Beyond 802.11: BLUETOOTH

- Bluetooth is a wireless LAN technology designed to connect devices of different functions such as telephones, notebooks, computers, cameras, printers, coffee makers, and so on.
- A Bluetooth LAN is an ad hoc network, which means that the network is formed spontaneously.

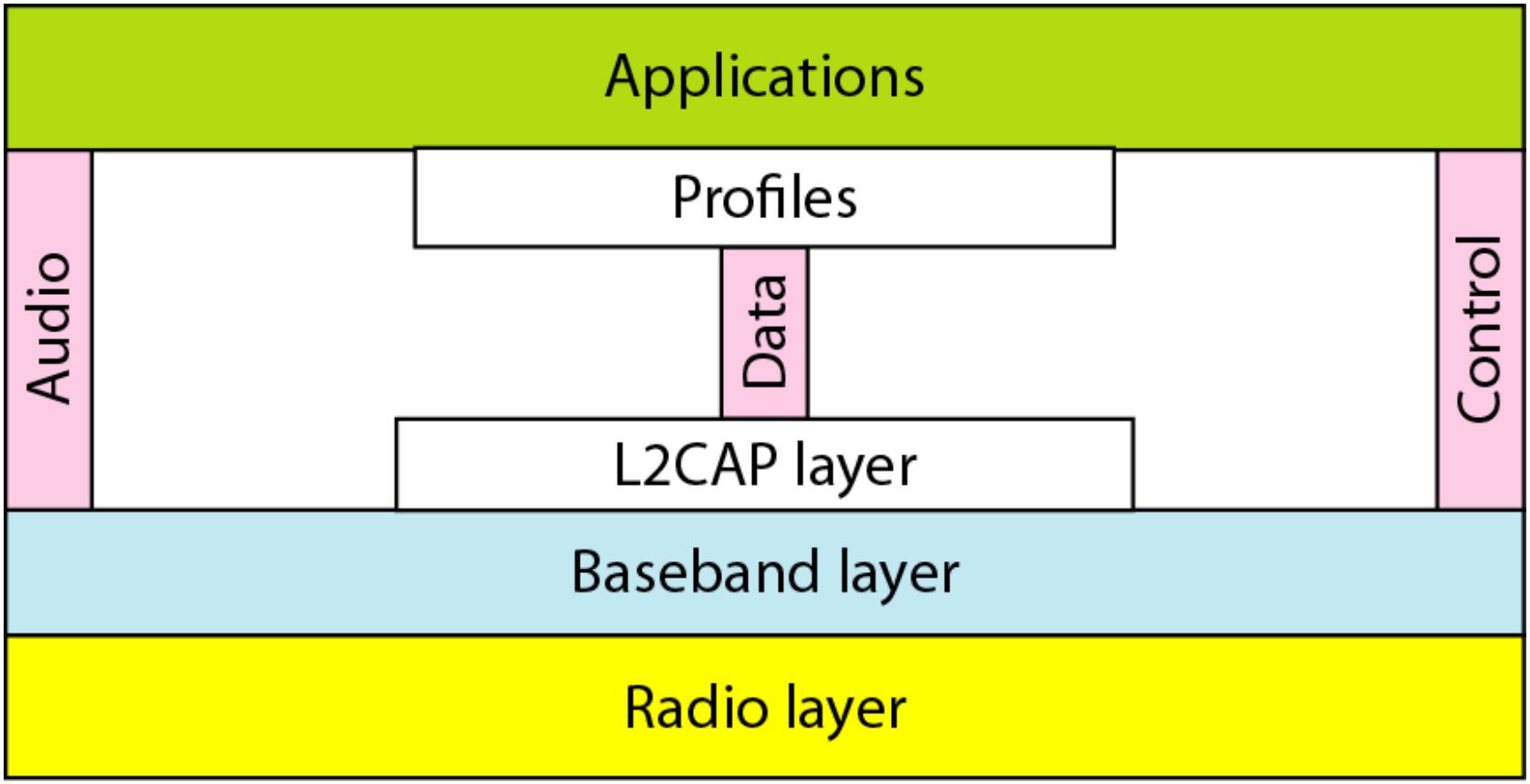
Some of the key issues:

Architecture
Bluetooth Protocol Stack
Baseband Layer
Transport Protocols

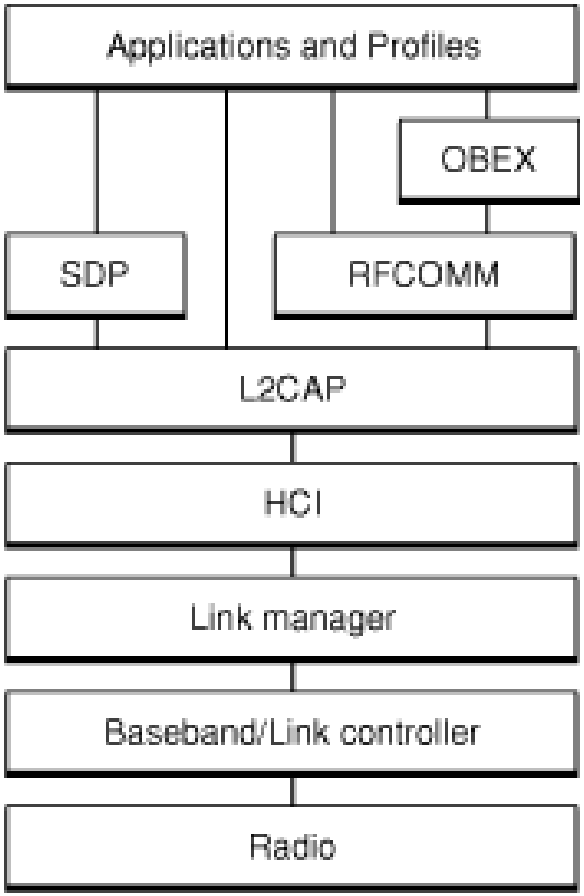
Bluetooth Architecture

- The Bluetooth architecture consists of a hardware-based radio system, and a software stack that specifies the linkages between layers.
- This supports flexibility in implementation across different devices and platforms. It also provides robust guidelines for maximum interoperability and compatibility.

Bluetooth layers



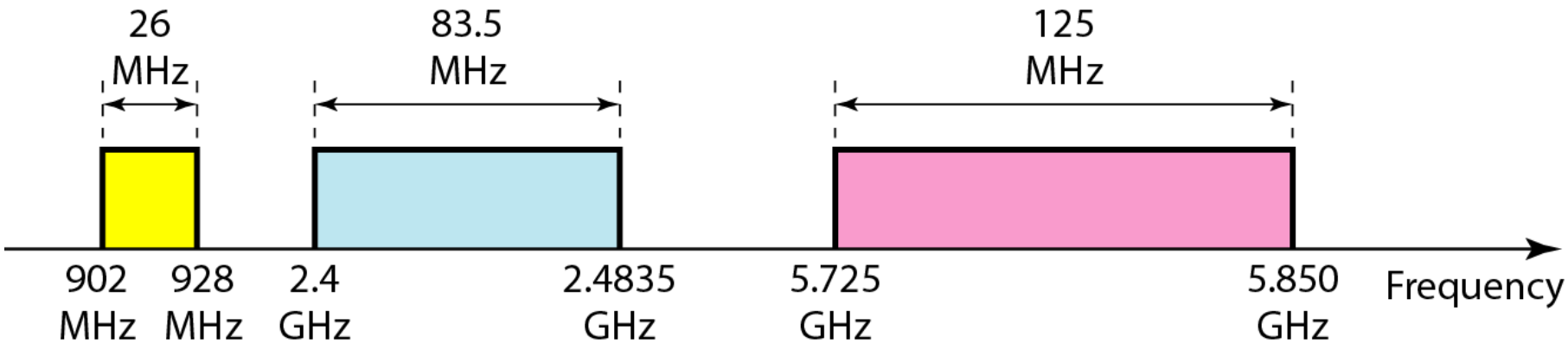
Bluetooth Protocol Stack



Lower Layers

- **Radio layer**
  - The radio module in a Bluetooth device is responsible for the modulation and demodulation of data into RF signals.
  - This layer describes the physical characteristics that a Bluetooth device's receiver-transmitter component must have.
  - These include modulation characteristics, radio frequency tolerance, and sensitivity level.

Industrial, scientific, and medical (ISM) band



Radio Layer

- Uses the 2.4 GHz ISM band, same as 802.11b/g
- Frequency bands and channel arrangement:
  - 2.400-2483.5GHz, f=2402 MHz+k, k=0,…,78
  - Lower (2MHz) and upper (3.5MHz) guard bands
- Transmitter:
  - Class 1: long range (≈ 100 m) devices, Poutmax = 20dBm (100mW)
  - Class 2: ordinary range (≈ 10 m) devices, Poutmax = 4dBm (2.5mW)
  - Class 3: short range (≈ 1 m) devices, Poutmax= 0dBm (1mW)
- Receiver:
  - Sensitivity level: Reference: -70dBm
  - Actual: “input level for which a raw BER of 0.1% is met”
  - Receiver: -70dBm or better with any Bluetooth transmitter
- Versions:
  - Version 1.2 (1 Mbps)
  - Version 2.0 + EDR (3 Mbps)
  - WiMedia Alliance (53 - 480 Mbps)

Frequency Hopping Spread Spectrum (FHSS)

- **FHSS** is a method of transmitting radio signals by rapidly switching a carrier between several frequency bands, using a pseudorandom sequence known to both the transmitter and receiver.
- Main advantages over a fixed-frequency transmission:
  - Highly resistant to narrowband interference. The process of re-collecting a spread signal spreads out the interfering signal, causing it to recede into the background.
  - Difficult to intercept. An FHSS signal simply appears as an increase in the background noise to a narrowband receiver.
  - An eavesdropper would only be able to intercept the transmission if they knew the pseudorandom sequence.

FHSS - Algorithm

- A FHSS communication session is initiated as follows:
  - The initiating device sends a request via a predefined frequency or control channel.
  - The receiving device sends back a random number, known as a seed.
  - The initiating device uses this number as a variable in a predefined algorithm, which calculates the sequence of frequencies that are to be used.
  - The initiating device sends a synchronization signal using the first frequency in the calculated sequence, thus acknowledging to the receiving device that it has correctly calculated the sequence.
  - The two devices can now communicate, and both devices change their frequencies using the calculated sequence, starting at the same point in time.

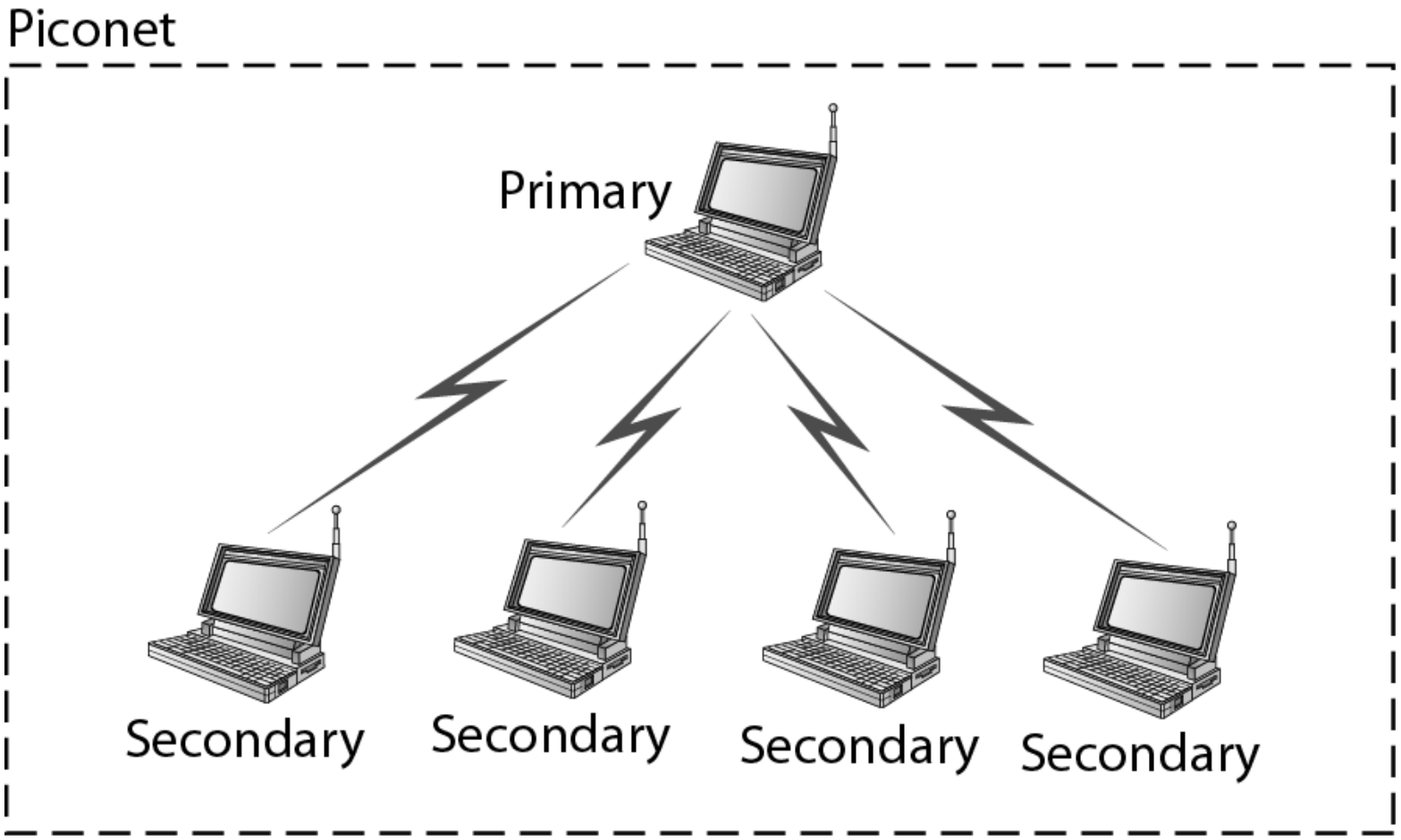
Baseband and Link Controller

- The Bluetooth specification doesn’t establish a clear distinction between the responsibilities of the baseband and those of the link controller.
- **Baseband Layer**
  - The best way to think about it is that the baseband portion of the layer is responsible for properly formatting data for transmission to and from the radio layer.
  - In addition, it handles the synchronization of links.
- **Link Controller Layer**
  - The link controller portion of this layer is responsible for carrying out the link manager’s commands and establishing and maintaining the link stipulated by the link manager.

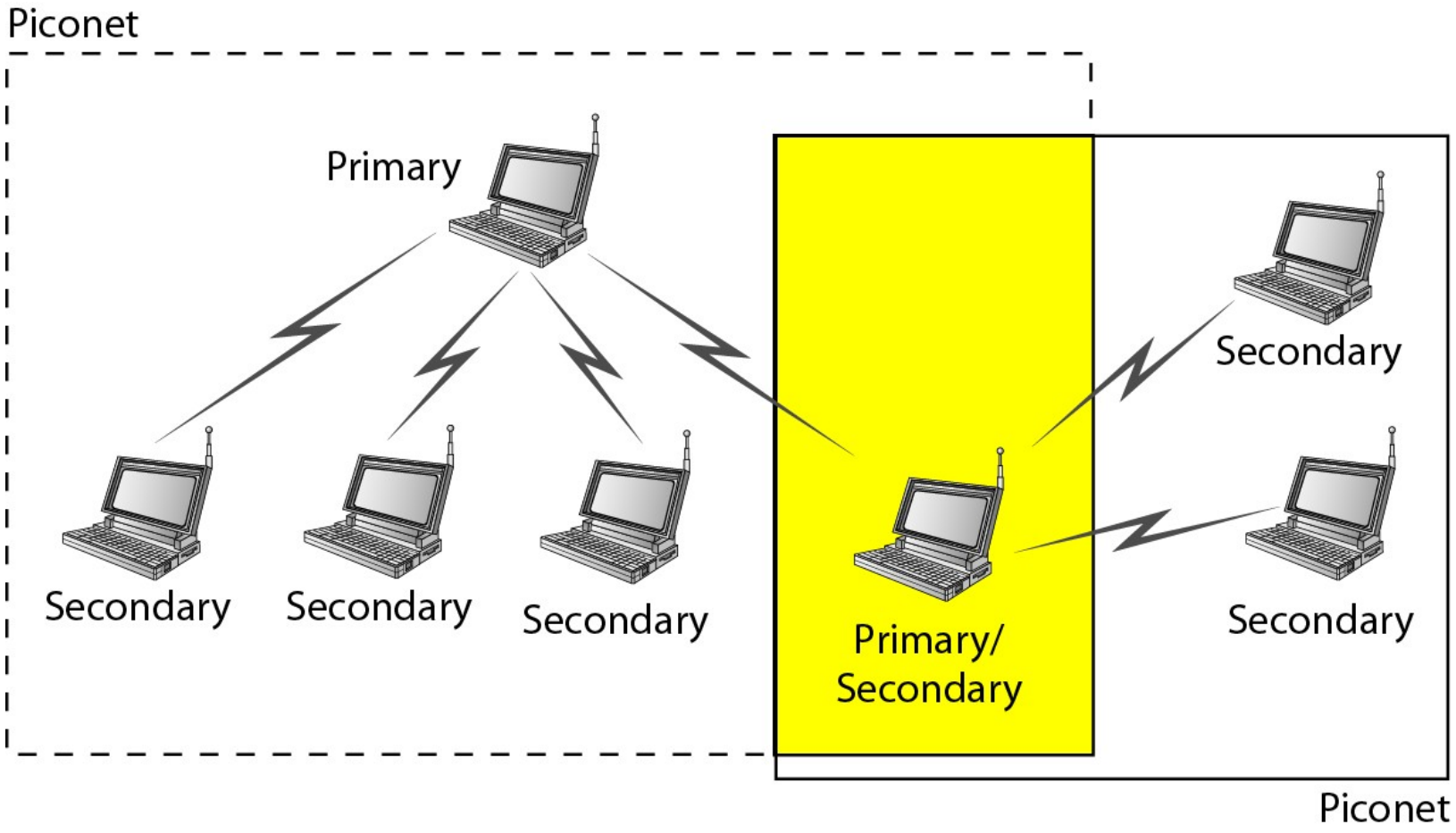
Baseband Layer

- Consists of Piconets and Scatternets:
- Up to 7 active slaves per piconet
- Up to 255 parked (synchronized) slaves per piconet
- Multiple piconets with overlapping coverage form a scatternet.
- Each piconet within a scatternet has its own hopping sequence.

Piconet


Piconet
Primary
Secondary
Secondary
Secondary
Secondary

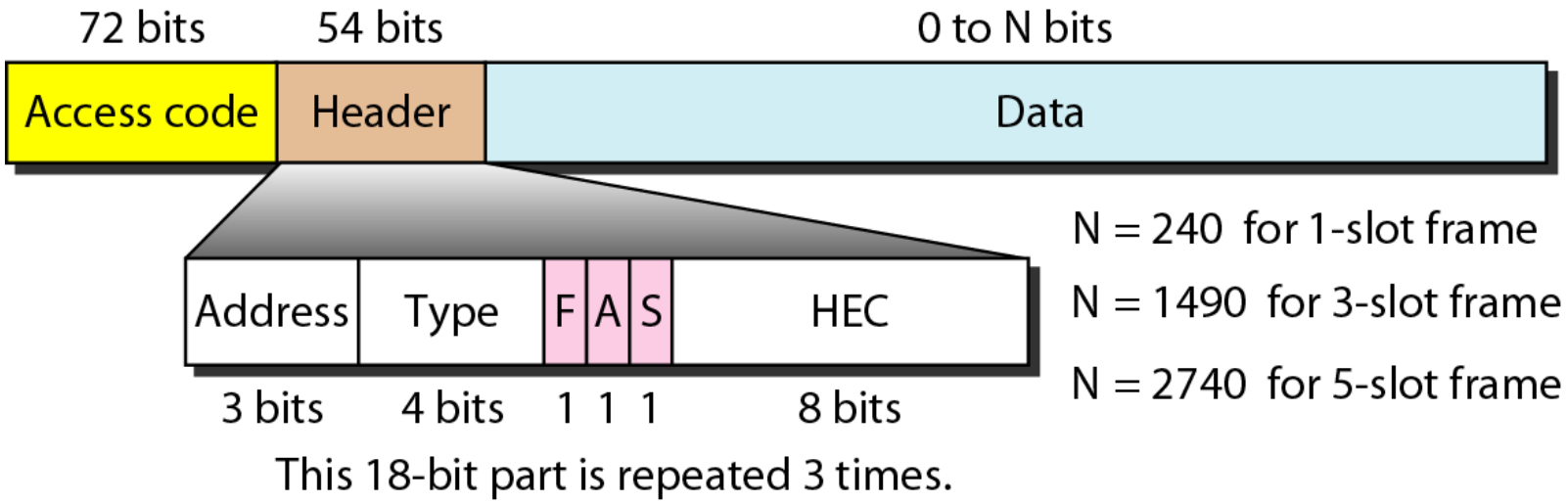
Scatternet


Piconet
Primary
Secondary
Secondary
Secondary
Primary/
Secondary
Secondary
Secondary
Piconet

Baseband

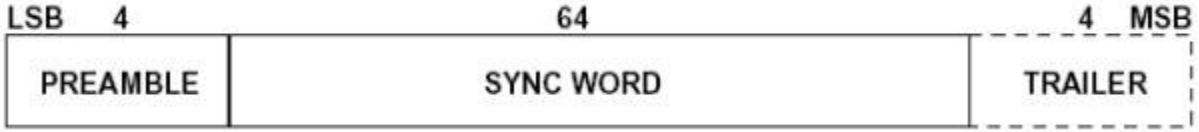
- There are two types of Bluetooth devices: master and slave. A unit is master if it is an originator and arbiter in a given piconet.
- The master device defines the frequency hop sequence and all data transfers in the piconet.
- Each slave device must to be synchronized with master’s clock.
- Each piconet has no more than seven active slave devices and many others may remain locked in so called parked state.
- Access to all these devices is controlled by the master device.
- A channel in Bluetooth is represented by a pseudo-random hopping sequence.
- The hopping sequence changes its value regularly and can be decoded only with the master’s clock and hopping sequence.

Piconet Data Frame Format



Access Code

- The Access Code is used for synchronization, DC offset compensation and identification:



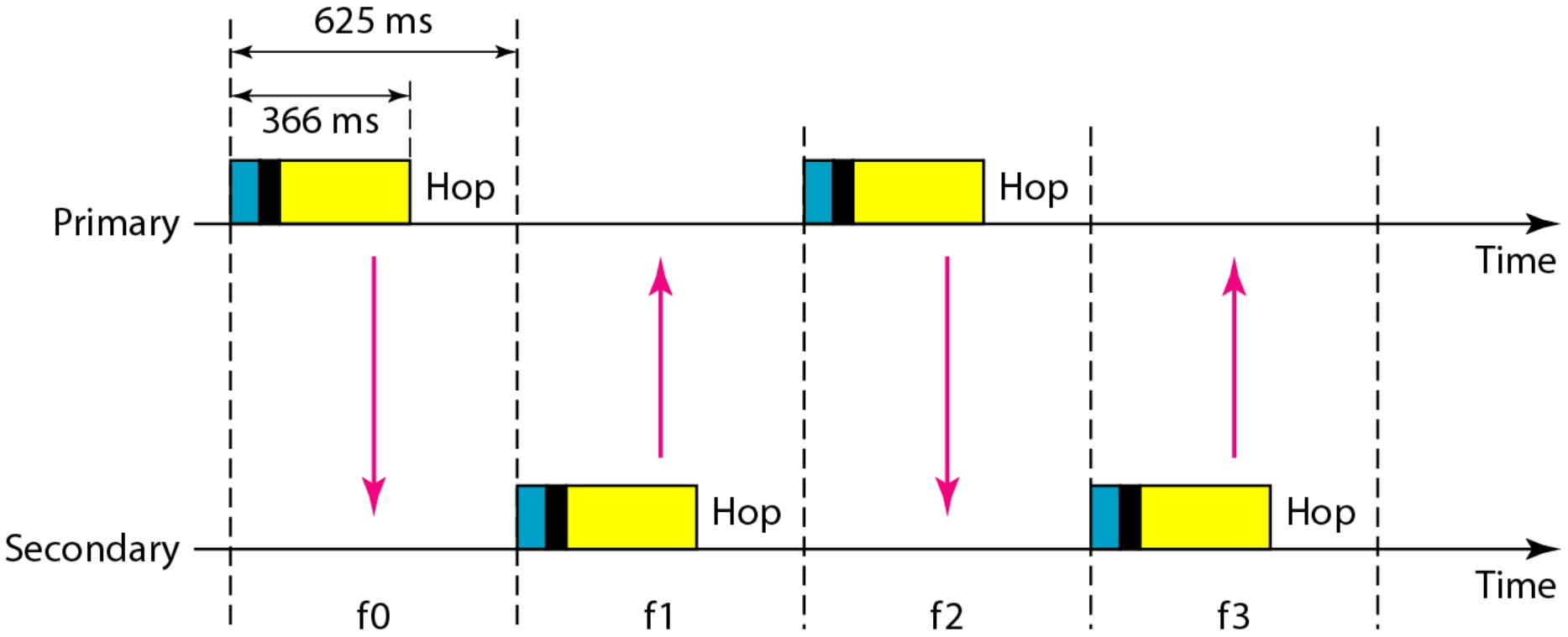
- There are three kinds of access codes:
- Channel Access Code (CAC) is used to identify the piconet. All packets sent through one channel of the piconet carries the address of the master device.
- Device Access Code (DAC) is used for special signaling procedures, such as paging and response to paging. A DAC for paging carries the address of paged device.
- Each BT device has a unique address called BD_ADDR, containing two parts:
  - company ID which is unique across the world
  - device ID which is unique within the products of the company.
- The Sync Word of the access code is derived from a BD_ADDR address using (64,30) expurgated block code with an overlay of an 64-bit full length PN sequence.

Header

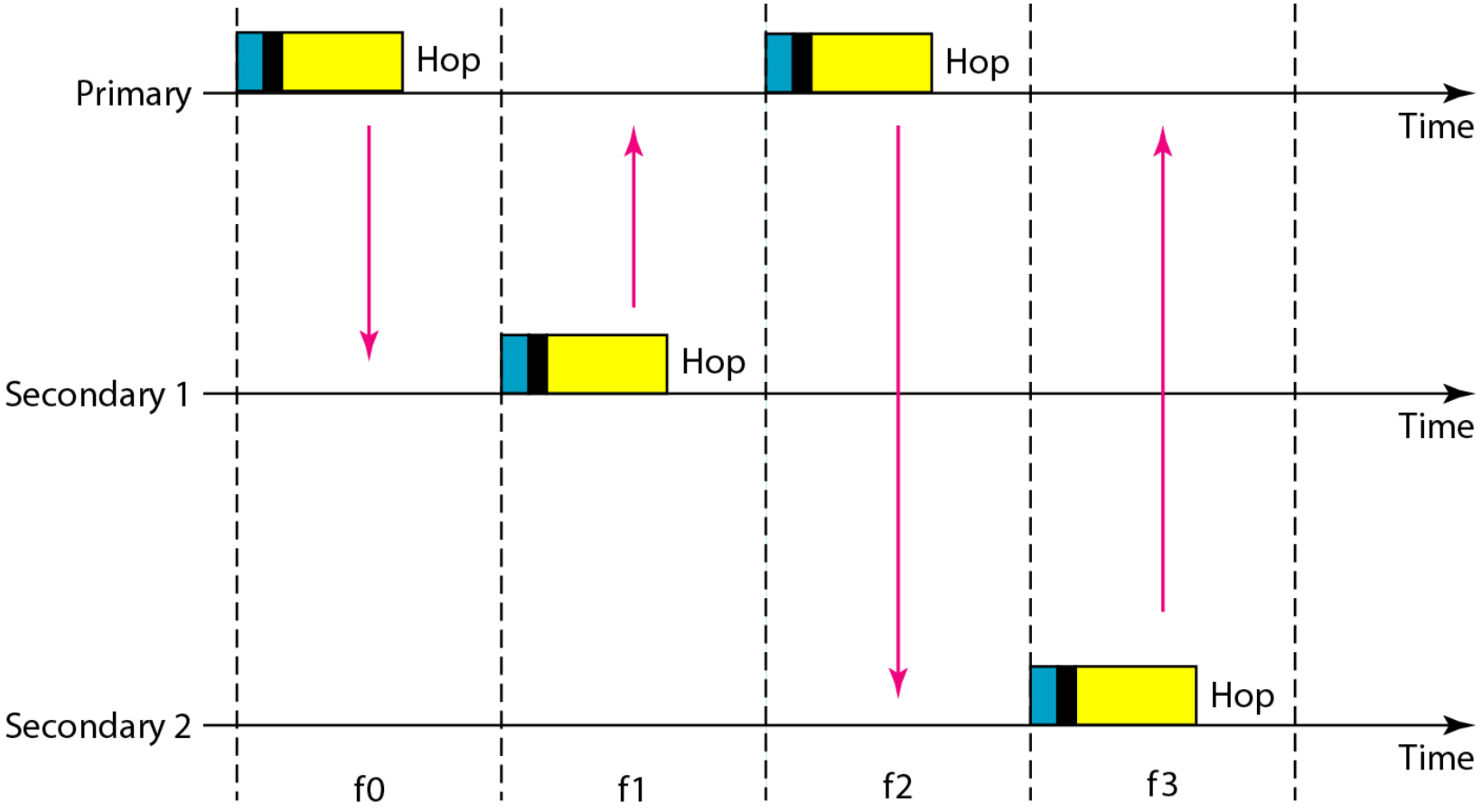
- The header details are as follows:

- AM_ADDR: temporary address assigned to active members of the piconet, used on all packets in both direction sent between the master and the addressed slave. An all-zero AM_ADDR is used to broadcast to all slaves.
- TYPE: type of packet. There are 12 types of packets for each SCO and ACL physical links, and four types of common control packets for both.
- FLOW: for flow control.
- ARQN: for ACK.
- SEQN: contains sequence number for packet ordering.
- HEC: header error check for header integrity.

Single-secondary communication



Multiple-secondary communication



Link Manager Protocol Layer (LMP)

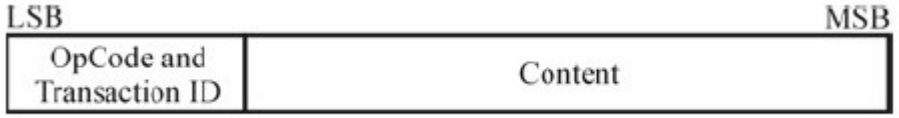
- The LMP translates the Host Controller Interface (HCI) commands it receives into baseband-level operations.
- LMP is the logical layer which manages all dedicated packet transfers between devices in a Bluetooth piconet.
- It is responsible for establishing and configuring links and managing power-change requests, among other tasks.
- The Bluetooth specification defines two types of links between Bluetooth devices:
  - **Synchronous, Connection-Oriented (SCO)**, for isochronous and voice communication using, for example, headsets
  - **Asynchronous, Connectionless (ACL)**, for data communication, such as the exchange of

Link Manager Protocol Layer (LMP)

- Each link type is associated with a specific packet type.
- A SCO link provides reserved channel bandwidth for communication between a master and a slave, and supports regular, periodic exchange of data with no retransmission of SCO packets.
- An ACL link exists between a master and a slave the moment a connection is established.
- The data packets Bluetooth uses for ACL links all have 142 bits of encoding information in addition to a payload that can be as large as 2712 bits.
- The extra amount of data encoding heightens transmission security. It also helps to maintain a robust communication link in an environment filled with other devices and common noise.

Link Manager Protocol Layer (LMP)

- The LMP registers store all the data for global settings and parameters of a device.
- An example is support of maximum time slots per packet, time intervals for dedicated data transmissions, etc.
- The transmission of all LMP messages is performed in one time slot.
- The structure of the LMP message is shown below:



Host Controller Interface (HCI) Layer

- This layer acts as a boundary between the lower layers of the Bluetooth protocol stack and the upper layers.
- The Bluetooth specification defines a standard HCI to support Bluetooth systems that are implemented across two separate processors.
- For example, a Bluetooth system on a computer might use a Bluetooth module‘s processor to implement the lower layers of the stack (radio, baseband, link controller, and link manager).
- It might then use its own processor to implement the upper layers (L2CAP, RFCOMM, OBEX, and selected profiles).

HCI Layer

- The HCI defines how a computer (**host**) interacts and communicates with a local BT adapter (**controller**).
- All communications that occur between the two are encapsulated within HCI packets.
- Because the Bluetooth HCI is well defined, we can design and implement drivers that handle different Bluetooth modules from different manufacturers.
- There are four types of HCI packets:
  - Command Packets
  - Event Packet
  - ACL Data Packet
  - Synchronous (SCO) Voice Data Packet

- HCI does not provide the ability to differentiate the four HCI packet types.
- Therefore, if the HCI packets are sent via a common physical interface, a HCI packet indicator has to be added according to table below

HCI packet type	HCI packet indicator
Command Packet	0x01
ACL Data Packet	0x02
SCO Data Packet	0x03
Event Packet	0c04

- The HCI packet indicator is sent immediately before the HCI packet.
- All four kinds of HCI packets have a length field, which is used to determine how many bytes are expected for the HCI packet.
- When an entire HCI packet has been received, the next HCI packet indicator is expected for the next HCI packet.

Command Packet

- Sent from the host to the controller, and Is used to control the adapter.
- Used to start a device inquiry, connect to a remote device, adjust connection parameters, etc.
- Packet contains:
  - Opcode that identifies the command type
  - Field specifying the total length
  - Parameter fields

Event Packet

- Generated by the controller and sent to the host whenever and event of interest occurs.
- Common events include a detected device, connection establishment, information about a local BT adapter.
- Packet contains:
  - The event code
  - Total length
  - Paremeters

ACL Data Packet

- ACL packets encapsulate data destined for or received a remote controller.
- In that sense it is a Transport protocol for higher level transport protocols such as:
  - ACL
  - L2CAP
  - RFCOMM
- Packet contains:
  - Connection handle identifying ACL connection for data
  - Packet Boundary (PB) flag whether packet data carries the start of a higher layer packet or is a fragment
  - Broadcast Flag (BC) identifying point-to-point data from Broadcast data
  - Total length field

SCO Voice Packet

- These are also encapsulated within HCI packets when transmitted from the host to the BT adapter.
- Characteristics are the same as the ACL data packets.

Upper Layers: Transport Protocols

- Bluetooth and TCP/IP are similar in that they involve using numerous different transport protocols, some of which are stacked on top of others.
- In TCP/IP, applications use either TCP or UDP, both of which rely on IP as an underlying transport.
- TCP provides a connection-oriented method of reliably sending data in streams, and UDP provides a thin wrapper around IP that unreliably sends individual datagrams of fixed maximum length.
- While Bluetooth does not have the exact equivalent protocols, it does provide protocols

The Radio Frequency Communications (RFCOMM) Protocol

- Provides roughly the same service and reliability guarantees as TCP.
- In general, applications that use TCP are concerned with having a point-to-point connection over which they can reliably exchange streams of data.
- If a portion of that data cannot be delivered within a fixed time limit, then the connection is terminated and an error is delivered.
- The biggest difference between TCP and RFCOMM from a network programmer's perspective is the choice of port number. Whereas TCP supports up to 65535 open ports on a single machine, RFCOMM only allows for 30

The Logical Link Control and Adaptation (L2CAP) Protocol

- L2CAP can be compared to UDP, which is a best-effort, packet-based protocol, but there are enough differences that the use cases for L2CAP are much broader than UDP.
- L2CAP, by default, provides a connection-oriented protocol that reliably sends individual datagrams of fixed maximum length.
- Can be configured for varying levels of reliability. To provide this service, the transport protocol employs a transmit/acknowledgement scheme, where unacknowledged packets are retransmitted based on one of three policies:
  - never retransmit
  - retransmit until total connection failure (the default)
  - drop a packet and move on to queued data if a packet hasn't been acknowledged after a specified time limit (0-1279 milliseconds). This is useful when data must be transmitted in a timely manner.

L2CAP data packet format

- The table below summarizes the characteristics of the protocols and their delivery semantics:

Requirement	Internet	Bluetooth
Reliable, streams-based	TCP	RFCOMM
Reliable, datagram	TCP	RFCOMM or L2CAP with infinite
Best-effort, datagram	UDP	L2CAP (0-1279 ms retransmit)

The Asynchronous Connection-Oriented Logical (ACL) Protocol

- All L2CAP connections are encapsulated within ACL connections.
- Since RFCOMM connections are transported within L2CAP connections, they are also encapsulated within ACL connections.
- ACL is similar to IP in that it is a fundamental protocol that is used to

The Synchronous Connection-Oriented (SCO) Logical Transport

- This is a best-effort packet-based protocol that is used exclusively to transmit voice-quality audio at 64 kbps.
- A typical application for this is a BT headset.
- SCO packets are not reliable and are not retransmitted.
- However, an SCO connection is guaranteed to have a 64 kbps transmission rate.
- To ensure that all SCO connections have this guarantee, no BT device is permitted to have more than 3 active SCO connections.

Service Discovery Protocol (SDP)

- SDP defines actions for both servers and clients of Bluetooth services. The specification defines a service as any feature that is usable by another (remote) Bluetooth device.
- A single Bluetooth device can be both a server and a client of services. An example of this is a PC itself. Using the file transfer profile the computer can browse the files on another device and allow other devices to browse its files.
- An SDP client communicates with an SDP server using a reserved channel on an L2CAP link to find out what services are available. When the client finds the desired service, it requests a separate connection to use the service.
- The reserved channel is dedicated to SDP communication so that a device always knows how to connect to the SDP service on any other device. An SDP server maintains its own SDP database, which is a set of service records that describe the services the server offers.
- Along with information describing how a client can connect to the service, the service record contains the service's **UUID**, or

Object Exchange (OBEX)

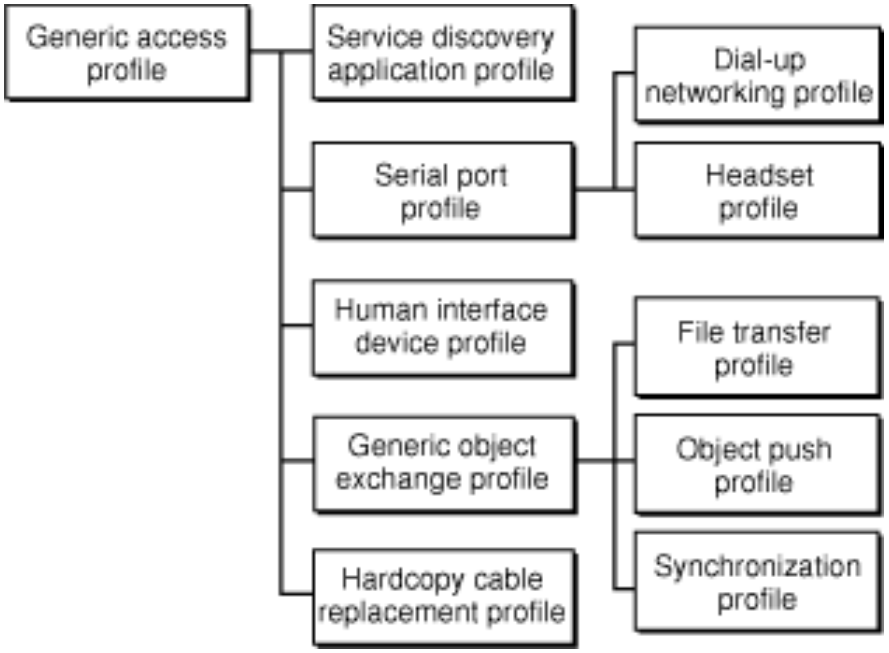
- OBEX is a transfer protocol that defines data objects and a communication protocol that two devices can use to easily exchange those objects.
- Bluetooth adopted OBEX from the IrDA IrOBEX specification because the lower layers of the IrOBEX protocol are very similar to the lower layers of the Bluetooth protocol stack.
- A Bluetooth device wanting to set up an OBEX communication session with another device is considered to be the client device.
- The client first sends SDP requests to make sure the other device can act as a server of OBEX services.
- If the server device can provide OBEX services, it responds with its OBEX service record. This record contains the RFCOMM channel number the client should use to establish an RFCOMM channel.
- Further communication between the two devices is conveyed in packets, which contain requests and responses, and data. The format of the packet is defined by the OBEX session protocol.
- Although OBEX can be supported over TCP/IP, this is not described in the Bluetooth specification.

The Bluetooth Profiles—A Hierarchy of Groups

- The Bluetooth specification defines a wide range of profiles, describing many different types of tasks.
- By following the profiles's procedures, developers can ensure that the applications they implement will work with any device that conforms to the BT specification.
- At a minimum, each profile specification contains information on the following topics:
  - **Dependencies on other profiles.** Every profile depends on the base profile, called the generic access profile, and some also depend on intermediate profiles.
  - **Suggested user interface formats.** Each profile describes how a user should view the profile so that a consistent user experience is maintained.
  - **Specific parts of the Bluetooth protocol stack used by the profile.** To perform its task, each profile uses particular options and parameters at each layer of the stack. This may include an outline of the required service record, if appropriate.

Profile Hierarchy

- Profiles are organized into a hierarchy of groups, with each group depending upon the features provided by its predecessor.
- The following diagram illustrates the dependencies of the Bluetooth profiles.



The Base (Generic) Profile

- At the base of the profile hierarchy is the Generic Access Profile (GAP), which defines a baseline means to establish a baseband link between Bluetooth devices.
- The GAP also defines:
  - Features that must be implemented on all Bluetooth devices
  - Generic procedures for discovering and linking to devices
  - Basic user-interface terminology
- All other profiles are based on the GAP, thus allowing each profile to take advantage of the features the GAP provides and ensuring a high degree of interoperability between applications

Remaining Profiles

- **Service Discovery Application Profile**
  - Describes how an application should use the SDP to discover services on a remote device.
  - Since any Bluetooth device should be able to connect to any other Bluetooth device, this profile requires that any application be able to find out what services are available on any Bluetooth device it connects to.
- **Human Interface Device (HID) profile**
  - Describes how to communicate with a HID class device using a Bluetooth link.
  - It describes how to use the USB HID protocol to discover a HID class device’s feature set and how a Bluetooth device can support HID services using the L2CAP layer.

- **Serial Port Profile**
  - Defines RS-232 serial-cable emulation for Bluetooth devices. This allows legacy applications to use Bluetooth as if it were a serial-port link, without requiring any modification.
  - The serial port profile uses the RFCOMM protocol to provide the serial-port emulation.
- **Dial-Up Networking (DUN) profile**
  - Built on the serial port profile and describes how a data-terminal device, such as a laptop computer, can use a gateway device, such as a mobile phone or a modem, to access a telephone-based network.
  - Like other profiles built on top of the serial port profile, the virtual serial link created by the lower layers of the Bluetooth protocol stack is transparent to applications using the DUN profile.
  - Thus, the modem driver on the data-terminal device is unaware that it is communicating over Bluetooth.

- **Headset Profile**
  - Describes how a Bluetooth-enabled headset should communicate with a computer or other Bluetooth device (such as a mobile phone).
  - When connected and configured, the headset can act as the remote device’s audio input and output interface.
- **Hardcopy Cable Replacement Profile**
  - Describes how to send rendered data over a Bluetooth link to a device, such as a printer.
  - Although other profiles can be used for printing, the HCRP is specially designed to support hardcopy applications.

- **Generic Object Exchange Profile**
  - Provides a generic blueprint for other profiles using the OBEX protocol and defines the client and server roles for devices.
  - As with all OBEX transactions, it stipulates that the client initiate all transactions.
  - These implementation details of the transactions are left to the profiles that depend on the generic object exchange profile, namely the object push, file transfer, and synchronization profiles.
- **Object Push Profile**
  - Defines the roles of push server and push client.
  - These roles are analogous to and must interoperate with the server and client device roles the generic object exchange profile defines.

- **File Transfer Profile**
  - Dependent on the generic object exchange profile.
  - It provides guidelines for applications that need to exchange objects such as files and folders, instead of the more limited objects supported by the object push profile.
  - The file transfer profile also defines client and server device roles and describes the range of their responsibilities in various scenarios.
  - For example, if a client wishes to browse the available objects on the server, it is required to support the ability to pull from the server a folder-listing object.
  - Likewise, the server is required to respond to this request by providing the folder-listing object.

- **Synchronization Profile**
  - Also dependent on the generic object exchange profile.
  - Describes how applications can perform data synchronization, such as between a personal data assistant (PDA) and a computer.
  - The synchronization profile focuses on the exchange of personal information management (PIM) data, such as a to-do list, between Bluetooth-enabled devices.
  - A typical usage of this profile would be an application that synchronizes your computer's and your PDA's versions of your PIM data.
  - Also describes how an application can support the automatic synchronization of data—in other words, synchronization that occurs when devices discover each other, rather than at a user's command.