UNIT III

Signal Encoding Techniques: Digital Data, Digital Signals-Digital Data, Analog Signals-Analog Data, Digital Signals-Analog Data, Analog Signals. Digital Data Communication Techniques: Asynchronous and Synchronous Transmission - Types of Errors –Error Detection –Error Correction-Line Configurations.

Signal Encoding Techniques

Introduction to Signal Encoding

Signal encoding is the process of converting data into a signal suitable for transmission over a communication channel. The goal is to transform digital data (bits) or analog data into a waveform that can be efficiently transmitted and reliably decoded at the receiver.

Encoding affects bandwidth usage, signal quality, error detection, synchronization, and power consumption.

Types of Signals

- **Analog Signals:** Continuous signals that vary smoothly over time (e.g., voice, music).
- **Digital Signals:** Discrete signals with distinct levels representing binary data (0s and 1s).

Types of Encoding Techniques

Encoding techniques are classified based on the type of signal and the encoding method:

A. Digital-to-Digital Encoding

This converts digital data into digital signals.

Purpose: To represent bits as specific voltage levels or pulses for transmission.

- Examples:
  - Unipolar
  - Polar
  - Bipolar
  - Manchester
  - Differential Manchester
  - AMI (Alternate Mark Inversion)

B. Analog-to-Digital Encoding

This converts analog signals into digital signals.

- Examples:
  - Pulse Code Modulation (PCM)
  - Delta Modulation (DM)

C. Digital-to-Analog Encoding

Converts digital signals into analog signals for transmission over analog channels.

- Examples:
  - ASK (Amplitude Shift Keying)
  - FSK (Frequency Shift Keying)
  - PSK (Phase Shift Keying)

D. Analog-to-Analog Encoding

Converts analog signals to analog signals with some modulation.

- Examples:
  - AM (Amplitude Modulation)
  - FM (Frequency Modulation)
  - PM (Phase Modulation)

Digital-to-Digital Encoding Techniques (Focus on common line codes)

Unipolar Encoding

- Uses one voltage level to represent binary 1, zero voltage for binary 0.
- Simple but has a large DC component (not good for AC coupling).
- No synchronization since long strings of 0s produce no signal.

Polar Encoding

- Uses two voltage levels: +V for 1 and -V for 0 (or vice versa).
- Better than unipolar since no DC component.
- More power efficient.

Bipolar Encoding (AMI - Alternate Mark Inversion)

- Zero represented by 0 volts.
- Ones alternate between positive and negative voltages (+V, -V, +V, ...).
- No DC component.
- Detects errors if two consecutive marks (1s) have the same polarity.
- Good for synchronization.

Manchester Encoding

- Combines clock and data by encoding bits with transitions:
  - 0 = high-to-low transition
  - 1 = low-to-high transition
- Self-clocking (synchronization easy).
- Doubles the bandwidth (each bit requires two signal intervals).
- No DC component.

Differential Manchester Encoding

- Always a transition at the middle of the bit interval.
- The presence or absence of a transition at the start indicates the bit:
  - Transition at start = 0
  - No transition at start = 1
- Self-clocking and robust to polarity reversals.

Analog-to-Digital Encoding Techniques

Pulse Code Modulation (PCM)

- Analog signal sampled at uniform intervals (Nyquist rate).
- Samples quantized to nearest value.
- Samples encoded into binary form.
- Used in telephony and digital audio.

Delta Modulation (DM)

- Encodes difference between successive samples instead of absolute value.
- Simple and requires less bandwidth but less accurate for rapidly varying signals.

Digital-to-Analog Encoding Techniques (Modulation)

Amplitude Shift Keying (ASK)

- Data represented by varying amplitude of carrier wave.
- 1 = high amplitude, 0 = low amplitude or zero.
- Simple but susceptible to noise.

Frequency Shift Keying (FSK)

- Data represented by changing frequency of carrier.
- 1 and 0 use different frequencies.
- More robust than ASK.

Phase Shift Keying (PSK)

- Data represented by changing phase of carrier wave.
- Common form: Binary PSK (BPSK) with two phases 0° and 180°.
- Robust and efficient.

Important Characteristics of Encoding Schemes

Characteristic	Explanation
Bandwidth Usage	Amount of spectrum required
Power Efficiency	Power required for transmission
Error Detection	Capability to detect errors
Synchronization	Ease of maintaining clock alignment
DC Component	Presence of zero frequency component affects line transmission (e.g., transformer coupling)
Complexity	Encoder/decoder complexity

Summary Table of Digital Line Encoding Techniques

Encoding Technique	Signal Levels	DC Component	Synchronization	Bandwidth Requirement	Notes
Unipolar	0, +V	Yes	Poor	Low	Simple but inefficient
Polar	+V, -V	No	Moderate	Moderate	Better power efficiency
Bipolar (AMI)	0, +V, -V	No	Good	Moderate	Error detection with violation
Manchester	Transition-based	No	Excellent	High (double bandwidth)	Self-clocking
Differential Manchester	Transition-based	No	Excellent	High	More robust than Manchester

Digital Data and Digital Signals

Introduction

In digital communication systems, **digital data** and **digital signals** are core components. Understanding their roles helps in grasping how computers, networks, and digital devices function.

What is Digital Data?

Definition:

Digital data refers to information represented in **discrete, binary form**, using a finite set of values (typically 0 and 1). It is the raw data that needs to be transmitted, stored, or processed.

Examples:

- Text ("Hello")
- Numbers (123)
- Audio/video files (in binary format)
- Sensor readings (digitized)

Characteristics:

Feature	Description
Discrete	Only specific values (e.g., 0, 1)
Binary	Usually represented in base-2 system
Quantized	Values are fixed, not continuous
Stored/Processed	In computers, memory, databases

What is a Digital Signal?

Definition:

Digital signals are physical waveforms or voltages that represent digital data. They carry the data over a transmission medium by switching between discrete voltage levels.

Examples:

- High voltage = 1, Low voltage = 0
- Serial data stream on a USB cable
- Waveform on a logic analyser

Characteristics:

Feature	Description
Discrete Levels	Two or more distinct voltage levels
Time-based	Signal changes at regular intervals
Pulse-shaped	Appears as square or rectangular waves
Transmittable	Used to transmit digital data over a medium

Relationship Between Digital Data and Digital Signals

Concept	Description
Source (Digital Data)	Original binary information (e.g., 10110010)
Medium (Digital Signal)	Electrical/optical pulses carrying that information
Encoding	Converts digital data into a digital signal format (line coding)
Decoding	Receiver interprets the signal and retrieves the original data

Example:
Text "A" → Binary: 01000001 → Voltage Pulses (digital signal) → Transmitted over wire → Received and decoded → "A"

Types of Digital Data

Type	Description	Example
Binary Data	0s and 1s	10101010
Text Data	Encoded using ASCII/Unicode	"Hello"
Numeric Data	Integers or floating points	42, 3.14
Multimedia Data	Images, audio, video in binary format	JPEG, MP3

Types of Digital Signals

➤ By Voltage Representation:

- **Unipolar:** Only one voltage (e.g., 0 = 0V, 1 = +5V)
- **Polar:** Two voltages (e.g., 0 = -5V, 1 = +5V)
- **Bipolar:** Uses three levels (e.g., -V, 0, +V)

➤ By Waveform:

- **NRZ (Non-return-to-zero)**
- **RZ (Return-to-zero)**
- **Manchester**
- **Differential Manchester**

Conversion Techniques

Conversion Type	Description	Example
Digital Data → Digital Signal	Line coding (e.g., NRZ, Manchester)	Ethernet
Analog Data → Digital Data	Sampling + Quantization (PCM)	Voice recording
Digital Data → Analog Signal	Modulation (ASK, FSK, PSK)	Modem
Analog Signal → Digital Signal	Not direct—requires digitization first	—

Digital Data vs Digital Signal — Comparison Table

Aspect	Digital Data	Digital Signal
Definition	Discrete information (binary format)	Electrical/optical pulses that carry data
Nature	Abstract/logical	Physical (waveform)
Representation	0s and 1s	Voltage levels, square waves
Purpose	To be stored, processed, or transmitted	To carry data across a medium
Example	10010110	Voltage waveform: High-Low-High

Applications

Digital Data:

- Software and applications
- Databases and websites
- Digital files (PDFs, media, spreadsheets)

Digital Signals:

- Computer buses (data transfer between CPU and RAM)
- Networking (Ethernet, Wi-Fi, USB)
- Microcontroller communication (SPI, I^2C)

Digital Data and Analog Signals

Introduction

In communication systems, data and signals are distinct but interconnected concepts:

- **Digital Data** is the *information* in binary form.
- **Analog Signal** is a *continuously varying waveform* used to transmit data over a medium.

This combination is important in systems like radio broadcasting, analog modems, and telecommunications.

What is Digital Data?

Definition:

Digital data refers to information represented using discrete symbols, typically binary digits (0 and 1). It is the format that computers and digital systems use to store, process, and transmit information.

Characteristics:

Feature	Description
Discrete	Only specific values (usually 0 and 1)
Binary Format	Based on base-2 numeral system
Digital Representation	Easily manipulated by computers
Error Detection	Can incorporate parity bits or checksums

Examples:

- Binary code: 01010101
- ASCII representation: A = 01000001
- Digital files: documents, images, audio (in compressed format)

What is an Analog Signal?

Definition:

An **analog signal** is a continuous waveform that varies smoothly over time and can represent a wide range of values. It is typically used to represent naturally occurring phenomena like sound, light, and temperature.

Characteristics:

Feature	Description
Continuous	Changes smoothly over time
Infinite Values	Can take any value in a given range
Sine Waveform	Often represented as a sine wave
Susceptible to Noise	Harder to filter or correct

Examples:

- Sound waves
- Electrical signals from microphones
- Radio signals
- Video signal before digitization

Relationship Between Digital Data and Analog Signals

In many systems, **digital data must be transmitted over analog channels**, like radio or telephone lines. This requires **modulation**:

Conversion Process:

Process	Description
Digital Data → Analog Signal	Requires **digital-to-analog modulation** (modem)
Analog Signal → Digital Data	Requires **sampling, quantization, and encoding**

Digital Data Transmitted via Analog Signals

Modulation Techniques (Digital Data → Analog Signal):

Type	Technique Name	Description
ASK	Amplitude Shift Keying	Varies amplitude of analog carrier
FSK	Frequency Shift Keying	Varies frequency of carrier signal
PSK	Phase Shift Keying	Varies phase of carrier signal

These techniques allow **binary data** to modulate an **analog carrier signal** for transmission over mediums like telephone lines, radio waves, or coaxial cables.

Digital Data vs. Analog Signals — Comparison Table

Feature	Digital Data	Analog Signal
Nature	Discrete	Continuous
Format	Binary (0 and 1)	Waveform (e.g., sine wave)
Values	Limited (e.g., 0 or 1)	Infinite values within a range
Processing	By computers and digital systems	By amplifiers, filters, analog circuits
Transmission	Needs encoding or modulation	Transmitted directly (e.g., audio signal)
Noise Sensitivity	Low (error correction possible)	High (degradation affects quality)

Applications

➤ **Digital Data:**

- Stored in computers, memory devices
- Used in software, databases, digital media
- Encoded into signals for transmission

➤ **Analog Signals:**

- Used in sound recording and playback
- Analog TV and radio broadcasting
- Transmitted in traditional telephone systems

Analog Data and Digital Signals

Introduction

In modern communication systems, **analog data** and **digital signals** are frequently combined, especially in digital communication systems that process natural signals like voice or video.

- **Analog Data** is the original, natural data (like sound or light) that varies continuously.
- **Digital Signal** is a discrete waveform used to transmit or represent data in binary form.

To use analog data in digital systems, it must first be **converted** into digital signals.

What is Analog Data?

Definition:

Analog data refers to information that is continuous in both time and value. It represents real-world phenomena such as sound, light intensity, temperature, or pressure.
Characteristics:

Feature	Description
Continuous	Exists at every instant of time
Infinite Values	Can take an infinite number of values
Natural Origin	Comes from real-world sources (e.g., sound)
Analog Representation	Often represented by sine waves

Examples:

- Human speech
- Music
- Temperature readings from a mercury thermometer
- Analog video (e.g., VHS tapes)

What is a Digital Signal?

Definition:

A **digital signal** is a sequence of voltage pulses that represent digital data using discrete values (typically two levels: 0 and 1). These signals are used to transfer or process binary information.

Characteristics:

Feature	Description
Discrete Values	Usually two levels (binary: 0 and 1)
Square Waveform	Abrupt changes in signal values
Synchronization	Timing based; each pulse occurs in time slots
Noise Resistant	Better immunity to distortion than analog

Relationship: Analog Data → Digital Signal

To transmit or process **analog data** using digital systems (like computers or digital networks), the data must be **converted** into a **digital signal** using the process of **digitization**.

Conversion Process (Analog Data → Digital Signal)

Process Steps:

Step	Description
1. Sampling	Measuring the amplitude of the analog signal at regular intervals
2. Quantization	Rounding off sampled values to nearest fixed level
3. Encoding	Converting quantized values into binary form
4. Line Coding	Converting binary data into a digital signal for transmission

This overall process is known as **Pulse Code Modulation (PCM)**.

Example:

Speech (analog) → Microphone → ADC (Analog-to-Digital Converter) → Binary data → Encoded into digital signal

Analog Data Transmitted as Digital Signals — Why?

Advantage	Explanation
Noise Resistance	Digital signals can be regenerated, reducing cumulative errors
Efficient Storage	Digital systems allow compact, lossless storage
Compression & Encryption	Easier to compress and secure digitally
Multiplexing	Multiple digital signals can be combined and transmitted
Compatibility with Modern Devices	Most devices (computers, phones) are digital

Analog Data vs Digital Signal — Comparison Table

Aspect	Analog Data	Digital Signal
Nature	Continuous	Discrete
Values	Infinite (real-world values)	Limited (usually 2: 0 and 1)
Representation	Sine wave, voltage levels	Square pulses, binary waveform
Example	Voice, video, temperature	01010101 (transmitted as voltage pulses)
Noise Sensitivity	High	Low (can regenerate signal)
Use in Systems	Needs conversion for digital systems	Ideal for digital storage/transmission

Applications

➤ Analog Data:

- Human voice in phone conversations
- Audio and video from cameras or microphones
- Environmental sensor readings (temperature, pressure)

➤ Digital Signal (carrying analog data):

- Voice-over-IP (Skype, WhatsApp)
- Digital music (MP3, CD audio)
- Digital video (streaming platforms like YouTube)
- Telemetry systems (IoT sensors to cloud platforms)

Key Concepts

Sampling Rate:

- Number of samples taken per second.
- Higher sampling rate → better quality representation of analog data.

Nyquist Theorem:

- To avoid information loss, sampling rate must be at least **twice** the highest frequency of the analog signal.

Bit Depth:

- Number of bits used per sample. Determines precision of the quantized value.

Analog Data Analog Signals

Introduction

Analog communication systems were the foundation of early telecommunication technologies. In this system:

- **Analog Data**: Real-world, continuously varying information.
- **Analog Signal**: A continuous waveform used to carry analog data over a communication medium.

What is Analog Data?

Definition:

Analog data refers to information that varies smoothly over time and can take on an **infinite number of values**. It is continuous both in **time** and **amplitude**.

Characteristics:

Feature	Description
Continuous	Exists at every instant of time
Variable	Can take on any value within a range
Natural Representation	Reflects real-world phenomena (e.g., sound, light)
Not Digital	Not represented in discrete binary form

Examples:

- Sound waves (human voice, music)
- Analog thermometer readings
- Light intensity
- Motion and pressure signals

What is an Analog Signal?

Definition:

An **analog signal** is a **continuous waveform** that varies in **amplitude**, **frequency**, or **phase** to represent information (often analog data).

Characteristics:

Feature	Description
Continuous	Infinite values over time
Sine Waveform	Often represented as smooth, periodic sine waves
Variable Parameters	Amplitude, frequency, and phase can change
Subject to Noise	Can be degraded by distortion and interference

Common Parameters:

- **Amplitude**: Height of the wave (signal strength)
- **Frequency**: Number of cycles per second (measured in Hz)
- **Phase**: Position of the wave in time

Relationship: Analog Data → Analog Signal

When analog data needs to be transmitted, it is **directly mapped** onto an analog signal using **analog modulation techniques**.

Analog Modulation Techniques

Analog signals can carry analog data using these **modulation methods**:

Technique	What it Varies	Description
Amplitude Modulation (AM)	Amplitude	Amplitude changes based on the data
Frequency Modulation (FM)	Frequency	Frequency changes to reflect data
Phase Modulation (PM)	Phase	Signal phase changes based on the data

Example:

Voice (analog data) → Modulates carrier wave using FM → Broadcast over radio → Demodulated at receiver → Original voice

Analog Data vs Analog Signal – Comparison Table

Feature	Analog Data	Analog Signal
Nature	Raw, continuous information	Waveform used to carry data
Representation	Physical quantities (e.g., sound, temp.)	Electrical/optical waveform
Variability	Varies over time	Varies in amplitude, frequency, phase
Usage	Source of information	Medium to transmit that information
Example	Voice	FM radio wave

Transmission of Analog Data using Analog Signals

Process:

1. **Source**: Analog data is generated (e.g., human speech).
2. **Modulation**: Data is encoded onto a carrier analog signal.
3. **Transmission**: Signal is sent through a medium (air, cable).
4. **Demodulation**: At the receiver, signal is decoded to retrieve analog data.

Real-World Example:

- FM Radio: Human voice → Modulated using FM → Transmitted → Received by radio → Demodulated → Sound from speaker.

Applications of Analog Data & Analog Signals

Field	Application Example
Telecommunications	Traditional telephone systems, AM/FM Radio
Broadcasting	Analog TV and radio
Instrumentation	Oscilloscopes showing analog signals
Sound Engineering	Analog microphones and speakers

Advantages and Disadvantages

Advantages:

- Simpler hardware
- Natural representation of real-world data
- Can directly interface with analog sensors (e.g., microphones)

Disadvantages:

- Prone to noise and distortion
- Limited signal quality over long distances
- Difficult to compress and encrypt
- Harder to multiplex many signals over one channel

Digital Data Communication Techniques

Introduction

Digital Data Communication Techniques involve methods used to transmit binary (digital) data between devices over a communication channel. These techniques ensure that the data is transmitted accurately, efficiently, and securely across networks or systems.

What is Digital Data?

Definition:

Digital data refers to data represented in discrete binary format — typically as 0s and 1s. This data needs to be transmitted over physical or wireless media using suitable communication techniques.

Digital Communication System: Basic Components

1. **Source** – Generates digital data (e.g., computer).
2. **Transmitter** – Encodes and modulates the data for transmission.
3. **Transmission Medium** – Cable, fiber, or wireless.
4. **Receiver** – Demodulates and decodes the signal.
5. **Destination** – Receives the original digital data.

Digital Data Communication Techniques Overview

There are two primary ways digital data is transmitted:

Technique	Description
Baseband Transmission	Transmit digital data directly as digital signals
Broadband (Carrier) Transmission	Use digital modulation to transmit data over analog channels

---

Baseband Transmission

Definition:

Transmission of digital data over a channel **without modulation**, using **digital signals** (like square waves).

Line Coding Techniques (Digital Data → Digital Signal):

Technique	Description
NRZ (Non-Return to Zero)	High = 1, Low = 0 (or vice versa)
RZ (Return to Zero)	Signal returns to zero between bits
Manchester Encoding	Bit = 0: High→Low, Bit = 1: Low→High
Differential Manchester	Always transitions; data based on phase
AMI (Alternate Mark Inversion)	0 = no voltage; 1 = alternating polarities

Characteristics:

- Suitable for **short distances**
- **Less bandwidth-efficient**
- **Simple and fast**

Broadband Transmission (Digital Modulation)

Definition:

Modulating a high-frequency **carrier signal** with digital data to enable **transmission over analog mediums** (telephone lines, radio waves).

Techniques (Digital Data → Analog Signal):

Type	Technique	Description
ASK	Amplitude Shift Keying	Amplitude changes based on bit (0 or 1)
FSK	Frequency Shift Keying	Frequency changes based on bit value
PSK	Phase Shift Keying	Phase changes represent data bits
QAM	Quadrature Amplitude Modulation	Combines ASK + PSK for higher bit rates

Multiplexing Techniques

Used to transmit multiple digital data streams over a single medium:

Technique	Description
TDM (Time Division Multiplexing)	Time slots assigned to each data source
FDM (Frequency Division Multiplexing)	Each source gets a separate frequency band
WDM (Wavelength Division Multiplexing)	Used in fiber optics; based on light wavelengths

---

Error Detection and Correction

Communication is prone to **noise** and **interference**, so techniques are needed to detect and correct errors:

Error Detection:

- **Parity Bit**
- **Checksum**
- **Cyclic Redundancy Check (CRC)**

Error Correction:

- **Hamming Code**
- **Reed-Solomon Code**
- **Forward Error Correction (FEC)**

Synchronous vs Asynchronous Transmission

Feature	Synchronous Transmission	Asynchronous Transmission
Timing	Requires clock synchronization	Independent timing
Speed	Faster	Slower
Overhead	Less overhead	More overhead (start/stop bits)
Example	Ethernet, DSL	Serial port, keyboard input

Protocols in Digital Data Communication

Protocol	Use Case
RS-232	Serial communication (COM ports)
USB	Data communication between peripherals
Ethernet	Local area networks
Wi-Fi (802.11)	Wireless communication

Asynchronous and Synchronous Transmission

Introduction

In digital data communication, **timing** between the sender and receiver is crucial. Based on how timing is coordinated, data transmission is categorized into:

- **Asynchronous Transmission**
- **Synchronous Transmission**

These techniques are essential for understanding how digital systems transmit data accurately and efficiently.

What is Asynchronous Transmission?

Definition:

Asynchronous transmission is a method of data transmission in which **each byte (or character) is sent individually** with **start and stop bits** to indicate the beginning and end of transmission.

Key Features:

- No shared clock between sender and receiver.
- Each data unit is **self-contained**.
- Common in **low-speed**, **intermittent** communication.

Structure:

```
css
CopyEdit
[Start Bit][Data Bits][Optional Parity Bit][Stop Bit(s)]
```

Example:

Sending the ASCII letter 'A' (binary 01000001) might be transmitted as:

```
scss
CopyEdit
Start(0) + 01000001 + Stop(1)
```

Use Cases:

- Serial ports (RS-232)
- Keyboard input
- Modems
- UART communication

What is Synchronous Transmission?

Definition:

Synchronous transmission sends **a continuous stream of data** along with timing information. Both sender and receiver **share a synchronized clock** or use special synchronization bytes/frames.

Key Features:

- Data sent in **blocks or frames**.
- Requires **synchronization** before data exchange.
- More efficient for **large volumes** of data.

Frame Structure (Simplified):

```
css
CopyEdit
[Header/SYNC][Data Block][Error Check]
```

Use Cases:

- High-speed communication
- Ethernet
- DSL, ISDN
- Digital telephony
- WANs and LANs

Comparison: Asynchronous vs Synchronous Transmission

Feature	Asynchronous Transmission	Synchronous Transmission
Clock Synchronization	Not required	Required
Data Sent	One byte or character at a time	Continuous block/stream
Start/Stop Bits	Used for each byte	Not needed per byte; sync is maintained overall
Transmission Speed	Slower (extra overhead)	Faster (less overhead)
Overhead	High (extra bits per character)	Low (sync overhead is less frequent)
Efficiency	Low for large data	High for large data
Error Detection	Simple (parity bits)	Advanced (CRC, checksums)
Examples	RS-232, keyboard, UART	Ethernet, USB, DSL

Advantages and Disadvantages

Asynchronous Transmission

Advantages	Disadvantages
Simple and cost-effective	Less efficient due to start/stop bits
Suitable for low-speed systems	Not suitable for high-speed systems
Ideal for infrequent transmissions	More error-prone for long sequences

Synchronous Transmission

Advantages	Disadvantages
Efficient for bulk data transfer	More complex (needs clock/sync mechanisms)
High-speed and reliable	Requires frame synchronization
Lower overhead per byte	Costlier hardware/software

Applications

➤ Asynchronous Transmission:

- Serial communication (RS-232)
- Keyboard/mouse input
- Microcontroller to sensor communication (UART)

➤ Synchronous Transmission:

- Computer networks (Ethernet)
- Digital telephony systems
- USB, SD cards
- DSL broadband

Real-World Analogy

Type	Analogy
Asynchronous	Sending separate letters, each with a salutation and signature
Synchronous	Having a phone call—both parties talk in sync continuously

Types of Errors

Introduction

In any data communication system, **errors** may occur due to various factors like noise, attenuation, distortion, or interference. These errors can **alter the original data**, leading to communication failures if not detected and corrected.

What is an Error?

Definition:

An **error** in data communication refers to the **corruption of bits** during transmission such that the **received data differs** from the **transmitted data**.

Classification of Errors

Errors can be broadly classified into:

Single-Bit Error

- **Definition**: Only **one bit** of the data unit is altered (flipped from 0 to 1 or vice versa).
- **Example**: Sent: `10010010` → Received: `10010000`
- **Causes**: Electrical noise, cross-talk
- **Easy to detect and correct**

Burst Error

- **Definition**: **Two or more bits** in a sequence are corrupted.
- **Length**: From first corrupted bit to the last corrupted bit.
- **Example**: Sent: `11010101` → Received: `11100101`
- **Causes**: Line disconnections, faulty hardware, signal fading
- **Harder to detect and correct**

Types of Errors Based on Cause

Type of Error	Description
Noise-Induced Error	Occurs due to electrical interference or random noise in the channel.
Attenuation Error	Signal weakens over distance, possibly misinterpreted.
Synchronization Error	Timing mismatch causes incorrect interpretation of bit streams.
Cross-talk Error	Signal from one channel interferes with another.

Examples

Error Type	Sent Data	Received Data	Description
Single-bit	`10110001`	`10110011`	Only 1 bit changed
Burst error	`11001010`	`11100100`	Multiple bits affected in a group

Detection and Correction

To ensure reliability, **error detection** and **error correction** mechanisms are used:

Error Detection Techniques:

- **Parity Bit**
- **Checksum**
- **Cyclic Redundancy Check (CRC)**

Error Correction Techniques:

- **Hamming Code**
- **Reed-Solomon Code**
- **Forward Error Correction (FEC)**

Comparison: Single-bit vs Burst Error

Feature	Single-bit Error	Burst Error
Affected Bits	Only one	Two or more consecutive bits
Frequency	Less common	More common in real scenarios
Detection/Correction	Easier	More complex
Main Cause	Short noise spikes	Channel imperfections, longer noise

Causes of Errors

Cause	Description
Thermal Noise	Random electrical disturbances in the transmission medium
Impulse Noise	Sudden spikes (e.g., from lightning or motors)
Crosstalk	Overlapping signals from adjacent lines
Distortion	Signal changes shape due to non-uniform transmission
Attenuation	Signal weakens over distance

Error Rate

Bit Error Rate (BER):

The **number of bit errors per unit time or per number of bits transmitted**.

BER=Number of ErrorsTotal Bits Sent\text{BER} = \frac{\text{Number of Errors}}{\text{Total Bits Sent}}BER=Total Bits SentNumber of Errors

Lower BER = More reliable transmission.

Error Detection and Error Correction

Introduction

During digital data transmission, **errors can occur** due to noise, interference, or signal degradation. To ensure **data integrity**, systems use **error detection and correction** methods.

What is Error Detection?

Definition:

Error Detection is the process of **identifying** whether transmitted data has been **corrupted** or altered during transmission.

- It does **not correct** the error, only identifies it.
- Used when **retransmission is possible**.

What is Error Correction?

Definition:

Error Correction not only detects the presence of an error but also **locates and corrects** it **without needing retransmission**.

- Used in **critical systems** where retransmission is not feasible (e.g., satellite communication).

Types of Error Handling Techniques

Error Detection Techniques

These methods help the receiver **identify** if the received data is incorrect.

Technique	Description
Parity Bit	Adds an extra bit to make the number of 1s even (even parity) or odd (odd parity)
Checksum	Adds all data segments, sends the sum (checksum) with the data
CRC (Cyclic Redundancy Check)	Uses polynomial division to detect burst errors
Hamming Distance	Measures difference between bit patterns; used to design detection/correction codes

---

Error Correction Techniques

These allow the receiver to **fix** errors without needing retransmission.

Technique	Description
Hamming Code	Can detect and correct single-bit errors
Reed-Solomon Code	Used for correcting burst errors (e.g., in CDs, DVDs, QR codes)
Forward Error Correction (FEC)	Adds redundant data to help correct errors at receiver
Automatic Repeat reQuest (ARQ)	Requests retransmission of corrupted frames (uses ACK/NACK)

Parity Bit – Example

➤ **Even Parity:**

If data = 1010001 → Number of 1s = 3 → Add 1 more 1 to make it even → 10100011

➤ **Odd Parity:**

If data = 1010001 → Number of 1s = 3 → Already odd → Add 0 → 10100010

Detects **single-bit errors**, but not all burst errors.

Checksum – Example

1. Divide data into segments (e.g., 8 bits each).
2. Add all segments using 1's complement arithmetic.
3. Send the **sum (checksum)** along with data.
4. Receiver adds all segments + checksum → If result is all 1s, data is correct.

CRC (Cyclic Redundancy Check)

How it Works:

1. Sender and receiver agree on a **divisor polynomial**.
2. Sender performs binary division and appends the **remainder (CRC bits)**.
3. Receiver performs the same division; if remainder = 0 → No error.

Very effective in detecting **burst errors**.

Hamming Code – Error Correction

Works by:

- Adding **redundant bits** at power-of-2 positions.
- These bits help determine the exact position of an error.
- Can **detect and correct single-bit errors**.

Example:

Data: 1011 → Encoded as: r1 r2 1 r3 0 1 1 → r1, r2, r3 are parity bits calculated based on positions.

Comparison: Detection vs Correction

Feature	Error Detection	Error Correction
Purpose	Detect errors in data	Detect and fix errors
Accuracy	Can detect single/burst errors	Can correct single/multiple errors (depending on code)
Complexity	Simple to implement	More complex
Retransmission Needed?	Yes (if error is found)	No (if correctable)
Used In	Ethernet, Internet protocols	Satellite, storage media, CDs

---

Automatic Repeat reQuest (ARQ)

Used for **reliable data transfer**, combines detection with retransmission.

ARQ Type	Description
Stop-and-Wait ARQ	Sender waits for acknowledgment after every frame
Go-Back-N ARQ	Sender sends multiple frames but retransmits from error
Selective Repeat ARQ	Only the erroneous frame is resent

Applications

- **Parity/Checksum**: TCP/IP, Serial data links
- **CRC**: Ethernet, USB, modems, hard disks
- **Hamming Code**: RAM memory, embedded systems
- **Reed-Solomon**: CDs, DVDs, QR codes, satellite comms
- **FEC**: Space missions, VoIP, video streaming

Line Configurations

Introduction

In data communication, **line configuration** (also called **connection configuration**) refers to the **physical or logical layout** that connects **two or more devices** in a communication network.

Types of Line Configurations

There are **two primary types** of line configurations:

A. Point-to-Point Configuration

B. Multipoint Configuration

Point-to-Point Line Configuration

Definition:

A **point-to-point** connection provides a **dedicated link** between **two devices**.

Characteristics:

- One **sender** and one **receiver**.
- The entire capacity of the channel is **reserved** for the two devices.
- Most **simple** and **reliable** configuration.

Diagram:

```
css
CopyEdit
Device A  ------------------  Device B
   <-------- Link -------->
```

Examples:

- USB connection between PC and printer
- Serial cable between two computers
- Telephone communication

Advantages:

- High data transmission speed
- No sharing or interference
- Easier error detection

Disadvantages:

- Wastes bandwidth if the link is idle
- Expensive for large-scale deployment

Multipoint Line Configuration

Definition:

A **multipoint** (or **multidrop**) connection is a link **shared by multiple devices**.

Characteristics:

- One link is shared by **two or more devices**.
- Devices must **coordinate** for data transmission.
- Can be either **centralized** or **distributed**.

Diagram:

```
lua
CopyEdit
Device A ----|
         |------ Channel ------|
Device B ----|                 |---- Device C
                          .
                          .
```

Types of Multipoint Configurations:

1. **Centralized** – One primary device, others are secondary.
2. **Distributed** – All devices equal, coordinate among themselves.

Examples:

- Mainframe with multiple terminals
- Classroom LAN
- Conference calls over a single frequency

Advantages:

- **Cost-effective** (fewer lines and ports)
- **Scalable**

Disadvantages:

- **Slower** (since bandwidth is shared)
- More **complex control logic**
- Prone to **data collision**

Comparison Table: Point-to-Point vs Multipoint

Feature	Point-to-Point	Multipoint
Number of Devices	Only 2 devices	More than 2 devices
Link Usage	Dedicated	Shared
Performance	Fast and efficient	Slower (shared bandwidth)
Cost	Higher (more links needed)	Lower (fewer cables/devices)
Error Control	Simple	Complex (must manage multiple devices)
Example	USB cable, modem	LAN, mainframe terminals

Use Cases and Applications

Line Configuration	Use Cases
Point-to-Point	Direct printer connection, Remote login (SSH), DSL
Multipoint	LANs, Bus networks, Classroom computers, IoT networks

Real-World Analogy

Configuration	Analogy
Point-to-Point	A **private phone call** between two people
Multipoint	A **group conference call** with multiple participants

Key Takeaways

- **Point-to-point** is simple and fast but not scalable.
- **Multipoint** is efficient and scalable but more complex.
- The choice depends on **cost**, **number of devices**, and **required performance**.