The Transport Layer

The **Transport Layer** is the **4th layer** of the OSI Model and plays a critical role in **end-to-end communication between devices**. It ensures that data is delivered **reliably, efficiently, and in the correct order** from a source application to a destination application.

1. Main Purpose of the Transport Layer

The Transport Layer acts as a **bridge between the Application Layer and the Network Layer**. Its job is to:

- Deliver data **from one process to another (process-to-process communication)**
- Ensure **reliable or fast delivery depending on protocol**
- Manage **data flow and errors**
- Break large data into smaller segments and reassemble them

2. Key Functions of the Transport Layer

a) Segmentation and Reassembly

- Large data from applications is divided into smaller chunks called **segments**
- Each segment is sent separately
- At the destination, segments are **reassembled in the correct order**

b) Service-Point Addressing (Port Numbers)

- Transport layer uses **port numbers** to identify applications

Example:

- Web server → Port 80 (HTTP)
- Email → Port 25 (SMTP)

So instead of just sending data to a device, it sends to a **specific application on that device.**

c) Multiplexing and Demultiplexing

- **Multiplexing** → Multiple applications send data through one network connection
- **Demultiplexing** → Data is delivered to the correct application at the receiver

d) Flow Control

- Prevents a fast sender from overwhelming a slow receiver
- Uses techniques like:
    - Sliding window

e) Error Control

- Detects and corrects errors
- Ensures:
    - No data loss
    - No duplication
    - Correct order

f) Congestion Control

- Controls traffic in the network to avoid overload
- Adjusts sending rate based on network conditions

3. Transport Layer Protocols

a) TCP (Transmission Control Protocol)

Transmission Control Protocol

Features:

- Connection-oriented
- Reliable (guarantees delivery)
- Error checking & correction
- Flow & congestion control

Used in:

- Web browsing (HTTP/HTTPS)
- Email
- File transfer

b) UDP (User Datagram Protocol)

User Datagram Protocol

Features:

- Connectionless
- Faster but **unreliable**
- No error recovery

Used in:

- Video streaming
- Online gaming
- Voice calls (VoIP)

4. TCP vs UDP

Feature	TCP	UDP
Connection	Yes	No
Reliability	High	Low
Speed	Slower	Faster
Error Checking	Yes	Minimal
Use Case	Web, Email, FTP	Streaming, Gaming

5. TCP Connection Process (3-Way Handshake)

Before data is sent using TCP, a connection must be established:

1. SYN → Client sends request
2. SYN-ACK → Server acknowledges
3. ACK → Client confirms

This is called the **3-way handshake**

6. Ports and Socket Concept

- A **socket = IP Address + Port Number**
- Example:
    - 192.168.1.1:80

This ensures communication goes to the **correct device AND correct application**

Transport Layer Addressing

In the **Transport Layer**, *addressing* means identifying the **exact application or service** on a device that should send or receive data—not just the device itself.

This is different from lower layers:

- Network Layer → identifies **devices** (IP address)
- Transport Layer → identifies **applications/processes** (port numbers)

1. Service-Point Addressing (Port Numbers)

The Transport Layer uses **port numbers** to ensure data is delivered to the **correct application**.

A **port number** is a **16-bit number** (range: 0–65535)

Categories of Port Numbers

Range	Type	Example
0 – 1023	Well-known ports	HTTP (80), FTP (21)
1024 – 49151	Registered ports	Apps like databases
49152 – 65535	Dynamic ports	Temporary client ports

2. Common Port Numbers

Application	Port	Protocol
HTTP	80	TCP
HTTPS	443	TCP
FTP	21	TCP
SMTP	25	TCP
DNS	53	TCP/UDP

---

3. Socket Address (Full Addressing)

A **complete Transport Layer address** is called a **socket**

Socket = **IP Address + Port Number**

Example:

- 192.168.1.10:80

This means:

- Device → 192.168.1.10
- Application → Web server (port 80)

4. How Addressing Works (Step-by-Step)

Sending Side:

1. Application generates data
2. Transport Layer adds **port number (source & destination)**
3. Data is passed to Network Layer

Receiving Side:

1. Transport Layer reads port number
2. Delivers data to correct application

5. Multiplexing & Demultiplexing Role

- **Multiplexing:** Many applications use different ports to send data simultaneously
- **Demultiplexing:** Incoming data is directed to the correct application using port numbers

6. TCP vs UDP in Addressing

Both:

- Transmission Control Protocol
- User Datagram Protocol

Use **port numbers**, but:

- TCP → reliable delivery
- UDP → fast delivery

Connection Establishment (Transport Layer)

Connection establishment is the process of creating a **reliable communication link** between two devices before actual data transfer begins. It is mainly associated with Transmission Control Protocol.

Unlike
User Datagram Protocol (which sends data immediately), TCP first **sets up a connection** to ensure reliability.

1. Why Connection Establishment is Needed

Before sending data, both sides must:

- Agree to communicate
- Synchronize sequence numbers

- Prepare buffers and resources

2. TCP 3-Way Handshake (Connection Establishment)

The process used is called the **3-way handshake**.

Step 1: SYN (Synchronize)

- Client sends a request to the server
- Indicates it wants to start communication

- Contains initial **sequence number**

Client → Server: **SYN**

Step 2: SYN-ACK (Synchronize + Acknowledge)

- Server responds with:
  - ACK (acknowledging client request)
  - Its own SYN (ready to communicate)

Server → Client: **SYN + ACK**

Step 3: ACK (Acknowledge)

- Client confirms the server's response

Client → Server: **ACK**

✅ Connection is now established, and data transfer begins

4. Key Features of Connection Establishment

a) Full-Duplex Communication

- Data can flow in both directions after connection setup

b) Sequence Number Synchronization

- Helps in:
  - Ordering data
  - Detecting lost packets

c) Reliability Setup

- Ensures both sides are ready before data starts flowing

5. Connection Establishment in UDP

For
User Datagram Protocol:

- No connection establishment
- No handshake

- Data is sent instantly

Faster but less reliable

Connection Release

Connection release in the transport layer is the process of terminating a logical connection between two endpoints, freeing resources like memory, buffers, and bandwidth. It uses orderly (four-way handshake/FIN) or abrupt (RST) methods to ensure data integrity. Synonyms include TCP connection termination, teardown, or closing a connection.

Key Aspects of Connection Release

i. **Orderly Release (Graceful):** Each direction of the full-duplex connection is closed independently. The standard TCP technique uses a four-step process where both sides send a FIN (Finish) packet and receive an ACK (Acknowledgment).

ii. **Abrupt Release (Abortive):** A connection is broken immediately using a RST (Reset) segment, often due to error conditions or user-initiated force-close. Unsent data may be discarded.

iii. Usage Examples:

iv. **Graceful Close:** When a web browser finishes downloading a page, it sends a `FIN` to the server, resulting in a graceful shutdown.

v. **Abrupt Termination:** If a server crashes or encounters a critical error, it sends an `RST` to immediately stop the connection.

vi. **Resource Management:** If a client becomes unreachable, the server times out and releases the connection.

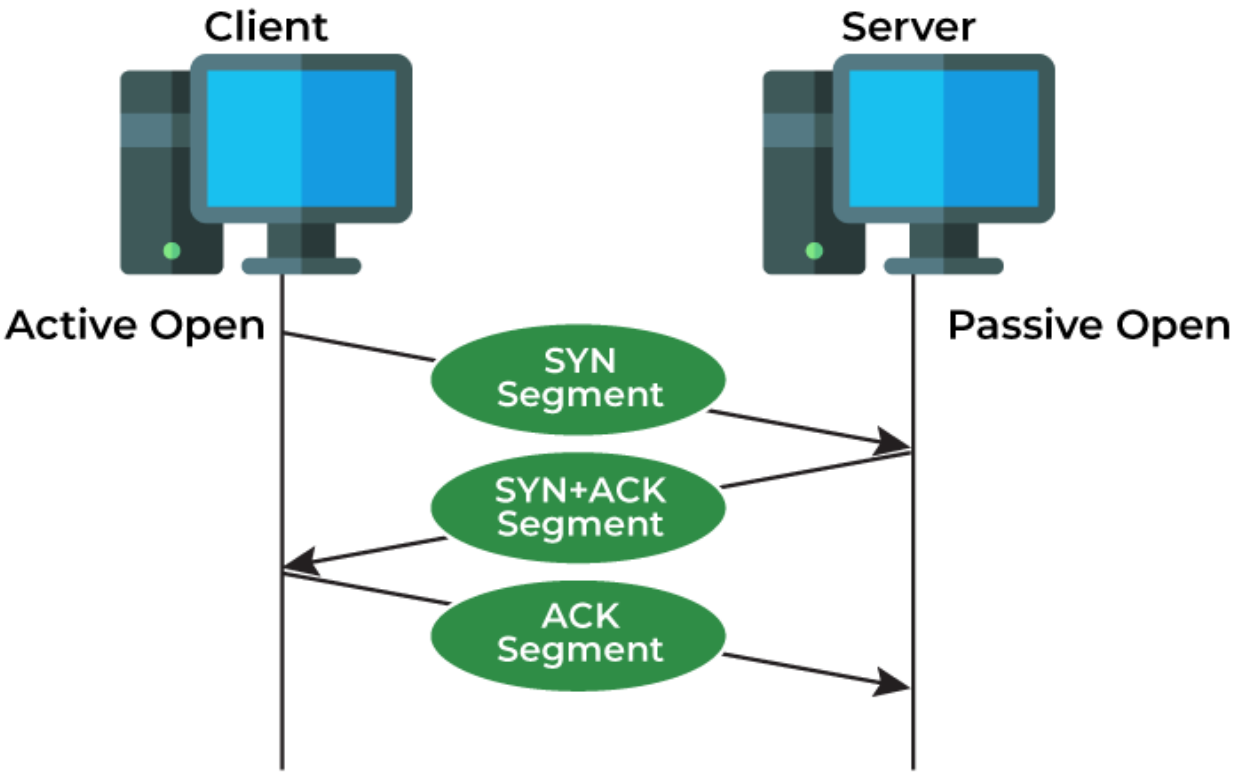
Steps in TCP Connection Termination

1. **FIN (Client):** Initiator sends a FIN segment, closing its sending side.
2. **ACK (Server):** Receiver sends an ACK, acknowledging the closure.
3. **FIN (Server):** Server sends its own FIN to close its side.
4. **ACK (Client):** Initiator sends the final ACK, closing the connection.

TCP AND UDP LAYERS

TCP is a Layer 4 protocol that ensures reliable, ordered, and error-checked data delivery between devices. It establishes connections using a three-way handshake and guarantees that all packets arrive correctly.

- Uses a three-way handshake to establish connections
- Breaks data into packets and reassembles them in order
- Retransmits lost or corrupted packets automatically
- Provides reliable, error-checked communication
- Commonly used for web browsing, email, and file transfers



Use Cases of TCP Protocol

TCP **(**Transmission Control Protocol**)** is one of the main parts of the internet which provides reliable and ordered data delivery. Here are some of its key use cases:

1. Web Browsing

- When you type a URL into your browser, your computer uses TCP to establish a connection with the web server.
- TCP ensures that the HTML, CSS, and JavaScript files that make up the webpage are delivered accurately and in the correct order.

2. Email

- Protocols like SMTP (Simple Mail Transfer Protocol) and IMAP (Internet Message Access Protocol) rely on TCP for sending and receiving emails.
- TCP guarantees that your emails are delivered completely and in the correct sequence.

3. File Transfer

- Protocols like FTP (File Transfer Protocol) and SFTP (Secure File Transfer Protocol) utilize TCP for transferring files between computers.
- TCP's reliability ensures that files are transferred accurately and without data corruption.

4. Remote Access

- Protocols like Telnet and SSH (Secure Shell) use TCP for remote access to other computers.
- TCP ensures that commands and data are transmitted reliably, allowing you to interact with remote systems securely.

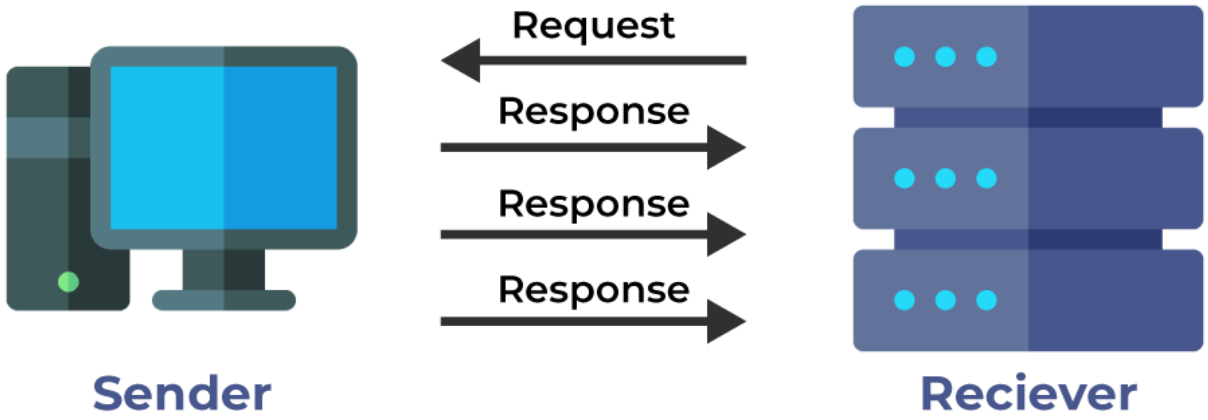
5. Online Banking and Financial Transactions

- TCP's reliability and security are crucial for online banking and financial transactions.
- It ensures that sensitive data is transmitted securely and accurately, preventing data loss or corruption.

What is User Datagram Protocol (UDP)?

User Datagram Protocol (UDP) is a Layer 4 protocol that sends data as independent datagrams without establishing a connection, offering no guarantees for delivery, order, or error correction. Its low overhead and minimal delay make it ideal for real-time applications where speed matters more than reliability.

- Connectionless communication (no handshake)
- Sends datagrams without delivery or ordering guarantees
- Very low latency and overhead
- No retransmission or error correction
- Used for streaming, online gaming, and VoIP



Use Cases of the UDP Protocol

UDP (User Datagram Protocol) is a connectionless protocol that prioritizes speed and efficiency over reliability. Here are some key use cases of the UDP:

1. Real-time Applications

- **Voice over IP (VoIP):** Services like Skype, Zoom, and Google Meet often utilize UDP for real-time voice and video communication. While some packet loss is acceptable, minimizing latency is crucial for a smooth conversation.
- **Online Gaming:** Many online games rely on UDP for fast, low-latency communication between players and game servers. This ensures responsiveness and prevents gameplay delays.
- **Video Conferencing:** Similar to VoIP, UDP is used for real-time video conferencing applications where timely delivery of video and audio streams is essential.

2. Streaming Media

- **Live Streaming:** Services like Twitch, YouTube Live, and Netflix use UDP for streaming audio and video content. While some packet loss is acceptable, UDP's speed and efficiency are crucial for delivering a smooth streaming experience.

3. Network Management Protocols

- **DNS (Domain Name System):** UDP is commonly used for DNS lookups, where quick responses are essential for resolving domain names into IP addresses.
- **SNMP (Simple Network Management Protocol):** This protocol is used for monitoring and managing network devices. UDP's speed and efficiency make it suitable for collecting performance data from network devices.
- **DHCP (Dynamic Host Configuration Protocol):** UDP is used for dynamically assigning IP addresses to devices on a network.

4. Broadcast and Multicast

- **Broadcast Applications:** UDP is well-suited for broadcast applications where a single message needs to be sent to multiple recipients simultaneously, such as network discovery protocols.
- **Multicast Applications:** UDP is used for multicast applications where a message needs to be sent to a specific group of recipients efficiently.

How to Choose Between TCP and UDP?

1. On the Basis of Reliability vs. Speed:

- **TCP** provides reliable, ordered, and error-checked delivery of data which makes it ideal for applications where accuracy matters more than speed (e.g., web pages, emails, file transfers).
- **UDP** offers faster, connectionless transmission without guaranteeing delivery or order, suitable for real-time applications like video streaming, online gaming, or voice calls, where speed and low latency are more important than perfect accuracy.

2. On the Basis of Connection Overhead:

- **TCP** establishes and maintains a connection between sender and receiver, adding overhead but ensuring stable data flow and retransmissions if packets are lost.
- **UDP** does not establish a dedicated connection, reducing latency and overhead, but leaving error-handling to the application if needed.

3. On the Basis of Use Case:

- **TCP** is best when demanding correctness such as web browsing, file downloads, and secure data transfers.
- **UDP** is preferred when you need minimal delay and can tolerate some data loss such as live broadcasts, online gaming, and VoIP.

TCP Sliding Window

The **TCP sliding window** is a mechanism used by
Transmission Control Protocol
to control **how much data can be sent before receiving an acknowledgment (ACK)**.

It is mainly used for:

- **Flow control** (avoid overwhelming receiver)
- **Efficient data transmission** (send multiple segments at once)

1. Basic Idea of Sliding Window

Instead of sending one segment and waiting:

TCP allows sending **multiple segments continuously**
The "window" defines **how many bytes can be sent at a time**

2. How Sliding Window Works

Key Concept:

- Sender maintains a **window of allowed data**
- As ACKs arrive, the window **slides forward**

Example:

- Window size = 4 segments
- Sender sends: 1, 2, 3, 4

- Receiver sends ACK for 1,2

Window slides → Sender can now send: 5, 6

3. Important Terms

a) Window Size

- Amount of data (in bytes) sender can transmit without ACK

- Controlled by receiver

b) Acknowledgment (ACK)

- Confirms received data

- TCP uses **cumulative ACK**

ACK = 5 means “received up to 4”

c) Sequence Numbers

- Identify order of bytes

- Used to track which data is sent/received

d) Sender Window vs Receiver Window

- **Sender Window** → Data allowed to be sent
- **Receiver Window** → Buffer space available

4. Types of Sliding Window Behavior

i. Normal Operation

- Data flows smoothly
- Window slides continuously

ii. Window Full

- Sender must **wait for ACK** before sending more

iii. Zero Window

- Receiver buffer is full
- Sender stops transmission temporarily

5. Flow Control Using Sliding Window

- Prevents fast sender from overwhelming slow receiver
- Receiver advertises **window size** in TCP header

Dynamic adjustment based on receiver capacity

6. Advantages of Sliding Window

i. Efficient data transmission
ii. Reduces waiting time
iii. Improves throughput

iv. Supports reliable communication

Conversion of Analog to Digital (ADC) and Digital to Analog (DAC)

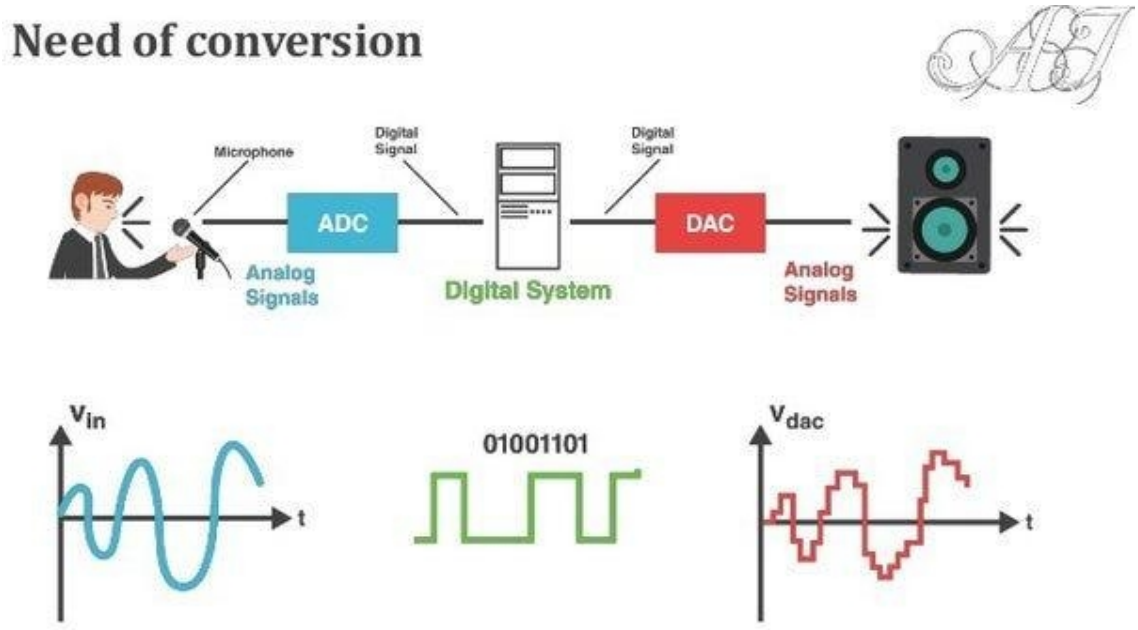
In communication systems, signals exist in two main forms:

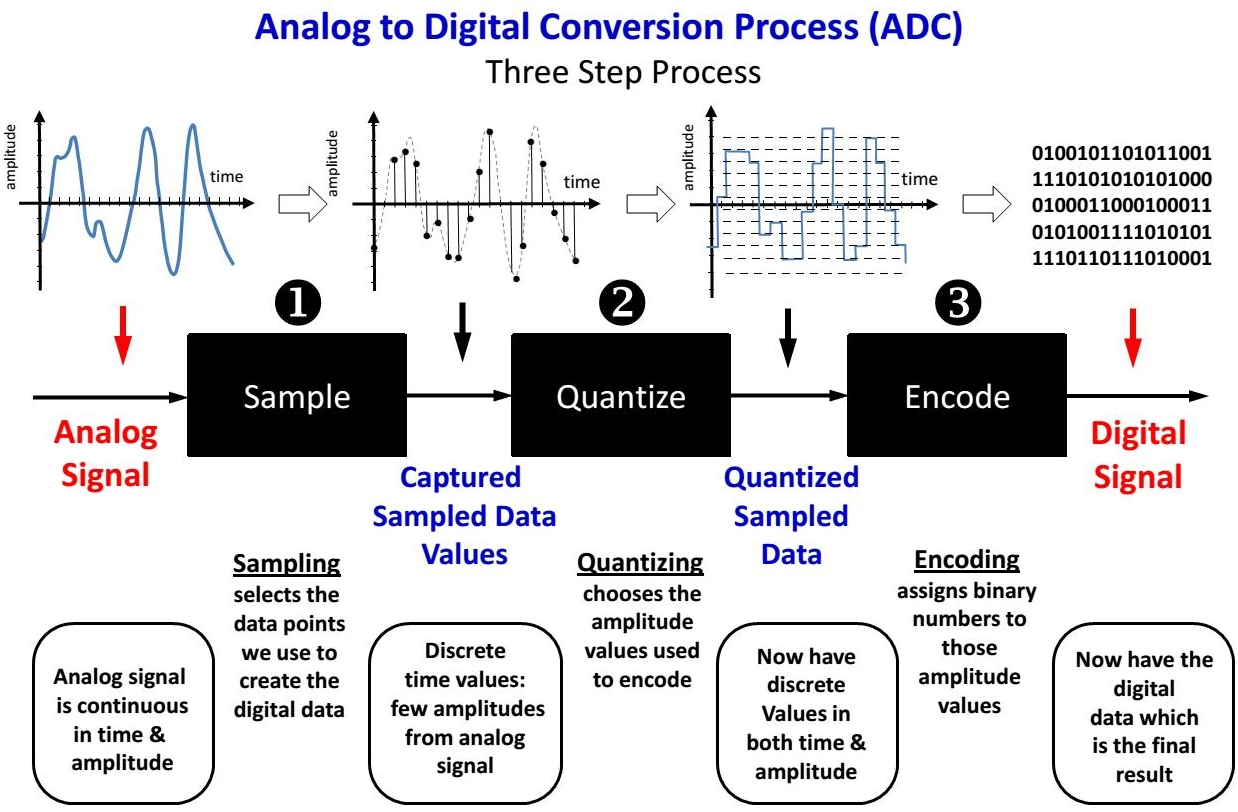
- **Analog signals** → Continuous (e.g., human voice)
- **Digital signals** → Discrete (0s and 1s used by computers)

To allow communication between real-world signals and digital systems, we use:

- Analog to Digital Converter
- Digital to Analog Converter

1. Analog to Digital Conversion (ADC)





Note: "Discrete time" corresponds to the timing of the sampling.

ADC converts **continuous analog signals into digital form (binary)**.

Steps in ADC Conversion

1. Sampling

- The analog signal is measured at **regular time intervals**
- Frequency of sampling must follow **Nyquist Theorem**:
    - Sampling rate ≥ 2 × highest signal frequency

Example: Voice signal sampled many times per second

2. Quantization

- Each sampled value is **rounded to the nearest level**
- Introduces **quantization error**

More levels = better accuracy

3. Encoding

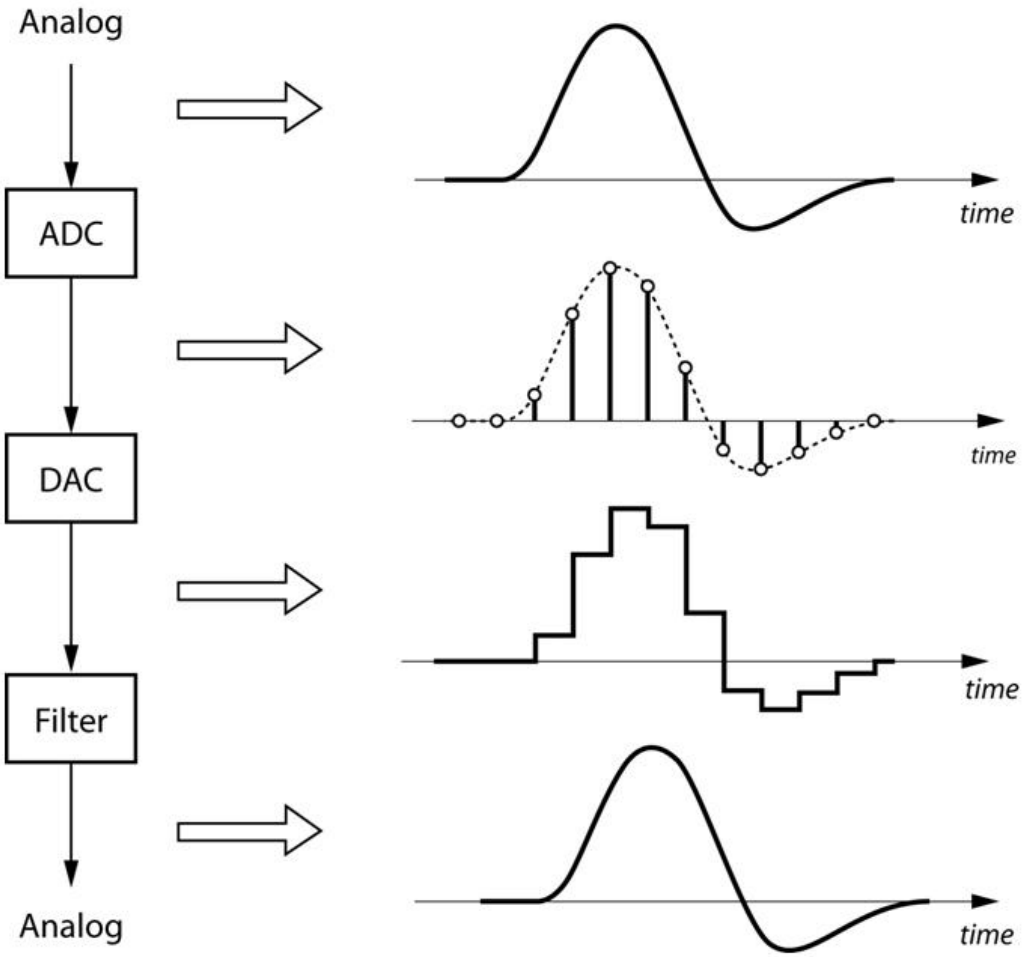
- Quantized values are converted into **binary numbers (0s and 1s)**

Output becomes digital data

Example of ADC

- Microphone converts sound → electrical signal → ADC → digital audio file

2. Digital to Analog Conversion (DAC)



DAC converts digital data back into analog signals.

Steps in DAC Conversion

1. Digital Input

- Binary data (0s and 1s) is received

2. Conversion to Staircase Signal

- Each binary value is converted into a **voltage level**
- Output looks like a **step (staircase) waveform**

3. Reconstruction (Filtering)

- A **low-pass filter** smooths the signal
- Produces a continuous analog waveform

Example of DAC

- Digital music file → DAC → speakers → sound waves

3. Differences between ADC and DAC

Feature	ADC	DAC
Function	Analog → Digital	Digital → Analog
Input	Continuous signal	Binary data
Output	Binary (0s and 1s)	Continuous signal
Process	Sampling, Quantization, Encoding	Reconstruction, Filtering
Example	Microphone	Speaker