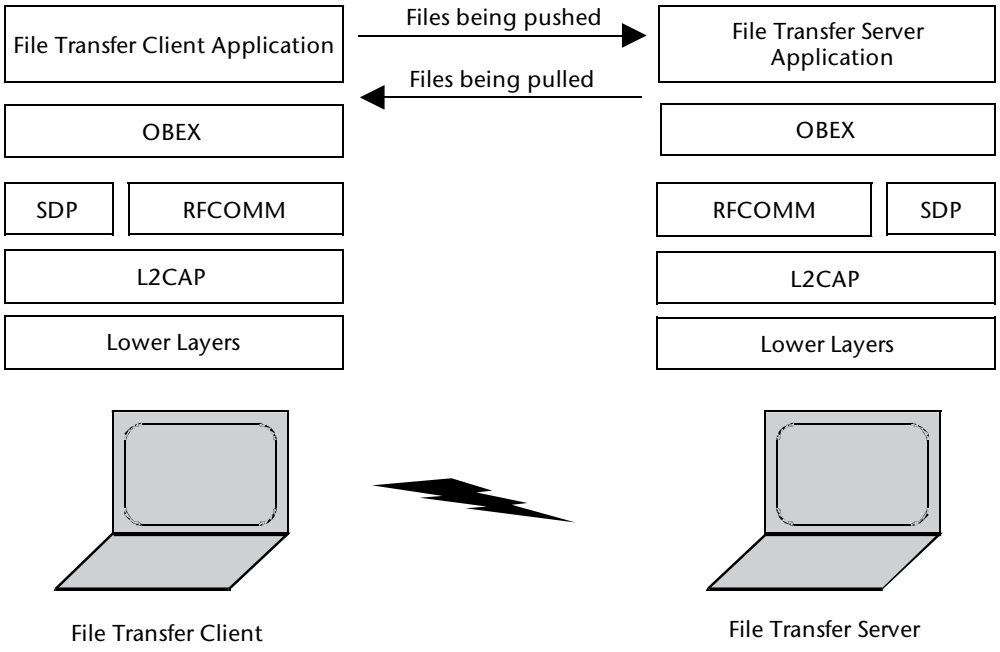

Figure 4.20 File transfer profile.

4.16 Generic Audio/Video Distribution Profile (GAVDP)

This profile defines the requirements for Bluetooth devices to support streaming channels for supporting audio/video distribution on ACL channels. It is dependent on GAP and uses AVDTP, L2CAP and lower layers.

GAVDP defines the following two roles. These are shown in Figure 4.21:

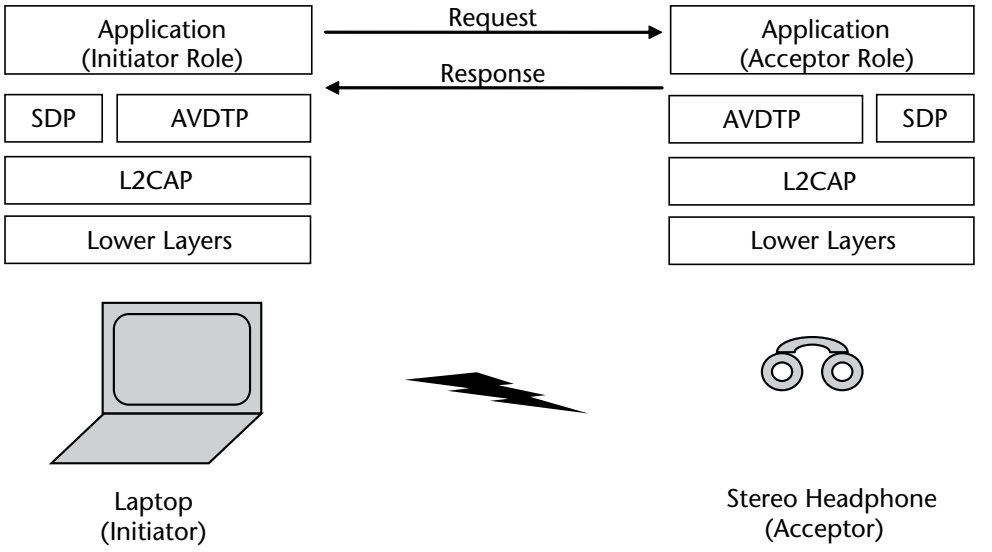

Figure 4.21 Generic audio video distribution profile.

- Initiator (INT): This is the device that initiates the GAVDP signaling procedures.
- Acceptor (ACP): This is the device that responds to the incoming requests from the INT.

A typical use case of the profile is the streaming of audio between a laptop and headphones. In this case, the laptop may act as the INT. It would send request to establish a streaming channel, negotiate parameters, control the stream etc. The headphones would act as the ACP and respond to the requests made by the INT.

GAVDP provides support for the following two scenarios:

- Setup the two devices for A/V data streaming and then create a connection between these two devices.
- Control the established streaming connection.

The detailed features and procedures supported by this profile are described in Table 4.7.

4.17 Advanced Audio Distribution Profile (A2DP)

This profile defines the requirements for Bluetooth devices to support high quality audio distribution. It uses the ACL channels for distribution of high quality audio. This is in contrast to the HF profile in which the SCO channels are used to transfer voice.

ACL channels are used because the bandwidth that is provided by SCO and eSCO channels is not sufficient to transfer the high quality audio data. In fact if raw audio data were to be streamed, then even the ACL channels don't have sufficient bandwidth. So, a codec is used to encode the data before sending and then decoding the data after it is received on the remote side.

Table 4.7 GAVDP Features and Procedures

Feature	*Procedure*	*Purpose*
Connection	Connection Establishment	This procedure is used when a device needs to create a connection with another device. It includes AVDTP procedures for finding the stream end points and getting the capabilities.
	Start Streaming	This procedure is used when both the devices are ready to start streaming and is used to start or resume streaming.
	Connection Release	This procedure is used to release the stream.
Transfer Control	Suspend	This procedure is used to suspend the A/V stream.
	Change Parameters	This procedure is used to change the service parameters of the stream.
Signaling Control	Abort	This procedure may be used to recover from a loss of signaling message.
Security Control	Security Control	This procedure is used to exchange security control messages between the two devices.

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).

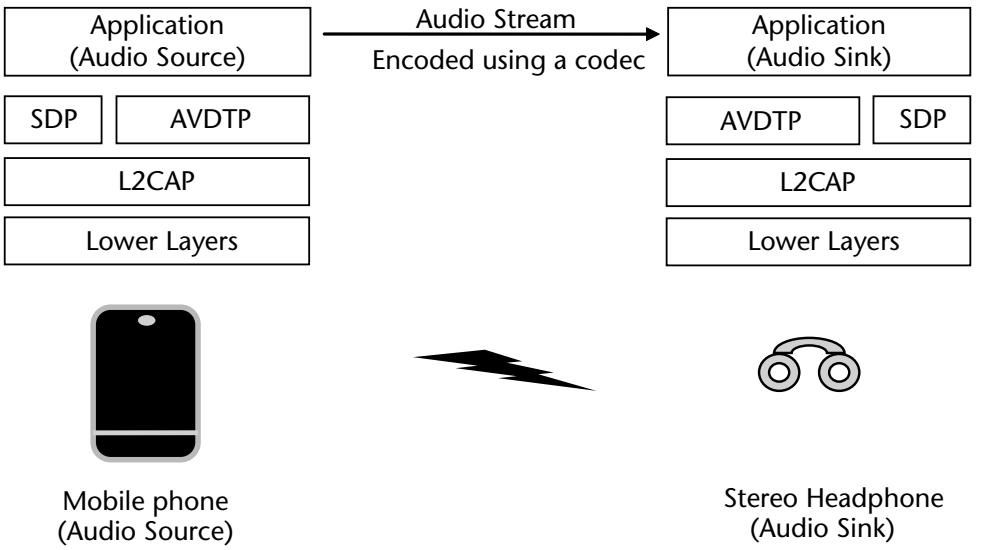

Figure 4.22 Advance audio distribution profile (A2DP).

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:

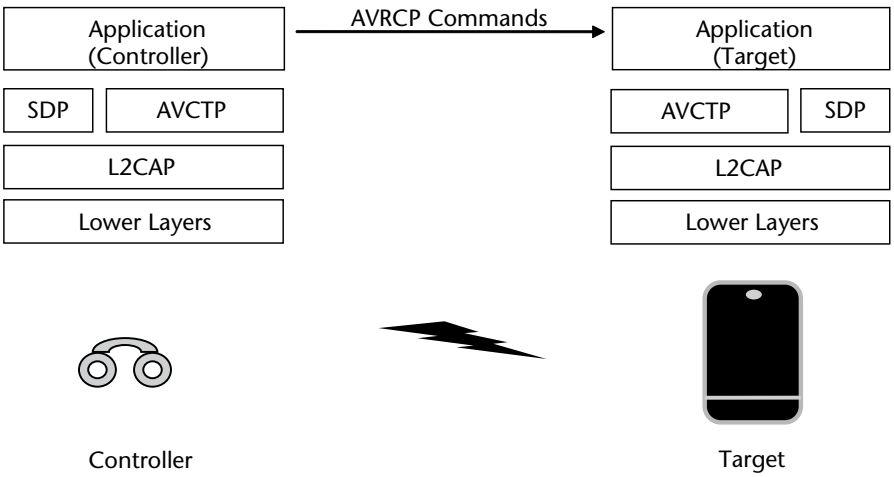

Figure 4.23 Audio/video remote control profile (AVRCP).

- 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).