Chapter 10

Bluetooth

Bluetooth is the oldest and the most pervasive technology to connect a wide range of devices and 'things' around us. Since its inauguration decades ago, it has gone through several upgrades and is continuing to play a dominant role in providing short-range connectivity for smart objects. In this chapter, we cover its history, markets and applications, followed by the core technologies behind the three generations of Bluetooth.

10.1 Bluetooth History

The history of Bluetooth started with Ericsson's Bluetooth Project in 1994 for radio communication between cell phones over short distances [NIST-BT]. It was named after Danish king Harald "Blatand" Gormsson (AD 940-981). He was fondly called Blatand, which blue-tooth in Danish, because of his dead tooth that looked blue [BT-SIG-ORG].

Intel, IBM, Nokia, Toshiba, and Ericsson formed Bluetooth SIG in May 1998 [NIST-BT,WIKI-BT]. Soon after, Version 1.0A of the specification came out in late 1999. IEEE 802.15.1, which was approved in early 2002, was based on Bluetooth [NIST-BT]. However, all later versions of Bluetooth were handled by Bluetooth SIG directly.

The key features of the original Bluetooth were low power, low cost, and small form factor. Bluetooth now comes built-in with many systems on chip and microcomputer boards, such as Intel Curie, Raspberry Pi, Arduino, and so on.

10.2 Wireless Personal Area Networks

Figure 9.1 shows the IEEE networking technologies stacked according to their communication ranges. At the bottom, we have the networks that cover the last 10m. These are collectively referred to as Wireless Personal Area Networks (WPANs), as historically these protocols were designed to serve devices within the vicinity of the person. In the IoT era, with the growing dependence on machine-to-machine communications, the name "personal" may not be very relevant for all scenarios, but the main criteria of 10m coverage will remain. Within WPAN, we have several competing solutions, such as Bluetooth, Zigbee, and Body Area Networks (BANs). In this chapter, we will focus on Bluetooth.

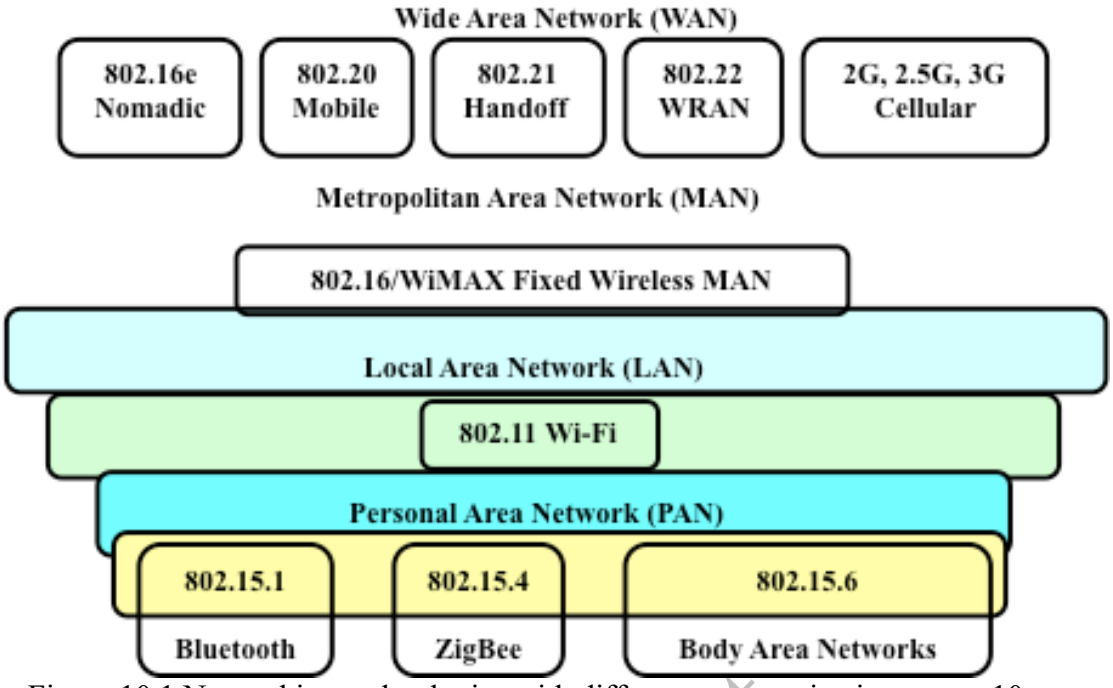


Figure 10.1 Networking technologies with different communication range. 10m or less technologies are at the bottom

All WPAN protocols follow a set of basic design principles:

Battery powered: The devices run on coin cell batteries with a couple of hundred mAh capacity, which has to last for a few years. Maximizing the battery life therefore is one of the major challenges.

Dynamic topologies: Because the devices have to conserve energy, they usually turn on for a short duration and then goes back to sleep. For example, a temperature monitor may wake up every 10 seconds and connects with the WiFi AP to send the temperature reading and then it goes back to sleep again. Therefore, connections are very short.

No infrastructure: They do not depend on any access point or base station.

Avoid Interference: These devices share the same ISM bands, such as 2.4GHz, with the high-power LAN devices, such as WiFi. How to avoid interference with such high-power communications in the same area therefore is a major issue to tackle.

Simple and Extreme Interoperability: As there are billions of devices, we have more variety than LAN or MAN. The interoperability challenge therefore is more severe than LAN or MAN.

Low-cost: Communication technology must be affordable as many low-cost devices, such as a $2 electric bulb may need such communication capabilities.

10.3 Bluetooth market

According to a recent report from Bluetooth SIG [BT-SIG], 48 billion devices will be connected to the Internet by 2021, of those 30% are forecasted to include Bluetooth technology. This includes a wide range of market segments including cars, wearables, factory instruments, and smart home products. The forecast further shows that Bluetooth shipments are expected to grow at a rate of 8% CAGR from 2019 to 2024.

10.4 Bluetooth Versions

Since the first release of Bluetooth 1.1 endorsed by the IEEE in 2002, there have been many updates over the years. The current version is 5.0. Table 10.1 provides a chronological list of Bluetooth versions and their capabilities [NIST-BT, BT-SIG]. Bluetooth versions prior to 4.0 are often referred to as Bluetooth Classic. Bluetooth 4.0 is also known as Bluetooth Smart and Bluetooth Low Energy (BLE).

Table 10.1 Chronological list of Bluetooth versions

Bluetooth Version	Description
Bluetooth 1.1 (2002)	IEEE 802.15.1-2002. Classical.
Bluetooth 1.2 (2003)	Adaptive frequency hopping to avoid frequencies with interference.
Bluetooth 2.0 + (2004)	Enhanced Data Rate (EDR); 3 Mbps using DPSK. Suitable for video applications. Reduced power due to reduced duty cycle.
Bluetooth 4.0 (2010)	Low energy. Smaller devices requiring longer battery life (several years). New incompatible PHY. A.k.a. Bluetooth Smart or BLE.
Bluetooth 5.0 (2016)	Make BLE go faster and farther.
Bluetooth 5.3 (2021)	Faster transitions between low and high duty cycle modes, enhanced key size control between host and controller, more efficient periodic advertisements, channel classification enhancement for arriving at more optimal channel map between peripheral and central devices.

10.5 Bluetooth Classic

We will start with Bluetooth 1.1 to understand the basic details of Bluetooth.

10.5.1 Bluetooth piconet

A Bluetooth network is called a piconet, which is formed by a *master* device communicating with one or more *slave* devices [PRAVIN2001]. Any Bluetooth device can become a master. Basically, the device that initiates the communication becomes the master and the devices that respond to the initial call becomes the slaves. For example, when a computer is turned on, it may advertise a message looking for a Bluetooth keyboard. If a nearby keyboard responds and subsequently pairs with the

computer, then the keyboard becomes a slave. Slaves can only transmit when requested by the master. Active slaves are polled by the master for transmissions. Slaves can only transmit/receive to/from the master, i.e., slaves cannot talk to another slave in the piconet. There can be up to 7 *active* slaves per piconet at a time.

Beyond the active slaves within a piconet, Bluetooth allocates an 8-bit parked address to any device wishing to join the piconet at some time in the future. This allows up to 255 *parked* slaves per piconet that sleep most of the time but may join the piconet from time to time. All parked stations are then uniquely identifying, and they are usually referred to using some mnemonic identifiers for human use. Any parked stations can join the piconet in 2ms and become active at any time. For other stations which are not parked yet, it usually takes much longer than 2ms to join. Figure 10.2 shows examples of Bluetooth piconets with both active and parked slaves.

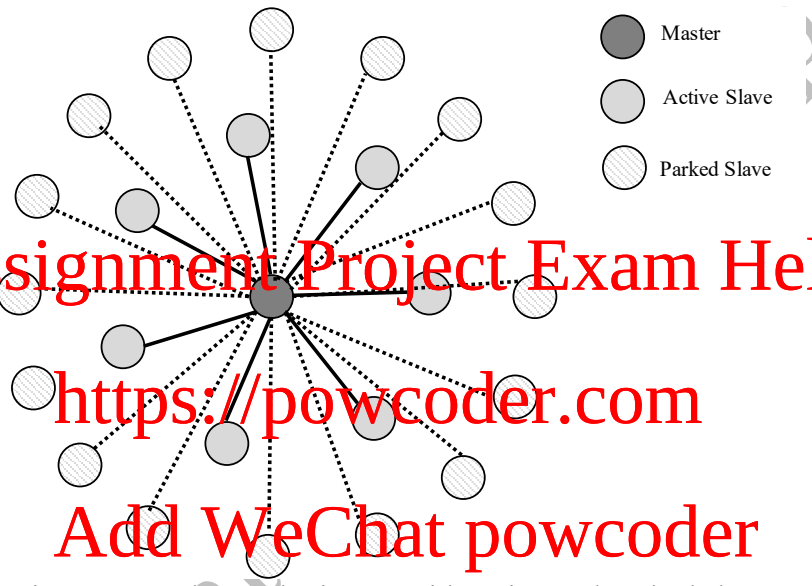


Figure 10.2 Bluetooth piconet with active and parked slaves.

For more densely deployed IoT scenarios, Bluetooth can use a more complex network topology, called *scatternet,* to allow a device to participate in multiple piconets as shown in Figure 10.3. However, for a device to participate in multiple piconets, it has to timeshare and must synchronize to the master of the current piconet, i.e., it can *be active* in only one piconet and in *park* mode in the other.

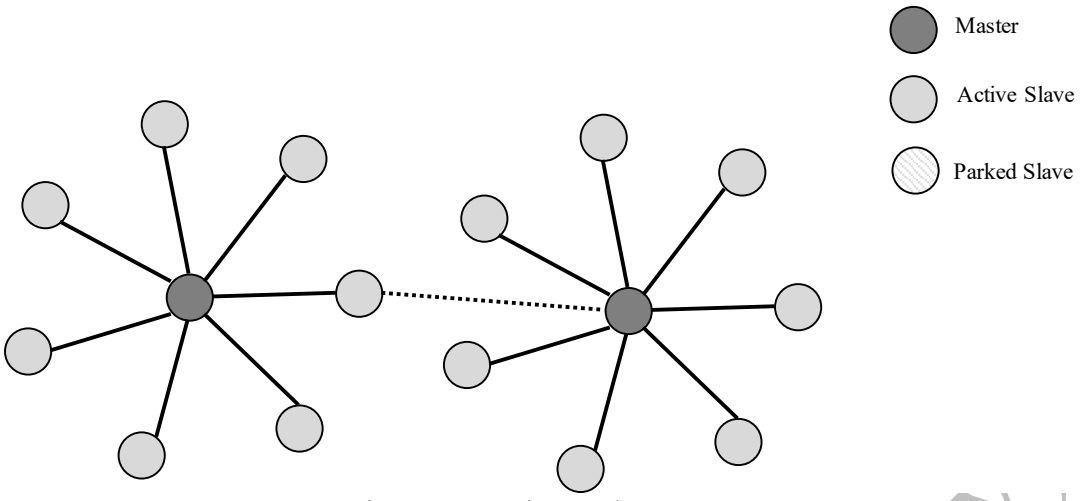


Figure 10.3 Bluetooth scatternet.

Note that there is no routing protocol defined, so nodes can only talk to other nodes which are directly within the Bluetooth communication range of about 10m.

10.5.2 Bluetooth spectrum and channels

Bluetooth operates in the same ISM band of 2.4 GHz as WiFi. Bluetooth classic divides the entire spectrum between 2402-2480 MHz (total 79 MHz) into 79 1-MHz channels as shown in Figure 10.4.

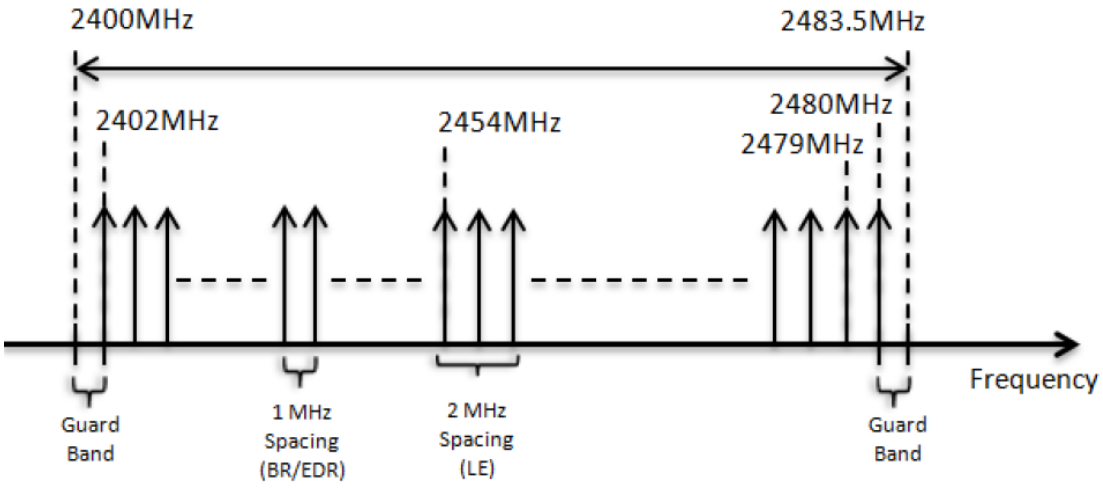


Figure 10.4 Bluetooth classic channels.

10.5.3 Modulation and data rates

Modulation and data rates for Bluetooth classic are classified into two groups, basic rate (BR) and enhanced data rate (EDR).

For BR, it uses a binary frequency shift keying to achieve only 1 bit per symbol. Bluetooth uses a symbol duration of 1μs, which gives 1Mbps for the BR. The EDR also uses 1μs symbols, but it supports more advanced modulations, namely μ/4-DQPSK with 2 bits/symbol and 8DPSK with 3 bits/symbol. Thus, under EDR, Bluetooth classic can deliver 2Mbps and 3Mbps data rates using μ/4-DQPSK and 8DPSK, respectively.

10.5.4. Frequency hopping

Unlike WiFi, Bluetooth constantly switches channel within the same connection to avoid interference with other nearby WiFi or Bluetooth communications using the same 2.4GHz band. Figure 10.5 shows how two Bluetooth networks can share the same frequency band without interfering with each other by hopping between channels (only 4 available channels are shown for illustration purposes). As we can see, the two networks are selecting different channels at each time slot, which avoids interference and collision.

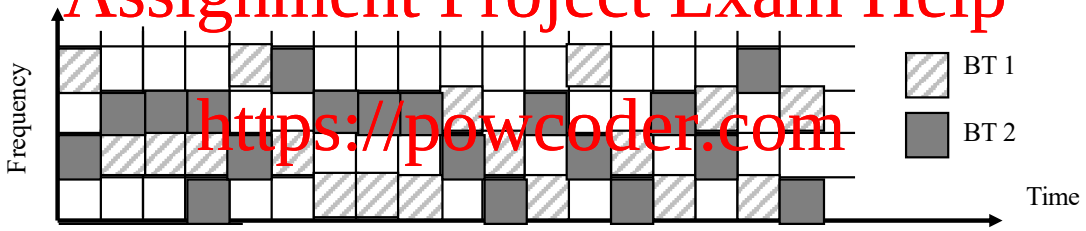


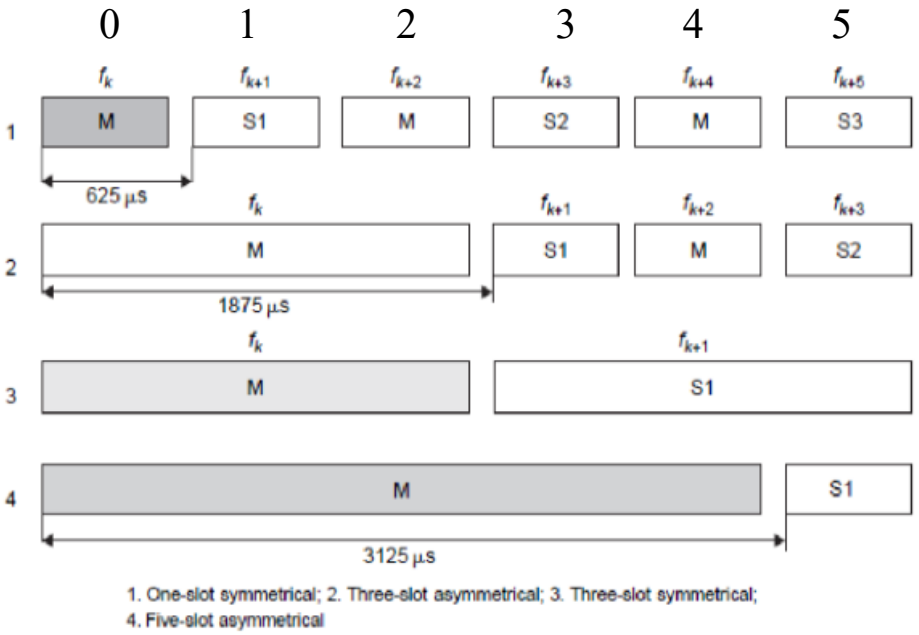
Figure 10.5 Two close by Bluetooth piconets sharing the same frequencies without interreference by hopping between channels; in this ideal hopping scenario, the same channel is never selected by both piconets.

So, how frequently Bluetooth switches the channel? Because it is rather complex to switch the channel within a packet transmission, Bluetooth only changes channels at packet boundaries effectively achieving a frequency hop per packet. No two successive packets are transmitted over the same channel. The effective frequency hopping rate therefore is a function of the packet duration.

In Bluetooth, time is slotted to 625μs using a 3200Hz clock, where 1 slot is determined by 2 clock ticks (1 clock tick = 312.5μs for a 3200Hz clock). Packet transmission can start only at the beginning of a time slot and it can last for 1, 3, or 5 slots. Other packet durations are not permitted.

The communication between the master and slave uses the entire frequency band, so both of them cannot transmit at the same time, i.e., full-duplex communication is not possible in Bluetooth. The master and the slave therefore alternate in using the channel, i.e., they implement half-duplex using time-division duplexing (TDD). Master-to-slave is called *downstream* and slave-to-master is called *upstream*. With TDD, downstream and upstream alternates in time.

The slots are numbered starting at zero. Master starts communicating first in slot 0 and the slaves can transmit right after receiving a packet from the master. Given that the allowed packet lengths are only 1, 3, or 5 slots, masters can only use the even numbered slots and the slaves the odd numbered slots. Frequencies switch only at the start of the slot that starts after a packet transmission is completed, which may not align with slot boundaries. Finally, the packet lengths between the master and the slaves may not have to be symmetrical. For example, it is perfectly OK for a master to use a short 1-slot packet, while a slave transmits a 3-slot packet. Figure 10.6 shows such symmetric vs. asymmetric packet lengths. Figures 10.7 illustrates that the minimum and maximum frequency hopping rates in Bluetooth are 320Hz (all packets are 5-slot) and 1600Hz (all packets are 1-slot).



1. One-slot symmetrical; 2. Three-slot asymmetrical; 3. Three-slot symmetrical;
4. Five-slot asymmetrical

M = master, S = slave

Figure 10.6 Frequency hopping for symmetric vs. asymmetric packet lengths

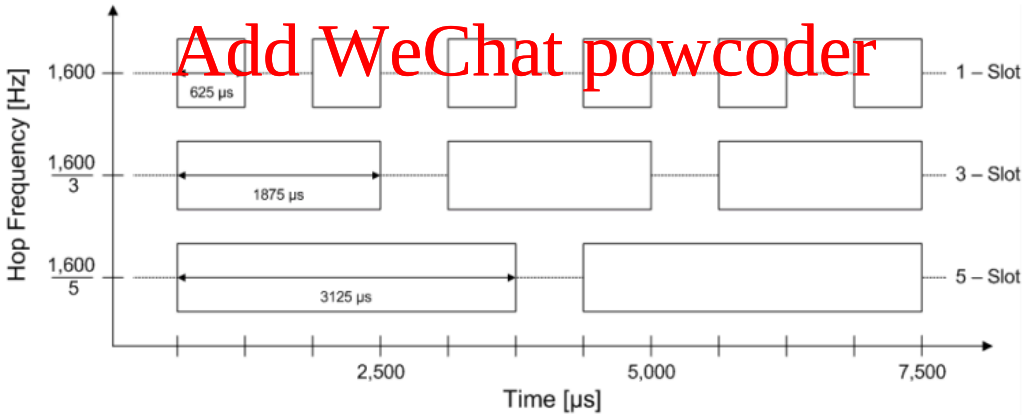


Figure 10.7 Dependency of hopping rate on the packet length. The maximum and minimum hopping rates are 1600Hz and 320Hz, respectively [BT-RS].

Example 10.1

Consider a Bluetooth link where the *master* always transmits 3-slot packets. The transmission from the master is always followed up by a single-slot transmission from

a *slave*. Assuming 625µs slots, what is the effective frequency hopping rate (# of hopping per second)?

Solution:

Given that frequency hopping cannot occur in the middle of a packet transmission, we only have 2 hops per 4 slots, or 1 hop per 2 slots.
The effective hopping rate = $1/(2\text{x}625\text{x}10^{-6})$ = 800 hops/s = 800Hz

---

Bluetooth is very popular for listening to music as well as for voice calls with earphones. Such traffic is synchronous generating packets at fixed intervals, where the interval depends on the audio codecs. To support such synchronous traffic, Bluetooth reserves slots ahead of time. For asynchronous traffic, the master simply polls each active station. Note that there is no contention avoidance mechanism; all traffic is controlled by the master. If there are contentions, packets get lost, which are eventually retransmitted by the higher layer.

10.5.5 Bluetooth Packet Format and MAC address

In Bluetooth, packets can be up to 5 slots long. Thus, with 625µs slots, a packet is allowed to last for a maximum of 625x5=3125µs. As BR and EDR have different data rates, their packet formats are also different. The packet format for BR is shown in Figure 10.8, which has only three fields [BT-NI]: It has a 68/72b access code, 54b control header and the rest is payload, which can be up to 2745b. The maximum size of a packet therefore can be 72+54+2745 = 2871b, which lasts for 2871µs when data is transmitted at the rate of 1Mbps.

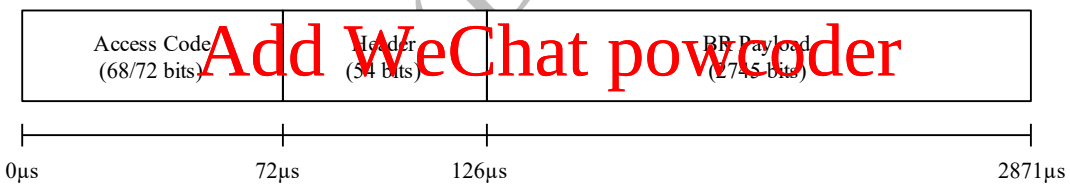


Figure 10.8 Bluetooth classic packet format for BR

There are 3 types of access codes: *Channel access code* (CAC) identifies the piconet, *Device access code* (DAC) is used for paging requests and response, and *inquiry access code* (IAC) can be used to discover particular Bluetooth devices in the vicinity. CAC is 72 bits long, but DAC and IAC can be either 68 or 72 bits long. There is an 18-bit header comprising member address (3b), type code (4b), flow control (1b), ack/nack (1b), sequence number (1b), and header error check (8b). This 18b Header is encoded using 1/3 rate coding resulting in 54b.

The packet format for EDR is more complex, which is shown in Figure 10.9. A notable feature of an EDR packet is that the modulation changes within the packet. While GFSK is used for the access code and header fields, it switches to DPSK (DQPSK for 2Mbps and 8DPSK for 3Mbps) after a guard interval lasting between 4.75µs to 5.25µs. EDR payload can accommodate more data than BR, but still fits within the maximum 5-slot due to higher data rates.

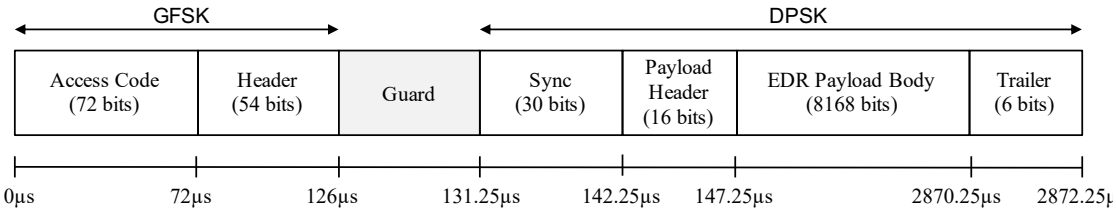


Figure 10.9 Bluetooth classic packet format for EDR

Example 10.2

How many slots are needed to transmit a Bluetooth Basic Rate packet if the payload is (a) 400 bits, (b) 512 bits, and (c) 2400 bits. Assume that the non-payload portions do not change.

Solution:

Bluetooth transmissions are 1, 3, or 5 slots (2, 4, 6, etc. not allowed)
Non-payload bits (max) = 54+72 = 126 bits
Each slot can carry 625 bits at most
(a) 400b payload results in 400+126 = 526b packet, which requires **1 slot**
(b) 512b payload results in 512+126 = 638b packet for which 2 slots would be sufficient, but will have to be padded for a **3-slot** transmission because 2-slot packets are not allowed
(c) 2400b payload results in 2400+126 = 2526b packet which fits in **5 slots**

Each Bluetooth device has a unique 48-bit MAC address included in the access code field of the packet header. As shown in Figure 10.10, the most significant 24 bits represent the OUI (Organization Unique Identifier) or the Vendor ID. Typically, the vendors convert each 4b into a decimal number and show the MAC address as a string of 12 decimal digits. For example, Figure 10.11 shows the label of a Bluetooth chip from Roving Networks with a MAC address of 000666422152 where 000666 would uniquely identify Roving Networks. Here, the decimal digits for the most significant bits are written from the left. While the main purpose of the Bluetooth MAC address is for identification and authentication, specific parts of it are also used to seed the frequency hopping pseudorandom generator for synchronizing the master and slave clocks as well as to pair the devices at the beginning, which we shall examine shortly.

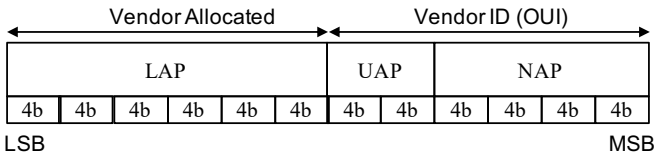


Figure 10.10 Bluetooth MAC address format

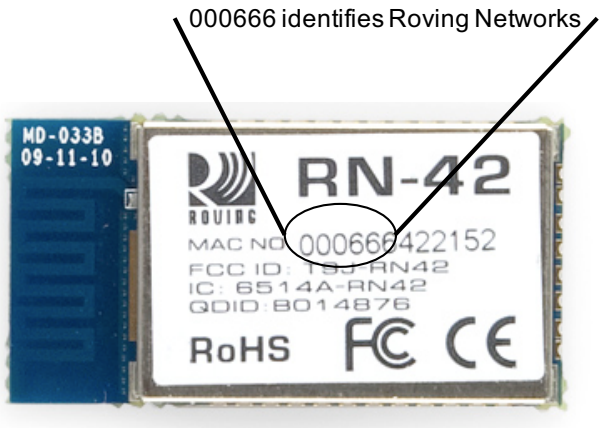


Figure 10.11 Example of a real Bluetooth MAC

10.5.6 Pseudorandom frequency hopping

In Bluetooth Classic, frequency hopping is defined by a pseudorandom generating algorithm seeded with the following values from the master device: UAP and LAP of the MAC address, and bits 1-26 (27 bits) of the 28-bit Bluetooth clock. As the master communicates these values to the slaves during connection set-up, the Master and the slaves generate the same frequency hopping patterns and switch to the same frequency values at every instant of the hop. As such, Bluetooth is both time and frequency synchronized at all times as illustrated in Figure 10.12.

Figure 10.13 shows an example trace of pseudorandom frequency hopping between a Bluetooth master and slave device that both uses single-slot packets and uses the first 16 channels for hopping. We can see that with single-slot packets, the master and slave take turns after every 612μs and switch randomly to a new channel within the 16-channel set.

The pseudorandom pattern has a finite length and hence technically it would run out of the original pattern and then repeat itself. To be more precise, with 27 clock bits to define the pattern, Bluetooth pseudorandom pattern would repeat itself after 2^{27} hops, which would take at least 23.3 hours to repeat at the maximum hopping rate of 1600Hz. In practice, the Bluetooth connections last much shorter than 23 hours, hence the pseudorandom sequence is not at risk of being repeated.

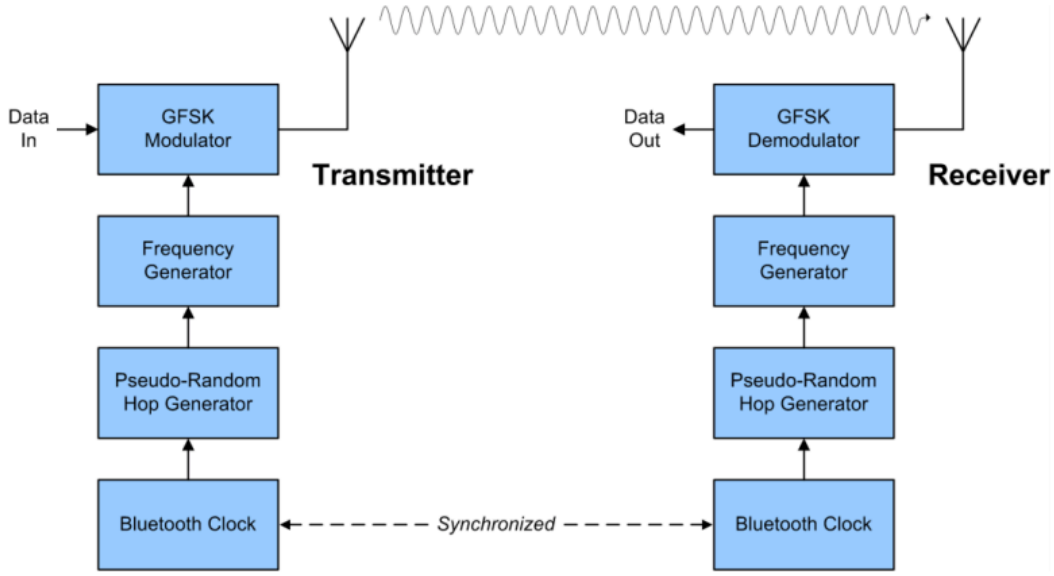


Figure 10.12 Time and frequency synchronization of Bluetooth [BT-RS]

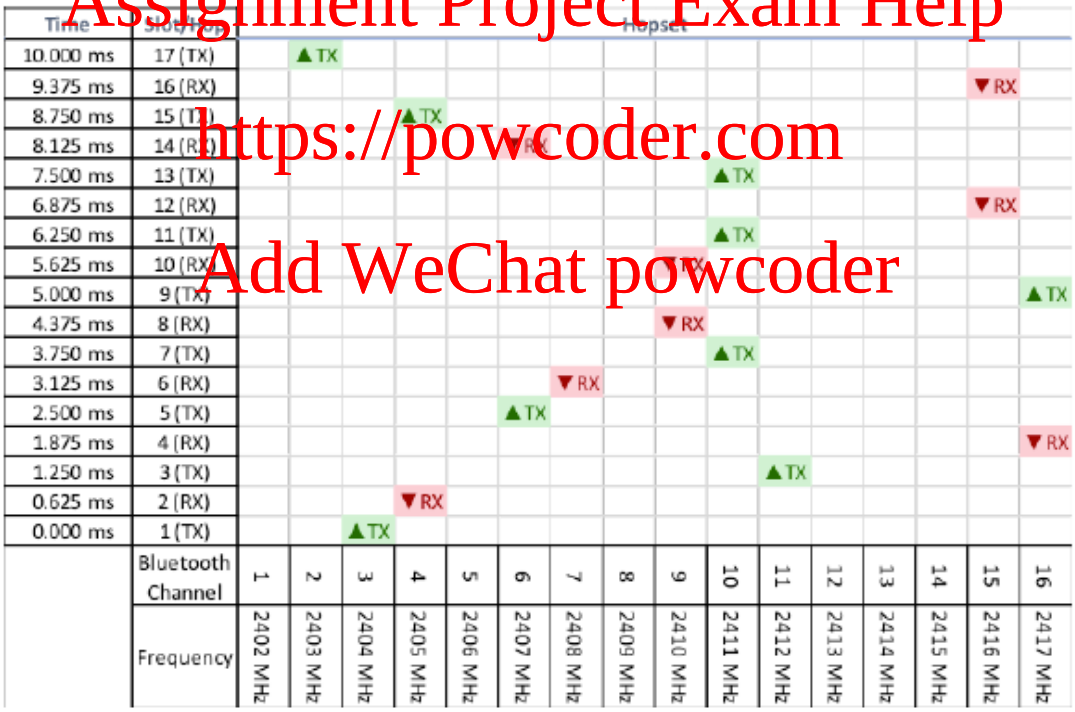


Figure 10.13 An example trace of Bluetooth pseudorandom frequency hopping

10.5.7 Adaptive frequency hopping

Because Bluetooth operates in the free unlicensed ISM spectrum, it is likely to face interference from other wireless sources, such as a 2.4GHz WiFi, sharing the same frequency of Bluetooth in the same time period. Especially, the interference from WiFi can be really harmful as WiFi uses much higher transmission power than Bluetooth. Therefore, some of the 79 channels can become unusable at certain times

and should be avoided by Bluetooth during frequency hopping. This is illustrated in Figure 10.14 where a WiFi transmission heavily interferes with the first 20 channels of Bluetooth. A Bluetooth device that always use the 79 channels for hopping would end up with transmitting some packets within those 20 interfered channels at some time causing packet and throughput loss.

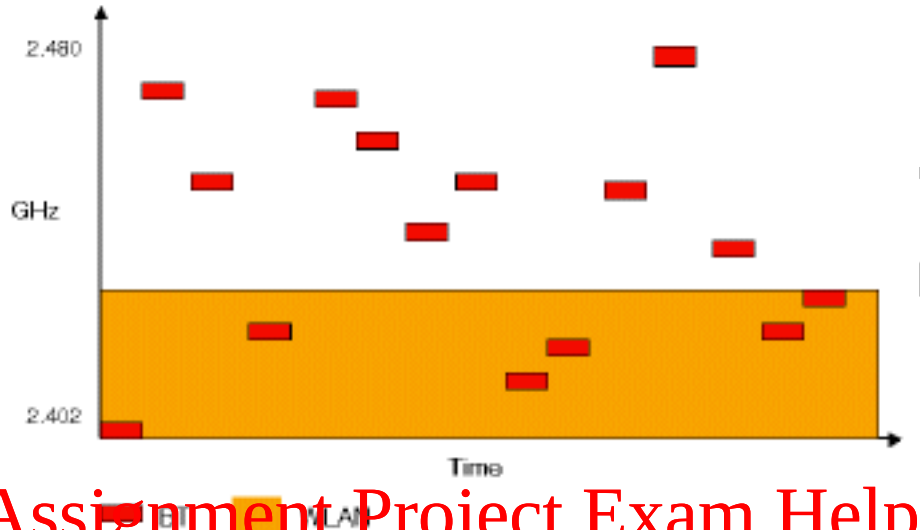


Figure 10.14 Collision with WiFi for non-adaptive Bluetooth hopping

The Adaptive Frequency Hopping (AFH) technique was proposed for Bluetooth to avoid hopping into channels that are experiencing interference. Basically, AFH requires a mechanism to measure channel states and mark interfering channels as bad channels when the interference is considered beyond certain threshold. How to implement this is left to the vendors, i.e., it is not part of the standard. Vendors usually measure metrics such as received signal strength (RSS) and signal-to-noise ratio (SNR) etc., to decide whether a channel is good or bad. Then, the hopping is constrained only within the good channels as illustrated in Figure 10.15. The standard specifies that a minimum of 20 channels are needed for hopping, i.e., a maximum of 59 channels can be marked as bad.

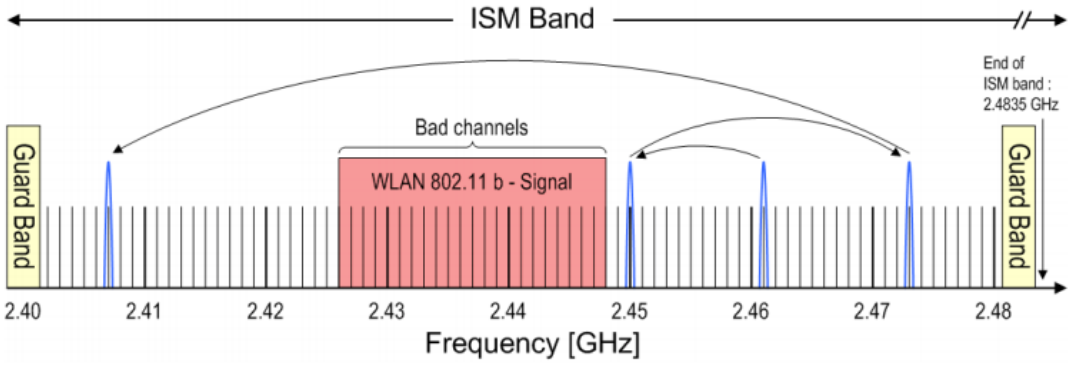


Figure 10.15 AFH Illustration: hopping only between good channels [BT-RS]

Because the interference can be dynamic, the set of good channels is likely to vary over time. Thus, the master maintains a channel map that marks good and bad channels and sends the map to the slave periodically. Figure 10.16 illustrates AFH when a WiFi 2.4GHz access point is operating over WiFi channel 6 nearby. Here the AFH successfully avoids hopping into any of the 25-45 Bluetooth channels which are marked as bad channels by the vendor's channel marker algorithm at that time.

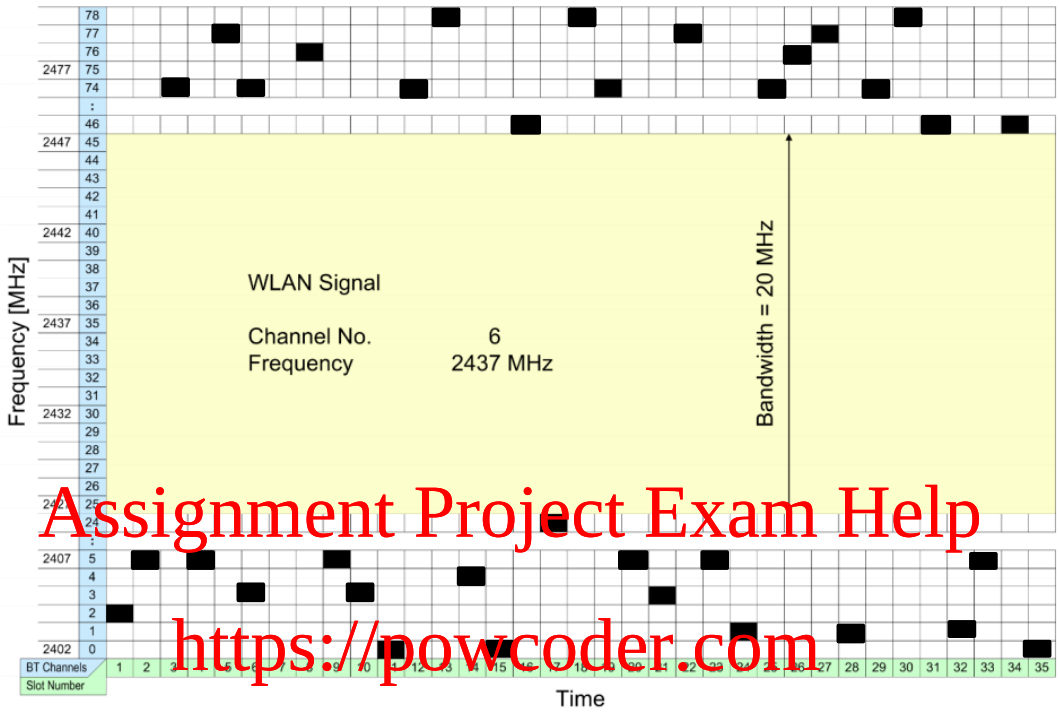


Figure 10.16 AFH Illustration [BT-RS]: Bluetooth avoids hopping into channels interfered by 2.4GHz WiFi operating over WiFi channel 6.

10.5.8 Bluetooth Operational States

There is a total of 8 distinct states as shown in Figure 10.17. These are grouped into four high level states, Disconnected, Connecting, Active, and Low Power.

Standby is the initial state when the station is disconnected but may try to connect later. There are two types of inquiry that can be used when trying to connect. The first one is called Inquiry state and the other is called Paging.

In the Inquiry state, master broadcasts an inquiry packet. Slaves scan for inquiries and respond with their address and clock after a random delay to avoid collision among many potential slaves responding at the same time. Master computes the clock offset for this slave.

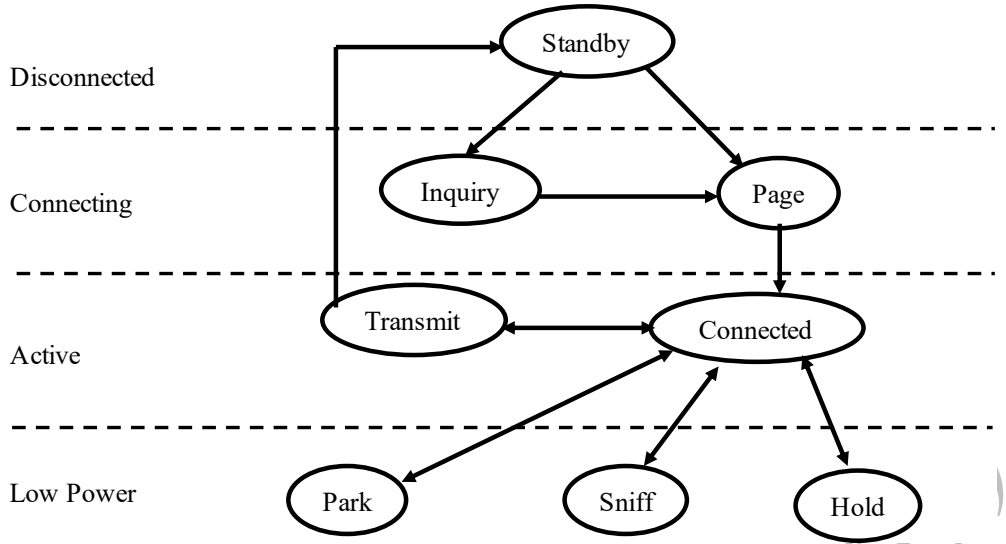


Figure 10.17 Bluetooth operational state

Master in Page state invites a slave to join the piconet. Slave enters *page response* state and sends *page response* to master including its device access code if it detected the page message. Master informs slave about its clock and address so that the slave can participate in piconet. Slave computes the clock offset.

After the page state, it transitions to the Connected state where a short 3-bit logical address (member address within control header field) is assigned for the slave. After the address assignment, the devices move to the Transmit state where they can transmit and receive a packet.

10.5.9 Bluetooth connection establishment procedure

The sequence of inquiry and paging messages exchanged between the master and the slave before the connection is established is shown in Figure 10.18. Note that the hopping sequence is only known to the master and the slave after the connection is established, which means that the inquiry and page messages has to be successfully received without exactly knowing the frequency used by the other party. This means that the master and the slave will have to guess the frequencies and try again ana again until successfully switching to the right frequencies. This makes the whole paging process indeterministic and could take several seconds before connection can be established. Let us take a closer look at how the frequency hopping is done during inquiry and paging.

First, the hopping set is constrained over a subset of 32 instead of the full 79 channels to increase the possibility of a frequency match between the transmitter and the receiver. These 32 channels are further divided into two 16-channel *trains*. For inquiry, each train is repeated 256 times before *switching* to the other train and must complete 3 train switches, i.e., 1st to 2nd to 1st and then back to the 2nd. Thus, each train is effectively repeated 256 x 2 times.

Master sends two inquiry/page packets using 2 different frequencies per slot, i.e., it hops in the middle of the slot (which leads to hopping in 312.5µs), and listens for

responses in both frequencies in the following slots. Thus, eventually 2 frequencies are covered in 2 slots.

Now, let us try to derive the worst possible time that may be needed by the inquiry process. It takes 16 x 625μs = 10ms for completing a train once. The maximum possible inquiry time is then obtained as 256x4x10ms = 10.24s. We should note that there would an additional paging time before a connection is finally established.

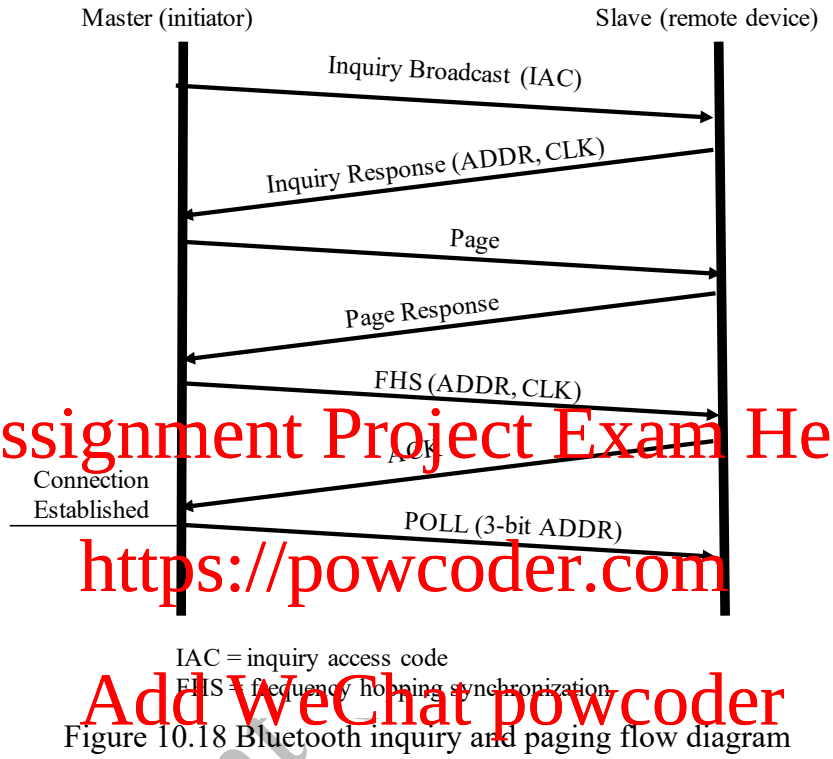


Figure 10.18 Bluetooth inquiry and paging flow diagram

10.5.10 Power saving states

Bluetooth has three inactive states to manage power saving:

Hold: In this state, no asynchronous communication can take place, but synchronous communication can continue. This means audio will not be disrupted, but the node will not initiate any other traffic, such as file transfer etc. Node can do something else though, such as scan, page or inquire. The station holds the 3-bit address as an active node of the piconet.

Sniff: It is a low-power mode to conserve energy. Slave does not continue with synchronous or asynchronous traffic but listens periodically after fixed sniff intervals. It keeps the 3-bit address.

Park: This is a very low-power mode. Slave gives up its 3-bit active member address and gets an 8-bit parked member address instead. It wakes up periodically and listens to the beacons broadcast by the Master.

10.5.11 Bluetooth Protocol Stack

Being developed almost entirely outside IEEE, Bluetooth has a slightly different shape and terminology for its protocol stack (see Figure 10.19). As we can see, it does not have a regular “layer 2”, “layer 3”, etc. stacked on top of each other. Rather, the Application can access many different services from the lower layer. Also, because there is no routing (all connections are direct single-hop), we do not have a network layer.

The bottommost layer, the RF layer, is the one that does frequency hopping. The actual bits are coded using Gaussian Frequency Shift Keying, which is basically a FSK, but the shape of the pulse is Gaussian as shown in Figure 10.20.

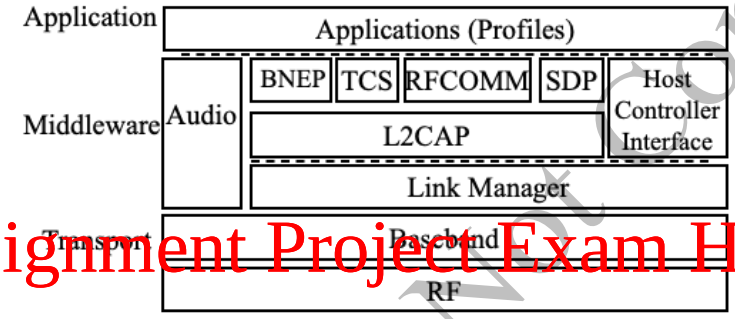


Figure 10.19 Bluetooth protocol stack

Figure 10.20 PHY coding in Bluetooth. Gaussian Frequency Shift Keying is different than Frequency Shift Keying.

Link manager negotiates parameters and sets up connections. Logical Link Control and Adaptation Protocol (L2CAP) is responsible for three tasks. It supports protocol multiplexing such as multiplexing BNEP, TCS, RECOMM, and SDP onto the logical link layer. It implements segmentation and reassembly. It also controls peak bandwidth, latency, and delay variation for different applications supporting quality of service for Bluetooth. Host Controller Interface is basically a hardware adaptation layer that enables the same software to run on different microchips.

When Bluetooth was designed, serial ports were major interfaces for many computers. Therefore, most chip manufacturers supported a serial interface to program their chips. RFCOMM layer presents a *virtual wireless* serial port capability for Bluetooth, so it can be connected to another RFCOMM. Thus, two Bluetooth devices could in practice establish a serial port connection between them over the air. In the modern age, serial ports are hardly used, but it can be utilized if necessary.

Every Bluetooth device provides a service. A Bluetooth mouse provides a mouse service, A Bluetooth keyboard provides a keyboard service, A Bluetooth speaker provides a speaker service and so on. Service Discovery Protocol (SDP) provides a standard way for devices to find such available services and their parameters so they can automatically connect to each other when turned on without human intervention.

Bluetooth can also be used to transmit standard IP data. The Bluetooth Network Encapsulation Protocol (BNEP) is used to encapsulate Ethernet/IP packets so they can be transmitted over Bluetooth.

Bluetooth also supports telephone. All modern cars are equipped with Bluetooth telephone control, so we can make or receive calls from our smartphone using buttons on the steering wheel. Telephony has audio as well as control signals. The telephony control is supported by the Telephony Control Specification (TCS) protocol. TCS support call control including group management (multiple extensions, call forwarding, and group calls).

Finally, we come to the Application Profiles. In Bluetooth, each application has a strict set of actions that it is allowed to do. For example, all actions of a headset application are defined in *headset profile* (HSP). Such strict application profiling is the key to Bluetooth's success in global and pervasive interoperability. Today we can buy a Bluetooth headset from any airport in the world and it works just fine with any mobile device from any manufacturers in any part of the world. Similarly, we have human interface device (HID) profile to wirelessly connect a range of user input devices such as mice, keyboards, joysticks, and even game controllers such as Wii PS3 controllers. Figure 10.X illustrates how HID profile is used within the Bluetooth protocol stack for connecting a wireless keyboard to the computer.

As Bluetooth became popular, number of profiles it had to support grew dramatically over the last few years. Table 10.21 inspects some of the typical Bluetooth profiles used in many products and services. With IoT, this list is expected to grow rapidly in the coming years to support communication with many different objects. For example, if we wish to connect a coffee mug to the Internet to detect object interactions for smart home residents of the future, then a coffee mug profile has to be defined.

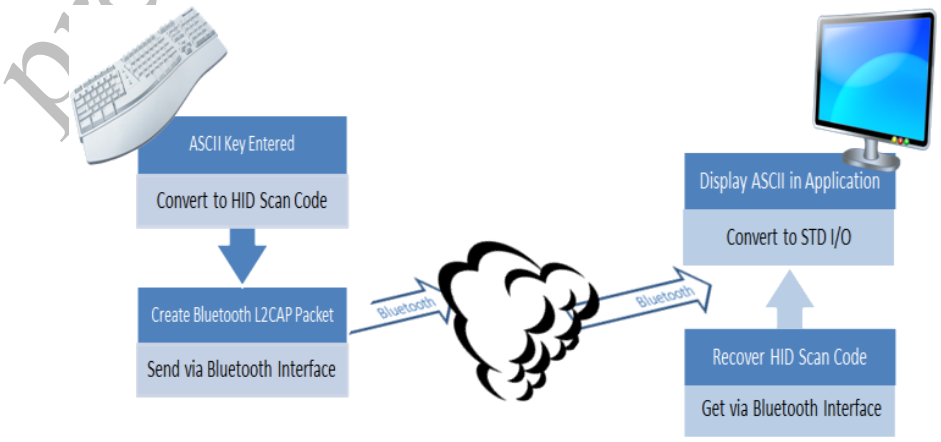


Figure 10.21 Connecting a wireless keyboard with HID Bluetooth profile [BT-SF].

Table 10.2 Examples of commonly used Bluetooth application profiles	
Bluetooth Profile	**Supporting Functions and Usage**
Headset Profile (HSP)	Defines a mono audio connection (mainly to support telephone calls) between a Bluetooth audio gateway, like a mobile phone, and a headset or earpiece. It also supports a single button to start or end a call, and the use of a volume control.
Hands-Free Profile (HFP)	Allows more extensive control of the mobile phone, such as dial numbers and voice control. Cars use this profile to integrate a mobile phone to the vehicle's audio system.
Message Access Profile (MAP)	Allows a car to access the text messages from the mobile phone.
Advanced Audio Distribution Profile (A2DP)	Supports stereo and multi-channel audio.
Human Interface Device Profile (HID)	Supports low latency and low power wireless connectivity for input devices such as mice, keyboards, joysticks, and game controllers.
AV Remote Control Profile (AVRCP)	Enables a smartphone to function as a remote control for computers and home theater equipment.
File Transfer Profile (FTP)	Once paired, a device can browse, modify, and transfer files and folders of another nearby device.

10.6 Bluetooth Low Energy a.k.a. Bluetooth 4.0

BLE is also known as Bluetooth Smart or Bluetooth 4.0. To understand the motivation behind development of BLE, we need to understand the new requirements imposed by IoT:

Low energy: It needs to reduce the energy consumption to 1% to 50% of Bluetooth classic so tiny sensors can be powered for years from a small coin cell battery without needing replacements.

Short broadcast messages: Instead of file transfers and audio streams, it needs to broadcast very short messages containing body temperature, heart rate, etc. for wearable devices, industrial sensors, and so on. 1Mbps data rate is required to quickly complete the transmit, but throughput is not critical.

Simplicity: Star topology is good enough. There is no need for scatternets.

Low cost: It must cost lower than Bluetooth classic as thousands and millions of devices will have to have them.

To meet these new requirements, NOKIA came up with a completely new protocol called WiBree, which shares the same 2.4GHz band with Bluetooth. Later this new protocol was accepted by the Bluetooth standard to be called Bluetooth Smart or BLE. Although both BLE and Bluetooth Classic share the same spectrum, they are not compatible. Many new Bluetooth chips are thus manufactured as dual-mode chips

implementing both the classic Bluetooth and the new protocol. All new smartphones support dual-mode Bluetooth. There are also Bluetooth chips that support only BLE at a much lower cost. These are used for devices such as Bluetooth location beacons, which broadcast a packet announcing the current location. Other nodes receiving that broadcast can then workout their location.

10.6.1 BLE Channels

BLE has a much more simplified channel structure as shown in Figure 10.22. Instead of using all 79 channels, it hops at every 2 MHz, which allows hoping over only 40 channels. Three channels, 37, 38, and 39, are reserved for advertising, while the rest, channels 0-36, can be used for data communications. The advertising channels are selected specially to avoid interference with popular WiFi channels, i.e., channels 1, 6, 11 as shown in Figure 10.23. Use of only 3 advertising channels significantly simplifies broadcasting and discovery for BLE compared to the complicated discovery process using 32 channels in BT Classic.

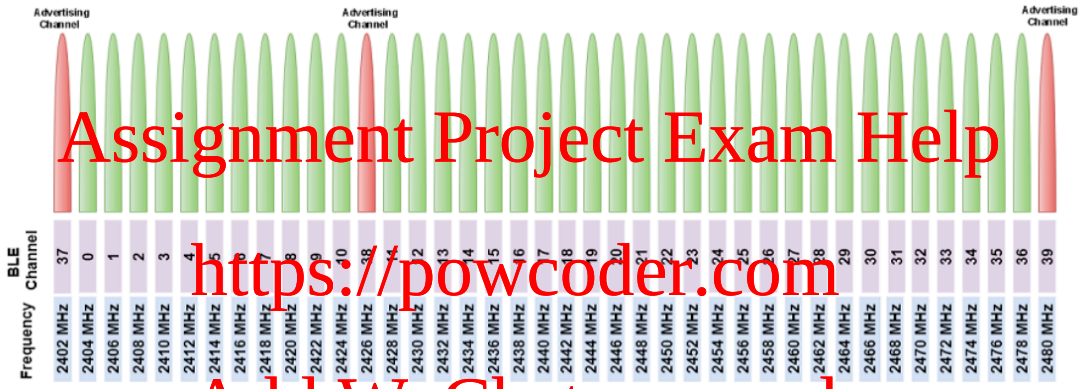


Figure 10.22 BLE channels

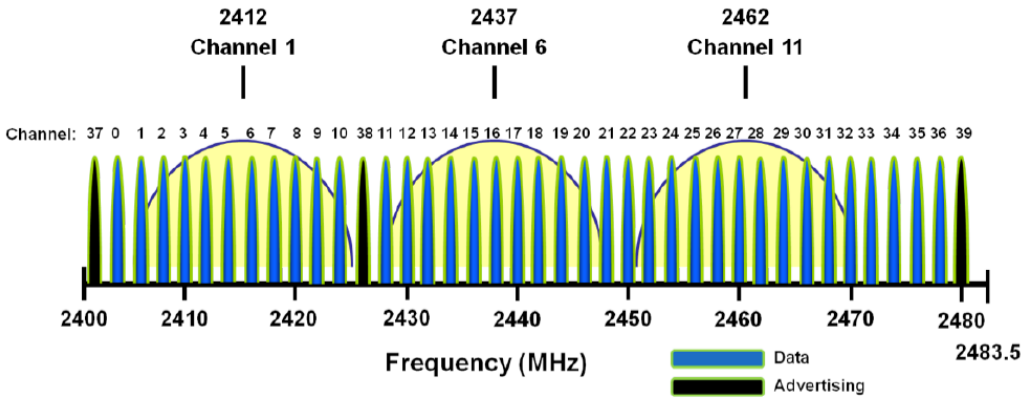


Figure 10.23 mapping between WiFi and BLE channels

10.6.2 BLE modulation and data rates

BLE uses binary GFSK over 2MHz channel, which allows more significant frequency separations for '0' and '1' compared to Bluetooth Classic that uses only 1MHz. The

wider separation of frequencies allows longer range with low power. BLE transmits 1 million symbols per second, which yields 1Mbps data rate for binary modulation.

10.6.3 BLE frequency hopping

In BLE, devices wake up periodically after every connection interval (CI) time; transmit some data (connection event) and then go back to sleep until the next connection event. A device sends a short blank packet if no data to send during a connection event but is allowed to send more than one packet during a connection event if data is pending. Connection interval time can vary from 7.5ms to 4s and is negotiated during connection set up. The device hops frequency (switch to different data channel) at the start of each event). Figure 10.24 illustrates the frequency hopping for BLE.

Figure 10.24 BLE connection events and connection intervals

BLE supports a simple frequency hopping algorithm a.k.a. Algorithm #1, that hops a fixed number of channels instead of selecting a random hop as used in BT Classic. Given that data channels range from 0-36, BLE uses the following equation to derive the next channel:

$$f_{k+1} = (f_k + h) \, mod \, 37 \qquad (10.1)$$

where h (hop increment) is a fixed value negotiated during connection setup.

Example 10.3

Assuming that two BLE devices negotiates a hop increment of 10 at the connection setup to generate the hopping frequencies during data transfers using Algorithm #1. What would be the BLE frequency selected after the fifth hop if Channel 0 was selected for the initial event?

Solution:

Initial channel: 0
Channel after the first hop: (0+10) mod 37 = 10
Channel after the second hop: (10+10) mod 37 = 20
Channel after the third hop: (20+10) mod 37 = 30

Channel after the fourth hop: (30+10) mod 37 = 3
Channel after the fifth hop: (3+10) mod 37 = 13

10.6.4 BLE services

The protocol stack for BLE has changed slightly from its predecessor Bluetooth Classic. The stack is shown in Figure 10.25. As we can see, there are basically three main layers, Controller at the bottom, the Host in the middle, and then Application at the top. Many of the functions are similar though. A new function we have is Generic Attribute Profile (GATT).

Recall that in Bluetooth Classic, the application profiles were very narrow and specific to a particular application, such as a headset or a keyboard and so on. It was adequate at the time because they had to define only a limited number of profiles. If we follow the same approach with IoT, we will have to define thousands of different profiles, which will not scale. This is why, the profile has to be more generic.

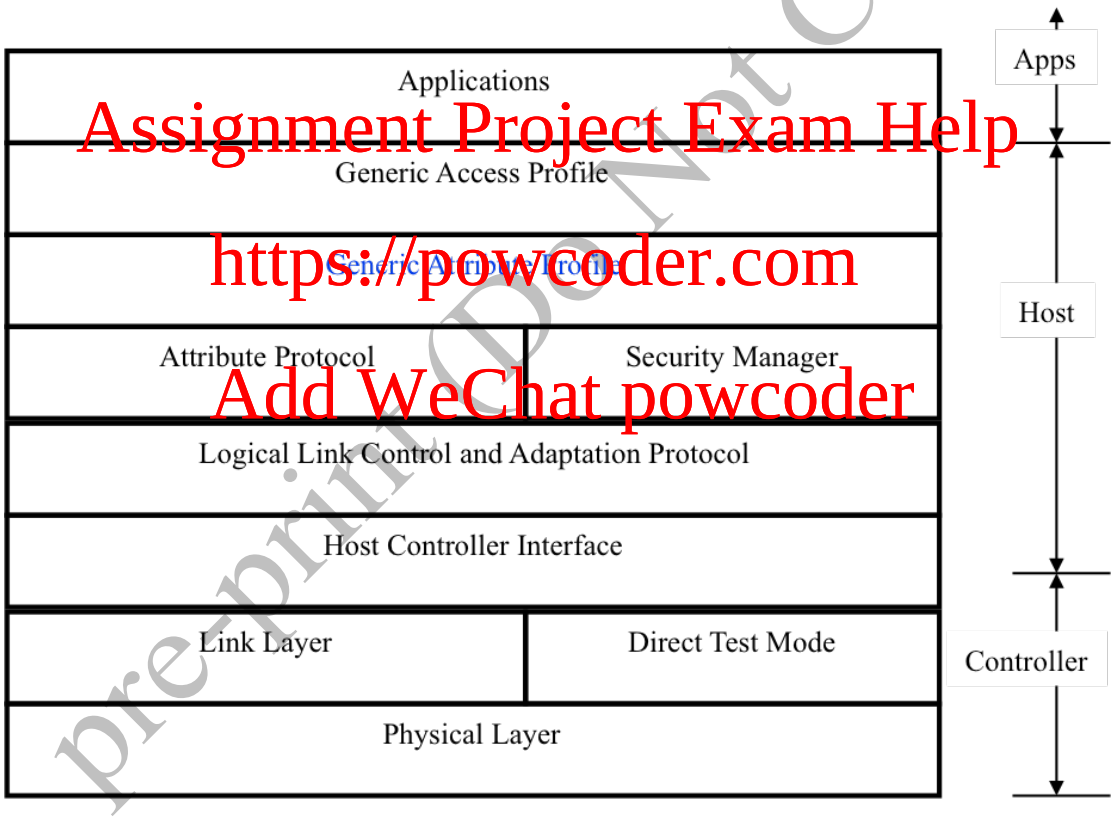


Figure 10.25 BLE protocol stack

Instead of defining specific profiles for specific applications, GATT introduces an attribute protocol that can define data formats and interfaces to suit any application. This is a major change in BLE that enables it to support large number of different objects and sensors in efficient and scalable manner.

Specifically, GATT uses a two-level hierarchy to structure the discovery and accessing BLE services. The first level is called *service*, which is uniquely identified

by a 16-bit universally unique ID (UUID) standardized by Bluetooth SIG. A set of sensor measurements called *characteristics* are then defined under each service. Characteristics are also uniquely defined using UUID. Both service and characteristic UUIDs are 16-bit numbers expressed in hexadecimal. This means in excess of 65,000 distinct services can be defined each with the capacity to define 65,000 characteristics. Table 10.3 shows some example BLE services and characteristics. For example, 0x1756 defines the environmental sensing service (ESS), which can measure temperature, humidity, pressure etc.

The sensors or peripheral devices are called GATT servers. They advertise the services they implement, so a GATT client, such as a mobile phone or a home IoT gateway can scan and identify the sensors of interest in the vicinity to collect data. Bluetooth SIG defines standard representations for exchanging sensing measurements between products from different manufacturers. Figure 10.26 illustrates how an IoT gateway can collect temperature data from a nearby sensor using BLE that implements the GATT ESS service.

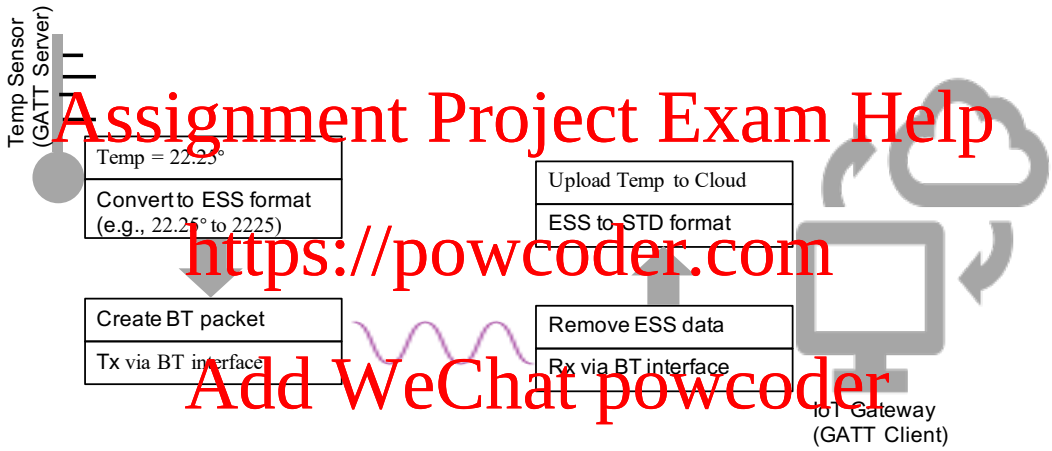


Figure 10.26 Temperature sensing using BLE environmental sensing service profile.

Table 10.3 Example of BLE GATT services and characteristics

Service	Characteristic	Format
Environmental Sensing Service (ESS) UUID = 0x1756	Temperature (2A1C)	16-bit little endian value representing the measured temperature. Unit: 0.01 deg C
	Humidity (2A6F)	16-bit little endian value representing the measured relative humidity. Unit: 0.01%
	Pressure (2A6D)	32-bit little endian value representing the measured pressure. Unit: 0.1 Pa (0.001 hPa)
Heart Rate Service (HRS) UUID = 0x180D	Hear Rate Measurement (2A37)	1-2B integer representing BPM

	Body Sensor Location (2A38)	1B integer as a code for standard locations on the body, e.g., 0x02 for WRIST.
Battery Service (BAS) (UUID = 0x180F)	Battery Level (2A19)	1B integer with the battery level as a percentage (takes a value between 0-100)

10.6.5 Bluetooth gateway Devices

In WiFi, devices talk directly to the Internet. But in Bluetooth, small sensors may talk to some intermediate devices, which can help forward requests or data to the cloud servers. Example of these intermediate devices, which become Bluetooth gateways, can be a smartphone for supporting fitness bands or smart watches as shown in Figure 10.27. tablets, laptops, desktops, or any dedicated computer that has Bluetooth can act as a gateway.

Figure 10.27 Bluetooth gateway devices

10.6.6. BLE Applications

There are many new BLE applications emerging. We discuss some of them here:

Proximity and location contexts: This BLE application helps users to discover their location contexts. For example, it can detect that the user is in the car, or in a specific room, or in a shopping mall near a perfume shop etc.

Object tracking: Small BLE tags can be attached to objects and animals to track them. For example, this application can help find keys, watches, phones, dogs, cats, children, and so on.

Health devices: Heart rate monitors, physical activities monitor, thermometer, etc. can all use BLE chips to send periodic measurements to a central server.

Sensors: Many sensors are used in industry and vehicles to constantly monitor the status of many critical components of the system for better management and improving efficiency, such as fuel efficiency of a car. For example, it can monitor

temperature, battery status, tire pressure, and so on. BLE can help these sensors to send the data to a nearby controller without having to layout complicated wiring.

Remote control: Many new types of remote controls are appearing in the market. For example, it can be used to open/close locks, turn on lights or changing the color of LED lights in the house, turn on the swimming pool heating from a nearby gateway connected to the Internet, etc.

Object interaction detection: For monitoring activities of elderly in home care, many objects, such as medicine box, toaster, shower tap, etc. can be fitted with Bluetooth Smart and sensors so when these objects are used by the person, a status is updated in the central server, which can be later analyzed to figure out what activities were completed by the person.

10.6.7 Bluetooth Beacons

Bluetooth Smart advertising by the peripherals was initially designed for connecting keyboards to the computer, headsets to the phone, phone to the car and son. Now there is a new use for it in marketing.

When the Bluetooth is turned on, it starts to advertise its presence on the advertising channels. Any other device in proximity then can pick up the signal and will detect the presence of the device. Now if the device happens to be your smartphone, then as soon as you walk into a store, the store server will immediately detect your presence. From the MAC address it can also find out whether you previously visited the store and bought something. Then it can send you some advertisements and coupons as shown in Figure 10.28.

Advertising packets consist of a header and a maximum of 27B of payload with multiple TLV-encoded data items. It may also include signal strength and distance to help you find out your precise location. For example, after some measurements, if it is established that a particular signal strength is received at a particular distance from the Bluetooth transmitter, then the receiving device will know from the signal strength whether it is at that distance from the transmitter. And if the transmitter also encodes its current location, then the receiver can find out its location. Such locations beacons, also known as BLE beacons, are now installed in many buildings to find people identify their position in an indoor map.

Beacons are also used for verification in electronic payments, such as PayPal. Geofencing is another use of beacons. New smartphones support this application, which allows to draw a perimeter on the map. When the dog or the child goes outside the perimeter, an alarm is turned to alter you.

Figure 10.28 Bluetooth coupons during shopping

10.7 Bluetooth 5

BLE (Bluetooth 4) was a major advancement compared to BT Classic in terms of reducing energy consumption and extending battery life. BLE, however, could not support high data rate applications, such as audio and file transfer (e.g., quick firmware updates), and the range was still limited for some new IoT applications. Bluetooth 5 extends BLE to realise a faster (2x) and longer range (4x) without compromising the battery life. Advertising and frequency hopping are also improved. Bluetooth 5 is seen as a significant new milestone in the evolution of Bluetooth, which is expected to support many new IoT markets in home and industrial automation, health and fitness tracking and so on.

10.7.1 Bluetooth 5 PHY

Bluetooth 5 [BT-SIG] introduces two new PHYs, one to support 2x data rate improvement and the other to provide 4x communication range compared to BLE. The PHY that provide 2x data rate increase is called 2M and the PHY providing extended range is known as Coded.

PHY 2M: The symbol duration is reduced to 55ns, which is half of BLE. This yields 2 million symbols per second, which can support 2Mbps with binary modulation. It uses the same GFSK modulation, but with higher frequency deviation to combat inter-symbol interference arising from shorter symbols. Specifically, it has frequency deviation of 370kHz (c.f. 180kHz in BLE 4) from the central frequency to denote '1' or '0' in FSK.

PHY Coded: It uses 1 million symbols per sec, the same as in BLE 4. However, to increase the range, data is coded with FEC. Two coding rates are used. The ½ rate cuts the data rate by half to 500Kbps but provides 2x range increase against BLE 4. The rate ¼ supports only 250Kbps, but achieves 4x range increase. Note that BLE 4 and BT Classic do not employ any FEC, i.e., they are not *coded*.

10.7.2 Advertising extensions

Bluetooth beacons is considered a major advertising use case for future smart cities and smart environments. Bluetooth 4, however, have limited advertising capabilities. Specifically, BLE allows just ID or URL to be advertised in the beacon due to limited advertising packet size (31 bytes payload). BLE also suffers from heavy load on its 3 advertising channels as every beacon has to be transmitted on all three channels.

Bluetooth 5 proposes three advertising extensions to address the limitations of Bluetooth 4.

Packet payload extension: The first extension Bluetooth 5 proposes is to allow advertising packets up to 255B payload, which enables smart devices and products to advertise many more things and status, such as a fridge can advertise its contents, temperature, expiry dates of sensitive items, etc.

Channel offload: With this extension, only the header is transmitted over advertising channels and the actual payload is offloaded to a data channel. Note that Bluetooth 4 reserves data channels only for data transfers during Connection Events when connections are established. Channel offload allows Bluetooth 5 to use of data channels in connectionless manner. Channel offloading is illustrated in Figure 10.29.

Packet chaining: Bluetooth 5 allows chaining multiple 255B packets together to carry a very large advertising message. The packet chaining is illustrated in Figure 10.30.

Figure 10.29 Advertising channel offload in Bluetooth 5

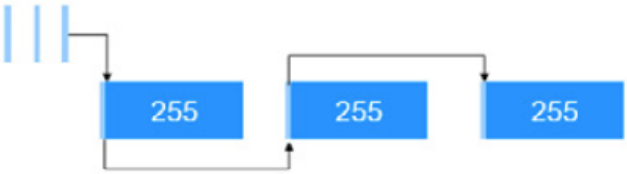


Figure 10.30 Advertising packet chaining in Bluetooth 5

10.7.3 Frequency hopping extension

Bluetooth 4 supports only a simple hopping algorithm, called Algorithm #1, which uses only a fixed hopping increment to switch channels. Fixed hopping increment limits the number of possible sequences to choose from. Bluetooth 5 supports pseudorandom hopping like the BT Classic using an algorithm called Algorithm #2, which allows large number of options for channel switching sequences.

10.8 Bluetooth 5.3

In 2021, the following new features were added to Bluetooth [BT-SIG] to further improve latency, battery life, security, and protection against interference.

Fast adjustment of duty cycling: As we have studied earlier in this chapter, Bluetooth version 4.0 introduced a parameter called Connection Interval (CI) to duty cycle the device for improved battery life. Given the inherent trade-off between low duty cycling (good battery life) and high throughput (lower latency), version 4.0 allowed the CI parameter to be selected from 7.5ms to 4s to address the needs of different applications. However, the CI value could only be negotiated during connection set up, which makes it difficult to switch between different duty cycling modes quickly.

In many applications, the devices need to change duty cycling rates to optimise the trade-off between battery life and latency or response time. For example, an environment monitor may start with a very low duty cycling rate to conserve battery, but when an event of interest is detected, it would like to switch to a high duty cycling rate to transfer the event related data at a faster rate to the server for immediate analysis of the event. Bluetooth 5.3 has added a parameter that now enables a device to change the CI value as a multiple of some base values within the connection. This allows devices to rapidly change duty cycling for better battery life as well as latency.

Peripheral input to adaptive frequency hopping: Before Bluetooth 5.3, only the central (master) device could decide the eligible list of frequencies (channel map) for adaptive hopping based on its own experience of interference. However, with longer ranges allowed by version 5.0, the peripheral devices (slaves) may experience different interference than the central device depending on the locations and environments. To achieve improved protection against all interference affecting the communication between the master and the slave, Bluetooth 5.3 includes protocol enhancement for the master to consult with the slave for deciding the final channel map to be used in frequency hopping.

Encryption key size control enhancement: with home and commercial building automation, Bluetooth is being used by a central controller to communicate with many peripheral devices such as door locks, privacy curtains, lights, and so on. As peripherals have better knowledge about the security need of the application, it is desirable for these devices to efficiently negotiate with the central controller about the size of the encryption keys. The new enhancement in version 5.3 allows a peripheral to convey minimum key lengths to the central controller. While this enhancement sounds trivial, it is designed to increase the competitiveness of Bluetooth in the competing market for automation.

More meta information for periodic advertising: Periodic advertising is one of the key applications of Bluetooth, where sensors and things fitted with a Bluetooth chip periodically broadcast their sensing data, such as temperature, pressure, etc., which are received and processed by nearby Bluetooth hosts to infer actionable information. In many cases, such periodic advertisements contain the same data (if there is no change in the status of the thing or environment from the last broadcast), which creates unnecessary processing load on the host. To address this situation, Bluetooth 5.3 adds a separate field in the advertising packet header to contain more meta

information about the content of the advertising packet. Specifically, this header can be used by the thing or sensor to indicate whether the packet content is the same as the previous one. Thus, the host can stop processing the rest of the packet as soon as the header says that it is a repeat information.

10.9 Chapter Summary

1. Bluetooth Classic uses frequency hopping over 79 1-MHz channels with 1, 3, and 5-slot packets.
2. Bluetooth 4 is designed for short broadcasts by sensors. 40 2-MHz channels are used with 3 channels reserved for advertising and 37 used for data transfers.
3. BT Classic uses flat application profiles to support different types of communication services, which requires different application profiles to be defined for different types of sensing and communications.
4. BLE has a hierarchical service structure to group many sensing measurements into a given service type, which scales for large variety of devices and services expected in the IoT era.
5. Bluetooth 5 extends BLE to support higher data rate and longer-range. It also has an improved advertising structure that allows advertisement of more comprehensive information and contents.

References

[NIST-BT] "Security of Bluetooth Systems and Devices," NIST, August 2012. https://csrc.nist.gov/csrc/media/publications/shared/documents/itl-bulletin/itlbul2012-08.pdf [accessed 26 October 2021]

[BT-SIG-ORG] "Origin of the Bluetooth name," Bluetooth Special Interest Group, https://www.bluetooth.com/about-us/bluetooth-origin/ [accessed 26 October 2021]

[WIKI-BT] Bluetooth Special Interest Group, Wikipedia. https://en.wikipedia.org/wiki/Bluetooth_Special_Interest_Group [accessed 26 October 2021]

[BT-SIG] Bluetooth Special Interest Group, https://www.bluetooth.com/.

[PRAVIN2001] P. Bhagwat, "Bluetooth: technology for short-range wireless apps," in *IEEE Internet Computing*, vol. 5, no. 3, pp. 96-103, May-June 2001, doi: 10.1109/4236.935183.

[BT-NI] "Introduction to Bluetooth Device Testing: From Theory To Transmitter and Receiver Measurements," National Instruments. https://download.ni.com/evaluation/rf/intro_to_bluetooth_test.pdf [accessed 26 October 2021]

[BT-RS] “Bluetooth Adaptive Frequency Hopping on a R&S CMW Application Note,” Rohde Schwarz. https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_application/application_notes/1c108/1C108_0e_Bluetooth_BR_EDR_AFH.pdf [accessed 26 October 2021]

[BT-SF] “Bluetooth HID Profile user Manual,” Sparkfun. https://cdn.sparkfun.com/datasheets/Wireless/Bluetooth/RN-HID-User-Guide-v1.0r.pdf [accessed 26 October 2021]

End of Chapter 10