Documentation:

**Before Bluetooth:**

* **Floppy disks** (3.5-inch and 5.25-inch) were widely used in the 1980s and 1990s for transferring small files.
* **CDs and later DVDs** allowed for larger file transfers and were common in the late 1990s and early 2000s.
* **Serial ports (RS-232)** and **parallel ports** were used to connect two computers directly.
* File transfer software like **LapLink** or **InterLink** enabled data exchange over these cables.
* In offices or homes with multiple computers, **Ethernet-based LANs** allowed file sharing using protocols like **FTP, SMB.**

**Human audible frequency:**

20 – 20,000 hz.

Bluetooth technology was introduced in **1998** (with development starting in the mid-1990s), and its purpose at that time was quite forward-thinking. Here's a breakdown of **why it was developed** and **what it was used for in its early days**:

🔍 **Why Bluetooth Was Developed (Mid-1990s)**

**Cable Replacement:**  
The main goal was to eliminate the need for physical cables between devices like mobile phones, laptops, headsets, and PDAs (Personal Digital Assistants).

The RS-232 standard was widely used as a computer serial port, catering to internet modems, printers, mice, data storage, and a host of other peripherals. As its proposed replacement, Bluetooth was designed as a flexible packet-based protocol with a wide selection of profiles to suit these applications and more. RS-232 was also rather power-hungry for a physical connection, so Bluetooth was built to require much less power.

**Low Power, Short Range:**  
It was designed to be low power and short range (typically 10 meters), making it ideal for portable devices.

**Universal Standard:**  
Companies like Ericsson, IBM, Intel, Nokia, and Toshiba formed the Bluetooth Special Interest Group (SIG) to create a universal wireless standard.

📱 **Early Use Cases (Late 1990s to Early 2000s)**

**Wireless Headsets**: Hands-free calling for mobile phones.

**Data Syncing**: Syncing contacts and calendars between phones and computers.

**File Transfer**: Sending images, ringtones, and small files between devices.

**Peripheral Connections**: Connecting keyboards, mice, and printers wirelessly.

* **WHY BLUETOOTH USES THE 2.4 GHZ BAND?**

**1. Globally Unlicensed Spectrum**

* The **2.4 GHz ISM band** is **license-free worldwide**, meaning manufacturers can use it without paying for spectrum licenses.
* This makes it ideal for **mass-market consumer devices** like phones, headsets, and keyboards.

**2. Balance of Range and Data Rate**

* **Lower frequencies** (e.g., 900 MHz) offer longer range but lower data rates.
* **Higher frequencies** (e.g., 5 GHz) offer higher data rates but shorter range and more signal attenuation.
* **2.4 GHz** provides a **good balance**: decent range (~10–100 meters) and sufficient data rate for Bluetooth’s needs.

**3. Small Antenna Size**

* The wavelength at 2.4 GHz (~12.5 cm) allows for **compact antennas**, which is ideal for small devices like earbuds, wearables, and phones.

**4. Interference Handling via Frequency Hopping**

* The 2.4 GHz band is crowded (used by Wi-Fi, microwaves, etc.), but Bluetooth uses **Frequency-Hopping Spread Spectrum (FHSS)**:
  + It rapidly switches frequencies (1600 hops/sec) across 79 channels.
  + This reduces interference and improves reliability.

**5. Cost and Compatibility**

* Using a common band reduces hardware costs and simplifies **interoperability** between devices from different manufacturers.
* **WHY NOT OTHER FREQUENCIES?**

| **Frequency** | **Reason Not Chosen** |
| --- | --- |
| **900 MHz** | Not globally available; larger antennas needed. |
| **5 GHz** | Higher attenuation, shorter range, more power consumption. |
| **Sub-GHz (<1 GHz)** | Lower data rates, not suitable for audio or file transfer. |

* **SHORT RANGE:**

🔹 Why Bluetooth is Called "Short-Range"

In wireless communication, technologies are generally grouped into three broad range categories:

| Category | Typical Range | Examples |
| --- | --- | --- |
| Short-Range | Up to ~100 meters | Bluetooth, Wi-Fi, Zigbee |
| Medium-Range | 100 meters to a few km | Cellular (3G/4G), LoRa |
| Long-Range | Several kilometres or more | Satellite, HF Radio |

So, "short-range" doesn't mean "shortest possible range", but rather that it's in the lowest tier of wireless communication ranges used for local or personal area networks (PANs).

🔹 Why NFC and RFID Aren’t in the Same Category

* NFC and passive RFID are often considered "proximity" or "contactless" technologies.
* They are not designed for continuous data exchange like Bluetooth.
* Their range is so limited that they form a subcategory under short-range, often called:
  + Proximity communication
  + Near-field communication

Summary

* Bluetooth is "short-range" because it fits within the 0–100 meter range category.
* NFC/RFID are even shorter, but they are not general-purpose communication protocols like Bluetooth.
* The classification is based on intended use, range, and data capabilities, not just raw distance.

\*-------------------------------------------------\*

MODULATION:

* **Modulation** is the process of varying one or more properties of a high-frequency periodic waveform (called the **carrier signal**) with a separate signal (called the **modulating signal**, typically containing the information to be transmitted).
* The **modulation concept** comes into consideration when the signal needs to be transmitted over a **long distance** through an antenna. Antenna helps transmit the signal over long distance.
* The modulation concept makes the communication purely **wireless and mobile**. And because of modulation, we can now roam freely without the fear of getting out of the communication grid.

(OR)

* **Modulation** is the process of varying one or more parameters of a carrier signal in accordance with the instantaneous values of the message signal.
* The message signal is the signal which is being transmitted for communication and the carrier signal is a high frequency signal which has no data but is used for long distance transmission.

**Key Parameters That Can Be Modulated:**

1. Amplitude – in Amplitude Modulation (AM)
2. Frequency – in Frequency Modulation (FM)
3. Phase – in Phase Modulation (PM)

**Purpose of Modulation:**

* To **transmit information** over long distances.
* To **match the signal** to the characteristics of the transmission medium.
* To **allow multiplexing** (transmitting multiple signals over the same channel).
* To **improve signal quality** and reduce interference.

BASEBAND SIGNAL

A signal consisting of significantly lower frequency (up to **10 kHz**) is known as a **baseband signal**. Example of the baseband signal is voice, audio and video signal.

The frequency range of **voice** signal is **300Hz to 3.5 kHz**.

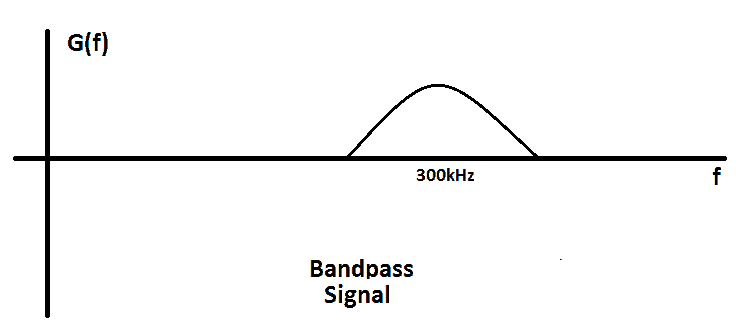
**Audio** signal’s frequency range is**20 Hz to 20 kHz**.

**Video** signal’s frequency range is **0Hz to 4.5 MHz**

All of these signals contain low frequencies (up to **10 kHz**), which makes them **baseband signals**. The baseband signal cannot be transmitted directly through the antenna. Thus, it gives rise to the **concept of modulation**.

BANDPASS SIGNAL

If a signal consists of significantly higher frequencies (Higher than **100 kHz**) then it is known as **Passband** or **BandPass** signal. **Bandpass signal** does not contain any frequency lower than **100 kHz**.

[[](https://www.electricaltechnology.org/wp-content/uploads/2018/12/Bandpass-signal.png)](https://www.electricaltechnology.org/wp-content/uploads/2018/12/Bandpass-signal.png)

**Bandpass** signal can be directly transmitted through the **antenna**.

Using the process of **modulation**, the signal with low frequency, also known as Baseband signal is converted into a signal with high frequency, also known as Bandpass signal.

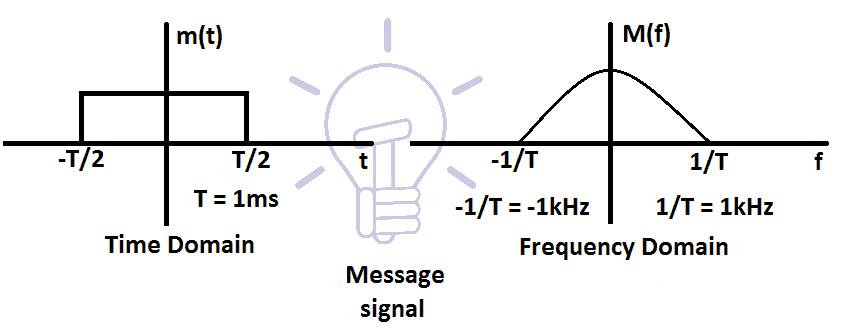
MESSAGE SIGNAL:

The signal that is used in modulating the carrier signal during modulation is called the **message signal**.

The message signal is a baseband signal. for example, voice, sound, video, images & data signals are baseband signals.

Suppose a baseband **message signal** **m(t)** is a rectangle signal with **T = 1ms**. To show its frequency spectrum, we need to do its Fourier transform.

We know that the Fourier transform of rectangle function is a sinc function but for the sake of understanding, we will just go with the spectrum given below. The spectrum of **m(t)** signal is given in the figure:

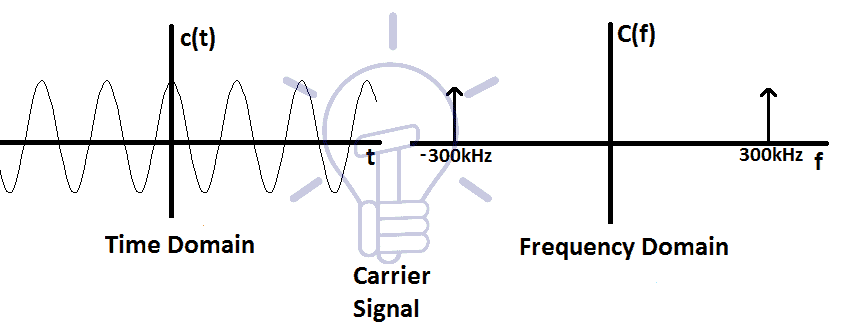
[[](https://www.electricaltechnology.org/wp-content/uploads/2018/12/Message-Signal.png)](https://www.electricaltechnology.org/wp-content/uploads/2018/12/Message-Signal.png)

The frequency **f** of **m(t)** ranges from **0 to 1Khz**, which makes it a **baseband signal**.

CARRIER SIGNAL:

The sinusoidal signal with a much higher frequency that is used in the modulation is called the **carrier signal**.

Let’s suppose a **carrier signal** **c(t)**, that is a sinusoidal signal with high frequency. And the frequency **fc** of the carrier signal is **300 kHz**. The time domain figure and the Fourier transform of the carrier signal are given in the figure below:

[[](https://www.electricaltechnology.org/wp-content/uploads/2018/12/Carrier-Signal.png)](https://www.electricaltechnology.org/wp-content/uploads/2018/12/Carrier-Signal.png)

As you can see, the spectrum of carrier signal only contains the frequency **300 kHz**. This makes it a **bandpass signal**. It is easily transmitted through the antenna.

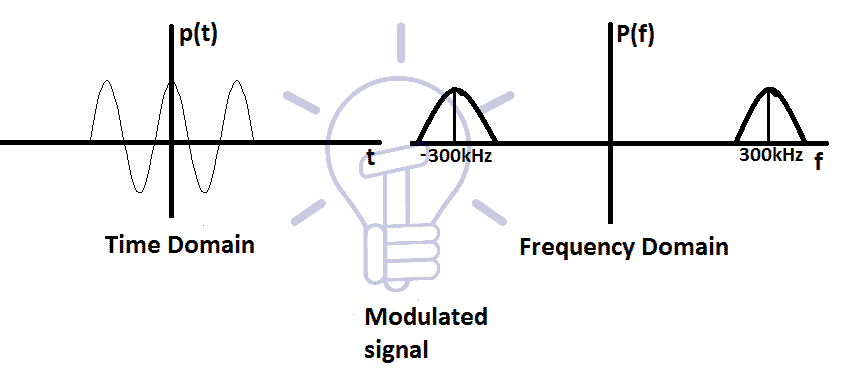
We need to transmit the **message signal** **m(t)** with the help of the **carrier signal c(t)**. To make it able to transmit through the antenna, we need to translate the message signal **m(t)** onto the carrier signal **c(t)**.

MODULATED SIGNAL

The resultant signal acquired after modulation of message and carrier signal is called **a modulated signal**.

A simple modulated signal is acquired by the **multiplication** of carrier and message signal. The resultant signal is the **modulated signal**.

Suppose the product signal **p(t)** is the modulated signal of **m(t) & c(t)**. It is a sinusoidal signal, where the amplitude of message signal **m(t)** is not **0**. The time domain and frequency domain figures of modulated signals are given below:

[[](https://www.electricaltechnology.org/wp-content/uploads/2018/12/Modulated-signal.png)](https://www.electricaltechnology.org/wp-content/uploads/2018/12/Modulated-signal.png)

The modulated signals **frequency** is shifted by the frequency of the carrier signal **fc**. The spectrum shows **2 sides** (negative and positive side)of the frequencies, which is nothing but a mirror of each other.

The modulated signal’s spectrum consists of **lower** and **higher side**. Lower and higher frequency is determined by **(fc-f)** and **(fc+f)** respectively.

This modulated signal is now a **bandpass signal,** and an antenna can easily transmit it.

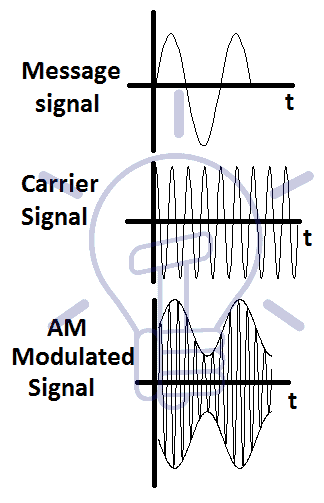
TYPES OF MODULATION

Basic types of Modulation are defined with figures below.

**Amplitude Modulation (AM)**

The type of modulation in which the **amplitude** of the carrier signal varies with respect to the amplitude of the message signal is called **Amplitude Modulation**.

The message signal’s information is stored in the **amplitude (envelope)** of the modulated signal.

[](https://www.electricaltechnology.org/wp-content/uploads/2018/12/AM-modulation.png)

**[Frequency Modulation (FM)](https://www.electricaltechnology.org/wp-content/uploads/2018/12/AM-modulation.png)**

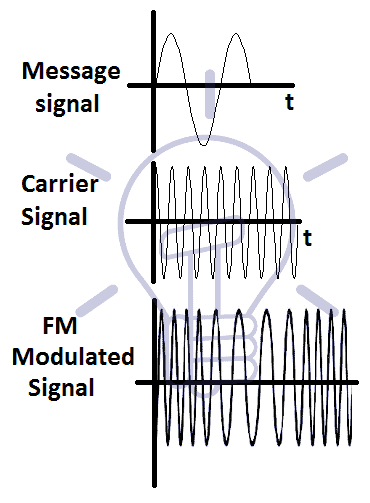
[The type of modulation in which the Frequency of the carrier signal varies with respect to the amplitude of the message signal is called Frequency modulation.](https://www.electricaltechnology.org/wp-content/uploads/2018/12/AM-modulation.png)

[The message signal’s information is stored in the frequency of the modulated signal.](https://www.electricaltechnology.org/wp-content/uploads/2018/12/AM-modulation.png)

**[Phase Modulation (PM)](https://www.electricaltechnology.org/wp-content/uploads/2018/12/AM-modulation.png)**

[The type of modulation in which the phase of the carrier signal varies linearly with respect to the amplitude of the message signal or data signal is called Phase modulation.](https://www.electricaltechnology.org/wp-content/uploads/2018/12/AM-modulation.png)

[The information of the message signal is stored in the phase of the modulated signal.](https://www.electricaltechnology.org/wp-content/uploads/2018/12/AM-modulation.png)

[[](https://www.electricaltechnology.org/wp-content/uploads/2018/12/FM-modulation.png)](https://www.electricaltechnology.org/wp-content/uploads/2018/12/FM-modulation.png)

**DIGITAL MODULATION:**

In Digital Modulation, only the carrier signal is analogue and the message signal is in digital form. In Digital Modulation a process called as Shift Keying is used.

**Shift Keying** means that the amplitude, frequency or phase of the carrier wave is shifted between two or more discrete values rather than varying continuously like Analog Modulation. Binary data requires two discrete levels of amplitude, frequency or phase for modulation called as Binary Shift Keying. A group of bits can be clubbed together to form M-ary Shift Keying.

There are mainly three types of Digital Modulation techniques. They are:

* Amplitude Shift Keying
* Frequency Shift Keying
* Phase Shift Keying

**Amplitude Shift Keying**

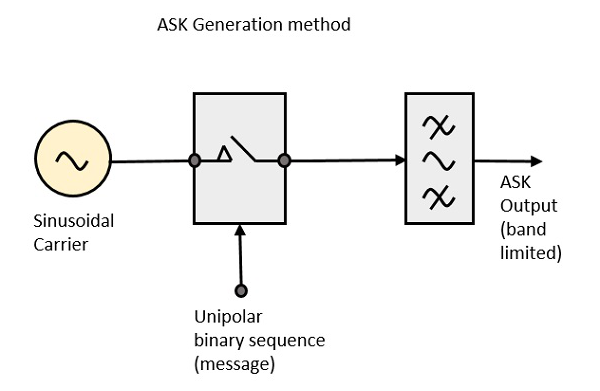
* **Amplitude Shift Keying (ASK)** is a type of Amplitude Modulation which represents the binary data in the form of variations in the amplitude of a signal.
* Any modulated signal has a high frequency carrier. The binary signal when ASK modulated, gives a **zero** value for **Low** input while it gives the **carrier output** for **High** input.
* The following figure represents ASK modulated waveform along with its input.



* To find the process of obtaining this ASK modulated wave, let us learn about the working of the ASK modulator.

**ASK Modulator**

The ASK modulator block diagram comprises of the carrier signal generator, the binary sequence from the message signal and the band-limited filter. Following is the block diagram of the ASK Modulator.



The carrier generator sends a continuous high-frequency carrier. The binary sequence from the message signal makes the unipolar input to be either High or Low. The high signal closes the switch, allowing a carrier wave. Hence, the output will be the carrier signal at high input. When there is low input, the switch opens, allowing no voltage to appear. Hence, the output will be low.

The band-limiting filter shapes the pulse depending upon the amplitude and phase characteristics of the band-limiting filter or the pulse-shaping filter.

**ASK Demodulator**

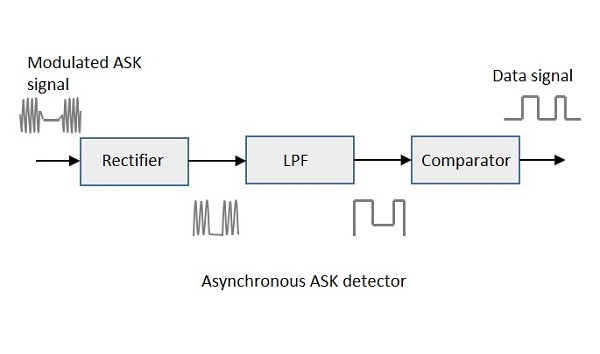
There are two types of ASK Demodulation techniques. They are −

* Asynchronous ASK Demodulation/detection
* Synchronous ASK Demodulation/detection

The clock frequency at the transmitter when matches with the clock frequency at the receiver, it is known as a Synchronous method, as the frequency gets synchronized. Otherwise, it is known as Asynchronous.

**Asynchronous ASK Demodulator**

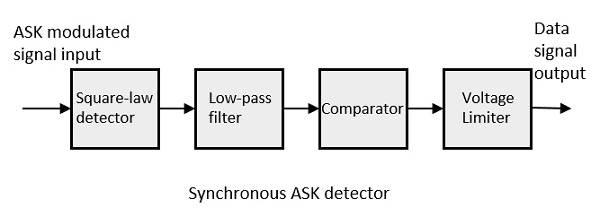
The Asynchronous ASK detector consists of a half-wave rectifier, a low pass filter, and a comparator. Following is the block diagram for the same.



The modulated ASK signal is given to the half-wave rectifier, which delivers a positive half output. The low pass filter suppresses the higher frequencies and gives an envelope detected output from which the comparator delivers a digital output.

Synchronous ASK Demodulator

Synchronous ASK detector consists of a Square law detector, low pass filter, a comparator, and a voltage limiter. Following is the block diagram for the same.



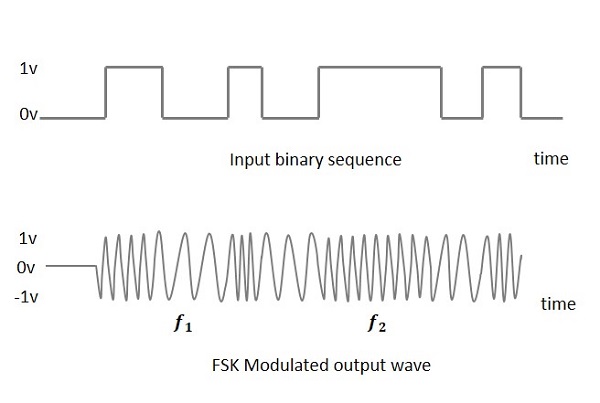
The ASK modulated input signal is given to the Square law detector. A square law detector is one whose output voltage is proportional to the square of the amplitude modulated input voltage. The low pass filter minimizes the higher frequencies. The comparator and the voltage limiter help to get a clean digital output.

**Frequency Shift Keying:**

**Frequency Shift Keying (FSK)** is the digital modulation technique in which the frequency of the carrier signal varies according to the digital signal changes. FSK is a scheme of frequency modulation.

The output of a FSK modulated wave is high in frequency for a binary High input and is low in frequency for a binary Low input. The binary **1s** and **0s** are called Mark and Space frequencies.

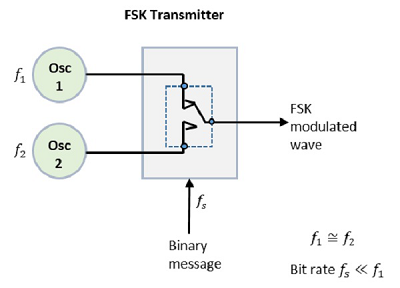
The following image is the diagrammatic representation of FSK modulated waveform along with its input.



To find the process of obtaining this FSK modulated wave, let us know about the working of a FSK modulator.

FSK Modulator

The FSK modulator block diagram comprises of two oscillators with a clock and the input binary sequence. Following is its block diagram.



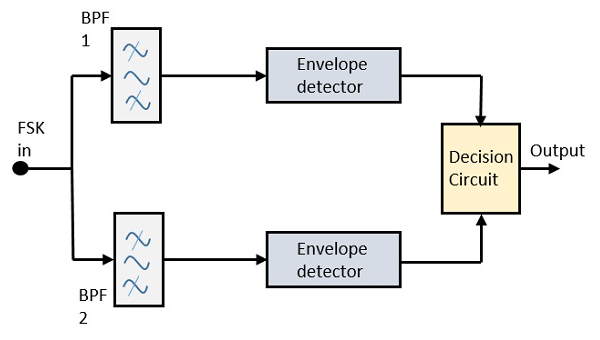
The two oscillators, producing a higher and a lower frequency signals, are connected to a switch along with an internal clock. To avoid the abrupt phase discontinuities of the output waveform during the transmission of the message, a clock is applied to both the oscillators, internally. The binary input sequence is applied to the transmitter so as to choose the frequencies according to the binary input.

**FSK Demodulator**

There are different methods for demodulating a FSK wave. The main methods of FSK detection are **asynchronous detector** and **synchronous detector**. The synchronous detector is a coherent one, while asynchronous detector is a non-coherent one.

**Asynchronous FSK Detector**

The block diagram of Asynchronous FSK detector consists of two band pass filters, two envelope detectors, and a decision circuit. Following is the diagrammatic representation.

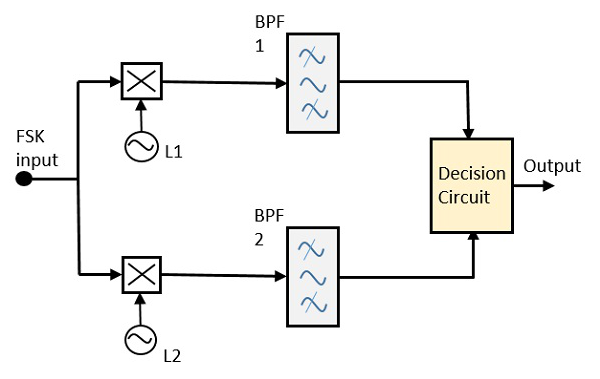


The FSK signal is passed through the two Band Pass Filters (BPFs), tuned to **Space** and **Mark** frequencies. The output from these two BPFs look like ASK signal, which is given to the envelope detector. The signal in each envelope detector is modulated asynchronously.

The decision circuit chooses which output is more likely and selects it from any one of the envelope detectors. It also re-shapes the waveform to a rectangular one.

**Synchronous FSK Detector**

The block diagram of Synchronous FSK detector consists of two mixers with local oscillator circuits, two band pass filters and a decision circuit. Following is the diagrammatic representation.



The FSK signal input is given to the two mixers with local oscillator circuits. These two are connected to two band pass filters. These combinations act as demodulators and the decision circuit chooses which output is more likely and selects it from any one of the detectors. The two signals have a minimum frequency separation.

For both of the demodulators, the bandwidth of each of them depends on their bit rate. This synchronous demodulator is a bit complex than asynchronous type demodulators.

**Digital Communication - Phase Shift Keying**

**Phase Shift Keying (PSK)** is the digital modulation technique in which the phase of the carrier signal is changed by varying the sine and cosine inputs at a particular time. PSK technique is widely used for wireless LANs, bio-metric, contactless operations, along with RFID and Bluetooth communications.

PSK is of two types, depending upon the phases the signal gets shifted. They are −

Binary Phase Shift Keying (BPSK)

This is also called as 2-phase PSK or Phase Reversal Keying. In this technique, the sine wave carrier takes two phase reversals such as 0° and 180°.

BPSK is basically a Double Side Band Suppressed Carrier (DSBSC) modulation scheme, for message being the digital information.

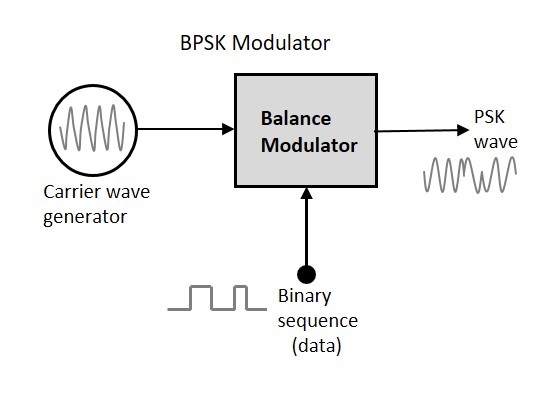
Quadrature Phase Shift Keying (QPSK)

This is the phase shift keying technique, in which the sine wave carrier takes four phase reversals such as 0°, 90°, 180°, and 270°.

If this kind of techniques are further extended, PSK can be done by eight or sixteen values also, depending upon the requirement.

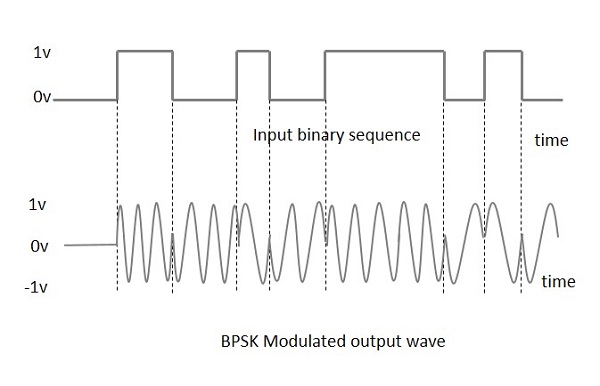
BPSK Modulator

The block diagram of Binary Phase Shift Keying consists of the balance modulator which has the carrier sine wave as one input and the binary sequence as the other input. Following is the diagrammatic representation.



The modulation of BPSK is done using a balance modulator, which multiplies the two signals applied at the input. For a zero binary input, the phase will be **0°** and for a high input, the phase reversal is of **180°**.

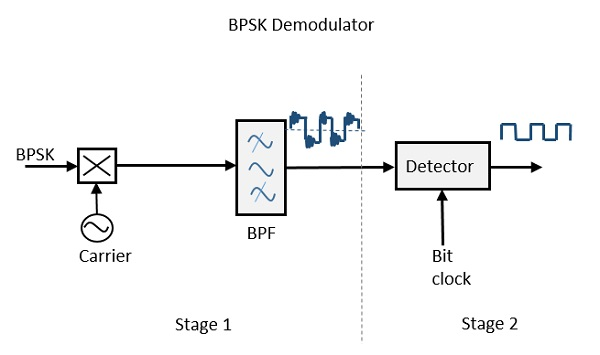
Following is the diagrammatic representation of BPSK Modulated output wave along with its given input.



The output sine wave of the modulator will be the direct input carrier or the inverted (180° phase shifted) input carrier, which is a function of the data signal.

**BPSK Demodulator**

The block diagram of BPSK demodulator consists of a mixer with local oscillator circuit, a bandpass filter, a two-input detector circuit. The diagram is as follows.



By recovering the band-limited message signal, with the help of the mixer circuit and the band pass filter, the first stage of demodulation gets completed. The base band signal which is band limited is obtained and this signal is used to regenerate the binary message bit stream.

In the next stage of demodulation, the bit clock rate is needed at the detector circuit to produce the original binary message signal. If the bit rate is a sub-multiple of the carrier frequency, then the bit clock regeneration is simplified. To make the circuit easily understandable, a decision-making circuit may also be inserted at the 2nd stage of detection.

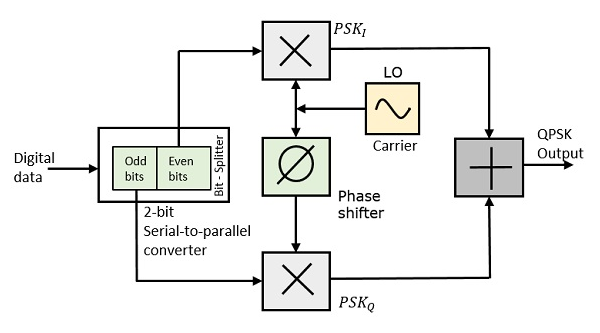
**Quadrature Phase Shift Keying**

The **Quadrature Phase Shift Keying (QPSK)** is a variation of BPSK, and it is also a Double Side Band Suppressed Carrier (DSBSC) modulation scheme, which sends two bits of digital information at a time, called as **bigits**.

Instead of the conversion of digital bits into a series of digital stream, it converts them into bit pairs. This decreases the data bit rate to half, which allows space for the other users.

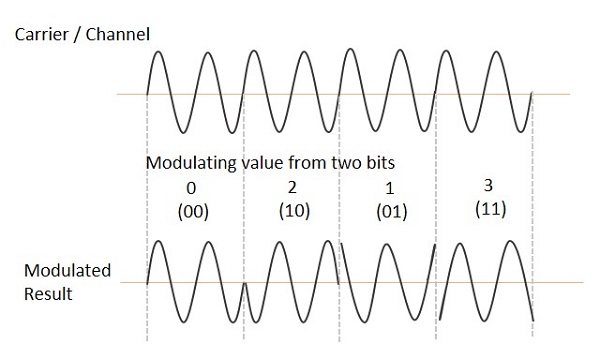
QPSK Modulator

The QPSK Modulator uses a bit-splitter, two multipliers with local oscillator, a 2-bit serial to parallel converter, and a summer circuit. Following is the block diagram for the same.



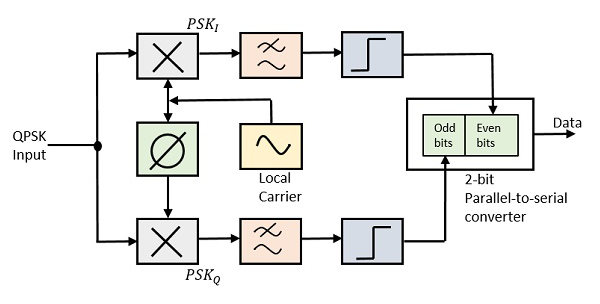
At the modulators input, the message signals even bits (i.e., 2nd bit, 4th bit, 6th bit, etc.) and odd bits (i.e., 1st bit, 3rd bit, 5th bit, etc.) are separated by the bits splitter and are multiplied with the same carrier to generate odd BPSK (called as **PSKI**) and even BPSK (called as **PSKQ**). The **PSKQ** signal is anyhow phase shifted by 90° before being modulated.

The QPSK waveform for two-bits input is as follows, which shows the modulated result for different instances of binary inputs.



**QPSK Demodulator**

The QPSK Demodulator uses two product demodulator circuits with local oscillator, two band pass filters, two integrator circuits, and a 2-bit parallel to serial converter. Following is the diagram for the same.

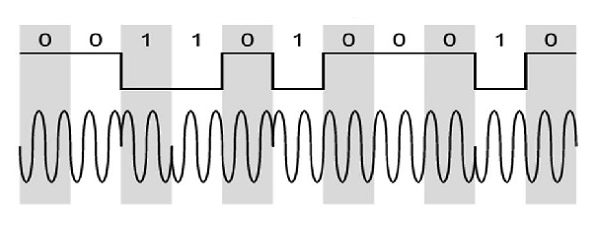


The two product detectors at the input of demodulator simultaneously demodulate the two BPSK signals. The pair of bits are recovered here from the original data. These signals after processing, are passed to the parallel to serial converter.

**Differential Phase Shift Keying**

In **Differential Phase Shift Keying (DPSK)** the phase of the modulated signal is shifted relative to the previous signal element. No reference signal is considered here. The signal phase follows the high or low state of the previous element. This DPSK technique doesnt need a reference oscillator.

The following figure represents the model waveform of DPSK.

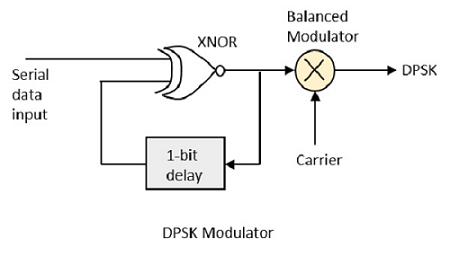


It is seen from the above figure that, if the data bit is Low i.e., 0, then the phase of the signal is not reversed, but continued as it was. If the data is a High i.e., 1, then the phase of the signal is reversed, as with NRZI, invert on 1 (a form of differential encoding).

If we observe the above waveform, we can say that the High state represents an **M** in the modulating signal and the Low state represents a **W** in the modulating signal.

DPSK Modulator

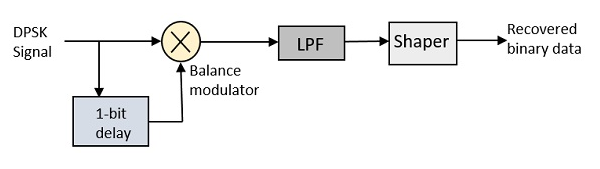
DPSK is a technique of BPSK, in which there is no reference phase signal. Here, the transmitted signal itself can be used as a reference signal. Following is the diagram of DPSK Modulator.



DPSK encodes two distinct signals, i.e., the carrier and the modulating signal with 180° phase shift each. The serial data input is given to the XNOR gate and the output is again fed back to the other input through 1-bit delay. The output of the XNOR gate along with the carrier signal is given to the balance modulator, to produce the DPSK modulated signal.

DPSK Demodulator

In DPSK demodulator, the phase of the reversed bit is compared with the phase of the previous bit. Following is the block diagram of DPSK demodulator.

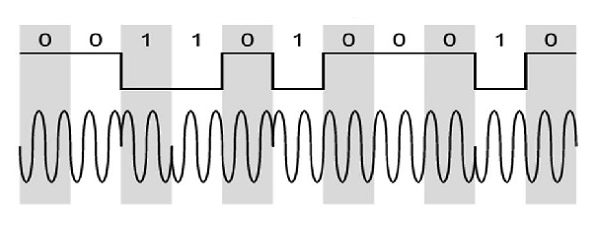


From the above figure, it is evident that the balance modulator is given the DPSK signal along with 1-bit delay input. That signal is made to confine to lower frequencies with the help of LPF. Then it is passed to a shaper circuit, which is a comparator or a Schmitt trigger circuit, to recover the original binary data as the output.

**DPSK:**

In **Differential Phase Shift Keying (DPSK)** the phase of the modulated signal is shifted relative to the previous signal element. No reference signal is considered here. The signal phase follows the high or low state of the previous element. This DPSK technique doesn’t need a reference oscillator.

The following figure represents the model waveform of DPSK.

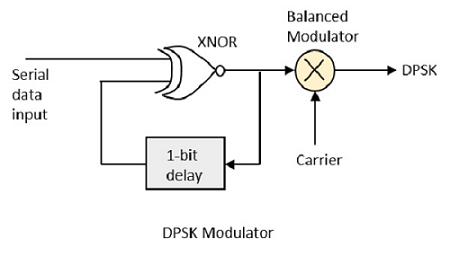


It is seen from the above figure that, if the data bit is Low i.e., 0, then the phase of the signal is not reversed, but continued as it was. If the data is a High i.e., 1, then the phase of the signal is reversed, as with NRZI, invert on 1 (a form of differential encoding).

If we observe the above waveform, we can say that the High state represents an **M** in the modulating signal and the Low state represents a **W** in the modulating signal.

**DPSK MODULATOR:**

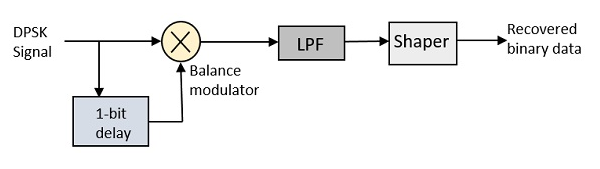
DPSK is a technique of BPSK, in which there is no reference phase signal. Here, the transmitted signal itself can be used as a reference signal. Following is the diagram of DPSK Modulator.



DPSK encodes two distinct signals, i.e., the carrier and the modulating signal with 180° phase shift each. The serial data input is given to the XNOR gate and the output is again fed back to the other input through 1-bit delay. The output of the XNOR gate along with the carrier signal is given to the balance modulator, to produce the DPSK modulated signal.

**DPSK Demodulator:**

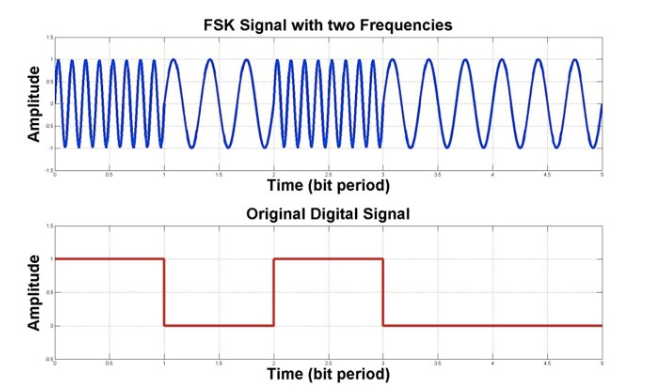
In DPSK demodulator, the phase of the reversed bit is compared with the phase of the previous bit. Following is the block diagram of DPSK demodulator.



From the above figure, it is evident that the balance modulator is given the DPSK signal along with 1-bit delay input. That signal is made to confine to lower frequencies with the help of LPF. Then it is passed to a shaper circuit, which is a comparator or a Schmitt trigger circuit, to recover the original binary data as the output.

**GFSK:**

**Gaussian Frequency Shift Keying or GFSK** is an extension of the FSK modulation scheme where the frequency of the modulated signal will not instantaneously change at the beginning of each symbol period of the binary data. As a result, the transition from bit 0 to bit 1 or vice-versa becomes smoother. In other words, the amplitude and phase variations of the modulated signal will be relatively less compared to traditional FSK. In principle, FSK is implemented using independent local oscillators separately for in-phase and Quadrature components and the oscillators will be switched at the beginning of each symbol period to generate the carrier frequency for modulation. In general, all independent oscillators will not be at the same amplitude and phase at the beginning of symbol period, and therefore this causes sudden and abrupt changes in frequency for every bit change of the transmitted signal. Thus, the modulated FSK signal will be very wide with non-negligible side-lobes. Figure illustrates this point.



If such a signal is received at the receiver, the distortion caused by the wireless channel along with other effects such as interference, thermal regeneration in receiver and others will degrade the performance of FSK, which does not allow the signal to be correctly decoded with precise and accurate amplitude and phase values after every symbol period.

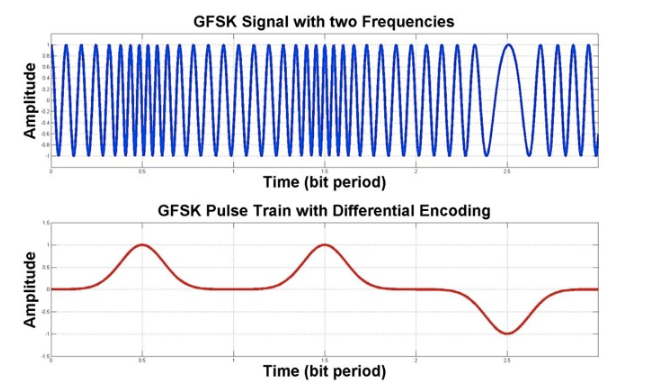
In the GFSK modulation scheme, a Gaussian filter is included before the baseband waveform signal, i.e. the data signal such as in Figure 1, is modulated by the FSK modulator circuit. The inclusion of this Gaussian filter before modulation is the difference between GFSK and FSK. A typical Gaussian filter is a filter whose impulse response is a Gaussian function as shown in Figure 2.

A diagram of a normal distribution

AI-generated content may be incorrect.

Impulse response of a Gaussian filter

Since the Gaussian function in time is still Gaussian in frequency domain, the frequency response of this filter is very narrow. When an input signal is passed through such a filter, the resulting spectral width of the filtered signal is reduced compared to FSK scheme with no filtering involved. As a result, any abrupt changes in frequency in FSK are filtered out, which make the transitions at the beginning of each symbol period relatively smoother than FSK. Figure 3 illustrates the response of GFSK signal.



 Gaussian filtered signal and GFSK modulated signal response

This method of filtering reduces sideband power or out-of-band spectrum, interference due to adjacent channels and the spectral width or bandwidth of FSK signal. However, this reduction in spectral width causes a spread in the time domain, thereby making it more likely to face intersymbol interference (ISI). Therefore, an careful design of Gaussian filter with optimal cut-off frequencies is very important to ensure that ISI-related effects can be minimized. Additionally, other robust signal processing and channel equalization techniques can also be used to overcome the effects of ISI.

This filtering stage before modulating the signal is also called pulse shaping as the data pulses are filtered out to produce a clean output signal with sharp rise and fall times, which helps in accurately determining the carrier frequency of the received signal. GFSK technique is useful and used in a wide range of wireless systems and technologies such as the Improved Layer 2 protocol, Bluetooth, IEEE 802.15.4, and [Z-wave](https://www.everythingrf.com/community/what-is-z-wave).

BLOCK DIAGRAM OF WIRELESS DIGITAL COMMUNICATION :-

source->channel encoder->modulator->transmitter->multiplexer->antenna =>

                                                            ||

                                                            channel medium =>

                                                                 ||

receiver <- channel decoder <- demodulator <-demultiplexer <-antenna

* **CHANNEL ENCODER:**

A **channel encoder** takes your original digital data and **adds extra bits** in a smart way. These extra bits help the receiver **find and fix errors** that might happen during transmission — like when the signal gets messed up by **noise**, **interference**, or **fading** in the air

* **MODULATOR:**

A **modulator** is a device or circuit in a communication system that **converts a baseband signal (like digital data or audio)** into a **form suitable for transmission** by varying a property of a high-frequency **carrier signal** — such as its amplitude, frequency, or phase — based on the input signal.

* **TRANSMITTER:**

A **transmitter** is an electronic device or circuit in a communication system that **processes and amplifies a modulated signal**, converting it into a form suitable for **radiation through an antenna** over a communication channel.

* **MULTIPLEXER:**

A **multiplexer (MUX)** is a digital circuit or device that **selects one input from multiple input signals** and **forwards it to a single output line**, based on control signals. It enables **multiple data sources** to share a single communication channel or resource efficiently.

* **ANTENNA:**

An **antenna** is a transducer that **converts electrical signals into electromagnetic waves** for transmission, and **electromagnetic waves back into electrical signals** for reception. It serves as the interface between a communication system and the propagation medium (such as air or space).

* **CHANNEL MEDIUM:**

In **wireless digital communication**, the **channel medium** refers to the **unguided physical environment** (such as air, vacuum, or space) through which **electromagnetic waves** propagate to carry digital information from the **transmitting antenna** to the **receiving antenna**. It acts as the **transmission path** for modulated radio frequency (RF) signals.

* **DEMULTIPLEXER:**

**Demultiplexing** is the process in a communication system where a **single composite signal** carrying multiple data streams is **separated back into its original individual signals**, based on control or addressing information. It is the reverse operation of multiplexing.

* **DEMODULATOR:**

A **demodulator** is a device or circuit in a communication system that **extracts the original baseband signal** (such as voice, video, or data) from a **modulated carrier wave** received over a communication channel. It performs the **inverse operation of modulation**.

* **CHANNEL DECODER:**

A **channel decoder** is a signal processing block at the receiver that takes the encoded and possibly corrupted data from the communication channel and attempts to reconstruct the original data by using the redundancy added by the channel encoder.

* **How Radio Waves Can Penetrate Through the Walls?**

Radio waves can penetrate through walls due to their **electromagnetic wave properties** and interactions with materials. Here's a technical explanation:

**1. Nature of Radio Waves**

* **Radio waves** are a type of **electromagnetic (EM) radiation** with frequencies ranging from **3 kHz to 300 GHz**.
* Bluetooth, Wi-Fi, and other wireless technologies typically use the **2.4 GHz** or **5 GHz** bands.

**2. Interaction with Materials**

When radio waves encounter a wall, several phenomena occur:

**a. Reflection**

* Part of the wave reflects off the surface.
* Depends on the **angle of incidence** and **material conductivity**.

**b. Refraction**

* The wave bends as it enters a different medium (e.g., from air to concrete).
* Governed by **Snell’s Law**.

**c. Diffraction**

* Waves bend around edges or through small openings.
* Enables signal to reach areas not in direct line-of-sight.

**d. Absorption**

* Some energy is absorbed by the wall material and converted to heat.
* Depends on the **dielectric properties** and **thickness** of the material.

**e. Transmission**

* The remaining energy passes through the wall.
* This is what allows radio waves to **penetrate**.

**3. Factors Affecting Penetration**

| **Factor** | **Effect** |
| --- | --- |
| **Frequency** | Lower frequencies (e.g., 900 MHz) penetrate better than higher ones (e.g., 5 GHz). |
| **Wavelength** | Longer wavelengths (lower frequency) diffract and penetrate more easily. |
| **Material Type** | Drywall, wood, and glass allow more penetration than concrete or metal. |
| **Wall Thickness** | Thicker walls absorb and attenuate more signal. |
| **Moisture Content** | Water absorbs radio waves, so wet walls reduce penetration. |

**4. Attenuation**

* The **attenuation** (signal loss) is measured in **decibels (dB)**.
* For example:
  + Drywall: ~3 dB loss
  + Brick: ~8–15 dB
  + Concrete: ~10–20 dB
  + Metal: ~30+ dB (often reflects completely)

**5. Mathematical Model (Simplified)**

The **transmitted power** (Pt) through a wall can be estimated using:

*Pt*​=*Pi*​⋅*e*−*αd*

Where:

* Pi : Incident power
* *α*: Attenuation coefficient (depends on material and frequency)
* d: Thickness of the wall
* **How a Bluetooth device can Share with Its Enabled Device of Size 4mb?**

**Scenario: Sending a 4 MB File via Bluetooth**

Bluetooth (especially **Bluetooth Classic**) supports file transfers using protocols like **OBEX** over **RFCOMM** (Serial Port Emulation). Here's how it works block by block:

**1. Source (Application Layer)**

* The file (e.g., image, document) is 4 MB = 4 × 1024 × 1024 = **4,194,304 bytes**.
* The application (e.g., file manager or Bluetooth sharing app) initiates the transfer.

**2. Channel Encoder (Link Layer + L2CAP + RFCOMM)**

**a. L2CAP (Logical Link Control and Adaptation Protocol)**

* Splits the 4 MB file into **smaller chunks** (typically 672 bytes per packet in Bluetooth Classic).
* Adds headers for reassembly and flow control.

**b. RFCOMM**

* Emulates serial communication.
* Adds framing and control information for reliable delivery.

**c. OBEX (Object Exchange Protocol)**

* Wraps the file data with metadata (e.g., file name, type).
* Handles session management and file transfer commands.

**3. Modulator**

* Bluetooth Classic uses **GFSK** (Gaussian Frequency Shift Keying) for Basic Rate (BR).
* For Enhanced Data Rate (EDR), it uses **π/4-DQPSK** or **8DPSK**.
* Converts digital bits into analog waveforms.

**4. Transmitter + Multiplexer**

* The modulated signal is prepared and slotted into time slots (Bluetooth uses **Time Division Duplexing**).
* Multiplexing allows multiple devices to share the same channel.

**5. Antenna**

* Transmits the signal over the **2.4 GHz ISM band** using **frequency hopping** (1600 hops/sec across 79 channels).

**6. Channel Medium**

* The signal travels through air.
* Bluetooth uses **FHSS (Frequency Hopping Spread Spectrum)** to reduce interference.

**7. Receiver Side**

**a. Antenna**

* Captures the incoming signal.

**b. Demultiplexer + Demodulator**

* Extracts the signal from the time slot.
* Converts analog waveforms back into digital bits.

**c. Channel Decoder (L2CAP + RFCOMM + OBEX)**

* Reassembles the chunks into the original 4 MB file.
* Checks for errors using CRC and retransmits if needed.
* OBEX finalizes the file transfer.

**8. Receiver (Application Layer)**

* The file is saved or opened on the receiving device.

**Transfer Time Estimation (Bluetooth Classic)**

* **Max payload per packet**: ~672 bytes (L2CAP)
* **Max data rate**: ~721 kbps (Bluetooth 2.0 + EDR)
* **Estimated time**:  
  (4,194,304 bytes×8​) / (721,000 bps) ≈ 46.5 seconds
* **What Are the Inputs for Multiplexer in Bluetooth?**

In Bluetooth, the multiplexer receives inputs from multiple logical data streams or protocol layers and combines them for transmission over a single physical radio channel. Here's a breakdown of the inputs to the multiplexer in Bluetooth:

**Inputs to the Multiplexer in Bluetooth**

1. L2CAP Channels

* L2CAP (Logical Link Control and Adaptation Protocol) provides multiple logical channels.
* Each channel carries data from a different higher-layer protocol.
* Inputs:
  + ATT (Attribute Protocol) – used in BLE for GATT profiles
  + SDP (Service Discovery Protocol)
  + RFCOMM – for serial port emulation
  + AVCTP/AVDTP – for audio/video control and streaming

2. RFCOMM Sessions

* Multiple virtual serial ports can be active.
* Each port is identified by a DLCI (Data Link Connection Identifier).
* These are multiplexed into a single RFCOMM stream.

3. SCO/eSCO Audio Streams (Bluetooth Classic)

* For voice data (e.g., during a call).
* These are time-sensitive and get dedicated slots.

4. ACL Data Streams

* Asynchronous Connection-Less (ACL) packets carry general data (e.g., file transfer, browsing).
* Multiple ACL streams can be active, each from a different application.

5. Control and Management Data

* Link management commands (e.g., pairing, encryption setup).
* Host Controller Interface (HCI) commands and events.

**How the Multiplexer Uses These Inputs**

* Assigns time slots to each stream (TDM).
* Tags each stream with channel identifiers (e.g., CID for L2CAP, DLCI for RFCOMM).
* Ensures QoS (Quality of Service) for time-sensitive data like audio.
* Prioritizes and schedules packets for transmission.

**Summary Table:**

| Input Source | Protocol Layer | Identifier Used | Purpose |
| --- | --- | --- | --- |
| GATT/ATT | L2CAP | CID | BLE attribute data |
| File Transfer | RFCOMM | DLCI | Serial port emulation |
| Audio Stream | SCO/eSCO | Slot assignment | Real-time voice |
| Control Commands | HCI | Opcode/Event ID | Device management |
| Service Discovery | SDP | CID | Finding services |

* **How The Bluetooth Changing Its Channels**

Bluetooth changes its channels using a technique called **Frequency Hopping Spread Spectrum (FHSS)**, and in modern versions, **Adaptive Frequency Hopping (AFH)**. Here's a technical explanation of how this channel switching works:

**1. Frequency Hopping Spread Spectrum (FHSS)**

* Bluetooth divides the 2.4 GHz ISM band into **79 channels** (for BR/EDR) or **40 channels** (for BLE).
* Each channel is **1 MHz** (BR/EDR) or **2 MHz** (BLE) wide.
* Devices **hop between channels** up to **1600 times per second** to avoid interference and improve reliability.

**2. Channel Selection Algorithm (CSA)**

Bluetooth uses a **Channel Selection Algorithm** to determine the next channel:

**🔹 CSA #1 (Legacy)**

* Uses a simple formula:

**Next Channel = (Current Channel + Hop Increment) mod N**

* + Where (N) is the number of available channels (e.g., 79).
  + The **hop increment** is derived from the master’s **Bluetooth Device Address (BD\_ADDR)**.

**🔹 CSA #2 (BLE 5.0+)**

* Uses a **permutation function** and **unmapped channel index**:
  + Provides better **randomness** and **collision avoidance**.
  + Takes into account the **channel map** (which channels are usable).

**CSA2 [CHANNEL SELECTION ALGORITHM]:**

Great! Let's dive into the **technical details of Channel Selection Algorithm #2 (CSA#2)** used in **Bluetooth Low Energy (BLE) 5.0 and later**.

**What is CSA#2?**

CSA#2 is a **deterministic, pseudo-random channel selection algorithm** designed to:

* Improve **channel distribution**
* Reduce **collisions**
* Support **Adaptive Frequency Hopping (AFH)**

It replaces the simpler CSA#1 used in earlier BLE versions.

**CSA#2: Step-by-Step Technical Breakdown**

Let’s denote:

* ch\_id: Channel Identifier (derived from the Access Address)
* event\_counter: Incremented for each connection event
* channel\_map: Bitmask of usable channels (37 bits for BLE)
* num\_used\_channels: Number of '1's in the channel map

**Step 1: Generate Unmapped Channel Index**

CSA#2 uses a **permutation function** to generate a pseudo-random number:

*prne*​=permute (*ch*\_*id*, *event*\_*counter*)

*unmapped*\_*channel*=*prne*​ mod 37

This gives a **uniform distribution** over all 37 data channels.

**Step 2: Remap to a Valid Channel**

If the unmapped\_channel is **not enabled** in the channel\_map, it is **remapped** to the next available channel using a **remapping table**.

channel\_index = remap (*unmapped*\_*channel*, *channel*\_*map*)

This ensures that only **usable channels** are selected, supporting **AFH**.

**Permutation Function Details**

The permutation function is a **bitwise mixing function** that uses:

* **XOR**, **bit shifts**, and **modulo operations**
* Ensures **pseudo-randomness** and **determinism**
* Based on the **Bluetooth Access Address** (unique per connection)

**Advantages of CSA#2**

| **Feature** | **Benefit** |
| --- | --- |
| Uniform Distribution | Reduces channel bias |
| AFH Support | Avoids bad channels dynamically |
| Deterministic | Devices stay synchronized |
| Efficient | Low computational overhead |

**3. Adaptive Frequency Hopping (AFH)**

* AFH dynamically **excludes bad channels** (e.g., those with high error rates or interference).
* The **master device** maintains a **channel map** (a bitmask of usable channels).
* This map is shared with the slave device via **LMP (Link Manager Protocol)** or **LL (Link Layer)** in BLE.
* The hopping sequence is then applied **only to the “good” channels**.

**4. Synchronization**

* The master and slave devices **synchronize their clocks** and **channel maps**.
* This ensures they **hop to the same channel at the same time**.

**5. Example (Simplified)**

Let’s say:

* Total channels: 79
* Hop increment: 23
* Current channel: 5

Then: **Next Channel** = (5+23) mod 79 = 28

If channel 28 is marked “bad” in AFH, the algorithm skips it and selects the next “good” one.

**What is a Channel Map?**

A **channel map** is a **bitmask** that represents the **availability of each data channel** in BLE. It is used by both the master and slave devices to **synchronize** their hopping behaviour and avoid "bad" channels.

* BLE uses **40 RF channels**:
  + **3 advertising channels** (37, 38, 39)
  + **37 data channels** (0–36)
* The channel map is a **37-bit field**:
  + Each bit corresponds to one data channel.
  + 1 = channel is **usable**
  + 0 = channel is **unusable** (due to interference, noise, etc.)

Eg: channel\_map = 1111111111111111111111111111111111100

This means channel 0 is disabled, and the rest are enabled.

**How It Works**

1. **Initialization**:
   * At connection setup, the master sends an initial channel map to the slave.
2. **Monitoring**:
   * Devices monitor **packet error rates (PER)**, **RSSI**, and **link quality**.
3. **Update**:
   * If a channel performs poorly, the master updates the channel map.
   * The new map is sent to the slave via the **Link Layer Control PDU**.
4. **Usage**:
   * The **Channel Selection Algorithm (CSA#2)** uses this map to:
     + Generate a pseudo-random channel index.
     + **Remap** it to a valid channel using the current channel map.

* **How Bluetooth is knowing there is a Channel with Interference?**

**1. Packet Error Rate (PER)**

* Bluetooth devices monitor the **success rate of packet transmissions**.
* If a high number of packets are **not acknowledged (NACK)** or **require retransmission**, the **PER increases**.
* A high PER on a specific channel suggests **interference or poor signal quality**.

**2. RSSI (Received Signal Strength Indicator)**

* RSSI measures the **power level** of received signals.
* If the RSSI is **too low**, it may indicate **weak signal** or **interference**.
* If RSSI is **high but packet errors are frequent**, it may indicate **co-channel interference** (e.g., from Wi-Fi).

**3. Link Quality (LQ)**

* LQ is a composite metric derived from:
  + Bit Error Rate (BER)
  + RSSI
  + PER
* A **declining LQ** on a specific channel signal that the channel is becoming unreliable.

**4. Adaptive Frequency Hopping (AFH) Feedback Loop**

* The **master device** maintains a **channel map**.
* When interference is detected:
  1. The master marks the channel as “bad”.
  2. It updates the channel map to **exclude** that channel.
  3. The new map is sent to the slave via **Link Layer Control PDUs**.
* This ensures both devices **avoid the bad channel** in future hops.

**5. Interference Sources Detected**

Bluetooth can detect interference from:

* **Wi-Fi (especially 2.4 GHz)**: overlapping channels
* **Microwave ovens**
* **Other Bluetooth devices**
* **Zigbee or proprietary 2.4 GHz systems**

**Summary Table**

| **Metric** | **What It Detects** | **Action Taken** |
| --- | --- | --- |
| PER | Packet loss | Channel marked bad |
| RSSI | Signal strength | Low RSSI = weak signal |
| LQ | Overall quality | Drop = interference |
| AFH | Channel exclusion | Updates channel map |

* **What Is Sensitivity?**

**Receiver sensitivity** is the **minimum signal strength (in dBm)** that a Bluetooth receiver can detect and still **correctly decode the data**.

* It’s usually expressed in **negative dBm** (e.g., **-90 dBm**).
* The **lower (more negative)** the number, the **better** the sensitivity.

**Role of Sensitivity in the Channel Medium**

1. **Determines Communication Range**
   * Better sensitivity allows devices to **receive weaker signals**, increasing the **effective range**.
   * For example:
     + A device with **-90 dBm** sensitivity can receive signals from farther away than one with **-80 dBm**.
2. **Affects Link Reliability**
   * In noisy or obstructed environments, a more sensitive receiver can **maintain a stable connection**.
   * Helps in **low-power** scenarios where the transmitter sends weak signals.
3. **Impacts Frequency Hopping**
   * During **frequency hopping**, the receiver must quickly lock onto the new frequency and detect the signal.
   * High sensitivity ensures **faster and more accurate synchronization**.
4. **Works with Adaptive Frequency Hopping (AFH)**
   * Sensitivity helps detect **interference** on channels.
   * Channels with poor signal quality (below sensitivity threshold) are marked as **bad** and avoided.

**------------------------------------------------------------------------------------------------**

* **Receiver Sensitivity**
* **Definition**: The **minimum signal strength** (in dBm) that a Bluetooth receiver can detect and still correctly decode the signal.
* **Typical Value**: Around **-70 dBm to -90 dBm** for Bluetooth devices.
* **Significance**:
  + A **more sensitive** receiver (e.g., -90 dBm) can detect **weaker signals**, which improves range and reliability.
  + However, it may also pick up more **noise and interference**.
* **Signal Threshold**
* **Definition**: A **predefined signal strength level** used to make decisions, such as:
  + Whether to accept or reject a connection.
  + Whether to switch channels (in Adaptive Frequency Hopping).
  + Whether to adjust transmission power.
* **Types of Thresholds**:
  + **RSSI Threshold**: Used to determine if a signal is strong enough to maintain a connection.
  + **Interference Threshold**: Used to detect if a channel is too noisy and should be avoided.
* **How They Work Together**
* If the **received signal strength (RSSI)** is **above the sensitivity** and **above the threshold**, the device can:
  + Maintain a stable connection.
  + Avoid unnecessary retransmissions.
* If the signal is **below the threshold** but **above sensitivity**, the device might:
  + Lower data rates.
  + Increase transmission power.
  + Switch to a better channel.

**HOW THE BLUETOOTH DEVICES CONNECT TO EACH OTHER:**

**1. Inquiry Process**

**Purpose**: To discover nearby Bluetooth devices.

* **Example**: You turn on Bluetooth on your mobile phone and make it discoverable.
* **Action**: The phone (inquiring device) sends out inquiry requests to find other Bluetooth devices within range.
* **Response**: The wireless earbuds (discoverable device) respond with their Bluetooth Device Address (BD\_ADDR) and other relevant information.

**2. Paging Process**

**Purpose**: To establish a connection with a specific device.

* **Example**: You select your wireless earbuds from the list of discovered devices on your phone.
* **Action**: The phone (master device) sends a paging message to the earbuds (slave device) using the BD\_ADDR obtained during the inquiry process.
* **Substates**:
  + **Page Substate**: The phone sends out paging messages.
  + **Page Scan Substate**: The earbuds listen for paging messages.
  + **Page Response Substate**: The earbuds respond to the paging message, indicating they are ready to connect.

**3. Connection Establishment**

**Purpose**: To complete the connection and start communication.

* **Example**: The phone and earbuds synchronize their communication parameters.
* **Action**: The devices exchange information to match frequency hopping patterns and timing of radio packets.
* **Result**: A link layer connection is established, allowing data transfer.

**Detailed Example**

1. **Inquiry**:
   * **Phone**: "Are there any Bluetooth devices nearby?"
   * **Earbuds**: "Yes, I'm here! My address is XX:XX:XX:XX:XX:XX."
2. **Paging**:
   * **Phone**: "Hey, earbuds at XX:XX:XX:XX:XX:XX, let's connect!"
   * **Earbuds**: "Got it, I'm ready to connect."
3. **Connection**:
   * **Phone and Earbuds**: "Let's synchronize our communication settings."
   * **Result**: "Connection established! We can now transfer audio data."

This process ensures that your phone and earbuds can find each other, establish a secure connection, and communicate effectively

To explain the **discovery and connectivity process of two Bluetooth devices** with respect to the **physical channel**, let’s break it down into **technical phases** that occur over the **Bluetooth physical layer** (radio interface):

**🔹 1. Inquiry Phase (Discovery)**

**Purpose:**

One device (the **inquirer**) searches for nearby Bluetooth devices (the **responders**).

**Physical Channel Behaviour:**

* Operates on **32 of the 79 RF channels** in the **2.4 GHz ISM band**.
* Uses a **dedicated inquiry hopping sequence**.
* Hops every **312.5 µs**, scanning **2 frequencies per 625 µs slot**.
* The inquirer sends **ID packets** (68-bit) repeatedly.
* Responders listen on a **deterministic schedule** (called the **inquiry scan window**).

**Key Technical Points:**

* **Inquiry packets** are sent using **GFSK modulation**.
* Devices use a **pseudo-random hopping sequence** derived from the inquirer’s address.
* Responders reply with an **FHS (Frequency Hop Synchronization)** packet containing their Bluetooth address and clock.

**🔹 2. Page Phase (Connection Establishment)**

**Purpose:**

The inquirer (now the **master**) pages a specific device (the **slave**) to establish a connection.

**Physical Channel Behaviour:**

* Uses **all 79 channels** but follows a **page hopping sequence** based on the slave’s address.
* The master sends **page packets** (ID or FHS) on this sequence.
* The slave listens on a **page scan window** using a known schedule.

**Key Technical Points:**

* The master sends an **ID packet** followed by an **FHS packet**.
* The slave responds with an **ID packet**, then an **ACK**.
* Once the slave acknowledges, both devices switch to a **new hopping sequence** for the connection.

**🔹 3. Connection Phase**

**Purpose:**

Devices exchange data using a **time-division duplex (TDD)** scheme.

**Physical Channel Behaviour:**

* Operates on **all 79 channels** using a **frequency hopping sequence** derived from the master's address and clock.
* Each time slot is **625 µs**.
  + Master transmits in **even-numbered slots**.
  + Slave transmits in **odd-numbered slots**.
* Supports **1-slot, 3-slot, or 5-slot packets**.

**Key Technical Points:**

* Modulation:
  + **GFSK** for Basic Rate (1 Mbps)
  + **π/4-DQPSK** (2 Mbps) and **8DPSK** (3 Mbps) for Enhanced Data Rate
* The hopping sequence ensures **robustness against interference** and **security**.

**🔄 Summary Timeline**

| **Phase** | **Channels Used** | **Hopping** | **Slot Duration** | **Modulation** | **Purpose** |
| --- | --- | --- | --- | --- | --- |
| Inquiry | 32 | 312.5 µs hop | 625 µs (2 hops/slot) | GFSK | Device discovery |
| Page | 79 | 625 µs hop | 625 µs | GFSK | Connection setup |
| Connection | 79 | 625 µs hop | 625 µs | GFSK / DPSK | Data exchange |

Architecture:

**GENERAL DESCRIPTION:**

* This part specifies the normal operation of a Bluetooth baseband. Bluetooth system provides a point-to-point connection or a point-to-multi point connection.
* In a point-to-point connection the physical channel is shared between two Bluetooth devices.
* In a point-to-multipoint connection, the physical channel is shared among several Bluetooth devices. Two or more devices sharing the same physical channel form a piconet. One Bluetooth device acts as the master of the piconet, whereas the other device(s) act as slave(s). Up to seven slaves can be active in the piconet. Additionally, many more slaves can remain connected in a parked state. These parked slaves are not active on the channel but remain synchronized to the master and can become active without using the connection establishment procedure. Both for active and parked slaves, the channel access is controlled by the master.
* Piconets that have common devices are called a scatternet, see (c) in Figure 1.1 on page 63. Each piconet only has a single master, however, slaves can participate in different piconets on a time-division multiplex basis. In addition, a master in one piconet can be a slave in other piconets. Piconets shall not be frequency synchronized and each piconet has its own hopping sequence.
* Data is transmitted over the air in packets. Two modes are defined: a mandatory mode called Basic Rate and an optional mode called Enhanced Data Rate. The symbol rate for all modulation modes is 1 Ms/s. The gross air data rate is 1 Mbps for Basic Rate. Enhanced Data Rate has a primary modulation mode that provides a gross air data rate of 2 Mbps, and a secondary modulation mode that provides a gross air data rate of 3 Mbps.
* The general Basic Rate packet format, each packet consists of 3 entities: the access code, the header, and the payload

|  |  |  |
| --- | --- | --- |
| ACCESS CODE | HEADER | PAYLOAD |

* The general Enhanced Data Rate packet format is shown in Figure 1.3. Each packet consists of 6 entities: the access code, the header, the guard period, the synchronization sequence, the Enhanced Data Rate payload and the trailer. The access code and header use the same modulation mode as for Basic Rate packets while the synchronization sequence, the Enhanced Data Rate payload and the trailer use the Enhanced Data Rate modulation mode. The guard time allows for the transition between the modulation modes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ACCESS CODE | HEADER | GUARD | SYNC | ENHANCED DATA RATE PAYLOAD | TRA ILER |
| **GFSK**  **DPSK** | | | | | |

**BLUETOOTH CLOCK**:

Every Bluetooth device shall have a native clock that shall be derived from a free running system clock. For synchronization with other devices, offsets are used that, when added to the native clock, provide temporary Bluetooth clocks that are mutually synchronized. It should be noted that the Bluetooth clock has no relation to the time of day; it may therefore be initialized to any value. The clock has a cycle of about a day. If the clock is implemented with a counter, a 28-bit counter is required that shall wrap around at 228-1. The least significant bit (LSB) shall tick in units of 312.5 μs (i.e. half a time slot), giving a clock rate of 3.2 kHz. The clock determines critical periods and triggers the events in the device. Four periods are important in the Bluetooth system: 312.5 μs, 625 μs, 1.25 ms, and 1.28 s; these periods correspond to the timer bits CLK0, CLK1, CLK2, and CLK12, respectively.

In the different modes and states a device can reside in, the clock has different appearances:

• CLKN native clock

• CLKE estimated clock

• CLK master clock

CLKN is the native clock and shall be the reference to all other clock appearances. In STANDBY and in Park, Hold, and Sniff modes the native clock may be driven by a low power oscillator (LPO) with worst case accuracy (+/- 250ppm). Otherwise, the native clock shall be driven by the reference crystal oscillator with worst case accuracy of +/-20ppm. The master shall never adjust its native clock during the existence of the piconet.

**BLUETOOTH DEVICE ADDRESSING:**

Each Bluetooth device shall be allocated a unique 48-bit Bluetooth device address (BD\_ADDR). This address shall be obtained from the IEEE Registration Authority. The address is divided into the following three fields:

• LAP field: lower address part consisting of 24 bits

• UAP field: upper address part consisting of 8 bits

• NAP field: non-significant address part consisting of 16 bits

The BD\_ADDR may take any values except the 64 reserved LAP values for general and dedicated inquiries.

**ACCESS CODES:**

In the Bluetooth system all transmissions over the physical channel begin with an access code. Three different access codes are defined, see also Section 6.3.1 on page 108:

• device access code (DAC)

• channel access code (CAC)

• inquiry access code (IAC)

* All access codes are derived from the LAP of a device address or an inquiry address. The device access code is used during page, page scan and page response substates and shall be derived from the paged device’s BD\_ADDR.
* The channel access code is used in the CONNECTION state and forms the beginning of all packets exchanged on the piconet physical channel.

The channel access code shall be derived from the LAP of the master’s BD\_ADDR.

* Finally, the inquiry access code shall be used in the inquiry substate. There is one general IAC (GIAC) for general inquiry operations and there are 63 dedicated IACs (DIACs) for dedicated inquiry operations.

The access code also indicates to the receiver the arrival of a packet. It is used for timing synchronization and offset compensation. The receiver correlates against the entire synchronization word in the access code, providing very robust signalling.

**PHYSICAL CHANNEL:**

* The lowest architectural layer in the Bluetooth system is the physical layer.
* All Bluetooth physical channels are characterized by the combination of a pseudo-random frequency hopping sequence, the specific slot timing of the transmissions, the access code and packet header encoding.
* For the basic and adapted piconet physical channels frequency hopping is used to change frequency periodically to reduce the effects of interference and to satisfy local regulatory requirements
* Two devices that wish to communicate use a shared physical channel for this communication. To achieve this, their transceivers must be tuned to the same RF frequency at the same time, and they must be within a nominal range of each other.
* Given that the number of RF carriers is limited and that many Bluetooth devices could be operating independently within the same spatial and temporal area there is a strong likelihood of two independent Bluetooth devices having their transceivers tuned to the same RF carrier, resulting in a physical channel collision.
* To mitigate the unwanted effects of this collision each transmission on a physical channel starts with an access code that is used as a correlation code by devices tuned to the physical channel. This channel access code is a property of the physical channel. The access code is always present at the start of every transmitted packet.
* Four Bluetooth physical channels are defined. Each is optimized and used for a different purpose.
* Two of these physical channels (the basic piconet channel and adapted piconet channel) are used for communication between connected devices and are associated with a specific piconet.
* The other two physical channels are used for discovering (the inquiry scan channel) and connecting (the page scan channel) Bluetooth devices.
* The synchronization scan physical channel is used by devices to obtain timing and frequency information about the Connectionless Peripheral Broadcast physical link or to recover the current piconet clock.
* A Bluetooth device can only use one of these physical channels at any given time. To support multiple concurrent operations the device uses time division multiplexing between the channels. In this way a Bluetooth device can appear to operate simultaneously in several piconets, as well as being discoverable and connectable.
* Whenever a Bluetooth device is synchronized to the timing, frequency and access code of a physical channel it is said to be 'connected' to this channel (whether or not it is actively involved in communications over the channel). At a minimum, a device need only be capable of connection to one physical channel at a time, however, advanced devices may be capable of connecting simultaneously to more than one physical channel, but the specification does not assume that this is possible.

**PHYSICAL CHANNEL DEFINITION:**

* Physical channels are defined by a pseudo-random RF channel hopping sequence, the packet (slot) timing and an access code. The hopping sequence is determined by the UAP and LAP of a Bluetooth device address and the selected hopping sequence. The phase in the hopping sequence is determined by the Bluetooth clock.
* All physical channels are subdivided into time slots whose length is different depending on the physical channel. Within the physical channel, each reception or transmission event is associated with a time slot or time slots. For each reception or transmission event an RF channel is selected by the hop selection kernel (see Section 2.6 on page 81). The maximum hop rate is 1600 hops/s in the CONNECTION state and the maximum is 3200 hops/s in the inquiry and page substates.
* The following physical channels are defined:

• basic piconet physical channel

• adapted piconet physical channel

• page scan physical channel

• inquiry scan physical channel

• synchronization scan physical channel

**1.BASIC PICONET PHYSICAL CHANNEL:**

* During the Connection state the basic piconet physical channel is used by default. The adapted piconet physical channel may also be used. The adapted piconet physical channel is identical to the basic piconet physical channel except for the differences.
* The basic piconet physical channel is defined by the Central of the piconet. The Central controls the traffic on the piconet physical channel by a polling scheme. The device that initiates a connection by paging is the Central. Once a piconet has been established, the Central and Peripheral may exchange roles.
* The basic piconet physical channel is characterized by a pseudo-random hopping through all 79 RF channels. The frequency hopping in the piconet physical channel is determined by the Bluetooth clock and BD\_ADDR of the Central. When the piconet is established, the Central's clock is communicated to the Peripherals. Each Peripheral shall add an offset to its native clock to synchronize with the Central's clock. Since the clocks are independent, the offsets must be updated regularly. All devices participating in the piconet are time-synchronized and hop-synchronized to the channel.

**Why 625 microseconds?**

The 625-microsecond duration is long enough to transmit data effectively while still allowing for frequent hops.

**TIME SLOT:**

A time slot is a fixed-duration interval used to organize and schedule transmissions between Bluetooth devices. In Bluetooth BR/EDR, each time slot is 625 microseconds (µs) long.

Bluetooth uses a **Time Division Duplex (TDD)** scheme, where devices take turns transmitting and receiving data in alternating time slots.

* **Master and Slave Roles**: In a piconet (Bluetooth network), the **master** device initiates communication and controls the timing.
* **Alternating Slots**:
  + The **master transmits** in **even-numbered slots** (0, 2, 4, …).
  + The **slave responds** in **odd-numbered slots** (1, 3, 5, …).

This ensures that both devices don’t transmit at the same time, avoiding collisions.

**TRANSMIT/RECEIVE:**

The Central transmission shall always start at even numbered time slots (CLK1=0) and the Peripheral transmission shall always start at odd numbered time slots (CLK1=1). Due to packet types that cover more than a single slot, Central transmission may continue in odd numbered slots and Peripheral transmission may continue in even numbered slots.

**2.ADAPTED PICONET PHYSICAL CHANNEL:**

* For devices that enter Connectionless Peripheral Broadcast mode, the device that transmits Connectionless Peripheral Broadcast packets is the Central of the piconet and any device that receives Connectionless Peripheral Broadcast packets is a Peripheral of the piconet.
* The adapted piconet physical channel uses the adapted channel hopping sequence.
* adaptive frequency hopping (AFH) enabled. There are two distinctions between basic and adapted piconet physical channels. The first is the same channel mechanism that makes the Peripheral frequency the same as the preceding Central transmission. The second aspect is that the adapted piconet physical channel may be based on less than the full 79 frequencies of the basic piconet physical channel. Each physical link on an adapted piconet physical channel may use a different set of frequencies.

**3.PAGE SCAN PHYSICAL CHANNEL:**

Although Central and Peripheral roles are not defined prior to a connection, the term Central is used for the paging device (that becomes a Central in the Connection state) and Peripheral is used for the page scanning device (that becomes a Peripheral in the Connection state).

|  |
| --- |
| HOPPING PATTERN:  a **hopping pattern** refers to the **sequence of frequency channels** that devices use to transmit and receive data. This is part of Bluetooth's **Frequency Hopping Spread Spectrum (FHSS)** technique, which helps reduce interference and improve security.   1. **Hopping Sequence**:    * Determined by the **master device’s Bluetooth address** and **clock**.    * Ensures that each connection has a **unique pseudo-random sequence**.    * The sequence is **deterministic** but appears random to outsiders. 2. **Types of Hopping Patterns**:    * **Inquiry Hopping**: Uses a subset of 32 channels.    * **Page Hopping**: Uses all 79 channels but in a pattern based on the slave’s address.    * **Connection Hopping**: Full 79-channel sequence based on the master’s address and clock. |

------------------------------------------------------------------------------------------------

**COLLISION DETECTION IN INQUIRY:**

Certainly! Let's delve into the technical details of the backoff procedure during the Bluetooth inquiry scan:

1. **Collision Detection**: During the inquiry scan, a Bluetooth device listens for inquiry responses from other devices. If it detects a collision (i.e., two or more devices transmitting simultaneously on the same frequency), it recognizes that the transmission was unsuccessful.
2. **Random Backoff Interval**: To avoid repeated collisions, the device employs a backoff procedure. This involves waiting for a random period before attempting to transmit again. The random interval is chosen from a predefined range, ensuring that the timing of retransmissions is staggered among different devices [[1]](https://staff.csie.ncu.edu.tw/jrjiang/publication/Analysis%20of%20Bluetooth%20Device%20Discovery%20and%20Some%20Speedup%20Mechanisms.pdf).
3. **Backoff Algorithm**: The backoff interval is typically determined using a pseudo-random number generator. The range of possible backoff intervals can be adjusted based on the number of collisions detected. For example, after the first collision, the device might choose a backoff interval from a smaller range. If collisions persist, the range can be increased to further reduce the likelihood of repeated collisions [[1]](https://staff.csie.ncu.edu.tw/jrjiang/publication/Analysis%20of%20Bluetooth%20Device%20Discovery%20and%20Some%20Speedup%20Mechanisms.pdf).
4. **Exponential Backoff**: In some cases, an exponential backoff algorithm is used. This means that the range of possible backoff intervals increases exponentially with each subsequent collision. This approach helps to quickly reduce the collision rate in environments with high levels of interference [[2]](https://www.khoury.northeastern.edu/home/rraj/Courses/6710/S10/Lectures/BluetoothMobileIP.pdf).
5. **Frequency Hopping**: During the backoff period, the device continues to follow the frequency hopping sequence. This means that when it retries the transmission, it will likely be on a different frequency, further reducing the chances of collision [[3]](https://people.cs.nycu.edu.tw/~yctseng/WSN2012-09/book-ch04-4%28bt-v1.2%29.ppt).
6. **Inquiry Scan Window**: The inquiry scan window is divided into multiple slots, each corresponding to a different frequency. By randomizing the backoff interval and continuing to hop frequencies, the device ensures that its retransmissions are spread out both in time and across different frequencies [[3]](https://people.cs.nycu.edu.tw/~yctseng/WSN2012-09/book-ch04-4%28bt-v1.2%29.ppt).

These mechanisms work together to ensure that even in environments with high levels of interference, Bluetooth devices can successfully complete the inquiry process and establish connections.

Would you like more details on any specific aspect of this procedure?

[[1]](https://staff.csie.ncu.edu.tw/jrjiang/publication/Analysis%20of%20Bluetooth%20Device%20Discovery%20and%20Some%20Speedup%20Mechanisms.pdf): Bluetooth Technology Website [[2]](https://www.khoury.northeastern.edu/home/rraj/Courses/6710/S10/Lectures/BluetoothMobileIP.pdf): Concurrent Interference Cancellation [[3]](https://people.cs.nycu.edu.tw/~yctseng/WSN2012-09/book-ch04-4%28bt-v1.2%29.ppt): [Analysis of Bluetooth Device Discovery](https://staff.csie.ncu.edu.tw/jrjiang/publication/Analysis%20of%20Bluetooth%20Device%20Discovery%20and%20Some%20Speedup%20Mechanisms.pdf)

**References**

[1] [Analysis of Bluetooth Device Discovery and Some Speedup Mechanisms](https://staff.csie.ncu.edu.tw/jrjiang/publication/Analysis%20of%20Bluetooth%20Device%20Discovery%20and%20Some%20Speedup%20Mechanisms.pdf)

[2] [Bluetooth and Mobile IP - Khoury College of Computer Sciences](https://www.khoury.northeastern.edu/home/rraj/Courses/6710/S10/Lectures/BluetoothMobileIP.pdf)

[3] [Several new features are introduced in the Bluetooth Core Specification ...](https://people.cs.nycu.edu.tw/~yctseng/WSN2012-09/book-ch04-4%28bt-v1.2%29.ppt)

Sure! Let's go through an example to illustrate how the backoff procedure works in the Bluetooth inquiry substate:

**Example Scenario**

Imagine three Bluetooth devices (A, B, and C) are trying to respond to an inquiry request at the same time. This can lead to collisions if they transmit their responses simultaneously on the same frequency.

**Step-by-Step Process**

1. **Initial Collision Detection**:
   * All three devices (A, B, and C) detect a collision because their responses interfere with each other.
2. **Random Backoff Interval**:
   * Each device selects a random backoff interval from a predefined range. Let's say the range is 0 to 10 milliseconds (ms).
   * Device A randomly selects 3 ms, Device B selects 7 ms, and Device C selects 2 ms.
3. **First Retransmission Attempt**:
   * After waiting for their respective backoff intervals, the devices attempt to retransmit their responses.
   * Device C transmits first (after 2 ms), followed by Device A (after 3 ms), and then Device B (after 7 ms).
4. **Collision Resolution**:
   * If Device C's transmission is successful, it will not need to retransmit.
   * If Device A and Device B still collide, they will detect the collision and enter another backoff period.
5. **Adjusting Backoff Range**:
   * After the second collision, the devices might increase the range of the backoff interval to reduce the likelihood of repeated collisions. For example, the new range could be 0 to 20 ms.
   * Device A randomly selects 12 ms, and Device B selects 18 ms.
6. **Second Retransmission Attempt**:
   * After waiting for their new backoff intervals, Device A and Device B attempt to retransmit.
   * Device A transmits after 12 ms, and Device B transmits after 18 ms.
7. **Successful Transmission**:
   * With the increased backoff range, the likelihood of collision is reduced, and both devices are more likely to successfully transmit their responses without interference.

**Exponential Backoff**

In environments with high levels of interference, an exponential backoff algorithm can be used. This means that the range of possible backoff intervals increases exponentially with each subsequent collision. For example:

* **First Collision**: Range = 0 to 10 ms
* **Second Collision**: Range = 0 to 20 ms
* **Third Collision**: Range = 0 to 40 ms

This exponential increase helps to quickly reduce the collision rate by spreading out the retransmission attempts over a wider time range.

**Summary**

By using a random backoff interval and adjusting the range based on the number of collisions, Bluetooth devices can effectively manage and resolve collisions during the inquiry substate. This ensures that devices can successfully complete the inquiry process and establish connections even in environments with high levels of interference.

**BLUETOOTH TRANSPORT:**

Great! Let's break down each component of the Bluetooth transport architecture:

**1. Physical Layer**

This is the lowest layer, dealing with the actual transmission of data over the air.

* **Physical Channels**: information related to frequency channels, time slotting, access code and packet header.
* **Physical Links**: These are the actual radio connections established between devices.
* **Physical Transports**: These describe how data is physically transmitted and received, including modulation and signal characteristics.

**2. Logical Layer**

This layer manages the logical aspects of data transport, ensuring efficient and reliable communication.

* **Logical Links**: These are virtual pathways that connect devices for data transfer. They include:
  + **ACL (Asynchronous Connection-Less)**: For general data transfer.
  + **SCO (Synchronous Connection-Oriented)**: For real-time data like voice.
  + **eSCO (Extended SCO)**: An enhanced version of SCO with better error handling.
* **Logical Transports**: These describe how different logical links interact and share resources. They help manage the complexity of shared resources like addresses and acknowledgment schemes.

**3. L2CAP Layer**

This layer handles the logical connections and data management between devices.

* **L2CAP Channels**: These facilitate the transfer of data packets between higher-level protocols and applications. They handle:
  + **Data Transfer**: Moving data between devices.
  + **Multiplexing and Segmentation**: Allowing multiple protocols to use the same connection and managing large data packets.
  + **Flow Control and Error Management**: Ensuring reliable data transfer.

**Summary**

* **Physical Layer**: Deals with actual data transmission over the air.
* **Logical Layer**: Manages logical data pathways and their interactions.
* **L2CAP Layer**: Handles logical connections and data management.

This layered approach ensures that Bluetooth communication is efficient, reliable, and compatible with various devices and applications.

Does this help clarify the Bluetooth transport architecture? Let me know if you have any more questions!

**CORE SYSTEM ARCHITECTURE:**

The Bluetooth Core system consists of two main parts:

1. **Host**: Manages higher-level protocols and services.
2. **Controller**: Manages the radio and low-level communication.

**Minimal Implementations**

1. **Bluetooth BR/EDR (Basic Rate/Enhanced Data Rate)**:
   * **Layers and Protocols**: Covers the four lowest layers and associated protocols.
   * **Service Layer Protocol**: Includes the Service Discovery Protocol (SDP).
   * **Profile Requirements**: Defined by the Generic Access Profile (GAP).
2. **Bluetooth LE (Low Energy)**:
   * **Layers and Protocols**: Also covers the four lowest layers and associated protocols.
   * **Service Layer Protocols**: Includes the Security Manager (SM) and Attribute Protocol (ATT).
   * **Profile Requirements**: Defined by the Generic Attribute Profile (GATT) and Generic Access Profile (GAP).

**Combined Implementations**

* **Bluetooth BR/EDR and LE**: Combines both minimal implementations described above.

**Additional Protocols**

* A complete Bluetooth application needs more service and higher layer protocols, which are defined in the Bluetooth specification but not detailed here.

**Bluetooth Core System Components**

1. **Controller**: Manages the radio and low-level communication.
   * **BR/EDR Controller**: Includes Link Manager, Link Controller, and BR/EDR Radio.
   * **LE Controller**: Includes Link Manager, Link Controller, and LE Radio.
2. **Host**: Manages higher-level protocols and services.
   * **BR/EDR Host**: Includes L2CAP, SDP, and GAP.
   * **LE Host**: Includes L2CAP, SMP, Attribute Protocol, GAP, and GATT.
   * **Combined Host**: Includes components from both BR/EDR and LE Hosts.

**Communication Interface**

* **Host-Controller Interface (HCI)**: A standard interface that allows the Host and Controller to communicate. This interface is optional but designed to ensure interoperability.

**Functional Blocks**

* **Functional Blocks**: Conceptual entities used to describe system behavior. They help in understanding how different parts of the Bluetooth system interact.

**Protocols and Services**

* **Core Protocols**: Include Radio (PHY), Link Control (LC), Link Manager (LM), Link Layer (LL), and L2CAP.
* **Service Layer Protocols**: Include Service Discovery Protocol (SDP) and Attribute Protocol (ATT).

**Services**

* **Service Access Points**: Points where services are accessed, shown as ellipses in diagrams.
* **Types of Services**:
  + **Device Control Services**: Modify device behavior and modes.
  + **Transport Control Services**: Manage traffic bearers (channels and links).
  + **Data Services**: Handle data transmission.

**Quality of Service (QoS)**

* **QoS Management**: Ensures that data is transmitted reliably and efficiently, managing resources like buffers and handling error detection and retransmission.

**Conformance Testing**

* **Testing**: Ensures that Bluetooth implementations comply with specifications. This involves controlling the device through interfaces like PHY, Direct Test Mode (DTM), and HCI.

**Summary**

* **Controller**: Manages low-level communication.
* **Host**: Manages higher-level protocols and services.
* **HCI**: Interface for Host-Controller communication.
* **Functional Blocks**: Conceptual entities for system behavior.
* **Core Protocols and Services**: Ensure interoperability and reliable communication.
* **QoS Management**: Maintains data transmission quality.
* **Conformance Testing**: Verifies compliance with specifications.

Does this help clarify the Bluetooth core system architecture? Let me know if you need more details on any specific part!

**BR/EDR Architecture:**

**Inquiry process:**

**Master to Slave:**

**1. Generic Access Profile (GAP)**

**Role**: GAP manages the overall process of device discovery, ensuring that the host can find nearby Bluetooth devices.

**Step**:

* **Initiation**: The host (e.g., a smartphone) initiates the inquiry process by sending an inquiry command to the controller.

**2. Device Manager**

**Role**: The Device Manager in the host handles the command and coordinates with the lower layers to start the inquiry process.

**Step**:

* **Command Handling**: The Device Manager receives the inquiry command and prepares to send it to the baseband layer.

**3. Baseband Layer**

**Role**: The baseband layer manages the physical channels and links, preparing the inquiry packets for transmission.

**Steps**:

* **Packet Preparation**: The baseband layer prepares the inquiry packets, which include the Inquiry Access Code (IAC).
* **Error Correction**: Ensures the integrity of the inquiry packets.
* **Data Whitening**: Randomizes the data to reduce patterns that could interfere with transmission.
* **Hop Selection**: Determines the frequency hopping pattern to avoid interference.
* **Physical Channel Selection**: The baseband layer selects the physical channels for the inquiry scan based on the frequency hopping pattern.
  + **Inquiry Scan Channels**: These channels follow a slower hopping pattern and use an access code to distinguish between occasional occupancy of the same radio frequency by two co-located devices using different physical channels.
* **Frequency Hopping**: Bluetooth uses a pseudo-random frequency hopping pattern. For the inquiry process, the baseband layer selects a sequence of 32 frequencies out of the 79 available in the 2.4 GHz ISM band.

**4. Physical Radio Layer**

**Role**: The physical radio layer handles the actual transmission of inquiry packets over the air.

**Steps**:

* **Modulation**: The inquiry packets are modulated using Gaussian Frequency-Shift Keying (GFSK) for BR or Phase-Shift Keying (PSK) for EDR.
* **Transmission**: The modulated inquiry packets are transmitted over the selected physical channels.

**Slave to Master:**

**1. Nearby Device Receives Inquiry Packets**

**Input**: Inquiry packets broadcasted by the host. **Process**:

* The nearby device (e.g., a headset) in discoverable mode receives the inquiry packets.
* The device processes these packets and prepares a response.

**2. Response Packet Preparation**

**Location**: Baseband layer of the nearby device.

**Input**: Inquiry packets received by the nearby device. **Process**:

* **Device Address**: The device includes its Bluetooth device address in the response packet.
* **Class of Device (CoD)**: The device may include its class of device, indicating its type and capabilities.
* **Other Information**: Additional information such as the device name or supported services may be included.

**3. Baseband Layer of the Nearby Device**

**Role**: The baseband layer manages the physical channels and links, preparing the response packets for transmission.

**Steps**:

* **Packet Preparation**: The baseband layer prepares the response packets.
* **Error Correction**: Ensures the integrity of the response packets.
* **Data Whitening**: Randomizes the data to reduce patterns that could interfere with transmission.
* **Hop Selection**: Determines the frequency hopping pattern for the response.

**4. Physical Radio Layer of the Nearby Device**

**Role**: The physical radio layer handles the actual transmission of response packets      over the air.

**Steps**:

* **Modulation**: The response packets are modulated using Gaussian Frequency-Shift Keying (GFSK) for BR or Phase-Shift Keying (PSK) for EDR.
* **Transmission**: The modulated response packets are transmitted over the air using the selected physical channels.

**5. Host Receives Response Packets**

**Input**: Response packets transmitted by the nearby device. **Process**:

* The host's physical radio layer receives the modulated response packets.
* The packets are demodulated and passed to the baseband layer.

**6. Baseband Layer of the Host**

**Role**: The baseband layer processes the received response packets.

**Steps**:

* **Error Correction**: Ensures the integrity of the received packets.
* **Data Whitening**: Reverses the whitening process to retrieve the original data.
* **Packet Processing**: Extracts the device address and other information from the response packets.

**7. Device Manager of the Host**

**Role**: The Device Manager handles the received response packets and updates the list of discovered devices.

**Steps**:

* **Device Enumeration**: Adds the discovered device to the list of nearby devices.
* **Service Management**: May initiate service discovery to find out what services the discovered device supports.
* Service discovery can be performed both during the inquiry stage and after a connection has been established. Performing service discovery after establishing a connection generally provides more detailed information and ensures secure communication. However, initial service discovery during the inquiry stage can help the host make efficient decisions about which devices to connect to.
  1. **Receiving the FHS Packet:**

**If EIR is enabled:**

**Input**: Inquiry packets broadcasted by the host. **Process**:

* The nearby device (e.g., a headset) in discoverable mode receives the inquiry packets.
* The device responds with an FHS packet, which includes basic information such as the device address and clock offset.

**2. Host Receives FHS Packet**

**Input**: FHS packet transmitted by the nearby device. **Process**:

* The host's physical radio layer receives the FHS packet.
* The packet is demodulated and passed to the baseband layer.

**3. Baseband Layer of the Host**

**Role**: The baseband layer processes the received FHS packet.

**Steps**:

* **Error Correction**: Ensures the integrity of the received packet.
* **Data Whitening**: Reverses the whitening process to retrieve the original data.
* **Packet Processing**: Extracts the device address and clock offset from the FHS packet.

**4. Device Manager of the Host**

**Role**: The Device Manager handles the received FHS packet and updates the list of discovered devices.

**Steps**:

* **Device Enumeration**: Adds the discovered device to the list of nearby devices.
* **Service Management**: May initiate service discovery to find out what services the discovered device supports.

**5. Extended Inquiry Response (EIR) Handling**

**Input**: EIR enabled, requiring additional information from the nearby device.

**Process**:

* **EIR Request**: The host sends an EIR request to the nearby device to obtain extended information.
* **EIR Response**: The nearby device responds with an EIR packet containing additional information such as the device name, supported services, and other attributes.

**6. Baseband Layer of the Nearby Device**

**Role**: The baseband layer of the nearby device prepares the EIR packet.

**Steps**:

* **Packet Preparation**: The baseband layer prepares the EIR packet with extended information.
* **Error Correction**: Ensures the integrity of the EIR packet.
* **Data Whitening**: Randomizes the data to reduce patterns that could interfere with transmission.
* **Hop Selection**: Determines the frequency hopping pattern for the response.

**7. Physical Radio Layer of the Nearby Device**

**Role**: The physical radio layer handles the actual transmission of the EIR packet over the air.

**Steps**:

* **Modulation**: The EIR packet is modulated using Gaussian Frequency-Shift Keying (GFSK) for BR or Phase-Shift Keying (PSK) for EDR.
* **Transmission**: The modulated EIR packet is transmitted over the air using the selected physical channels.

**8. Host Receives EIR Packet**

**Input**: EIR packet transmitted by the nearby device.

**Process**:

* The host's physical radio layer receives the modulated EIR packet.
* The packet is demodulated and passed to the baseband layer.

**9. Baseband Layer of the Host**

**Role**: The baseband layer processes the received EIR packet.

**Steps**:

* **Error Correction**: Ensures the integrity of the received packet.
* **Data Whitening**: Reverses the whitening process to retrieve the original data.
* **Packet Processing**: Extracts the extended information from the EIR packet.

**10. Device Manager of the Host**

**Role**: The Device Manager handles the received EIR packet and updates the list of discovered devices with extended information.

**Steps**:

* **Device Enumeration**: Updates the discovered device entry with extended information.
* **Service Management**: May initiate further service discovery based on the extended information.

**Paging process:**

**1. Initiation of Paging**

**Input**: The host (e.g., a smartphone) has discovered a nearby device during the     inquiry process and has the device address.

**Process**:

* The host decides to establish a connection with the discovered device.
* The Device Manager in the host sends a paging command to the controller, including the target device's address.

**2. Baseband Layer of the Host**

**Role**: The baseband layer manages the physical channels and links, preparing the paging packets for transmission.

**Steps**:

* **Packet Preparation**: The baseband layer prepares the paging packets, which include the Device Access Code (DAC) and the target device's address.
* **Error Correction**: Ensures the integrity of the paging packets.
* **Data Whitening**: Randomizes the data to reduce patterns that could interfere with transmission.
* **Hop Selection**: Determines the frequency hopping pattern for the paging process.

**3. Physical Radio Layer of the Host**

**Role**: The physical radio layer handles the actual transmission of paging packets over the air.

**Steps**:

* **Modulation**: The paging packets are modulated using Gaussian Frequency-Shift Keying (GFSK) for BR or Phase-Shift Keying (PSK) for EDR.
* **Transmission**: The modulated paging packets are transmitted over the air using the selected physical channels.

**4. Nearby Device Receives Paging Packets**

**Input**: Paging packets broadcasted by the host.

**Process**:

* The nearby device (e.g., a headset) receives the paging packets.
* The device processes these packets and prepares a response.

**5. Response Packet Preparation by Nearby Device**

**Location**: Baseband layer of the nearby device.

**Input**: Paging packets received by the nearby device.

**Process**:

* **Device Address**: The device includes its Bluetooth device address in the response packet.
* **Clock Offset**: The device includes its clock offset to synchronize communication.
* **Other Information**: Additional information such as supported services may be included.

**6. Baseband Layer of the Nearby Device**

**Role**: The baseband layer manages the physical channels and links, preparing the response packets for transmission.

**Steps**:

* **Packet Preparation**: The baseband layer prepares the response packets.
* **Error Correction**: Ensures the integrity of the response packets.
* **Data Whitening**: Randomizes the data to reduce patterns that could interfere with transmission.
* **Hop Selection**: Determines the frequency hopping pattern for the response.

**7. Physical Radio Layer of the Nearby Device**

**Role**: The physical radio layer handles the actual transmission of response packets over the air.

**Steps**:

* **Modulation**: The response packets are modulated using Gaussian Frequency-Shift Keying (GFSK) for BR or Phase-Shift Keying (PSK) for EDR.
* **Transmission**: The modulated response packets are transmitted over the air using the selected physical channels.

**8. Host Receives Response Packets**

**Input**: Response packets transmitted by the nearby device.

**Process**:

* The host's physical radio layer receives the modulated response packets.
* The packets are demodulated and passed to the baseband layer.

**9. Baseband Layer of the Host**

**Role**: The baseband layer processes the received response packets.

**Steps**:

* **Error Correction**: Ensures the integrity of the received packets.
* **Data Whitening**: Reverses the whitening process to retrieve the original data.
* **Packet Processing**: Extracts the device address, clock offset, and other information from the response packets.

**10. Link Manager Protocol (LMP)**

**Role**: The Link Manager Protocol (LMP) handles the setup of the physical link, including authentication and encryption.

**Steps**:

* **Link Setup**: LMP negotiates link parameters such as authentication and encryption.
* **Synchronization**: LMP synchronizes the devices' clocks and frequency hopping patterns.

**11. Connection Established**

**Output**: A connection is established between the host and the nearby device.

**Link Manager protocol (LMP):**

After the Link Manager Protocol (LMP) has successfully established a connection, the process continues with several steps to ensure the connection is configured, authenticated, and ready for data transmission. Here's a detailed step-by-step explanation:

**1. Link Configuration**

**Role**: The Link Manager Protocol (LMP) configures the link parameters.

**Steps**:

* **Slot Offset Request**: The host requests the slot offset to synchronize communication slots.
* **Clock Offset Request**: The host requests the clock offset to synchronize device clocks.
* **Timing Accuracy Information Request**: Ensures accurate timing for communication.

**2. Authentication**

**Role**: LMP handles the authentication process to ensure secure communication.

**Steps**:

* **Challenge-Response**: The verifier (host) sends an LMP\_au\_rand PDU containing a random number (challenge) to the claimant (nearby device).
* **Response Calculation**: The claimant calculates a response using the challenge, its BD\_ADDR, and a secret key.
* **Response Verification**: The verifier checks if the response is correct.
* **Mutual Authentication**: Both devices authenticate each other to ensure secure communication.

**3. Encryption**

**Role**: LMP sets up encryption to protect data transmitted over the link.

**Steps**:

* **Change Link Key**: The host and nearby device agree on a new link key.
* **Encryption Mode Request**: The host requests to enable encryption.
* **Encryption Key Exchange**: The devices exchange encryption keys.
* **Encryption Activation**: Encryption is activated for the link.

**4. Role Switch (Optional)**

**Role**: LMP can switch the roles of the devices (master/slave) if needed.

**Steps**:

* **Role Switch Request**: The host requests a role switch.
* **Role Switch Confirmation**: The nearby device confirms the role switch.
* **Role Switch Execution**: The roles are switched, and the devices continue communication with the new roles.

**5. Quality of Service (QoS) Configuration**

**Role**: LMP configures the Quality-of-Service parameters to ensure optimal communication.

**Steps**:

* **QoS Request**: The host requests specific QoS parameters.
* **QoS Negotiation**: The devices negotiate the QoS parameters.
* **QoS Confirmation**: The QoS parameters are confirmed and applied.

**6. SCO Link Setup (Optional)**

**Role**: LMP sets up Synchronous Connection-Oriented (SCO) links for audio data transmission.

**Steps**:

* **SCO Link Request**: The host requests to set up an SCO link.
* **SCO Link Confirmation**: The nearby device confirms the SCO link setup.
* **SCO Link Activation**: The SCO link is activated for audio data transmission.

**7. Data Transmission Setup**

**Role**: The Logical Link Control and Adaptation Protocol (L2CAP) sets up logical channels for data transmission.

**Steps**:

* **Channel Setup**: L2CAP sets up logical channels for data transfer.
* **Data Packetization**: Data is divided into packets suitable for transmission.
* **Transmission**:
  + **SCO Links**: Used for transmitting audio data.
  + **ACL Links**: Used for transmitting control signals and other data.

**8. Data Transmission**

**Role**: The Baseband layer and Physical Radio layer handle the actual transmission of data packets.

**Steps**:

* **Baseband Processing**: The Baseband layer handles error correction, data whitening, and framing.
* **Modulation**: The data is modulated using Gaussian Frequency-Shift Keying (GFSK) for BR or Phase-Shift Keying (PSK) for EDR.
* **Transmission**: The modulated data packets are transmitted over the physical channel.

**Logical Link Control and Adaptation Protocol (L2CAP):**

**L2CAP** is a protocol in the Bluetooth host stack that:

* Provides **logical channels** over the physical ACL (Asynchronous Connection-Less) link.
* Supports **multiplexing**, **segmentation/reassembly**, and **QoS**.
* Acts as a bridge between higher-layer protocols (like SDP, RFCOMM, ATT/GATT) and the lower HCI/controller layers.

**Step-by-Step: L2CAP Operation After Connection**

**1. HCI Notifies Host of Connection**

* **Event**: HCI\_Connection\_Complete
* **Effect**: Host stack activates L2CAP for the new connection.

**2. L2CAP Channel Manager Starts**

**Role**: Manages logical channels over the ACL link.

**Responsibilities**:

* Assigns **Channel Identifiers (CIDs)**.
* Handles **channel creation, configuration, and teardown**.
* Routes data to the correct higher-layer protocol (e.g., SDP, ATT).

**Steps**:

1. Receives connection info from HCI.
2. Creates a **signalling channel** (CID = 0x0001) for control messages.
3. Sends **L2CAP Connection Request** to the peer device.
4. Waits for **L2CAP Connection Response**.

**3. L2CAP Resource Manager Engages**

**Role**: Manages **buffering, segmentation, reassembly**, and **flow control**.

**Responsibilities**:

* Allocates memory and buffers for each channel.
* Segments large SDU (Service Data Units) into MTU-sized PDUs.
* Reassembles incoming PDUs into complete SDUs.
* Ensures flow control and QoS compliance.

**Steps**:

1. Checks if the incoming/outgoing data fits within the **MTU (Maximum Transmission Unit)**.
2. If not, segments the data and queues it for transmission.
3. Monitors buffer usage and applies flow control if needed.

**4. Channel Configuration**

    Once the peer accepts the connection:

* **L2CAP Configuration Request** is sent (e.g., MTU size, flush timeout).
* **L2CAP Configuration Response** is received.
* Channel is now **open and ready** for data transfer.

**5. Data Transfer Begins**

          Higher-layer protocols (e.g., ATT for BLE, RFCOMM for serial) send data to L2CAP.

* L2CAP:
  + Assigns the correct CID.
  + Segments the data (if needed).
  + Sends it to the controller via HCI.

**Summary of Key Components:**

| **Component** | **Role** | **Key Functions** |
| --- | --- | --- |
| **L2CAP Channel**  **Manager** | Logical channel control | Channel setup, teardown, CID assignment |
| **L2CAP Resource**  **Manager** | Data handling | Segmentation, reassembly, buffering, flow  control |
| **HCI** | Transport | Sends L2CAP packets to/from controller |
| **Controller** | Physical link | Transmits data over ACL link |

**What Is CID and PSM in L2CAP?**

**CID (Channel Identifier):**

* A **CID** uniquely identifies a **logical channel** between two Bluetooth devices.
* It is used in **every L2CAP packet** to indicate **which channel** the packet belongs to.
* **CID = 0x0001** is **reserved** for the **Signalling Channel**.
* Other CIDs (e.g., 0x0040, 0x0041, etc.) are dynamically assigned for **application-specific channels** like RFCOMM, AVDTP, etc.

**PSM (Protocol/Service Multiplexer)**

* **PSM** identifies the **higher-layer protocol** that is using L2CAP.
* It is used **only during the connection request** to indicate what protocol the new channel is for.
* Examples:
  + 0x0001 → L2CAP signalling
  + 0x0003 → RFCOMM
  + 0x0019 → AVDTP (used in A2DP)

**How They Are Used in the signalling Channel**

**🔹 When the signalling Channel Is Created:**

* **CID = 0x0001** is automatically used by both devices.
* **PSM = 0x0001** is implied (not explicitly sent in every packet) because this channel is reserved for signalling.

**🔹 In Each L2CAP Packet:**

* The **CID field** in the L2CAP header is set to 0x0001 to indicate the packet is for the signalling channel.
* The **PSM is not included** in every signalling packet — it’s only used in **connection requests** to indicate which protocol a new dynamic channel is being requested for.

**Example: L2CAP Connection Request Packet**

When the host wants to open a new channel for RFCOMM:

* It sends an **L2CAP Connection Request** over the **signalling channel (CID = 0x0001)**.
* The payload includes:
  + **PSM = 0x0003** (for RFCOMM)
  + **Source CID** (e.g., 0x0040) — the CID the host will use for this new channel.

The peer responds with:

* **Destination CID** (e.g., 0x0041) — the CID it will use for the same channel.

**What Happens After L2CAP Channel Setup?**

              Once the signalling channel has successfully negotiated and opened a dynamic channel (e.g., for RFCOMM, AVDTP, ATT), the system transitions into **active data communication**. Here's how the process unfolds:

**Step 1: Higher-Layer Protocol Activation**

* Based on the **PSM** used during the connection request, the corresponding **higher-layer protocol** is activated:
  + **PSM = 0x0003** → RFCOMM (Serial Port Emulation)
  + **PSM = 0x0019** → AVDTP (Audio/Video Streaming)
  + **PSM = 0x001F** → ATT (Attribute Protocol for BLE)

             These protocols now use the **assigned CID** to send and receive data.

**Step 2: Data Transmission Begins**

* Each data packet sent by the host includes:
  + **CID**: Identifies the logical channel (e.g., RFCOMM channel).
  + **Payload**: Contains the protocol-specific data (e.g., OBEX file data, audio frames).
* The **L2CAP Resource Manager**:
  + Segments large payloads into MTU-sized chunks.
  + Buffers and queues them for transmission.
  + Reassembles incoming chunks into complete messages.

**Step 3: HCI and Controller Transmission**

* L2CAP packets are passed to the **HCI layer**, which wraps them into **ACL data packets**.
* These are sent to the **controller**, which:
  + Uses the **baseband** to frame and schedule transmission.
  + Uses the **physical radio** to transmit over the air.

**Step 4: Receiver Processing**

* The **slave device** receives the ACL packets.
* Its controller:
  + Demodulates and error-checks the packets.
  + Passes them to its L2CAP layer.
* L2CAP:
  + Uses the **CID** to route the packet to the correct protocol handler.
  + Reassembles and delivers the data to the appropriate application.

**TRANSPORT LAYERS:**

Communication happening between master and slaves in a piconet can be 'connection-oriented' or 'connection-less’.

The communication can be called as connection-oriented only if a dedicated session is created at both the ends for communication. Dedicated session is created only when negotiation happens between the master and the salve for its (session's) creation and the set of parameters that identify the dedicated session belongs to that of both the master and the particular slave.

The communication can be called as connection-less when there is no creation of dedicated session between master and the particular slave (as parameters that identify the session for transaction will only be that of masters own or slave's own and not of both).

Hence ACL, SCO, e-SCO are connection oriented logical transports where dedicated session for transaction is created at both master and slave while ASB and CSB are connection-less logical transports having session for transaction but the parameters do not uniquely identify the session and hence never is 'dedicated'.

**CREATION OF SESSION FOR DEFAULT ACL LOGICAL TRANSPORT:**

During Connection establishment procedure, when master sends FHS packet contain LT\_ADDR as one of its field, an ACL logical transport session is created at both the aspiring master and paged device (aspiring slave). ACL session at both the ends is identified by Frequency Hopping Sequence (FHS), clock of the master, channel access code, primary LT\_ADDR and packet type.

The same instance when default ACL logical transport is created, ACL-C logical link as a sub session within the ACL logical transport session exist. Through this ACL-C logical link sub-session LM talks with its counterpart in the other Bluetooth device to negotiate for creation of ACL-U logical link sub-session. When both LMs exchange LMP\_setup\_complete PDU, session for ACL-U logical link is established. LLID field identifies the sub-session of ACL-U or ACL-C in default ACL session.

**Creation of session for SCO, e-SCO logical transports:**

On demand from the application, link manager of either the master or the slave device sends LMP\_SCO\_link\_req. On reception of LMP\_accepted by LM of the other device, session for transaction for SCO logical transport is created at both end devices, which shares the LT\_ADDR of default ACL logical transport and is uniquely identified by SCO handle, Tsco, Dsco fields and packet type.

The same is the case with creation of e-SCO logical transport session, where either the master or the slave device sends LMP\_eSCO\_link\_req and on reception of LMP-accepted by LM of the other device, session for transaction for e-SCO logical transport is created at both end devices. E-SCO logical transport is uniquely identified with eSCO handle, eSCO LT\_ADDR, De-sco, Te-sco, We-sco and packet type {M to S and S to M}.

**Creation of session for ASB logical transport:**

After the procedure of connection-establishment ( i.e. creation of session for ACL logical transport with both ACL-C and ACL-U sub-sessions) a session, by default is created for ASB logical transport at master and slave side having LT\_ADDR 0. Since creation of ASB logical transport session happens without LMP signalling, it cannot be considered as dedicated session (no negotiation). Any slave active in the piconet can receive the packet sent with LT\_ADDR as 0, since a session for ASB is present in all Bluetooth devices participating in the piconet.

ASB AND APB ARE SAME.

**Creation of session for CSB logical transport:**

Session for CSB logical transport is created by the master’s (sender’s) LM on demand from the host and session for CSB logical transport is created by the slave (receiver) whenever configured for reception of Profile Broadcast Data. Session for CSB logical transport is uniquely identified with CSB LT\_ADDR assigned by master's LM, but there is no LMP PDU to negotiate between master and the slave in the piconet for the creation of CSB logical transport at the slave end and hence there is a different mechanism involved.

Synchronization scan physical channel is used for this purpose. Master shall enter into synchronization train sub-state to transmit synchronization train packets, (uses a subset of RF channels). Slave shall enter synchronization scan sub-state and wait in a single frequency to receive synchronization train packets. This operation happens only on synchronization scan physical channel.

Synchronization scan physical channel is used for this purpose. Master shall enter into synchronization train sub-state to transmit synchronization train packets, (uses a subset of RF channels). Slave shall enter synchronization scan sub-state and wait in a single frequency to receive synchronization train packets. This operation happens only on synchronization scan physical channel.

The contents of synchronization train packet broadcasted by master is CSB LT\_ADDR, CSB interval, CLK of master, AFH channel map, and master BD\_ADDR. The packet type of this synchronization train packet is basic rate ACL DM3 packet with LT\_ADDR as 0, as it is to be received by any slave willing to establish CSB logical transport. Access code is derived from BD\_ADDR of master and the packet is transmitted on synchronization scan physical channel.

Session for CSB logical transport is created by master when host demands controller to send Profile Broadcast Data and hence CSB logical transport is created by Master's LM and enters into synchronization train sub-state. Slave has to be configured to receive Profile Broadcast Data and may periodically enter into synchronization scan sub-state to receive train packets to create session for CSB. On reception of train packet, slave creates a session for CSB and goes back to its previous state. The session created in slaves will all bare the same CSB LT\_ADDR and is not dedicated for a master – particular slave transaction.

CSB can exist in a slave even without necessarily having default ACL logical transport and hence it's an advantage to have theoretically limit-less slaves waiting for profile broadcast data which will be sent by master periodically. The period (CSB interval) is communicated in synchronization train packet so that configured CSB slaves can wakeup only during those intervals to listen for the packets from the master.

**TRAFFIC BEARERS:**

* Bluetooth provides the standard traffic bearers optimized for different data types.
* Applications use these bearers via logical links and logical transports.
* **Logical links** are the virtual links established over the physical links. They are used to manage and transmit data between Bluetooth devices.
* **Logical Transport** refers to the methods of transferring data over a physical links. It defines how logical links are mapped onto the physical links.

**A traffic bearer is a defined path that moves data across Bluetooth Layers. It provides:**

* Correct formatting for data.
* Suitable timing or delay constraints.
* Appropriate level of reliability.

**NAMING CONVENTION OF LOGICAL LINKS:**

Logical links are named based on:

* The transport Type: ACL, SCO, APB.
* A Suffix indicating data type:

C- Control messages (Eg. LMP)

U- User data (via L2CAP)

S- Stream (raw audio or unformatted data).

B- Broadcast

For e.g.

ACL-C: ACL transport used for control (LMP Messages)

SCO-S: SCO transport used for streaming audio.

**MAPPING APPLICATIONS TRAFFIC TO BEARERS:**

Each application traffic type is matched to a bearer based on:

* Required reliability
* Latency sensitivity
* Framing needs
* Constants vs variable rate

**FLEXIBILITY IS MAPPING:**

It is allowed to use alternative bearers as long as the application data characteristics as preserved.

FOR EG:

🡪In a BR/EDR piconet, with one-to-one peripheral the central could send L2CAP broadcasts over ACL-U instead of APB-U.

🡪**WHY?** Because in point-to-point setup, broadcasting is not needed and using ACL-U saves bandwidth.

**WHAT CHARACTERISTICS ARE MATTER FOR MAPPING:**

-The Bluetooth core doesn’t care about what the original data was, it only cares about.

* Is it reliable or not?
* Is it framed or unframed?
* Does it need low latency or constant rate?
* Is it for one peer or for broadcast?

**FRAMED DATA TRAFFIC:**

Framed data traffic means the data in:

* Organizes into discrete chunks or frames.
* Each frame has clear boundaries
* Frames are delivered as into the remote devices.

Types of L2CAP channels:

* 1. CONNECTION ORIENTED L2CAP CHANNELS (ACL, SCO):
* Used for unicast traffic (point to point)
* Created through as L2CAP connection procedure.
* Offer context and control, such as:

🡪Quality of service settings

🡪Flow control

🡪Error control

* 1. CONNECTIONLESS L2CAP CHANNELS (APB):
* Used to broadcast or simple unicast traffic.
* No need to open a connection, hence:

🡪Lower Latency

🡪Simpler setup

**USE OF QOS IN L2CAP:**

L2CAP channels in BR/EDR can have QOS settings that define:

* + - Whether the data is isochronous (has a lifetime, useful for live audio)
    - A time constraint (should arrive X ms)
    - Reliability level (retry until correct)

Eg.

For voice: Deliver within 20ms for discard

For file: Deliver without error, even if it takes longer.

**FIXED L2CAP CHANNELS:**

These are predefined and created automatically when logically links are established.

**USED FOR: LOGICAL LINKS USED:**

Signalling ACL-U

Connectionless data ACL-U, APB-U

ATT ACL-U

**Role of L2CAP channel manger:**

* Managing all L2CAP channels
* Deciding which logical links to use
* Possibly multiplexing multiple channels over a single logical link.

**WHAT IS UNFRAMED DATA TRAFFIC?**

Unframed data= Continuous stream of data without explicit “frame” boundaries handled by Bluetooth.

Often used for real time streams like voice, where data must be sent at a fixed rate and timing, not waiting for a full frame.

In Bluetooth, this kind of data bypass L2CAP & is sent directly over baseband logical links.

**WHAT APPLIES TO BR/EDR?**

* + 1. SCO-S (Synchronous Connection-oriented Stream)
* Use for constant-rate unframed voice streams.
* Very low-latency (fixed intervals)
* No retransmissions / not very reliable, but fast.
* Used in HSP (Head set Profile) & HFP (Hands-free profile)
* Only supports Basic Rate (no EDR)
  + 1. eSCO-S (extended SCO- Stream)
* An enhanced over SCO:
  + - More bitrate
    - Some error correction/ retransmissions
    - Supports EDR
* Still used for voice applications e.g. High- quality calls.
* Also uses constraints interval scheduling locked to piconet clock.
  1. Profile Broadcast Data (PBD) logical link
* Similar to SCO/eSCO:

🡪Constant-rate, isochronous

🡪 fixed intervals, no multiplexing.

* But broadcasts to multiple receivers instead of point to point.
* Used in br/edr for broadcasting.
* Still br/edr specific, not part of Bluetooth BLE.
* For unframed traffic, L2CAP is skipped.
* Applications send the stream directly via the logical link (SCO-S)

**BR/EDR RELIABILITY:**

Links are reliable and which are not, based on ARQ [Automatic Repeat request] concept.

**HOW BLETOOTH BR/EDR ENSURES RELIABILITY (LAYER BY LAYER):**

1. Baseband layer

This is the layer closest to the physical radio hardware.

* Uses FEC (Forward Error Correction) to fix some errors in the packet before its even checked.
* Uses HEC (Header Error Check) and CRC (Cyclic Redundancy Check) to detect errors that FEC couldn’t fix.
* Some packets (like DM1, DM3, DM5) use FEC for both header and payload for extra protection.

1. ARQ Protocol on ACL Links

* ACL (Asynchronous Connection Oriented): used for most data.
* Uses ARQ (Automatic Repeat request): If an error is detected 🡪receiver rejects it 🡪 transmitter resends it.
* Makes ACL links reliable.
  + - If the data is time-sensitive, it may skip retransmission if it’s too late to be useful.

1. Modified ARQ on eSCO Links

* eSCO is used for voice.
* Supports limited retransmissions
* Helps voice sound better but still drops old packets to keep it real-time.

1. L2CAP Layer

* Adds extra error control:

🡪if a packet is missed or not delivered properly by baseband, L2CAP detects it.

🡪It can ask for retransmission again to fix any missing data.

🡪Especially useful when baseband fails to detect a rare error.

* + - This makes the system reliable enough for apps like file transfers, OBEX, messaging, etc…

  2. Stream Links (e.g. SCO/eSCO)
* These don’t guarantee full reliability.
* Sometimes reliable sometimes not depends on:
  + - Signal quality
    - Packet size
    - Whether retransmissions are used (only eSCO supports some)

Voice data is time-sensitive, so it often accepts a little loss for lower delay.

**What is the QoS Option in Bluetooth BR/EDR?**

The **QoS (Quality of Service) option** in Bluetooth BR/EDR allows devices to **negotiate how data should be sent**—whether it should be fast, reliable, bursty, or just best-effort. This is especially important when multiple devices are communicating and resources like bandwidth and time slots are limited.

**QoS Option Format (Option Type = 0x03)**

This option is used in **L2CAP configuration requests/responses** to describe how traffic should flow between devices. It includes several fields:

**Flags (1 byte)**

* Reserved for future use. You can ignore this for now.

**2. Service Type (1 byte)**

This tells the device what kind of service is expected:

* 0x00 – **No traffic**: No data will be sent.
* 0x01 – **Best Effort** (default): Try your best, but no guarantees.
* 0x02 – **Guaranteed**: Must meet the specified QoS parameters.

If you don’t specify anything, **Best Effort** is assumed.

**3. Token Rate (4 bytes)**

* Think of this as the **average speed** at which the app wants to send data (in bytes per second).
* If set to 0x00000000, it means “I don’t care.”
* If set to 0xFFFFFFFF, it means “Give me as much bandwidth as possible.”

🧠 **Analogy**: Imagine a faucet dripping water steadily—that’s your token rate.

**4. Token Bucket Size (4 bytes)**

* This defines how much **data can be sent in a burst**.
* If the app wants to send a lot of data quickly (like a video frame), it needs a bigger bucket.
* 0x00000000 = no burst allowed (default).
* 0xFFFFFFFF = as big a burst as possible.

🧠 **Analogy**: The bucket stores water (data). You can pour it all at once if needed.

**5. Peak Bandwidth (4 bytes)**

* This is the **maximum speed** at which data can be sent back-to-back.
* Helps the system allocate resources efficiently.

🧠 **Analogy**: This is like the **maximum width of the pipe**—how fast you can pour the bucket.

**6. Latency (4 bytes)**

* The **maximum delay** allowed for sending data (in microseconds).
* Important for real-time applications like voice.

**7. Delay Variation (4 bytes)**

* Also known as **jitter**—how much the delay can vary.
* Lower variation is better for smooth audio/video.

**In Bluetooth terms:**

1. **Device A** sends an L2CAP\_CONFIGURATION\_REQ packet with its preferred QoS settings.
2. **Device B** responds with an L2CAP\_CONFIGURATION\_RSP:
   * If it agrees, it confirms the settings.
   * If it disagrees, it suggests alternative values.
3. This back-and-forth continues until both devices **agree on a common set of parameters**.

**Why is this important?**

* It ensures **compatibility** between devices.
* It helps **optimize performance** based on what each device can handle.
* It allows for **flexibility**—devices can request more or less strict QoS depending on the application (e.g., voice vs file transfer).

**PHYSICAL LINK: 🡪500**

A physical link represents a Baseband connection between devices. A physical link is always associated with exactly one physical channel. Physical links have common properties that apply to all logical transports on the physical link.

Other than the Connectionless Peripheral Broadcast physical link, common properties of physical links are:

* Power control (see [Vol 2] Part C, Section 4.1.3)

• Link supervision (see Section 3.1 and [Vol 2] Part C, Section 4.1.6)

• Encryption (see [Vol 2] Part H, Section 4 and [Vol 2] Part C, Section 4.2.5)

• Channel quality-driven data rate change (see [Vol 2] Part C, Section 4.1.7)

• Multi-slot packet control (see [Vol 2] Part C, Section 4.1.10)

and for physical links on the adapted piconet physical channel:

AFH channel map (see Section 2.3 and [Vol 2] Part C, Section 4.1.4)

**POWER CONTROL:**

If the received signal characteristics differ too much from the preferred value of a Bluetooth device, it may request an increase or a decrease of the other device’s transmit power level. A device's transmit power level is a property of the physical link, and affects all logical transports carried over the physical link. Power control requests carried over the default ACL-C logical link shall only affect the physical link associated with the requesting device’s default ACL-C logical link; they shall not affect the power level used on the physical links to other Peripherals.

Two power control mechanisms are specified: legacy power control (see Table 4.3) and enhanced power control (see Table 4.4 and Section 4.1.3.1).

|  |  |  |
| --- | --- | --- |
| M/O | PDU | Contents |
| O(9) | LMP\_INCR\_POWER\_REQ | Reserved |
| O(9) | LMP\_DECR\_POWER\_REQ | Reserved |
| O(18) | LMP\_MAX\_POWER | none |
| O(18) | LMP\_MIN\_POWER | none |

* + - Legacy power control PDU

|  |  |  |
| --- | --- | --- |
| M/O | PDU | Contents |
| O(58) | LMP\_POWER\_CONTROL\_REQ | Power\_Adj\_req |
| O(58) | LMP\_POWER\_CONTROL\_RES | Power\_Adj\_Rsp |

* + - Enhanced power control PDU
* The power adjustment requests may be made at any time using the legacy power control mechanism following a successful Baseband Paging procedure and before the link manager supported features responses have been processed.
* After the Link Manager supported features, responses have been processed, if both devices support enhanced power control (see Section 4.1.3.1) then only enhanced power control shall be used.
* Otherwise, if either device supports only the legacy power control mechanism, then only legacy power control shall be used.
* If a device does not support power control requests this is indicated in the supported features list and thus no power control requests shall be sent after the supported features response has been processed.
* Prior to this time, a power control adjustment might be sent, and if the recipient does not support power control, it shall send LMP\_MAX\_POWER in response to LMP\_INCR\_POWER\_REQ and LMP\_MIN\_POWER in response to LMP\_DECR\_POWER\_REQ or shall send LMP\_NOT\_ACCEPTED with the Error\_Code Unsupported LMP Feature (0x1A) in response to either PDU.
* Upon receipt of an LMP\_INCR\_POWER\_REQ PDU or LMP\_DECR\_POWER\_REQ PDU the output power shall be increased or decreased one step. See [Vol 2] Part A, Section 5.2 for the definition of the step size.

If the receiver of LMP\_INCR\_POWER\_REQ is at maximum power LMP\_MAX\_POWER shall be returned. The device shall only request an increase again after having requested a decrease at least once. If the receiver of LMP\_DECR\_POWER\_REQ is at minimum power then LMP\_MIN\_POWER shall be returned and the device shall only request a decrease after having requested an increase at least once.

**Initiator**

**LMP**

**LMP**

**Initiator**

**LMP**

**LMP**

**ENHANCED POWER CONTROL:**

Enhanced power control shall only be used when both devices support the enhanced power control LMP feature. Legacy power control shall not be used when both devices support the enhanced power control LMP feature.

**SENDING ENHANCED POWER CONTROL REQUESTS**

* To adjust the remote device's output power, a device shall send the LMP\_POWER\_CONTROL\_REQ PDU with the Power\_Adj\_Req parameter.
* The Power\_Adj\_Req parameter may be set to: one step up, one step down, or all the way to the max power level. The remote device shall respond with an LMP\_POWER\_CONTROL\_RES PDU. The responder shall transmit the LMP\_POWER\_CONTROL\_RES PDU at the new power level. Upon reception of the LMP\_POWER\_CONTROL\_RES PDU, the initiating device should restart any processes used to determine whether additional power level changes are required.
* If the receiver of the LMP\_POWER\_CONTROL\_REQ PDU has indicated that the output power levels for all of the supported modulation modes are at maximum the requesting device shall only request an increase again after having requested a decrease at least once. If the receiver of the LMP\_POWER\_CONTROL\_REQ PDU has indicated that the output power levels for all of the supported modulation modes are at minimum the requesting device shall only request a decrease after having requested an increase at least once.
* A new LMP\_POWER\_CONTROL\_REQ PDU shall not be sent until an LMP\_POWER\_CONTROL\_RES PDU has been received.
  1. **RESPONDING TO ENHANCED POWER CONTROL REQUESTS**
* When a device receives an LMP\_POWER\_CONTROL\_REQ PDU with the Power\_Adj\_Req parameter set to "increment one step," all supported modulations that are not at the maximum level shall be increased one step.
* Implementations shall not violate the relative power level between modulations (see [Vol 2] Part A, Section 5.2).
* Note: See [Vol 2] Part A, Section 5.2 for requirements on the relative power levels of different modulation modes.
* The responding LM shall send the LMP\_POWER\_CONTROL\_RES PDU to indicate the status for every modulation. The Power\_Adj\_Rsp parameter has three 2-bit fields indicating the status for each modulation mode: GFSK, π/4-DQPSK and 8DPSK. Each 2-bit field shall be set to one of the following values: not supported, changed one step, max power, or min power. The changed one step value shall only be used when the power level for that modulation mode has not reached the minimum or maximum level.
* The not supported value shall be used for each unsupported modulation type.
* The responder shall transmit the LMP\_POWER\_CONTROL\_RES PDU at the new transmit power level and shall not change its power level until requested by the remote device by a subsequent LMP\_POWER\_CONTROL\_REQ PDU.

**Initiator**

**LMP**

**LMP**

**Initiator**

**LMP**

**LMP**

1. **LINK SUPERVISION:**
   1. **LINK SUPERVISION FOR ACTIVE PHYSICAL LINKS:**

* A connection can break down due to various reasons such as a device moving out of range, encountering severe interference or a power failure condition. Since this can happen without any prior warning, it is important to monitor the link on both the Central and the Peripheral side to avoid possible collisions when the logical transport address (see Section 4.2) is reassigned to another Peripheral.
* To be able to detect link loss, both the Central and the Peripheral shall use a link supervision timer, T supervision. Upon reception of a valid packet header with one of the Peripheral's addresses (see Section 4.2) on the physical link, the timer shall be reset. If at any time in Connection state, the timer reaches the supervisionTO value, the connection shall be considered disconnected. The same link supervision timer shall be used for SCO, eSCO, and ACL logical transports.
* The timeout period, supervisionTO, is negotiated by the Link Manager. Its value shall be chosen so that the supervision timeout will be longer than hold and sniff periods.

**2.2. LINK SUPERVISION FOR CONNECTIONLESS PERIPHERAL BROADCAST PHYSICAL LINKS**

For Connectionless Peripheral Broadcast physical links, only the Receiver side monitors the link. To detect link loss, the Receiver shall use a link supervision timer, TCPB\_Supervision. Each Connectionless Peripheral Broadcast Receiver shall reset the timer upon reception of a Connectionless Peripheral Broadcast packet with a valid packet header. If at any time in Connectionless Peripheral Broadcast mode of the Connection state, the timer reaches the CPB\_supervisionTO value, the connection shall be considered disconnected.

For each Receiver, the timeout period, CPB\_supervisionTO, can be provided by the Host (see Section B.1.7).

* + - The Link\_Supervision\_Timeout parameter is used by the BR/EDR Controller to monitor link loss. If, for any reason, no packets are received from that Connection\_Handle for a duration longer than the Link\_Supervision\_Timeout, the connection shall be disconnected. The same timeout value is used for both synchronous and ACL connections for the device specified by the Connection\_Handle.
    - **Note: Setting the Link\_Supervision\_Timeout to No link supervision timeout (0x0000) will disable the Link\_Supervision\_Timeout check for the specified Connection\_Handle.**

**Link\_Supervision\_Timeout:**

**Size: 2 octets**

|  |  |
| --- | --- |
| **Value** | **Parameter Description** |
| 0x0000 | No link supervision timeout. |
| N = 0xXXXX | Range: 0x0001 to 0xFFFF  Default: 0x7D00  Mandatory Range: 0x0190 to 0xFFFF Time = N × 0.625 ms  Time Range: 0.625 ms to 40.9 s  Time Default: 20 s |

1. **CHANNEL QUALITY DRIVEN DATA RATE CHANGE:**

* The data throughput for a given packet type depends on the quality of the RF channel. Quality measurements in the receiver of one device can be used to dynamically control the packet type transmitted from the remote device for optimization of the data throughput.
* Device A sends the LMP\_AUTO\_RATE PDU once to notify device B to enable this feature. Once enabled, device B may request packet type(s) that A should transmit by sending the LMP\_PREFERRED\_RATE PDU.
* This PDU has a parameter which determines the preferred error coding (with or without 2/3 FEC) and optionally the preferred size in slots of the packets. Device A is not required to change to the packet type specified by this parameter.
* Device A shall not send a packet that is larger than max slots (see Section 4.1.10) even if the preferred size is greater than this value.
* The Data\_Rate parameter includes the preferred rate for Basic Rate and Enhanced Data Rate modes.
* When operating in Basic Rate mode, the device shall use bits 0 to 2 to determine the preferred data rate.
* When operating in Enhanced Data Rate mode, the device shall use bits 3 to 6 to determine the preferred data rate.
* For devices that support Enhanced Data Rate, the preferred rates for both Basic Rate and Enhanced Data Rate modes shall be valid at all times.
* These PDUs may be sent at any time after connection setup is completed.

|  |  |  |
| --- | --- | --- |
| M/O | PDU | Contents |
| O(10) | LMP\_AUTO\_RATE | None |
| O(58) | LMP\_PREFERRED\_RATE | Data\_rate |

**Initiator**

**LMP**

**LMP**

**Initiator**

**LMP**

**LMP**

1. **CONTROL OF MULTI-SLOT PACKETS:**

The number of consecutive slots used by a device on an ACL-U logical link can be limited. It does not affect traffic on the eSCO links where the packet sizes are defined as part of link setup. A device allows the remote device to use a maximum number of slots by sending the PDU LMP\_MAX\_SLOT providing a Max\_Slots parameter. Each device can request to use a maximal number of slots by sending the PDU LMP\_MAX\_SLOT\_REQ providing a Max\_Slots parameter. After a new connection (as a result of page or page scan), or after a role switch, the value shall be 1 slot. These PDUs can be sent at any time after connection setup is completed.

|  |  |  |
| --- | --- | --- |
| M/O | PDU | Contents |
| M | LMP\_MAX\_SLOT | Max\_Slots |
| M | LMP\_MAX\_SLOT\_REQ | Max\_Slots |

1. **ENCRYPTION:**

* Two encryption mechanisms are defined: E0 encryption (legacy) and AES-CCM encryption.
* If both devices support the Secure Connections (Controller Support) and Secure Connections (Host Support) features, then AES-CCM shall be used when encryption is enabled.
* If at least one device does not support both the Secure Connections (Controller Support) and Secure Connections (Host Support) features, then E0 shall be used when encryption is enabled.
* In order for the Central to use the same encryption parameters for all Peripherals in the piconet where E0 encryption would be used it shall issue a temporary key, K\_temp.
* The Central shall make this key the current link key for all Peripherals in the piconet where E0 encryption would be used before encryption is started, see Section 4.2.4. This is required if broadcast packets are to be encrypted.

Note: Packets encrypted with broadcast encryption can not be received by Peripherals that have AES-CCM encryption enabled. When the local Controller supports Secure Connections and there are not any Peripherals in the piconet that do not support Secure Connections, broadcast packets will not be encrypted and may be received by Peripherals that support Secure Connections.

**LOGICAL TRANSPORTS:**

Between Central and Peripheral(s), different types of logical transports may be established.

Five logical transports have been defined:

• Synchronous Connection-Oriented (SCO) logical transport

• Extended Synchronous Connection-Oriented (eSCO) logical transport

• Asynchronous Connection-Oriented (ACL) logical transport

• Active Peripheral Broadcast (APB) logical transport

• Connectionless Peripheral Broadcast (CPB) logical transport.

The synchronous logical transports are point-to-point logical transports between a Central and a single Peripheral in the piconet. The synchronous logical transports typically support time-bounded information like voice or general synchronous data. The Central maintains the synchronous logical transports by using reserved slots at regular intervals. In addition to the reserved slots the eSCO logical transport may have a retransmission window after the reserved slots.

The ACL logical transport is also a point-to-point logical transport between the Central and a Peripheral. In the slots not reserved for synchronous logical transport(s), the Central can establish an ACL logical transport on a per-slot basis to any Peripheral, including the Peripheral(s) already engaged in a synchronous logical transport.

The APB logical transport is used by a Central to communicate with active Peripherals.

The CPB logical transport is used by a Central to send profile broadcast data to zero or more Peripherals.

**Logical transport address (LT\_ADDR):**

Each Peripheral active in a piconet is assigned a primary 3-bit logical transport address (LT\_ADDR). The all-zero LT\_ADDR is reserved for APB broadcast messages. The CPB logical transport uses a single non-zero LT\_ADDR. The Central does not have an LT\_ADDR. A Central's timing relative to the Peripherals’ distinguishes it from the Peripherals. A secondary LT\_ADDR is assigned to the Peripheral for each eSCO logical transport in use in the piconet. The secondary LT\_ADDR shall not be 0. Only eSCO traffic (i.e. NULL, POLL, and one of the EV packet types as negotiated at eSCO logical transport setup) may be sent on these LT\_ADDRs. ACL traffic (including LMP) shall always be sent on the primary LT\_ADDR. A Peripheral shall only accept packets with matching primary or secondary LT\_ADDR and broadcast packets. The LT\_ADDR is carried in the packet header (see Section 6.4). The LT\_ADDR shall only be valid for as long as a Peripheral is connected. As soon as it is disconnected, the Peripheral shall lose all of its LT\_ADDRs.

The primary LT\_ADDR shall be assigned by the Central to the Peripheral when the Peripheral is activated. This is either at connection establishment or role switch, when the primary LT\_ADDR is carried in the FHS payload.

At any given time an LT\_ADDR (other than the special case of the all-zero LT\_ADDR) is either unused or is used for exactly one of the three purposes of the primary address for a Peripheral, a secondary address for eSCO traffic, or for a CPB logical transport. Therefore, allocating a secondary LT\_ADDR for an eSCO logical transport, or reserving an LT\_ADDR for the CPB logical transport, reduces the maximum number of active Peripherals possible in the piconet.

* + 1. **Synchronous logical transports:**

The first type of synchronous logical transport, the SCO logical transport, is a symmetric, point-to-point transport between the Central and a specific Peripheral. The SCO logical transport reserves slots and can therefore be considered as a circuitswitched connection between the Central and the Peripheral. The Central may support up to three SCO links to the same Peripheral or to different Peripherals. A Peripheral may support up to three SCO links from the same Central, or two SCO links if the links originate from different Centrals. SCO packets are never retransmitted.

The second type of synchronous logical transport, the eSCO logical transport, is a pointto-point logical transport between the Central and a specific Peripheral. eSCO logical transports may be symmetric or asymmetric. Similar to SCO, eSCO reserves slots and can therefore be considered a circuit-switched connection between the Central and the Peripheral. In addition to the reserved slots, eSCO supports a retransmission window immediately following the reserved slots. Together, the reserved slots and the retransmission window form the complete eSCO window.

* + 1. **Asynchronous logical transport:**

In the slots not reserved for synchronous logical transports, the Central may exchange packets with any Peripheral on a per-slot basis. The ACL logical transport provides a packet-switched connection between the Central and all active Peripherals participating in the piconet. Both asynchronous and isochronous services are supported. Only a single ACL logical transport shall exist between any two devices. For most ACL packets, packet retransmission is applied to assure data integrity.

ACL packets not addressed to a specific Peripheral (LT\_ADDR=0) are considered as broadcast packets and should be received by every Peripheral except Peripherals with only a CPB logical transport. If there is no data to be sent on the ACL logical transport and no polling is required, no transmission is required.

The Link Manager Protocol (LMP) and the Link Controller (LC) work together to manage and control Bluetooth connections. Here's a detailed explanation of how they communicate and how the LC applies the parameters negotiated by the LMP:

**Communication Between LMP and LC**

1. **Role of LMP**:
   * The LMP is responsible for high-level link management tasks such as authentication, encryption, power control, and negotiation of connection parameters. It operates at a higher layer than the LC and uses the links provided by the LC to communicate with the LMP on the remote device.
2. **Role of LC**:
   * The LC handles low-level operations such as packet acknowledgment, frequency hopping synchronization, and link establishment/disconnection. It directly interacts with the physical layer to manage the actual transmission and reception of data packets.

**How LMP and LC Interact**

1. **LMP Commands**:
   * The LMP sends commands to the LC to perform specific actions. These commands include instructions for setting up connections, adjusting power levels, and managing link states. The LMP uses the ACL-C (Asynchronous Connection-Less Control) logical link to send these commands.
2. **LC Execution**:
   * The LC receives these commands and executes the necessary actions at the physical layer. For example, if the LMP negotiates a change in transmission power, the LC adjusts the power settings accordingly.
3. **Feedback Loop**:
   * The LC provides feedback to the LMP about the status of the link and the success of the executed commands. This feedback helps the LMP make informed decisions about further link management actions.

**Applying Negotiated Parameters**

1. **Parameter Negotiation**:
   * During the connection setup and maintenance phases, the LMP negotiates various parameters with the remote device's LMP. These parameters include connection intervals, power levels, and security settings.
2. **Parameter Application**:
   * Once the parameters are negotiated, the LMP sends the relevant commands to the LC to apply these settings. For example, if a new connection interval is agreed upon, the LMP instructs the LC to adjust the timing of packet transmissions to match the new interval.
3. **Continuous Monitoring**:
   * The LC continuously monitors the link quality and other relevant metrics. If any issues are detected, the LC can provide feedback to the LMP, which may then renegotiate the parameters or take corrective action.

**Example Scenario**

Imagine two Bluetooth devices establishing a connection. The LMP on Device A negotiates the connection parameters with the LMP on Device B. Once agreed upon, Device A's LMP sends commands to its LC to apply these parameters. The LC then manages the physical link according to the negotiated settings, ensuring that the connection remains stable and efficient.

------------------------------------------------------------------------------------------------

**LOW POWER MODE:**

**SNIFF MODE**

* SNIFF MODE Since many Bluetooth devices operate on battery power, it is important that mechanisms are available to minimize current drain during periods of inactivity.
* Sniff mode provides the mechanism to define extended periods of time during which the master and slave devices will not exchange ACL packets. Synchronous packet transmissions, such as SCO and eSCO, are not affected by sniff mode. Sniff mode scheduled absences may allow the local device to conserve power by turning off its radio and entering a low power mode. Sniff mode may also be used by a device that is a participant in multiple piconets (often called scatternet) when it needs to perform some over-the-air action in another piconet.

The following terminology is important when discussing sniff mode, some of the concepts are difficult to express in words so examples will follow the definitions:

* Anchor point – The slot boundary that defines the period when a slave device will listen for a transmission from its master device. The actual point in time where the slave enables its receiver will be derived from the anchor point by subtracting half of the uncertainty window duration. This means the uncertainty window length depends on the duration of link inactivity and the accuracy of the slot clocks on each device.
* Tsniff – Is the number of slots between adjacent anchor points. Since the master can only send data to the slave during even numbered slots, this value must be an even integer greater than zero.
* Dsniff – Is an offset value that must be an even integer between zero and (Tsniff - 2). An anchor point must satisfy the relationship:

• **Dsniff = Clock\_Value (bits 26-1) mod Tsniff**

* Nsniff attempt – Is the number of master-to-slave slots, including the anchor point, that the slave shall listen for transmissions from the master. Since this value represents only slots with a clock value divisible by 2, its value must be in the range: (1≤ N≤ Tsniff)
* Nsniff timeout – Is the number of master-to-slave slots that the slave continues to listen for master-to-slave packets after the slave has received a packet containing its own LT\_ADDR. This window allows exchanges to continue once data starts flowing without having to wait for the next anchor point. The sniff timeout counter will restart when a packet is received or, if the exchange continues long enough, at the next sniff anchor point.

**SNIFF PARAMETER DIAGRAMS:**

Some of the following timing diagrams are designed to illustrate concepts and are not representative of sensible sniff parameters. There would be little point in setting the sniff interval to six, for example. However, it is rather easy to create a timing diagram that shows six slots. A discussion of realistic sniff parameters is included following the parameter definition diagrams.

A diagram of a mathematical equation

AI-generated content may be incorrect.

Figure 4: illustrates a Tsniff interval of 6. Here, the initial sniff anchor point is defined by the relationship:

**Clock26-1 mod Tsniff = 0**

Equation 5: Tsniff relation to clock

To put it another way, when the slot clock value is evenly divisible by 6, then that slot is the sniff anchor point. The careful reader will notice that the equation uses only 26 bits of the 28 bit Bluetooth clock. The high order bit, bit 27, is ignored. In the core spec there are actually two equations listed, but they both ignore bit 27. The low order bit, bit 0, is not used as the clock runs at twice the slot rate. Slots are 625 μs but the clock period is 312.5 μs. Since sniff is only concerned with full slots, the low order bit is not considered.

After the initial anchor point is determined, subsequent anchor points are calculated by adding the fixed interval Tsniff to the clock value of the current sniff anchor point. Since the clock can roll over, the “Clock modulo Tsniff = 0” relationship may not be maintained. In other words, the value of Dsniff may change when the clock rolls over.

This is not the complete story. Consider the case where a master device has two slaves and both slaves need to be put into sniff mode with Tsniff = 6. Equation 5 implies that the anchor points for both devices would coincide. Since the master can only address one slave in any given slot, this would not work. Fortunately the sniff feature also provides for an offset, Dsniff, within the Tsniff interval that covers this situation. So in actuality, equation 5 is a specific case, the more general case is shown below in equation 6.

**Clock26-1 mod Tsniff = Dsniff**

Equation 6: Tsniff relation to clock

Figure 5 illustrates the same sniff connection that is represented in Figure 4, but with the addition of a second device connection using Tsniff = 6 and Dsniff = 2.

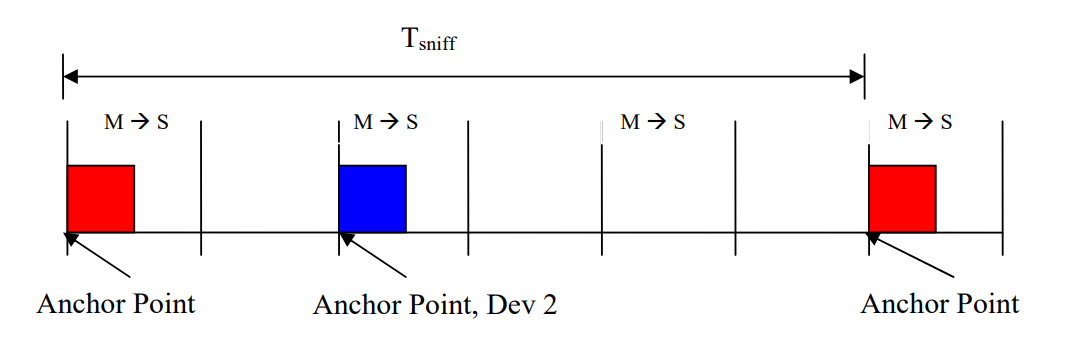


Figure 5: Add second connection to show Dsniff=0 (red) and Dsniff=2 (blue)

Sniff is designed to save power by reducing the amount of link maintenance data exchanged between the two devices and to allow PMP devices to be absent from one piconet while they are participating in another piconet. In these cases sniff anchor points may be one or two seconds apart. If a sniff anchor point transaction is lost, possibly due to the master being unexpectedly busy in a second piconet, or a dropped master to slave packet, the slave will need to wait for another sniff interval to pass before it can communicate with the master. In order to improve link robustness, the sniff setup includes a parameter that changes the anchor point concept from one slot to several slots.

A black and white image of a rectangular object

AI-generated content may be incorrect.

Figure 6: Sniff attempts illustrated.

Figure 6 illustrates a link with a Tsniff value greater than 6, since another anchor point is not visible in the diagram, and an Nsniff attempt of 3. Important points to note are:

• Nsniff attempt is a count of master to slave slots

• Nsniff attempt includes the anchor point so it must be set to 1 or more

• The maximum value of sniff attempt is Tsniff / 2

• The slave is required to listen at each M -> S slot during the Nsniff attempt period.

Just as Nsniff attempt may be used to increases the robustness of reestablishing a connection at a sniff anchor point, Nsniff timeout may be used to insure that devices can complete any required data transfers before they return to a low power state awaiting the next sniff anchor point. The following diagrams will illustrate all of these concepts.

A graph with arrows and letters

AI-generated content may be incorrect.

Figure 7: Anchor point with no communication from master.

In Figure 7, the slave receives no data from the master in the designated slot. Since the Nsniff attempt value is 1, the slave does not need to continue to listen in the next master to slave slot. Instead it resumes low power mode and waits for the next sniff anchor point when it will wake up and listen again for a packet from the master.

A graph with text and symbols

AI-generated content may be incorrect.

Figure 8: Anchor point with no data.

In Figure 8, the master sends a POLL packet at the anchor point. The slave answers with a NULL packet. Since no ACL data was transmitted, the slave is allowed to go back to low power mode and await the next sniff anchor point. But note, the rules do say that the slave may continue to listen at master to slave slots.

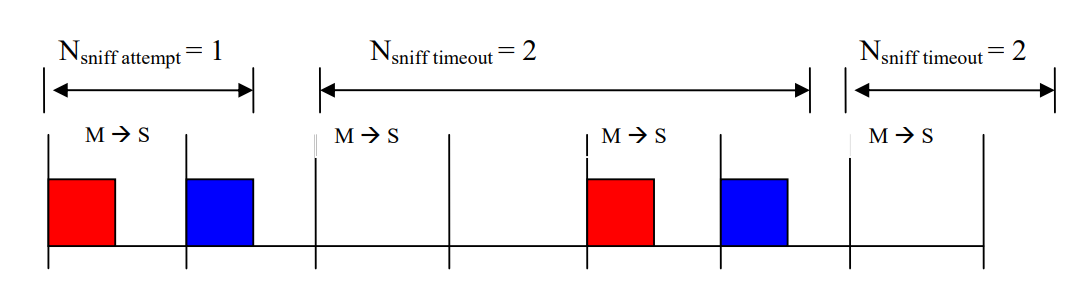


Figure 9:Anchor point with ACL data.

In Figure 9 the master sends ACL data to the slave and the slave responds with ACL data for the master. At this point, since the Nsniff timeout value is 2, the slave will continue to listen for more traffic from the master. The slave receives another ACL data packet from the master and responds in kind. Since data was exchange, the timeout counter is restarted and the slave will continue to listen for data until the timeout expires with no master to slave ACL data received.

A diagram with text and numbers

AI-generated content may be incorrect.

Figure 10: Anchor point with one data exchange.

Figure 10 illustrates a data timeout. The master and slave exchange data at the anchor point. The slave continues to listen for 2 more master-to-slave slots. Seeing no traffic from the master, the slave resumes low power mode until the next sniff anchor point.

A black and white diagram

AI-generated content may be incorrect.

Figure 11: Anchor point during data exchange.

In Figure 11, the two devices have continued to exchange data through an entire Tsniff interval. When the anchor point arrives all sniff values are reset and the link is waiting for the anchor point packet. Note that the sniff timeout still had one slot to go, but was terminated at the anchor point. Once the next anchor point arrives, the behaviors defined by Figure 7 through Figure 10 are again applicable.

**REQUESTING SNIFF MODE:**

The preceding discussion of sniff parameters describes the link manager view of sniff. Remember, it is ultimately up to the LM software to negotiate the details of link management including reserved slot allocations for synchronous data, and certain sniff parameters. The application code makes requests of the link manager and the link manager negotiates the details with the remote device’s link manager.

In systems where the optional HCI layer is employed, Sniff mode is requested by the host stack using the HCI command HCI\_Sniff\_Mode. The required parameters are:

• Connection\_Handle – a two octet number that identifies the affected ACL connection

• Sniff\_Max\_Interval – the maximum number of slots that the link may remain in sniff mode or, using the LMP terms, the maximum value of Tsniff. This value must be an even number as it relates to master to slave slots, which are always even. This value must be greater than the sniff minimum interval and mandatory range is between 0x6 and 0x540. This equates to 3.75 msec through 840 msec. Implementations may allow values outside the mandatory range.

• Sniff\_Min\_Interval – the lower acceptable value for Tsniff. The value range is the same as for the max interval above, but it must be less than the specified max interval.

• Sniff\_Attempt – the number of masters to slave slots, including the sniff anchor slot, that slave device will monitor for transmissions. Cannot exceed Tsniff / 2.

• Sniff\_Timeout – when > 0, the number of masters to slave slots that the slave will continue to listen after receiving a master packet containing ACL data.

Note that, since the application layer cannot know what allocations are being made by the LM, it is up to the LM to negotiate the values for Tsniff and Dsniff, so they are not a part of the HCI parameter set for sniff.

**SNIFF SUB-RATING:**

Sniff sub-rating (SSR) provides a means to further reduce power consumed by link management. SSR allows either device to increase the time between sniff anchor points. While this change will reduce the responsiveness of the link, it also reduces the number of packets that are exchanged to maintain the link and thus reduces power consumption.

SSR is particularly useful for devices that have periods of activity separated by long periods of inactivity. A computer mouse would be a good example. A user working with a word processor might use the mouse to open a document and position the cursor. Once that is accomplished the user might use only the keyboard for an extended time, never touching the mouse. During these periods of inactivity, the mouse can move into SSR mode and reduce its power consumption.

SSR parameters are established by the use of the HCI command HCI\_Sniff\_Subrating. The key parameters are:

**Maximum\_Latency:** Is specified in units of baseband slots (625 μs). This value is used by the link manager (LM) layer to calculate the value of max\_sniff\_subrate, which is sent as a parameter of the LMP command used to start sniff subrating. Note: The core specification refers to this value as max\_sniff\_subrate, maximum\_sniff\_subrate, or max\_subrate, depending on which section you reference.

While these terms all mean the same thing, this document will use only max\_sniff\_subrate to avoid confusion.

**Minimum\_Remote\_Timeout and Minimum\_Local\_Timeout**: Is specified in units of baseband slots (625 μs). This value is used by the link manager (LM) layer to determine when to transition a device from sniff mode to sniff sub-rating mode.

At the baseband level, the max\_sniff\_subrate is an integer multiplier applied to the basic Tsniff value. For example, if Tsniff is 20 slots and max\_sniff\_subrate is 4, then when the device enters sniff subrating mode each anchor point will be separated by 80 slots.

**ESTABLISHING SNIFF SUB-RATING MODE:**

Sniff sub-rating mode is initiated when one of the devices sends the LMP command

LMP\_sniff\_subrating\_req and the other device responds with LMP\_sniff\_subrating\_res. In this exchange, the link master device shall provide a Bluetooth clock value known as the sniff sub-rating instant. This clock value must be in the future and will be the point at which sniff sub-rating mode begins.

These PDUs also exchange the max\_sniff\_subrate values used by each device. This number is calculated and must not cause the time between anchor points to exceed either the HCI-supplied maximum latency or the link supervision timeout value. If the value exceeds the latter, the link will timeout and disconnect waiting for the next anchor point. If the master needs to change the link supervision time out (LSTO) value, and the new value is less than the sniff sub-rating anchor point interval, the master must exit sniff sub-rating before setting the LSTO value and then re-establish sniff sub-rating.

**MOVING BETWEEN SNIFF AND SNIFF SUB-RATING MODES**

Each device can move between sniff mode and sniff sub-rating mode independently of the other device. Since all sub-rating anchor points are also sniff anchor points, the devices will eventually reconnect.

As a part of establishing sniff sub-rating, each device will provide a time-out value via the HCI layer. When a device enters sniff mode a timer will be started. When the timer exceeds the sub-rating time-out value, the device will transition to sniff sub-rating mode. Since the two devices may have different timeout values, the current mode of each device is independent of the mode of the other device.

A device shall transition from sniff subrating mode to sniff mode whenever it receives ACL-C data, used to carry LMP signalling between devices, or ACL-U data, used to carry L2CAP data between devices, from its partner device. This transition to sniff mode will continue until the sniff timeout timer returns the device to sniff subrating mode.

A slave device shall temporarily exit sniff subrating mode and enter sniff mode while waiting for baseband acknowledgement of a transmitted packet.

------------------------------------------------------------------------------------------------

**HOLD MODE:**

During the Connection state, the ACL logical transport to a Peripheral can be put in a Hold mode. In Hold mode the Peripheral temporarily shall not support ACL packets on the channel. Any synchronous packet during reserved synchronous slots (from SCO and eSCO links) shall be supported. With the Hold mode, capacity can be made free to do other things like scanning, paging, inquiring, or attending another piconet. The device in Hold mode can also enter a low-power sleep mode. During Hold mode, the Peripheral keeps its logical transport address(es) (LT\_ADDR).

Prior to entering Hold mode, Central and Peripheral agree on the time duration the Peripheral remains in Hold mode. A timer shall be initialized with the holdTO value. When the timer is expired, the Peripheral shall wake up, synchronize to the traffic on the channel and will wait for further Central transmissions.

------------------------------------------------------------------------------------------------

**PAYLOAD HEADER – GUARD TIME:**

Sure! Let's break this down in simple terms, especially in the context of **Bluetooth Enhanced Data Rate (EDR)** communication.

**🟦 What is Bluetooth EDR?**

Bluetooth EDR is a faster version of Bluetooth that allows quicker data transfer by using more advanced modulation techniques than the basic version.

**🟦 What is "Guard Time"?**

Think of **guard time** as a **short pause or buffer** between two parts of a Bluetooth signal. It's like a small gap to make sure everything is aligned properly before continuing.

**🟦 Where does this guard time happen?**

In Bluetooth EDR packets:

* The **header** (the beginning part of the packet) is sent using a basic modulation method called **GFSK**.
* The **payload** (the actual data) is sent using a faster method (like π/4-DQPSK or 8DPSK).

The **guard time** is the tiny pause between:

* The **end of the header** (last GFSK symbol), and
* The **start of the synchronization sequence** (which helps the receiver lock onto the new faster signal).

**🟦 Why is this important?**

Switching from one modulation type to another (GFSK to EDR) needs a little time to stabilize. The guard time ensures:

* The receiver has time to adjust.
* There’s no overlap or confusion between the two parts of the signal.

**🟦 How long is this guard time?**

It must be between **4.75 microseconds and 5.25 microseconds** — a very short but precise window.

**PAYLOAD – SYNCHRONIZATION SEQUENCE:**

Let’s simplify this technical Bluetooth EDR (Enhanced Data Rate) explanation step by step:

**🔷 What’s Happening Here?**

This section describes how **Bluetooth EDR packets** transition from the **header** (sent using GFSK modulation) to the **payload** (sent using faster DPSK modulation). To make this transition smooth and accurate, a **synchronization sequence** is used.

**🔷 Key Concepts Explained Simply**

**1. Symbol Timing Alignment**

* The **symbol timing** (the rhythm of the signal) at the start of the synchronization sequence must be **very closely aligned** (within ±0.25 microseconds) with the timing of the last GFSK symbol.
* This ensures a **smooth handoff** from the header to the payload.

**2. Synchronization Sequence**

* This is a **special 11-symbol pattern** that helps the receiver lock onto the new modulation format (DPSK).
* It starts with a **reference symbol** (called **Sref**) and is followed by **10 DPSK symbols** (**S1 to S10**).
* The **phase changes** (how the signal rotates) between these symbols are predefined and follow this pattern:

φ1 to φ10 =

3π/4, -3π/4, 3π/4, -3π/4, 3π/4, -3π/4, -3π/4, 3π/4, 3π/4, 3π/4

These phase changes help the receiver recognize and sync with the signal.

**3. Bit Sequences for Generating the Sync Pattern**

* To create this synchronization sequence, the transmitter **prepends specific bits** to the data.
* These bits are different depending on the modulation:
  + For **π/4-DQPSK**:  
    0,1,1,1,0,1,1,1,0,1,1,1,1,1,0,1,0,1,0,1
  + For **8DPSK**:  
    0,1,0,1,1,1,0,1,0,1,1,1,0,1,0,1,1,1,1,1,1,0,1,0,0,1,0,0,1,0

These bits are **not part of the actual data** — they’re just used to generate the sync pattern.

**🔷 Visual Summary (Text-Based)**

[Header (GFSK)] --5 µs Guard Time--> [Sref][S1][S2]...[S10] --> [Payload (DPSK)]

↑

Sync starts here

* **Sref**: Reference symbol (starting point)
* **S1 to S10**: DPSK symbols with specific phase changes (φ1 to φ10)

**COEXISTENCE AND COLLOCATION 328**

🡪Bluetooth devices operate in the unlicensed 2.4 GHz Industrial, Scientific and Medical (ISM) band. Many other technologies utilize the ISM band, including wireless LAN, cordless phones, and microwave ovens. The ISM band is also close enough to other frequency bands that Bluetooth devices may be an interferer of or a victim of other technologies.

🡪Radios may be collocated or non-collocated. The term "collocated" is a loose one - in this specification, collocated radios are assumed to be in the same product (a Multi-Radio Terminal or MRT) and may have mechanisms to coordinate their activity in order to mitigate interference.

🡪Determining the amount of expected isolation between radios is important for choosing an appropriate coexistence mechanism. With sufficient isolation, frequency division duplexing (FDD) techniques are the most efficient. With insufficient isolation or a shared antenna, time division duplexing (TDD) techniques need to be used. In many cases, a combination of FDD and TDD techniques are required to achieve acceptable levels of performance.

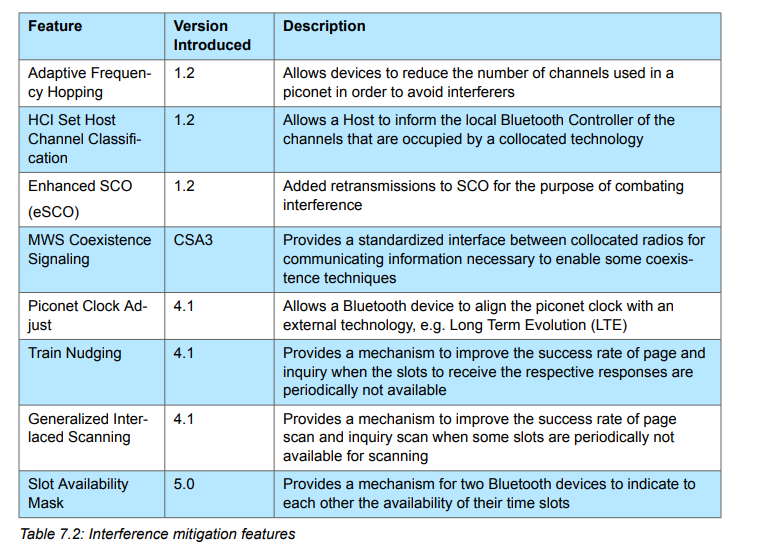
🡪This specification supports a variety of features that help mitigate interference to other devices and to minimize interference from other devices. Broadly, the types of solutions fall into the following categories:

A screenshot of a computer

AI-generated content may be incorrect.

**Core features supporting coexistence and collocation:**

There are features in the specification to specifically target the reduction of interference from collocated or non-collocated devices.



**ENCRYPTION**

* **Encryption key size:**

In the Bluetooth specification, the terms you're asking about—**L\_sug\_c**, **L\_min\_p**, and **L\_max\_p**—refer to **encryption key sizes** negotiated between the Central and Peripheral devices during the encryption setup phase.

Here’s what each term means:

**🔑 Key Size Negotiation Terms**

* **L\_sug\_c**:  
  This is the **suggested key size** proposed by the **Central** device.
  + Initially, it is set to L\_max\_c, which is the **maximum key size** the Central supports.
* **L\_min\_p**:  
  This is the **minimum key size** that the **Peripheral** device is willing to accept.
* **L\_max\_p**:  
  This is the **maximum key size** that the **Peripheral** can support.

**🔄 Negotiation Logic**

When the Central sends its suggested key size (L\_sug\_c) to the Peripheral:

* If the suggested size is **within the range** supported by the Peripheral: $ L*{\text{min}*p} \leq L*{\text{sug}*c} \leq L*{\text{max}*p} $ then the Peripheral **accepts** it by sending an LMP\_ACCEPTED PDU.
* If not, the Peripheral may **reject** it or propose a different size, depending on the protocol version and capabilities.

**📏 Why This Matters**

* This negotiation ensures that both devices agree on a **secure and compatible encryption key size**.
* Larger key sizes (e.g., 16 bytes or 128 bits) offer **stronger security**, but both devices must support it.
  + - This part of the Bluetooth specification describes how the **encryption key size negotiation** continues if the initial suggestion from the Central is not accepted by the Peripheral.

Let’s walk through it step by step:

**🔁 Two-Way Negotiation Process**

1. **Initial Proposal by Central**:
   * The Central sends an LMP\_ENCRYPTION\_KEY\_SIZE\_REQ PDU with its suggested key size (L\_sug\_c).
2. **Peripheral Evaluates**:
   * If the Peripheral **accepts** the key size (i.e., it falls within its supported range), it replies with LMP\_ACCEPTED, and the key size is agreed upon.
   * If **not**, the Peripheral can **respond with its own LMP\_ENCRYPTION\_KEY\_SIZE\_REQ**, suggesting a different key size.
3. **Central Evaluates Peripheral’s Suggestion**:
   * Now the Central checks whether the Peripheral’s suggested key size is acceptable (i.e., within the Central’s own supported range).
   * If it is, the Central replies with LMP\_ACCEPTED.
4. **Repeat Until Agreement or Failure**:
   * This back-and-forth continues until:
     + **An agreement is reached**, in which case the **last suggested key size** (from the last LMP\_ENCRYPTION\_KEY\_SIZE\_REQ) is used.
     + **No agreement is possible**, meaning the supported ranges of the two devices do not overlap. In that case, encryption cannot proceed.

**✅ Key Takeaway**

This negotiation ensures that **both devices agree on a secure and compatible encryption key size**, and it allows for **flexibility** if the initial suggestion isn’t acceptable.

**TEMPORARY KEY:**

You're asking about a detailed process in the Bluetooth specification for generating and using a **temporary link key** during a session. Let me explain it step by step in **simple terms**:

**🔐 Why Use a Temporary Link Key?**

* Normally, Bluetooth devices use a **semi-permanent link key** for encryption.
* Sometimes, during a session, they need a **temporary key** for added security or flexibility.
* This temporary key is **not stored permanently** and is used only during the current session.

**🧩 Step-by-Step Breakdown**

**1. Central Creates a Temporary Key (K\_temp)**

* The Central generates two random numbers: RAND1 and RAND2.
* It uses a cryptographic function E22 to combine them: $ K\_{\text{temp}} = E22(\text{RAND1}, \text{RAND2}, 16) $
* This ensures the key is **secure and truly random**, even if the random number generator isn't perfect.

**2. Central Sends a Third Random Number (RAND)**

* This RAND is sent to the Peripheral.
* Both devices use E22 again, but this time with the **current link key** and RAND to compute an **overlay**: $ \text{Overlay} = E22(\text{current link key}, \text{RAND}, 16) $

**3. Central Sends Encrypted Temporary Key**

* The Central encrypts K\_temp by XORing it with the overlay: $ \text{Encrypted } K*{\text{temp}} = K*{\text{temp}} \oplus \text{Overlay} $
* It sends this to the Peripheral.

**4. Peripheral Recovers K\_temp**

* The Peripheral knows the overlay and can decrypt the message to get K\_temp.

**5. Authentication**

* Both devices perform **mutual authentication** using K\_temp to confirm they both have the same key.
* This step ensures the key exchange was successful and secure.

**6. ACO Is Not Updated**

* ACO (Authenticated Ciphering Offset) is normally updated during authentication.
* In this case, it is **not updated**, because the original ACO is needed if the devices switch back to the semi-permanent key later.

**🔒 Encryption Activation**

* The Central sends a command to **activate encryption**.
* Before doing this, it sends a **shared random number** called EN\_RAND to all Peripherals.

**7. Each Peripheral Computes Encryption Key**

* Each Peripheral uses:
  + K\_temp (temporary key)
  + EN\_RAND (shared random number)
  + COF (Ciphering Offset, derived from Central’s Bluetooth address)
* They use another function E3 to compute the encryption key: $ K*{\text{enc}} = E3(K*{\text{temp}}, EN\_{\text{RAND}}, COF) $

**📝 Important Notes**

* The temporary key is used **only during the session**.
* The ACO from this authentication is **not used** to compute the encryption key.
* This process ensures **secure communication** even if the semi-permanent key is compromised.

**PROFILE VS PROTOCOL:**

In Bluetooth (BT), profiles and protocols serve different but complementary roles in enabling communication between devices. Here's a clear breakdown of the differences:

🔧 Protocol

* Definition: A protocol defines the rules and formats for data exchange between Bluetooth devices.
* Function: It handles how data is transmitted, routed, and managed.
* Examples:
  + L2CAP (Logical Link Control and Adaptation Protocol): Manages data multiplexing and segmentation.
  + RFCOMM: Emulates serial ports over Bluetooth.
  + ATT (Attribute Protocol): Used in Bluetooth Low Energy (BLE) for data exchange.
  + GATT (Generic Attribute Profile): Built on top of ATT, used for discovering services and characteristics.

📱 Profile

* Definition: A profile defines a specific use case or application for Bluetooth communication.
* Function: It specifies what a device can do and how it should behave in a particular scenario.
* Examples:
  + A2DP (Advanced Audio Distribution Profile): For streaming high-quality audio.
  + HFP (Hands-Free Profile): For voice calls in car systems.
  + HID (Human Interface Device Profile): For keyboards, mice, game controllers.
  + HRP (Heart Rate Profile): For fitness devices.

🧩 How They Work Together

* Protocols are the building blocks.
* Profiles are the blueprints that use those blocks to create specific functionalities.

For example:

* The Heart Rate Profile (HRP) uses the GATT protocol to define how a heart rate monitor communicates with a smartphone.