## Data Communication

**1. Bits**

A **bit** (binary digit) is the **smallest unit of data** in a computer.

* It can have **only two values**:  
  **0 or 1**
* These represent **off/on**, **false/true**, or **low/high voltage**

Example:

1 bit → 0 or 1

8 bits → 1 byte

**2. Numbers (Binary, Decimal, Hex)**

Computers use **binary**, while humans use **decimal**.

| **System** | **Base** | **Example** |
| --- | --- | --- |
| Decimal | 10 | 25 |
| Binary | 2 | 11001 |
| Hexadecimal | 16 | 0x19 |

Example conversion:

Binary: 11001

Decimal: 25

Binary is used because electronic circuits reliably detect **two states**.

**3. ASCII**

**ASCII** (American Standard Code for Information Interchange) maps **characters to numbers**.

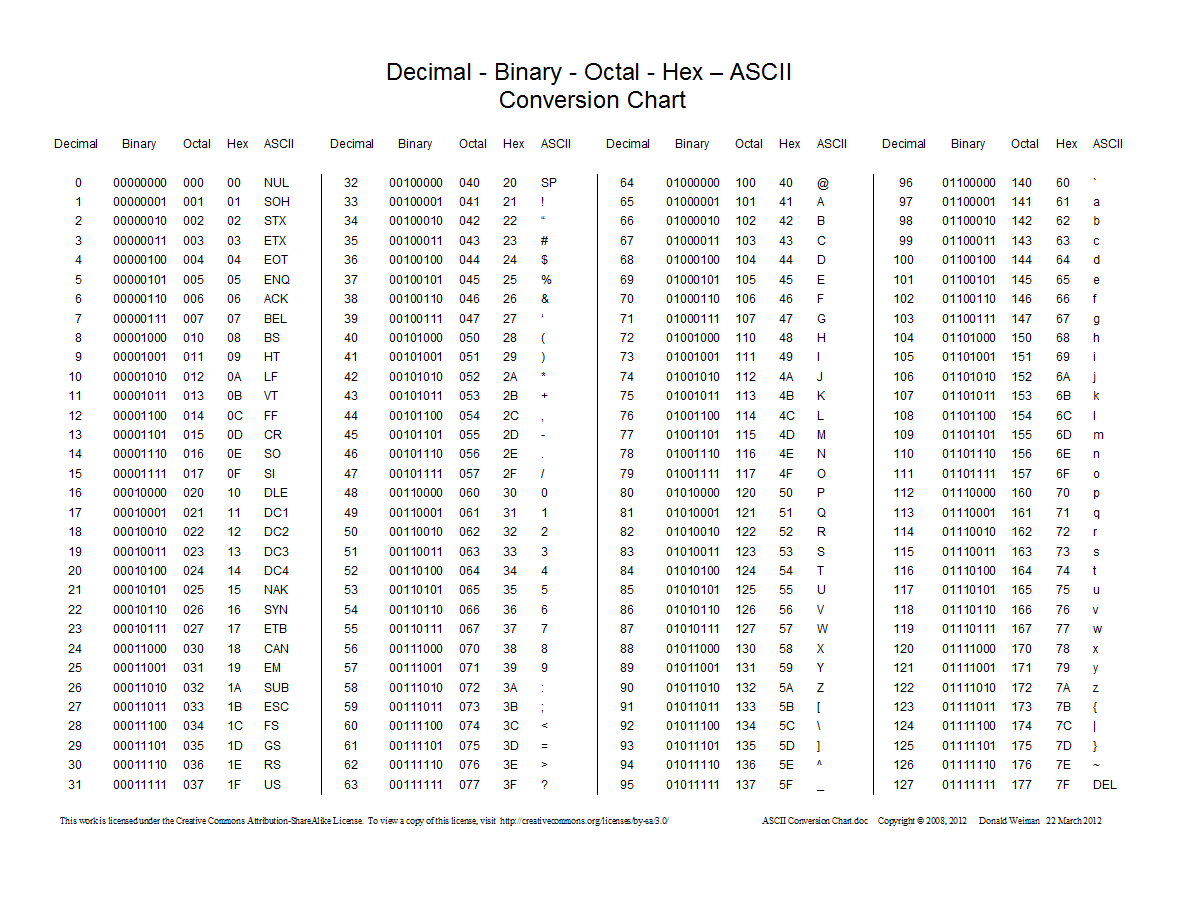
Example:

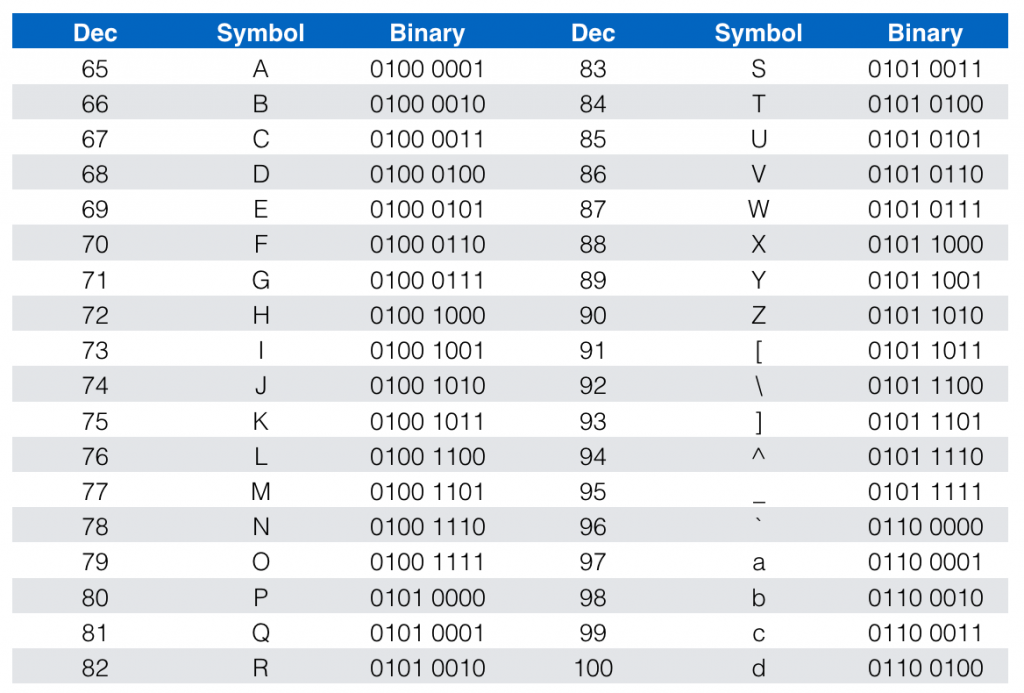
'A' → 65 → 01000001

'a' → 97 → 01100001

'1' → 49 → 00110001

* Uses **7 bits** (originally)
* Can represent **128 characters**
* Modern systems use **Unicode / UTF-8**, which extends ASCII



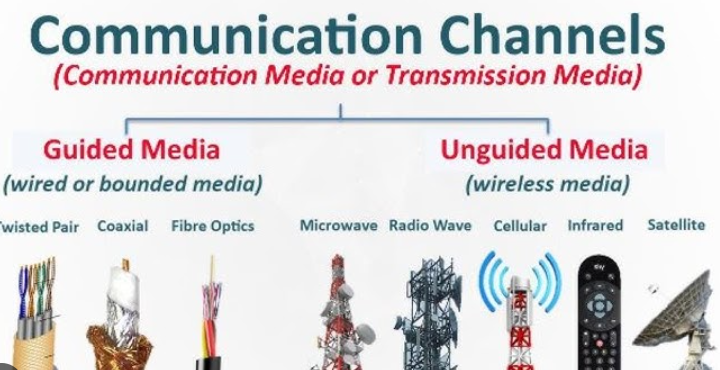


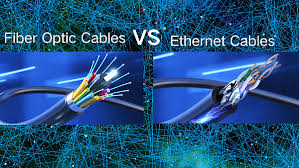
**4. Communication Mediums**

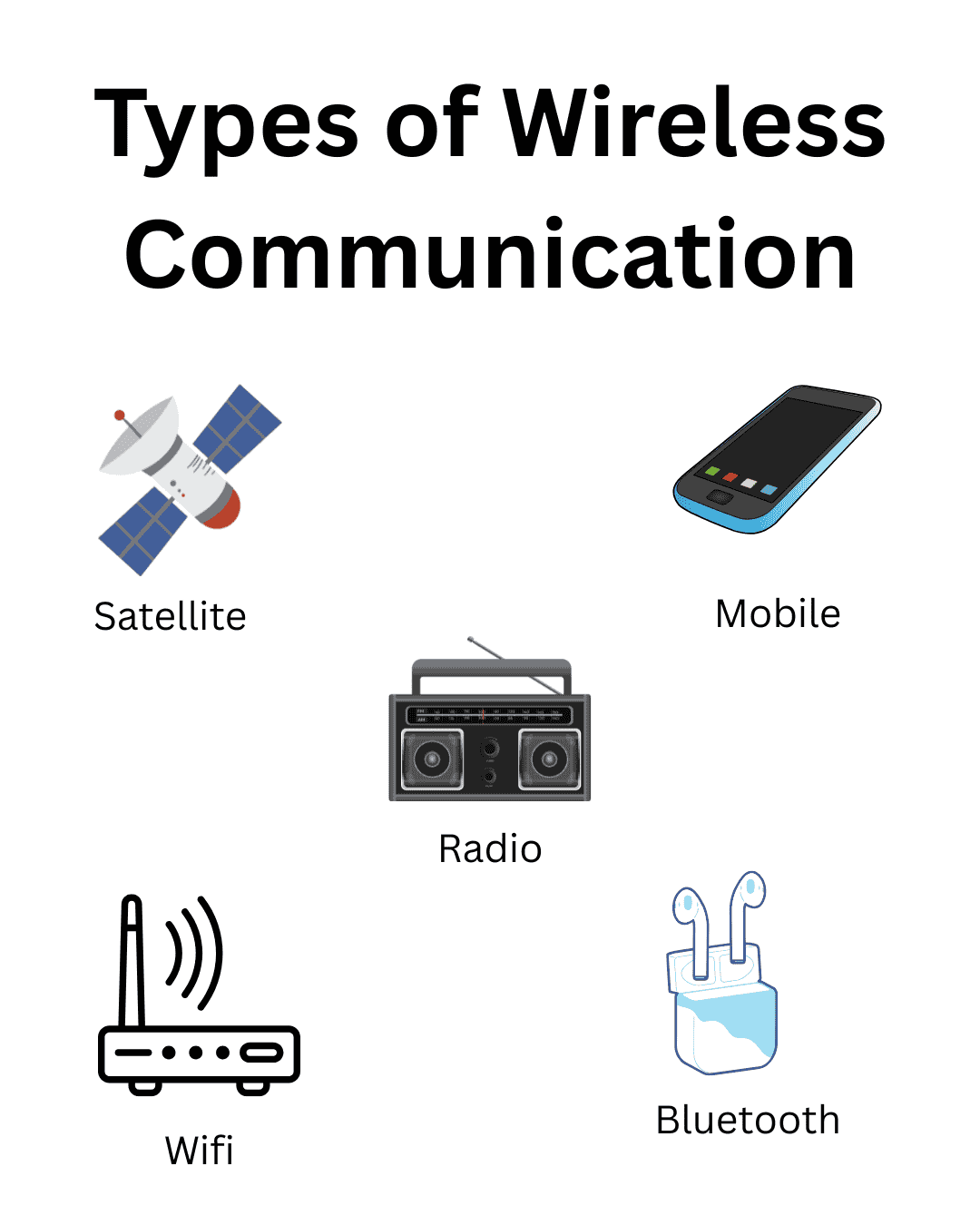
A **communication medium** is **how data travels** from sender to receiver.

**Types:**

1. **Wired**
   * Twisted Pair (Ethernet)
   * Coaxial cable
   * Fiber optic (light signals)
2. **Wireless**
   * Wi-Fi
   * Bluetooth
   * Cellular (4G / 5G)
   * Satellite







**5. Bitrate**

**Bitrate** = **how fast data is transmitted**

Measured in:

* bps (bits per second)
* Kbps, Mbps, Gbps

Example:

10 Mbps = 10 million bits per second

Higher bitrate → faster downloads & streaming  
Lower bitrate → buffering, lag

**6. Delay (Latency)**

**Delay** is the **time it takes for data to travel** from sender to receiver.

Measured in **milliseconds (ms)**.

Types of delay:

* **Propagation delay** – distance
* **Processing delay** – device handling
* **Queuing delay** – waiting in buffers
* **Transmission delay** – sending bits

Example:

Ping = 30 ms → very responsive

Ping = 300 ms → noticeable lag

**7. Packet Loss**

Data is sent in **packets**.  
**Packet loss** happens when packets **fail to reach their destination**.

Causes:

* Network congestion
* Weak wireless signal
* Faulty hardware

Effects:

* Video freezes
* Choppy audio
* Retransmissions → slower speed

Example:

2% packet loss → noticeable problems

**8. Protocols**

A **protocol** is a **set of rules** for communication.

They define:

* How data is formatted
* How it’s sent
* How errors are handled

Examples:

* HTTP – web pages
* FTP – file transfer
* SMTP – email
* DNS – name to IP lookup

Without protocols, devices would not understand each other.

**9. TCP/IP vs OSI Models**

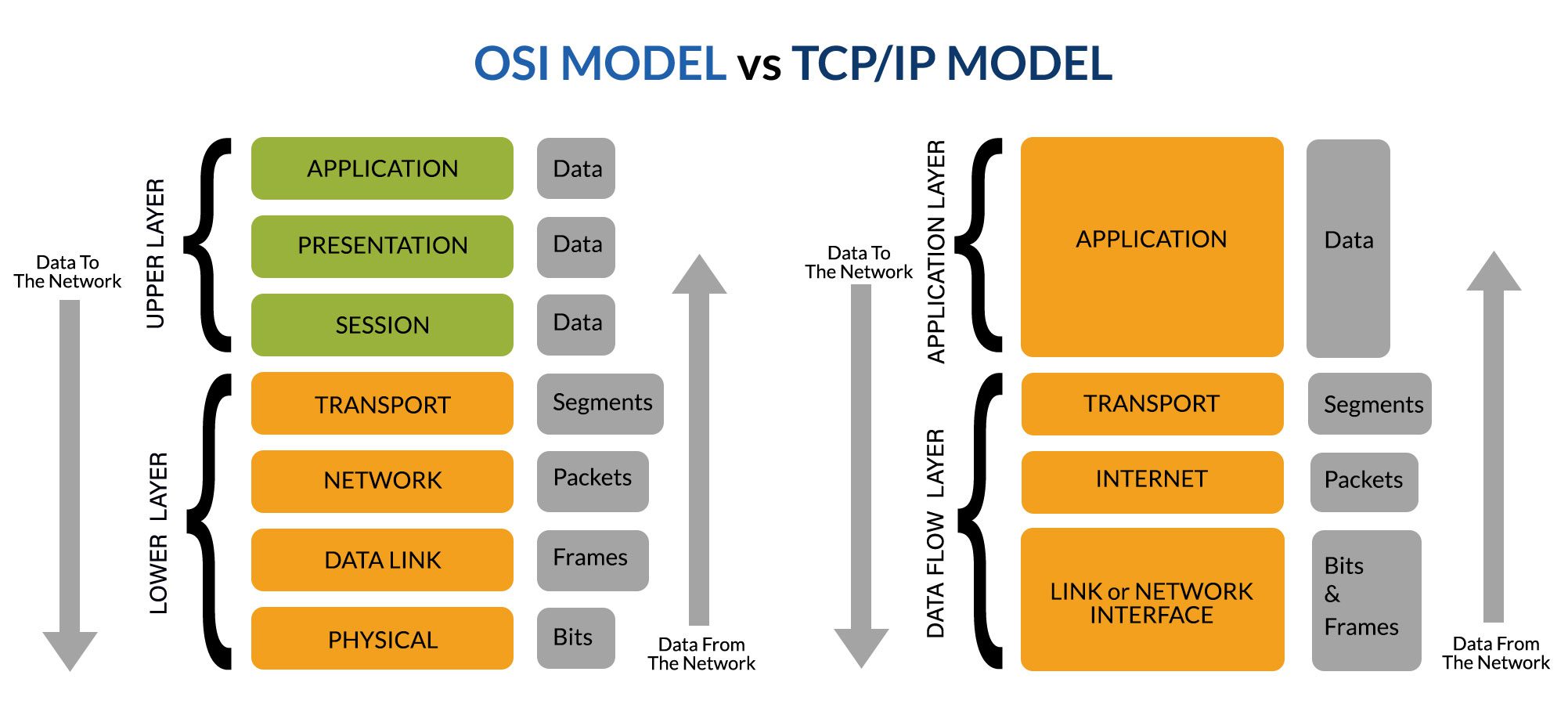
These are **network architecture models**.

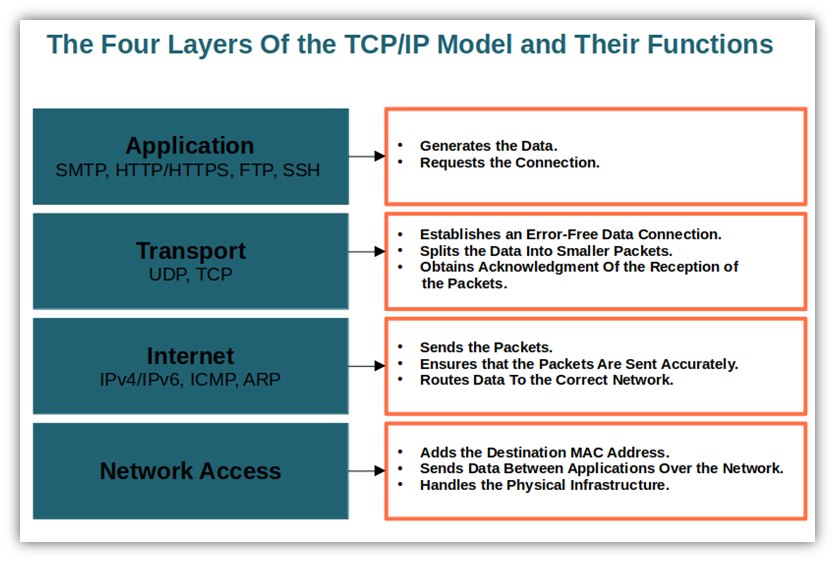
**OSI Model (7 Layers – theoretical)**

1. Application
2. Presentation
3. Session
4. Transport
5. Network
6. Datalink
7. Physical

**TCP/IP Model (4 Layers – practical)**

1. Application
2. Transport
3. Internet
4. Link or Network





**Key Differences:**

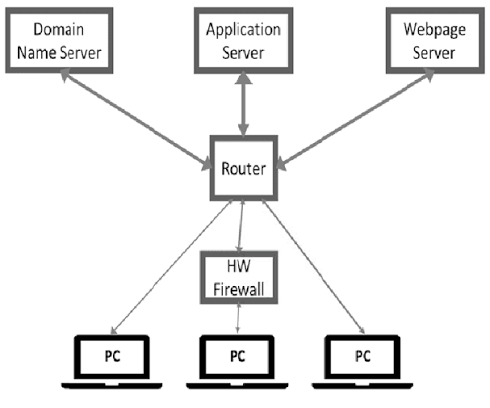
| **OSI** | **TCP/IP** |
| --- | --- |
| Conceptual | Practical |
| 7 layers | 4 layers |
| Teaching model | Real-world Internet |

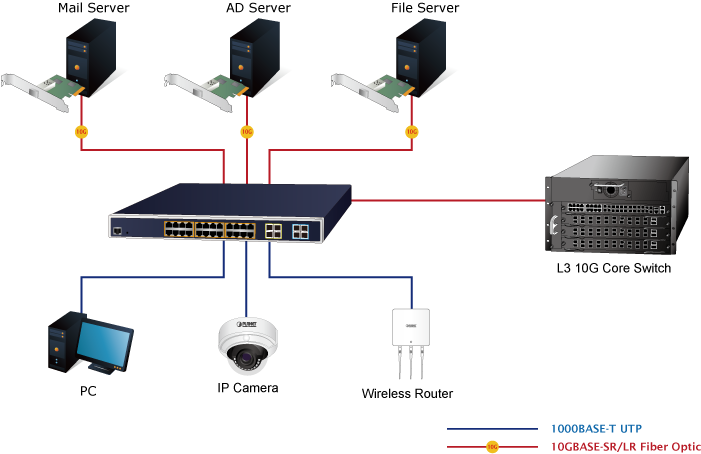
**10. Internet Components**

The Internet is made of **interconnected hardware and software**.

**Key Components:**

* **Clients** (PCs, phones)
* **Servers** (websites, cloud)
* **Routers** (direct traffic)
* **Switches** (local networks)
* **ISPs** (connect users)
* **Cables & wireless links**



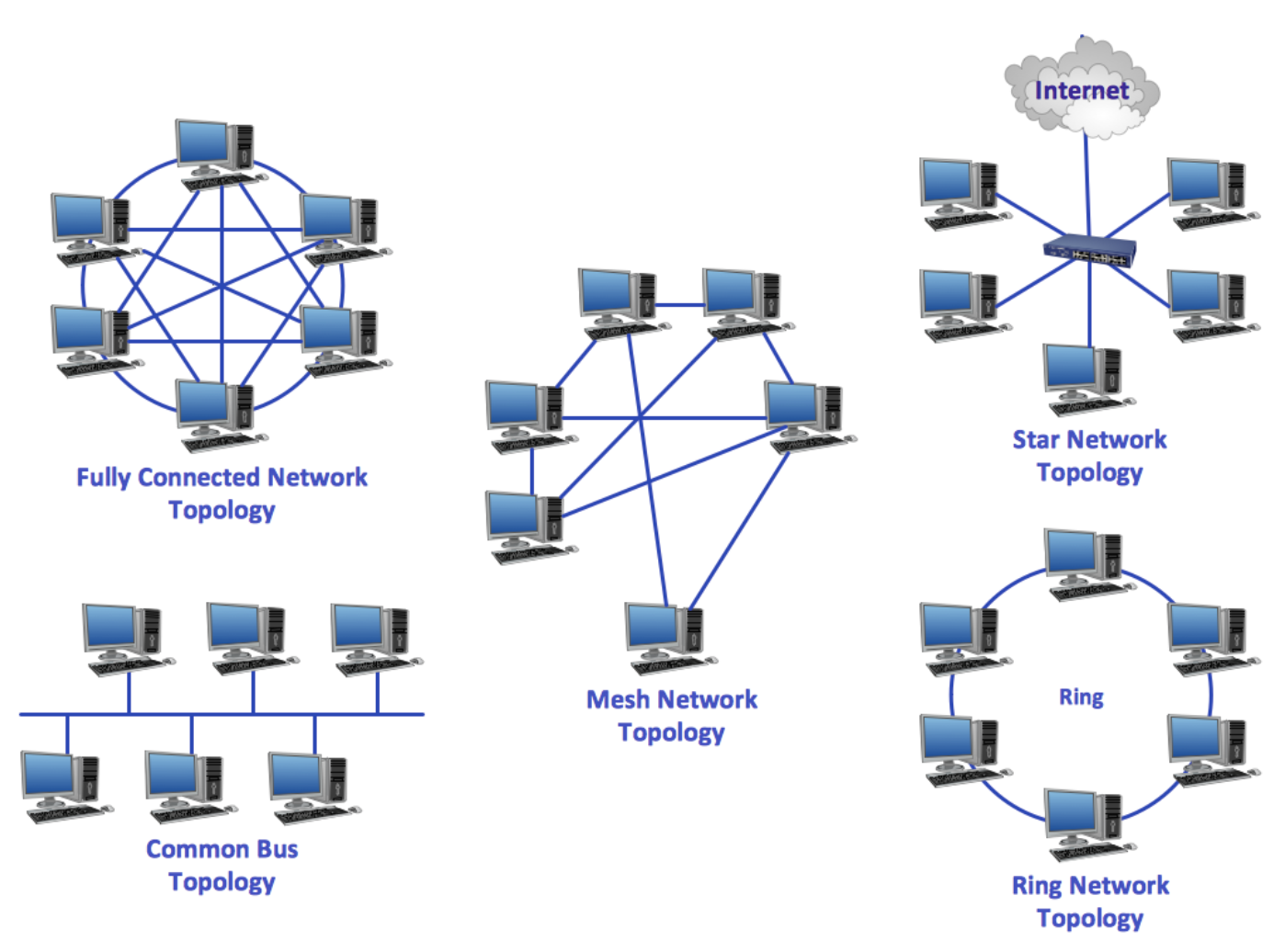


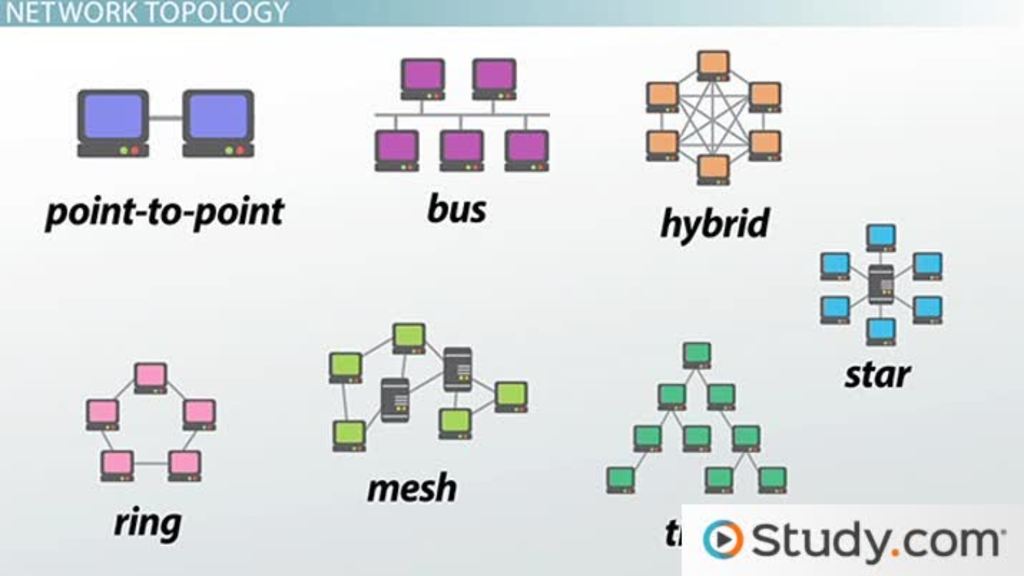
**11. Network Topologies**

A **topology** is how devices are arranged.

**Common Topologies:**

1. **Bus** – one main cable
2. **Star** – central hub/switch
3. **Ring** – circular path
4. **Mesh** – multiple paths
5. **Tree** – hierarchical





**Most modern networks:**

* **Star topology**
* **Mesh backbone** for redundancy

**How Everything Fits Together**

Bits → Numbers → Characters (ASCII)

→ Packets → Protocols

→ Mediums → Bitrate & Delay

→ Internet Components → Global Communication

## Application Layer

The **Application Layer** is the **topmost layer** of the network model and provides **network services directly to end-user applications**. It acts as the **interface between user software and the network**.

**Key functions:**

* Enables user applications to access network services
* Defines communication rules for applications
* Handles data formatting at the application level

**Common application-layer protocols:**

* **HTTP / HTTPS** – web browsing
* **FTP** – file transfer
* **SMTP / POP3 / IMAP** – email
* **DNS** – domain name resolution

In short, the Application Layer allows users to **send emails, browse websites, and transfer files over a network**.

**1. Client–Server Architecture**

**Client–server architecture** is a model where:

* **Clients** request services
* **Servers** provide services

**How it works**

1. Client sends a request (e.g., “Give me a webpage”)
2. Server processes it
3. Server sends back a response

Examples:

* Web browser → Web server
* Email app → Mail server
* Mobile app → Cloud backend

**Characteristics**

* Centralized control
* Easier maintenance
* Scalable (many clients, powerful servers)

**2. IP Addresses & Ports**

**IP Address**

An **IP address** uniquely identifies a device on a network.

Types:

* **IPv4** → 192.168.1.1
* **IPv6** → 2001:db8::1

Think of it as a **house address** on the Internet.

**Ports**

A **port number** identifies **which application/service** on a device is communicating.

Think of ports as **doors** in the house.

Common ports:

| **Service** | **Port** |
| --- | --- |
| HTTP | 80 |
| HTTPS | 443 |
| FTP | 21 |
| SMTP | 25 |
| DNS | 53 |

Example:

IP: 142.250.72.206

Port: 443

→ Google HTTPS server

**3. HTTP: Persistent vs Non-Persistent**

**HTTP** is the protocol used for web communication.

**Non-Persistent HTTP**

* **One TCP connection per object**
* Connection closed after each request

Example:

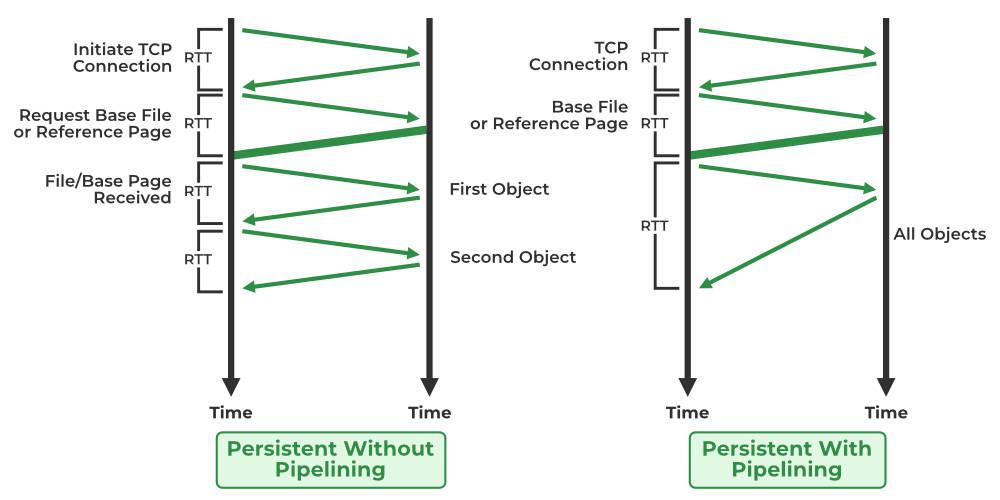
* HTML page
* Image
* CSS  
  → Each needs a new connection

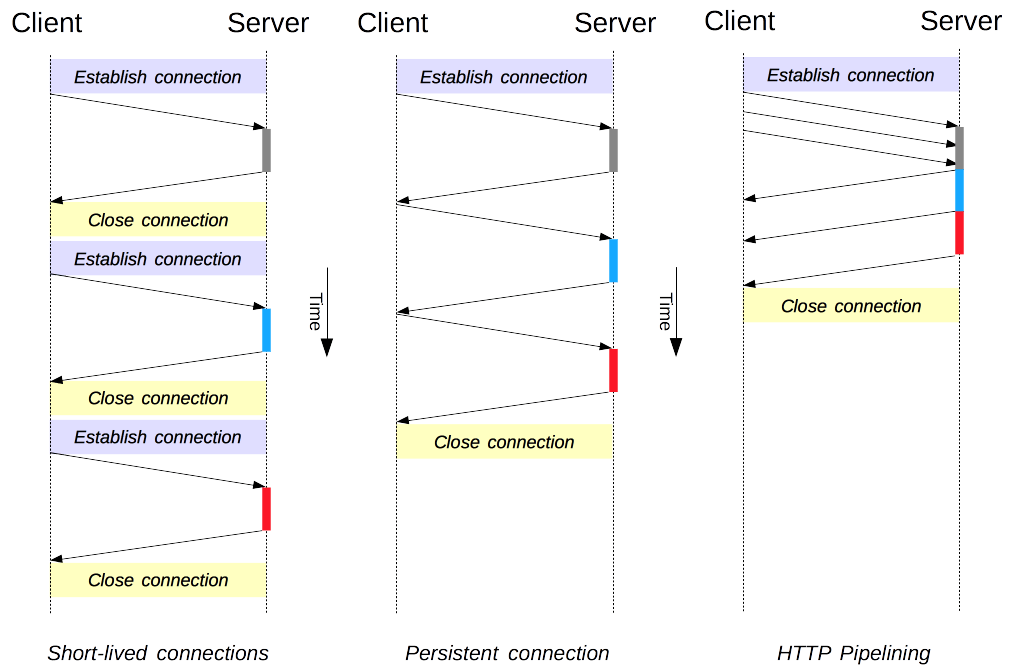
❌ Inefficient  
❌ High delay

**Persistent HTTP**

* **Single TCP connection reused**
* Multiple requests sent over same connection

✅ Faster  
✅ Less overhead  
✅ Used by modern browsers (HTTP/1.1+)





**4. Cookies & Web Caching**

**Cookies**

**Cookies** are small pieces of data stored on the client by a website.

Used for:

* Login sessions
* Shopping carts
* Personalization
* Tracking

How it works:

1. Server sends cookie
2. Browser stores it
3. Browser sends it back with future requests

Example:

Cookie: session\_id=AB123

**Web Caching**

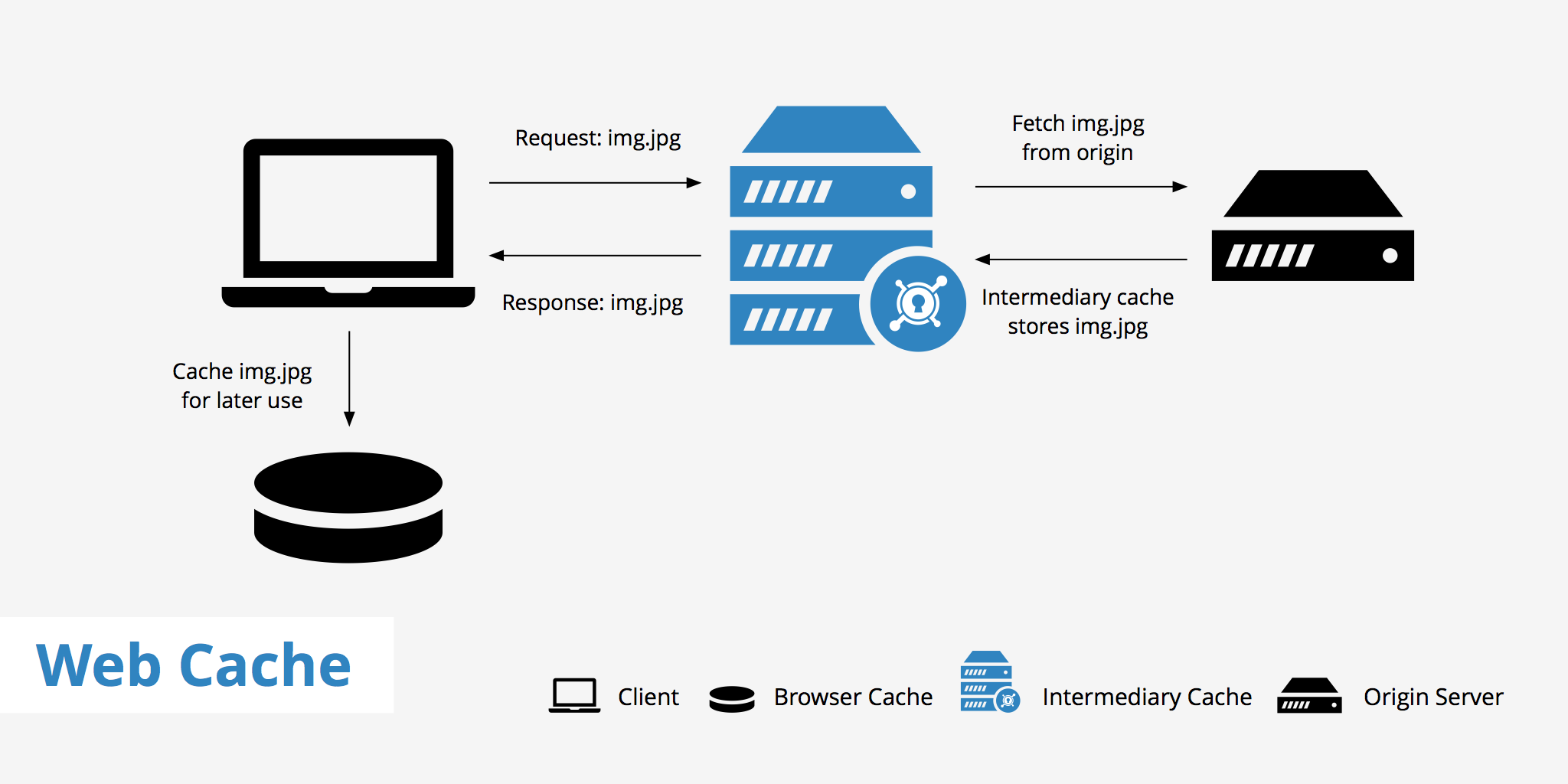
A **web cache** stores copies of frequently requested content.

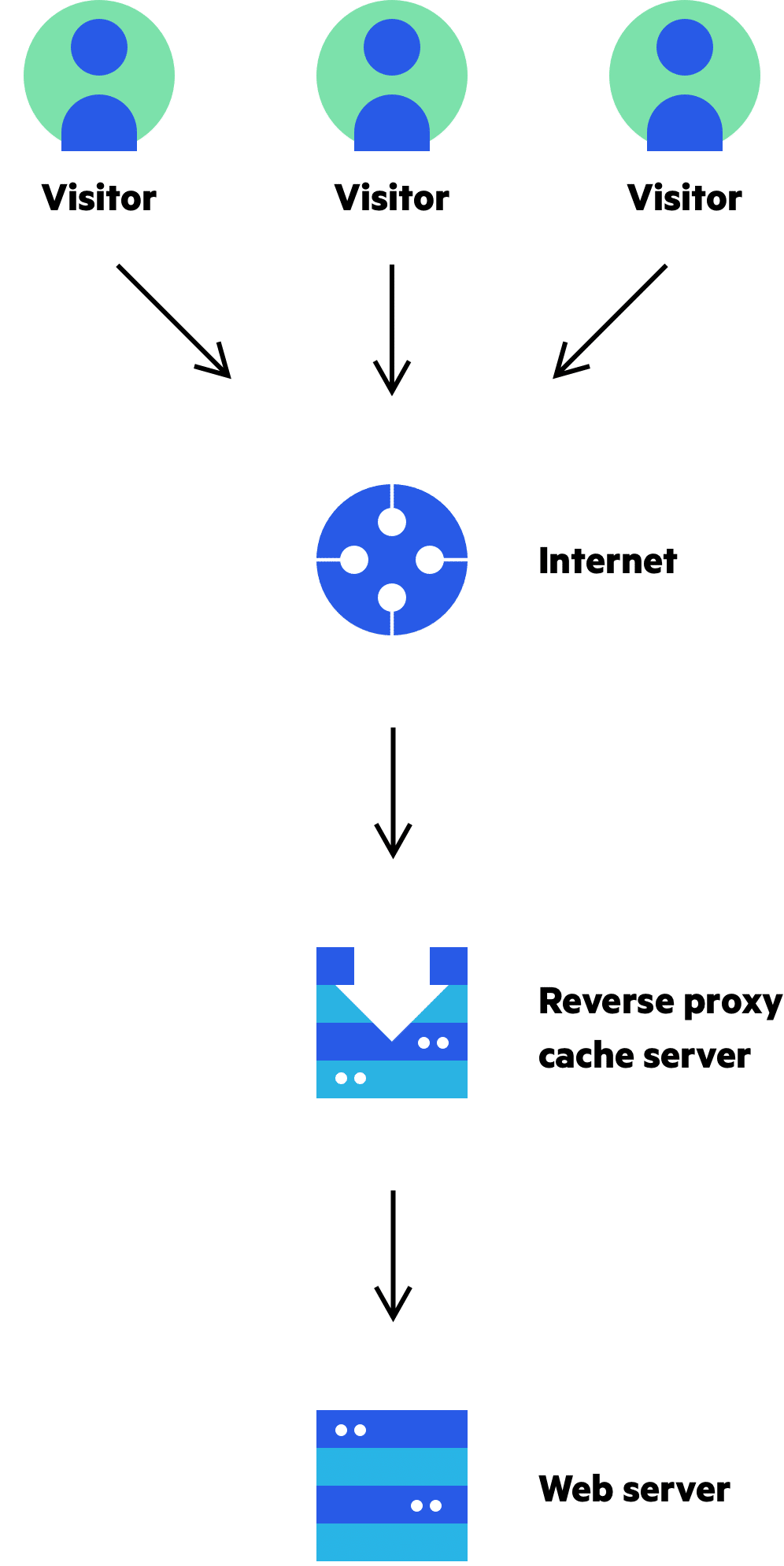
Types:

* Browser cache
* Proxy cache
* CDN cache

Benefits:

* Faster page loads
* Reduced server load
* Reduced network traffic





**5. DNS Hierarchy & Caching**

**DNS (Domain Name System)** translates:

www.example.com → 93.184.216.34

**DNS Hierarchy**

DNS is organized as a **tree structure**:

1. **Root servers**
2. **Top-Level Domain (TLD)** servers  
   (.com, .org, .edu)
3. **Authoritative name servers**
4. **Local DNS resolver**

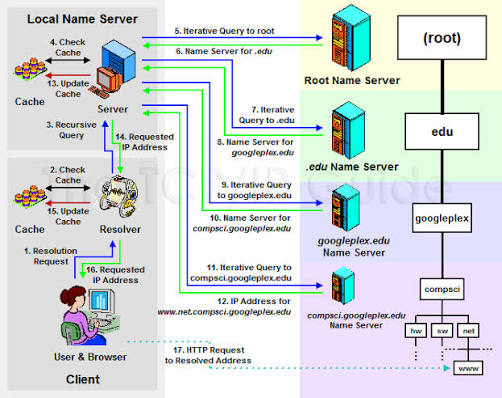
Query flow:

Client → Local DNS

→ Root → TLD → Authoritative

→ IP address returned





**DNS Caching**

To improve speed:

* DNS responses are cached at:
  + Browser
  + OS
  + Local DNS server

Each record has a **TTL (Time To Live)**.

Benefits:

* Faster lookups
* Less DNS traffic
* Reduced load on root servers

**One-Page Summary**

| **Topic** | **Key Idea** |
| --- | --- |
| Client–Server | Clients request, servers respond |
| IP Address | Identifies device |
| Port | Identifies service |
| HTTP Persistent | One connection, many requests |
| HTTP Non-Persistent | One connection per request |
| Cookies | Client-side state |
| Web Caching | Store content for speed |
| DNS Hierarchy | Distributed name system |
| DNS Caching | Faster name resolution |

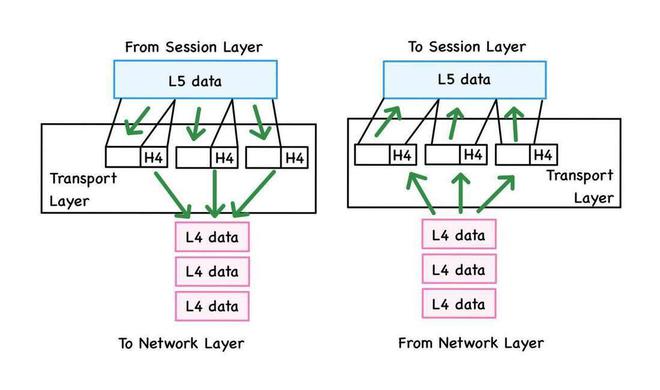
## Transport Layer

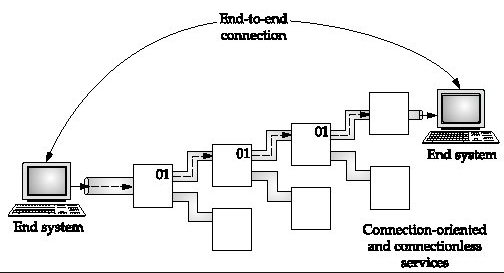
The **Transport Layer** provides **end-to-end communication** between applications running on different hosts.

**Main responsibilities:**

* Process-to-process delivery (using ports)
* Segmentation & reassembly
* Reliable or unreliable delivery
* Flow control
* Congestion control

Examples of transport protocols: **TCP** and **UDP**

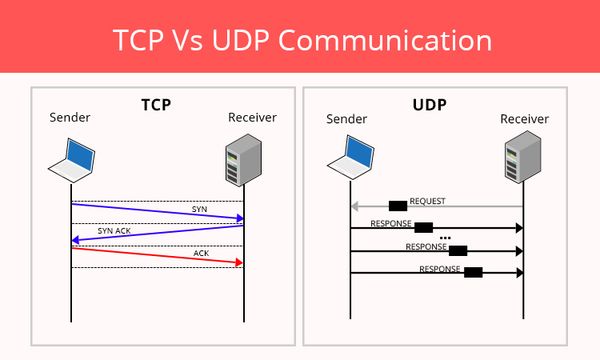


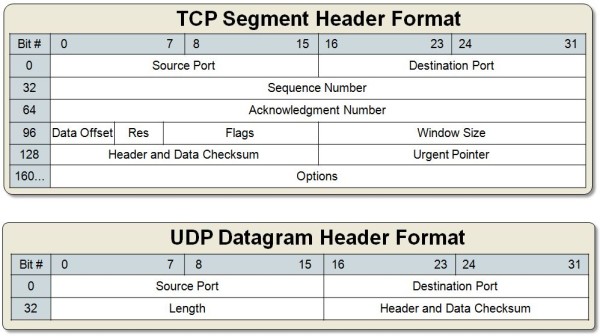


**TCP vs UDP**

| **Feature** | **TCP** | **UDP** |
| --- | --- | --- |
| Connection | Connection-oriented | Connectionless |
| Reliability | Reliable | Unreliable |
| Ordering | In-order delivery | No ordering |
| Flow control | Yes | No |
| Congestion control | Yes | No |
| Speed | Slower | Faster |

**TCP uses:** Web (HTTP/HTTPS), Email, File transfer  
**UDP uses:** Video streaming, Online gaming, VoIP, DNS





**Multiplexing & Demultiplexing**

These allow **multiple applications** to use the network **simultaneously**.

**Multiplexing (Sender side)**

* Collects data from many applications
* Adds **port numbers**
* Sends segments to the network

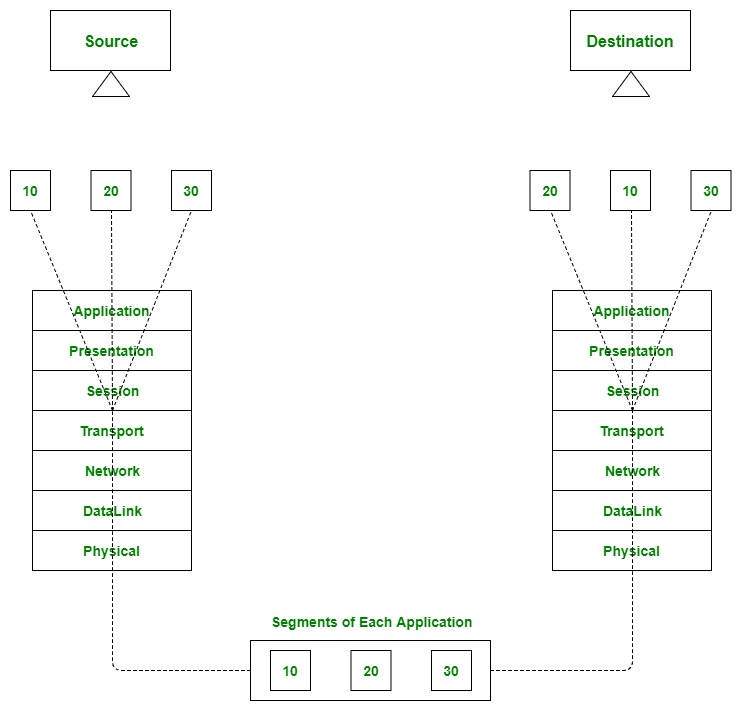
**Demultiplexing (Receiver side)**

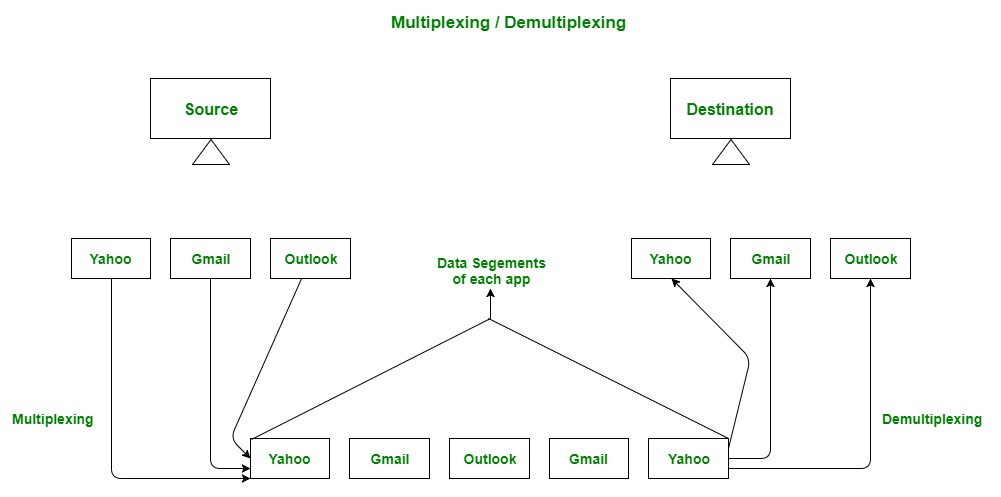
* Uses destination port number
* Delivers data to the correct application

Example:

Browser (port 5001) + Email (port 5002)

→ One network interface





**Reliable Data Transfer**

**Reliable data transfer** ensures data arrives:

* Without errors
* In correct order
* Without loss

TCP achieves reliability using:

* Sequence numbers
* Acknowledgements (ACKs)
* Retransmissions
* Timers
* Checksums

If data is lost or corrupted → **retransmit**

UDP does **not** provide reliability.

**Flow Control vs Congestion Control**

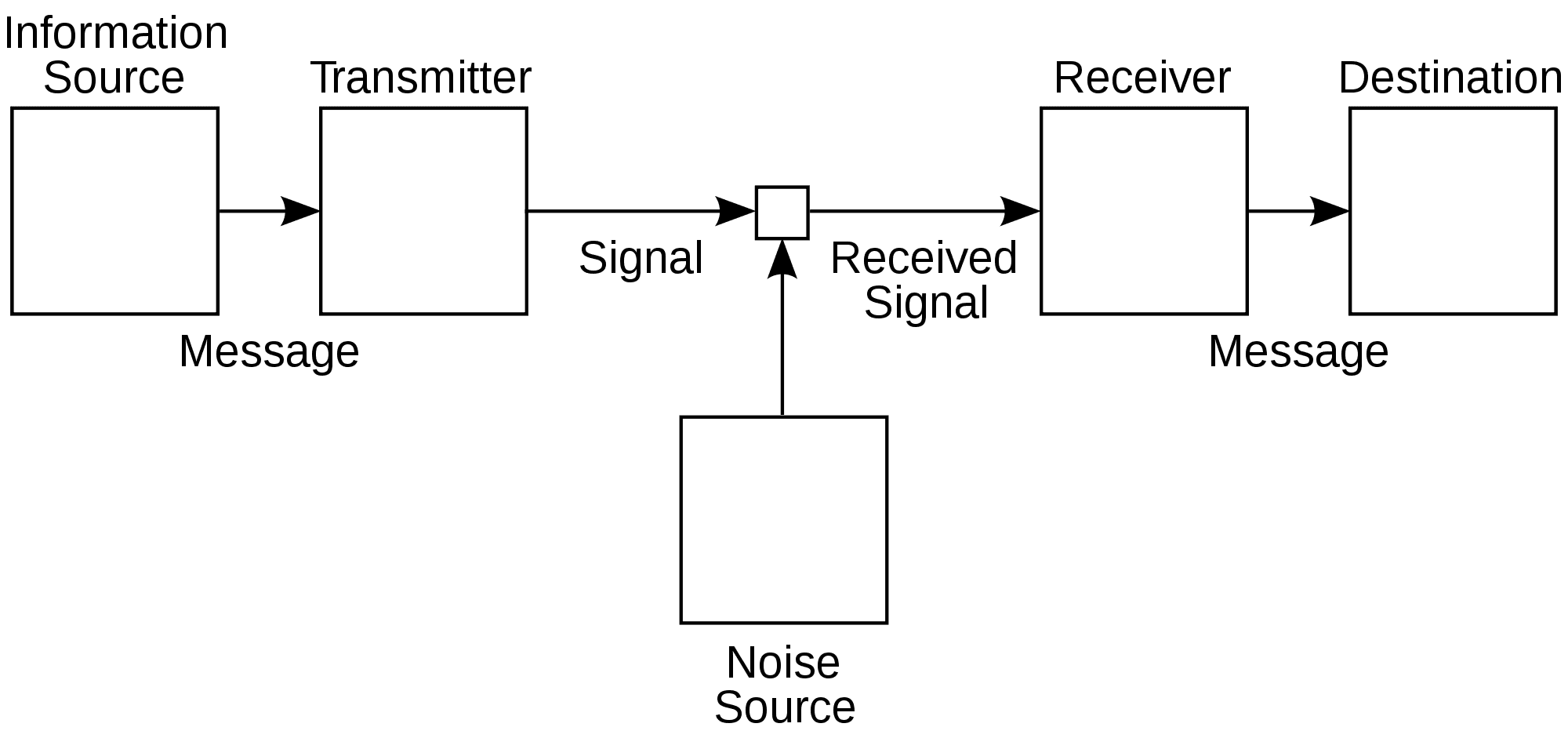
**Flow Control**

* Prevents **sender from overwhelming receiver**
* Based on receiver’s capacity
* Implemented using **receiver window (rwnd)**

**Congestion Control**

* Prevents **network from being overloaded**
* Based on network conditions
* Implemented using **congestion window (cwnd)**

| **Aspect** | **Flow Control** | **Congestion Control** |
| --- | --- | --- |
| Protects | Receiver | Network |
| Concern | Receiver buffer | Network traffic |
| Used by | TCP | TCP |





**TCP Congestion Control Phases**

TCP dynamically adjusts sending rate using these phases:

**1. Slow Start**

* Starts with small congestion window (cwnd)
* cwnd **doubles every RTT**
* Rapid growth

**2. Congestion Avoidance**

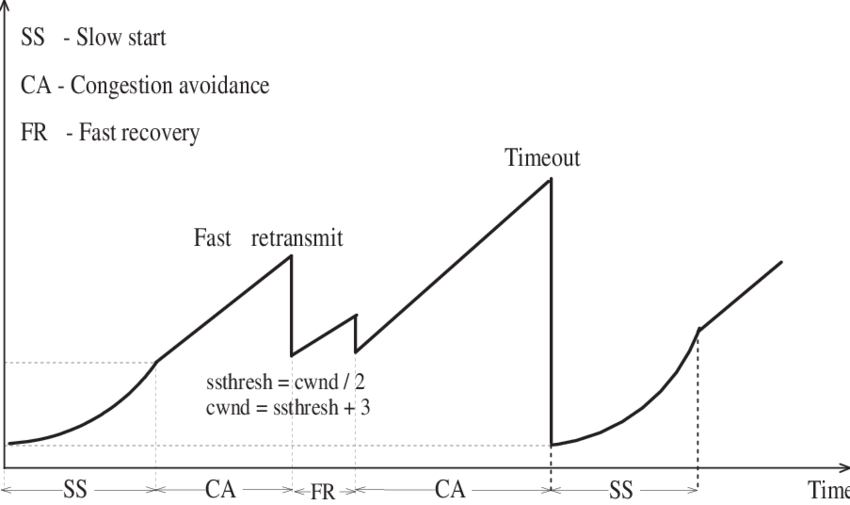
* cwnd increases **linearly**
* Prevents sudden congestion

**3. Fast Retransmit**

* Triggered by **3 duplicate ACKs**
* Retransmits lost segment immediately

**4. Fast Recovery**

* Reduces cwnd (not to 1)
* Continues congestion avoidance



**Quick Exam Summary (2–3 lines)**

* The **Transport Layer** provides end-to-end communication using TCP or UDP.
* **TCP** ensures reliable, ordered, congestion-controlled delivery; **UDP** is fast and connectionless.
* **Multiplexing/demultiplexing** use port numbers, while **flow control** protects the receiver and **congestion control** protects the network.

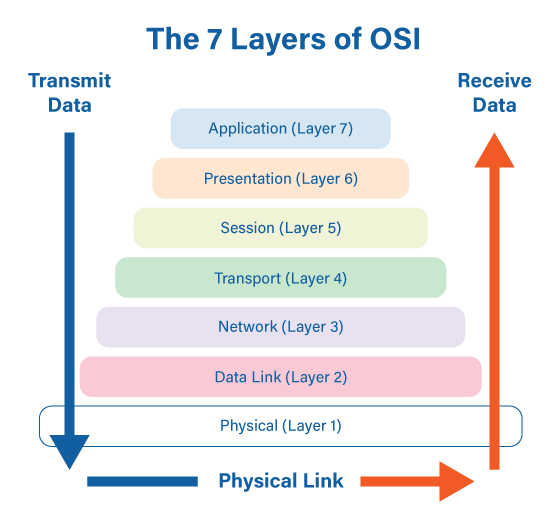
## Network Layer

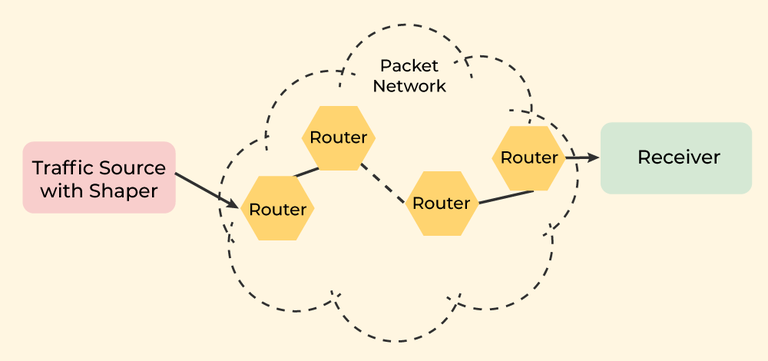
The **Network Layer** is responsible for **delivering packets from the source host to the destination host across multiple networks**.

**Key responsibilities:**

* Logical addressing (IP addresses)
* Routing (path selection)
* Packet forwarding
* Fragmentation (if needed)

The core protocol of this layer is **IP (Internet Protocol)**.





**Data Plane vs Control Plane**

**Data Plane**

* Handles **actual packet forwarding**
* Works **locally on routers**
* Uses forwarding tables

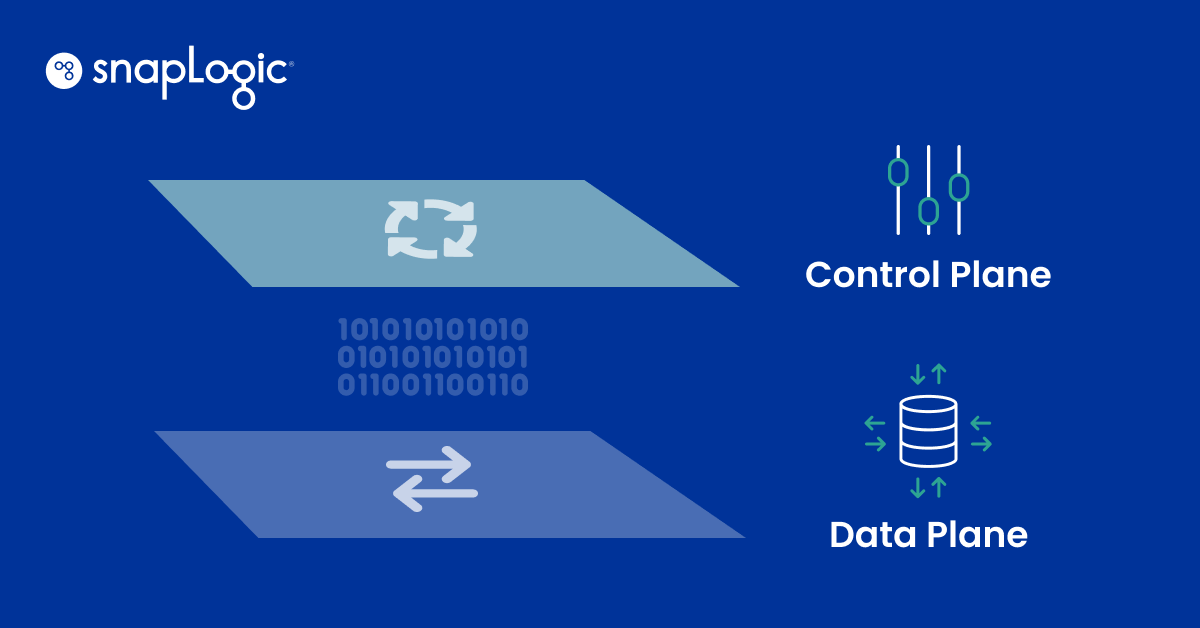
Example:  
“Send this packet out of interface 2”

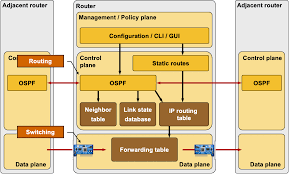
**Control Plane**

* Decides **how packets should be routed**
* Runs **routing algorithms & protocols**
* Builds routing tables

Example:  
“Which path is best to reach this destination?”

| **Aspect** | **Data Plane** | **Control Plane** |
| --- | --- | --- |
| Function | Forward packets | Compute routes |
| Scope | Per router | Network-wide |
| Speed | Fast | Slower |





**IP Addressing**

An **IP address** uniquely identifies a device on a network.

**IPv4**

* 32 bits
* Written in dotted decimal

192.168.1.1

Split into:

* **Network part**
* **Host part**

**IPv6**

* 128 bits
* Solves address exhaustion

**Subnetting**

**Subnetting** divides a large network into **smaller sub-networks**.

**Why subnet?**

* Efficient IP usage
* Reduced broadcast traffic
* Better security & management

Example:

192.168.1.0/24

→ can be split into two /25 subnets

**CIDR (Classless Inter-Domain Routing)**

CIDR replaces class-based addressing.

Format:

IP / prefix length

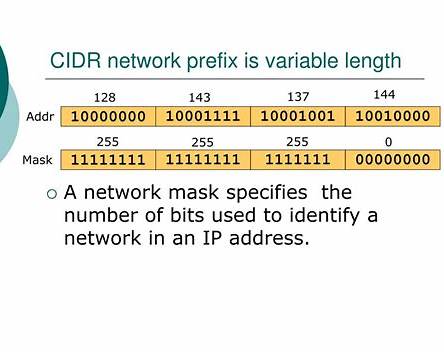
Example:

192.168.1.0/24

Benefits:

* Flexible network sizes
* Reduces routing table size
* Efficient IP allocation





**DHCP & NAT**

**DHCP (Dynamic Host Configuration Protocol)**

Automatically assigns:

* IP address
* Subnet mask
* Default gateway
* DNS server

Process (DORA):

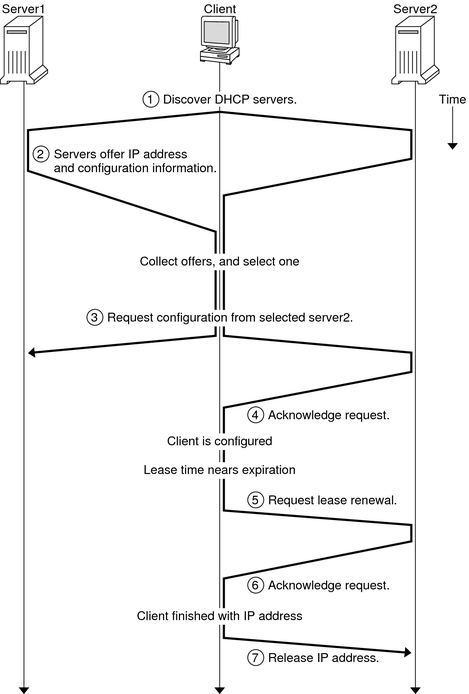
1. Discover
2. Offer
3. Request
4. Acknowledge

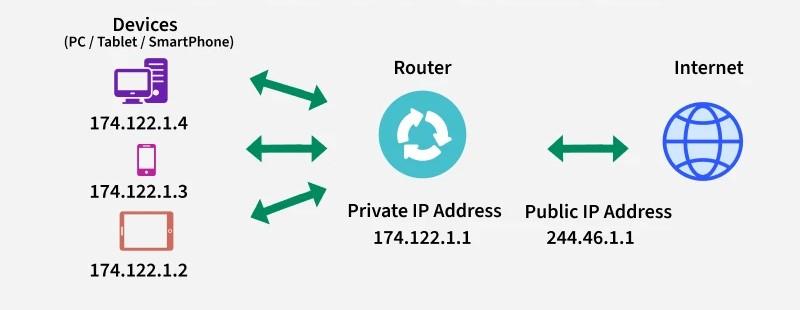
**NAT (Network Address Translation)**

Maps **private IPs** to **public IPs**.

Why NAT?

* Saves public IP addresses
* Adds basic security
* Allows multiple devices to share one public IP





**Routing Algorithms**

Routing algorithms determine the **best path** for packets.

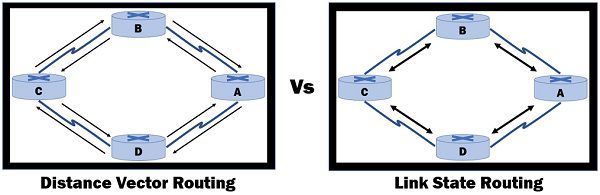
**1. Distance Vector**

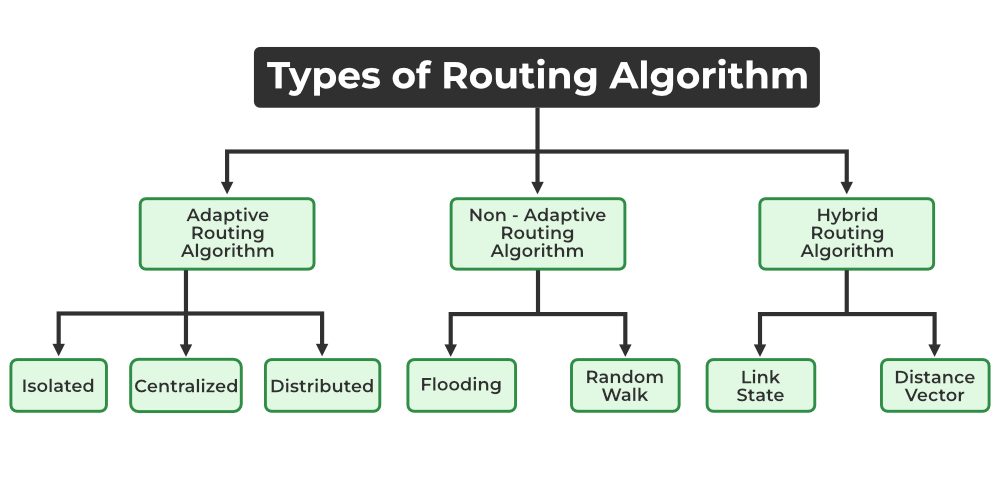
* Uses neighbor information
* Bellman-Ford algorithm
* Example: RIP
* Problem: count-to-infinity

**2. Link State**

* Uses full network topology
* Dijkstra’s algorithm
* Example: OSPF
* Faster convergence

| **Algorithm** | **Knowledge** | **Speed** |
| --- | --- | --- |
| Distance Vector | Partial | Slower |
| Link State | Global | Faster |





**BGP & Autonomous Systems**

**Autonomous System (AS)**

An **AS** is a group of networks under **one administrative control**.

Example:

* ISP
* Large enterprise
* University network

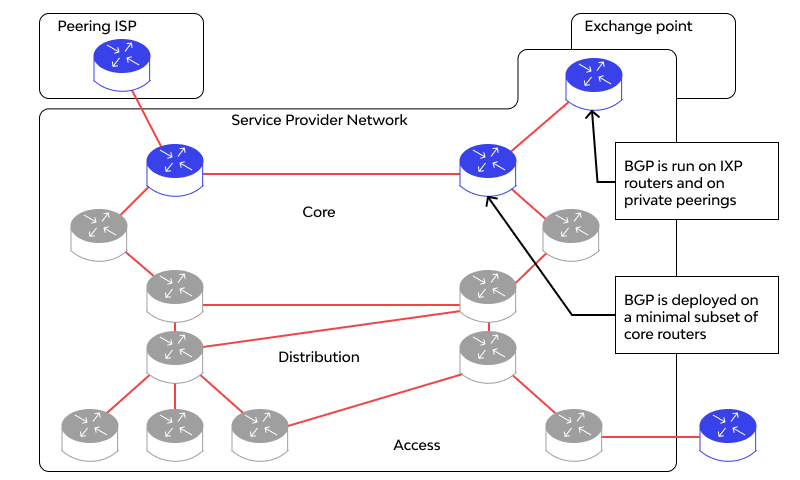
Each AS has a unique **AS number (ASN)**.

**BGP (Border Gateway Protocol)**

* Routing protocol **between Autonomous Systems**
* Internet’s **core routing protocol**
* Uses **path-vector** algorithm
* Policy-based routing (not shortest path)

BGP decides:

* Which AS path to use
* Which routes to advertise or block





**One-Page Exam Summary**

* The **Network Layer** provides logical addressing and routing across networks.
* The **data plane** forwards packets, while the **control plane** computes routes.
* **CIDR, subnetting, DHCP, and NAT** improve address efficiency.
* **Routing algorithms** find paths, and **BGP** connects Autonomous Systems across the Internet.

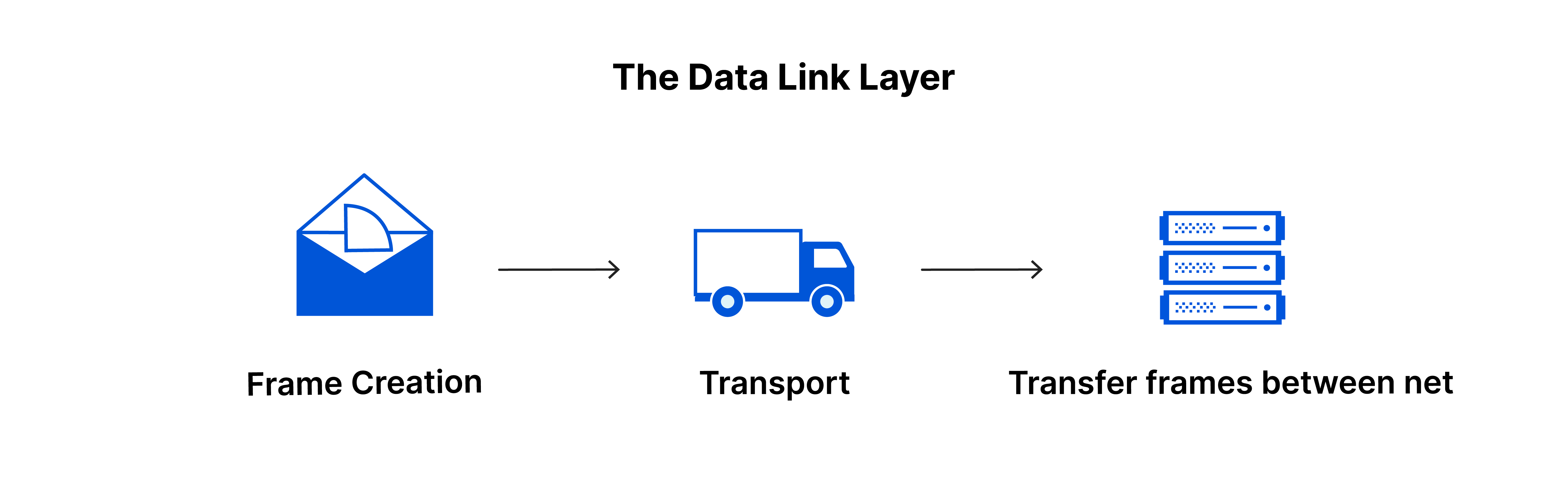
## Link Layer

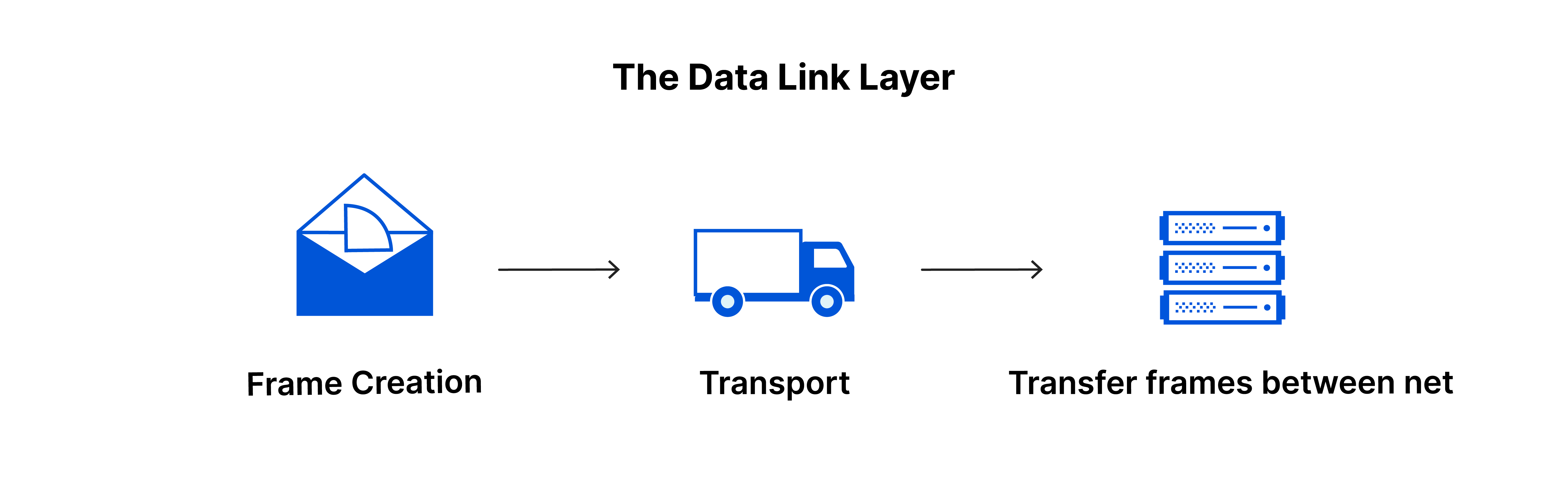
The **Link Layer** (also called the **Data Link Layer**) is responsible for **node-to-node delivery** of data over a **single physical link**.

**Main responsibilities:**

* Framing
* MAC (hardware) addressing
* Error detection
* Medium access control
* Reliable delivery over a local link

It works **between two directly connected devices** (host ↔ switch, switch ↔ router).





**Framing & Error Detection**

**Framing**

**Framing** divides the continuous bit stream into **frames** so receivers can identify:

* Start of data
* End of data

Each frame contains:

* Header (addresses, control info)
* Payload (actual data)
* Trailer (error detection bits)

**Error Detection**

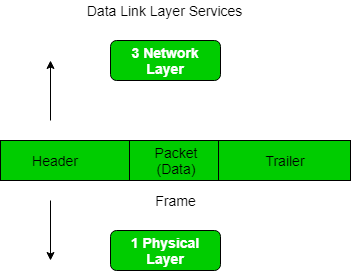
Used to detect **bit errors** during transmission.

Common techniques:

* **Parity bit**
* **Checksum**
* **CRC (Cyclic Redundancy Check)** – most widely used

If an error is detected:

* Frame is discarded
* Upper layers may request retransmission



**MAC Address vs IP Address**

| **Aspect** | **MAC Address** | **IP Address** |
| --- | --- | --- |
| Type | Physical address | Logical address |
| Assigned by | Manufacturer | Network / ISP / DHCP |
| Length | 48 bits | 32 bits (IPv4) |
| Scope | Local network | Global (Internet) |
| Changes? | No | Yes |

Example:

MAC: 00:1A:2B:3C:4D:5E

IP: 192.168.1.10

👉 **MAC = who you are**  
👉 **IP = where you are**

**Ethernet Switching**

An **Ethernet switch** connects devices in a **LAN** and forwards frames using **MAC addresses**.

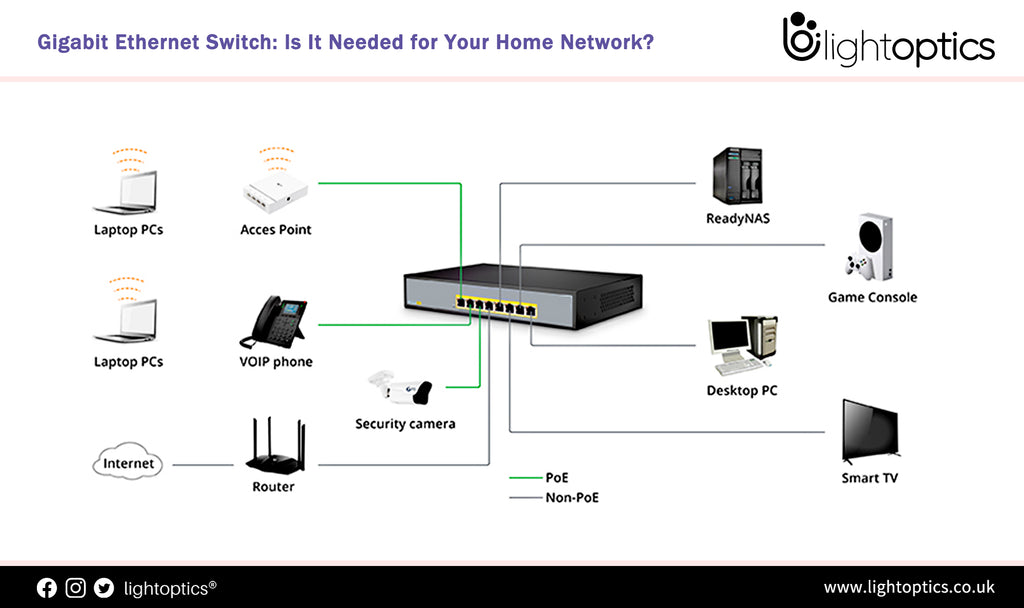
**How switching works:**

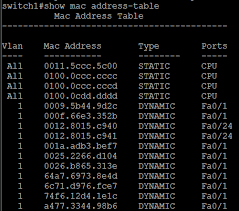
1. Switch learns MAC addresses from incoming frames
2. Builds a **MAC address table**
3. Forwards frames only to the correct port

Benefits:

* Reduces collisions
* Improves performance
* Full-duplex communication

If destination MAC is unknown → **broadcast**





**ARP (Address Resolution Protocol)**

**ARP** maps an **IP address to a MAC address** inside a local network.

**Why ARP is needed**

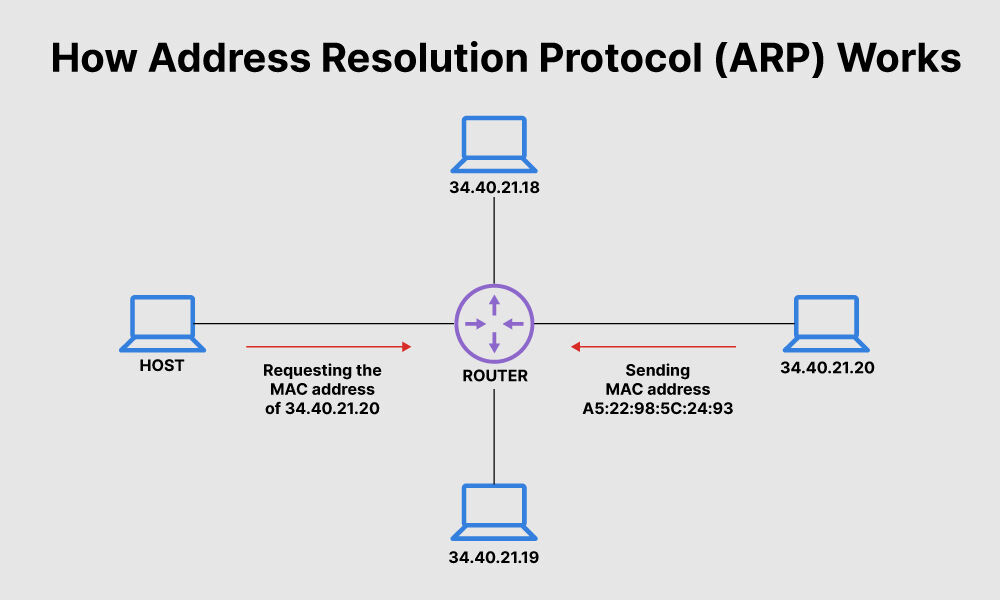
* IP is used for routing
* MAC is needed for actual delivery

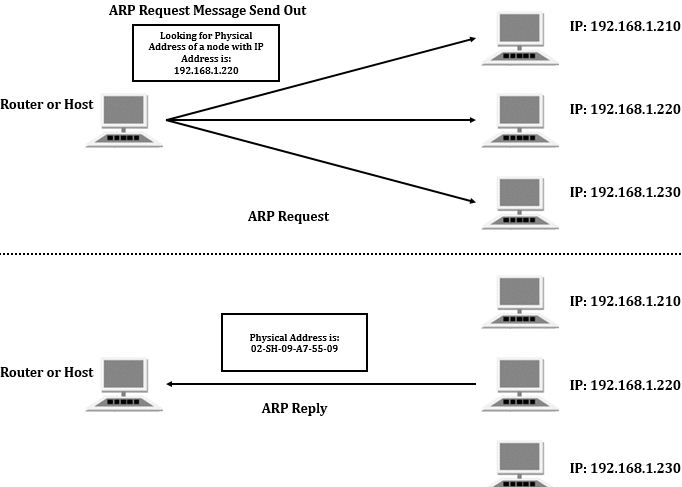
**ARP Process:**

1. Sender broadcasts: *Who has IP X?*
2. Target replies with its MAC address
3. Sender stores mapping in **ARP cache**

Example:

IP: 192.168.1.1 → MAC: AA:BB:CC:DD:EE:FF





**Quick Exam Summary (Perfect for 5–10 marks)**

* The **Link Layer** ensures reliable frame delivery over a single link using MAC addresses.
* **Framing** organizes data, while **error detection** identifies corrupted frames.
* **Ethernet switches** forward frames based on MAC tables.
* **ARP** resolves IP addresses to MAC addresses for local delivery.

## Wireless Networking (Overview)

**Wireless networks** transmit data using **radio waves** instead of cables.

Examples:

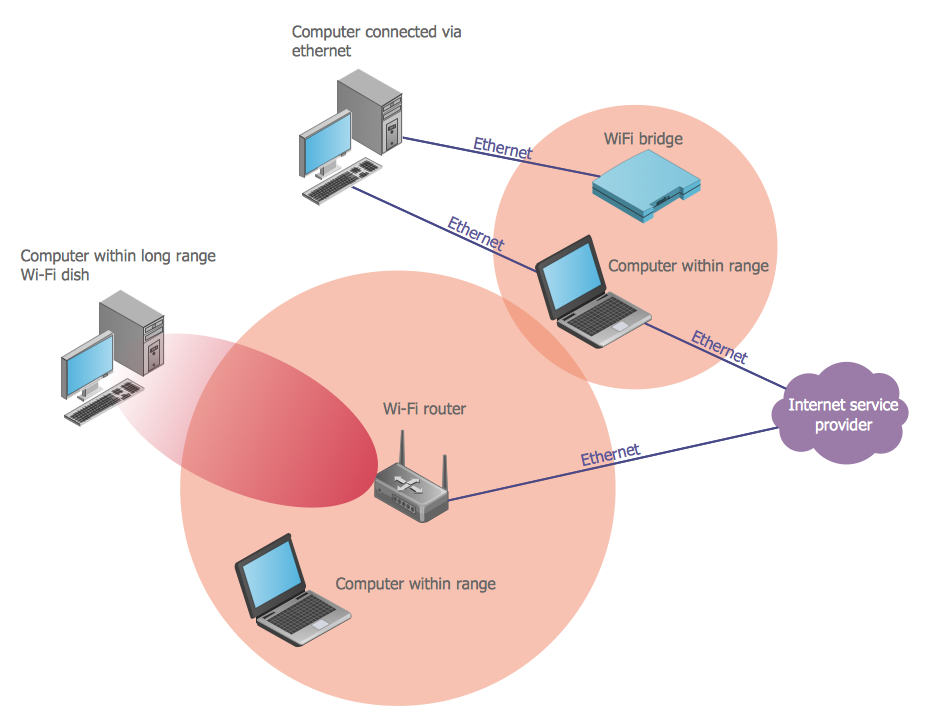
* Wi-Fi
* Bluetooth
* Cellular (4G/5G)
* Industrial wireless networks

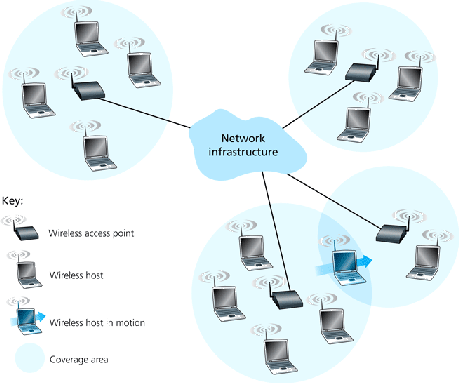
**Advantages:**

* Mobility
* Easy deployment

**Challenges:**

* Interference
* Security risks
* Variable signal quality





**Wi-Fi Challenges & Hidden Terminal Problem**

**Wi-Fi Challenges**

* Shared medium (many devices compete)
* Signal attenuation & interference
* Security threats (eavesdropping)
* Unpredictable delays

**Hidden Terminal Problem**

Occurs when **two devices cannot hear each other**, but both can communicate with the same access point.

Example:

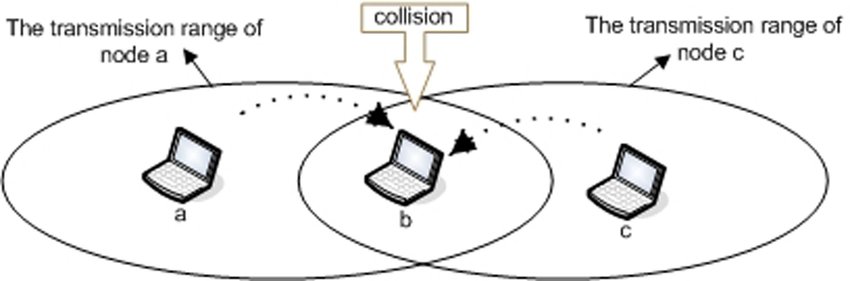
Node A → AP ← Node C

(A and C can’t hear each other)

Both transmit simultaneously → **collision at AP**

Result:

* Data corruption
* Retransmissions
* Reduced throughput



**Collision Avoidance (CSMA/CA)**

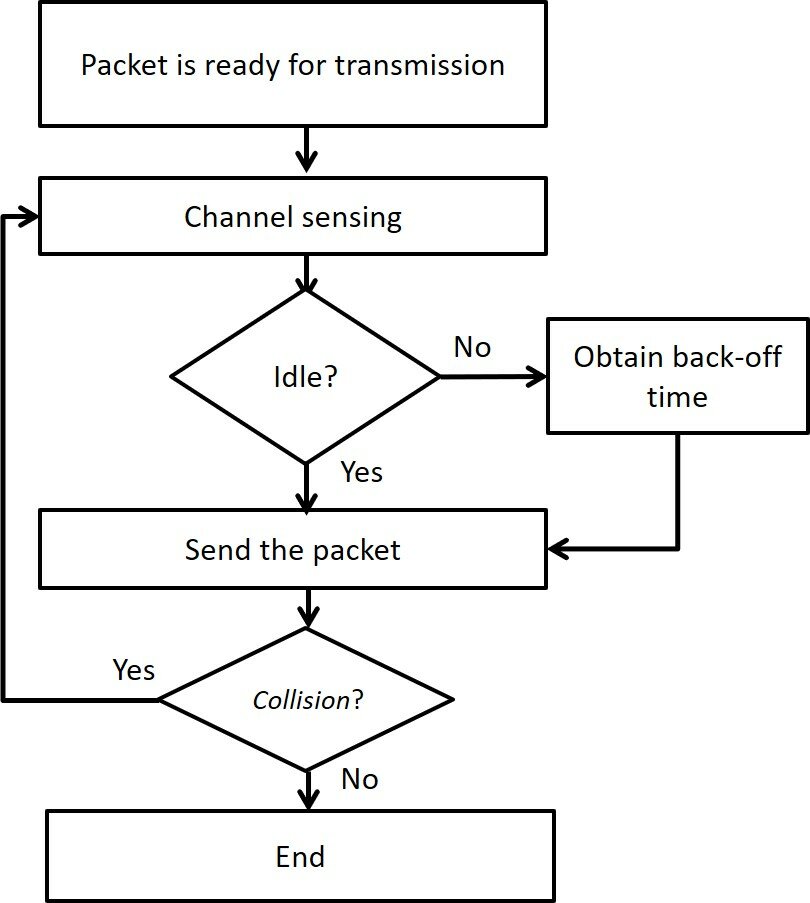
Wireless networks **cannot detect collisions**, so they use **collision avoidance**.

**CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)**

Steps:

1. Listen before transmitting
2. If channel is busy → wait (random backoff)
3. Optional RTS/CTS handshake
4. Transmit data
5. Receive ACK

This **reduces** collisions but does not eliminate them.





**Cryptography & Key Exchange**

**Cryptography**

Cryptography protects data by ensuring:

* **Confidentiality**
* **Integrity**
* **Authentication**

Types:

* **Symmetric encryption** – same key (fast)
* **Asymmetric encryption** – public/private keys (secure)

**Key Exchange**

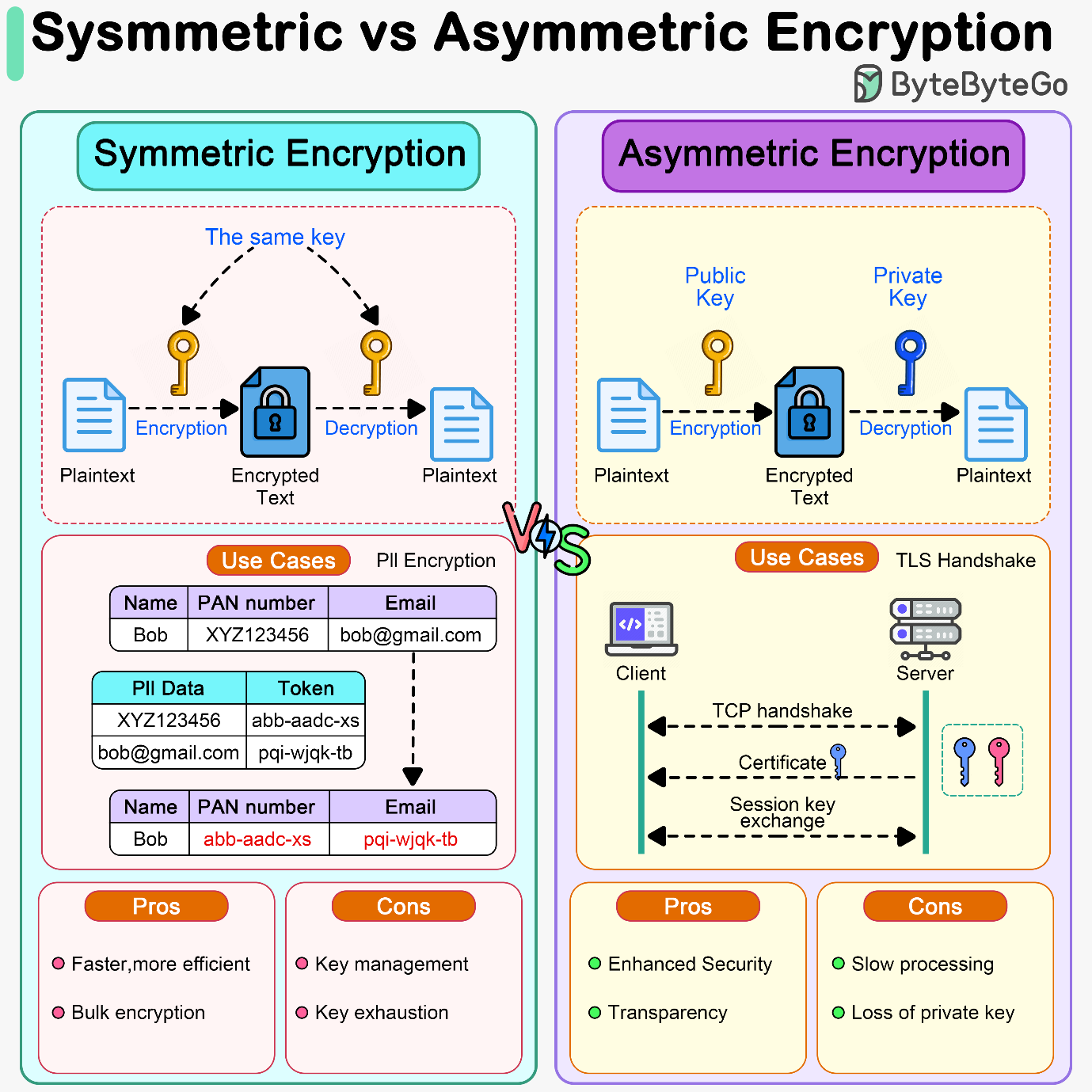
Key exchange allows two parties to **securely share a secret key** over an insecure network.

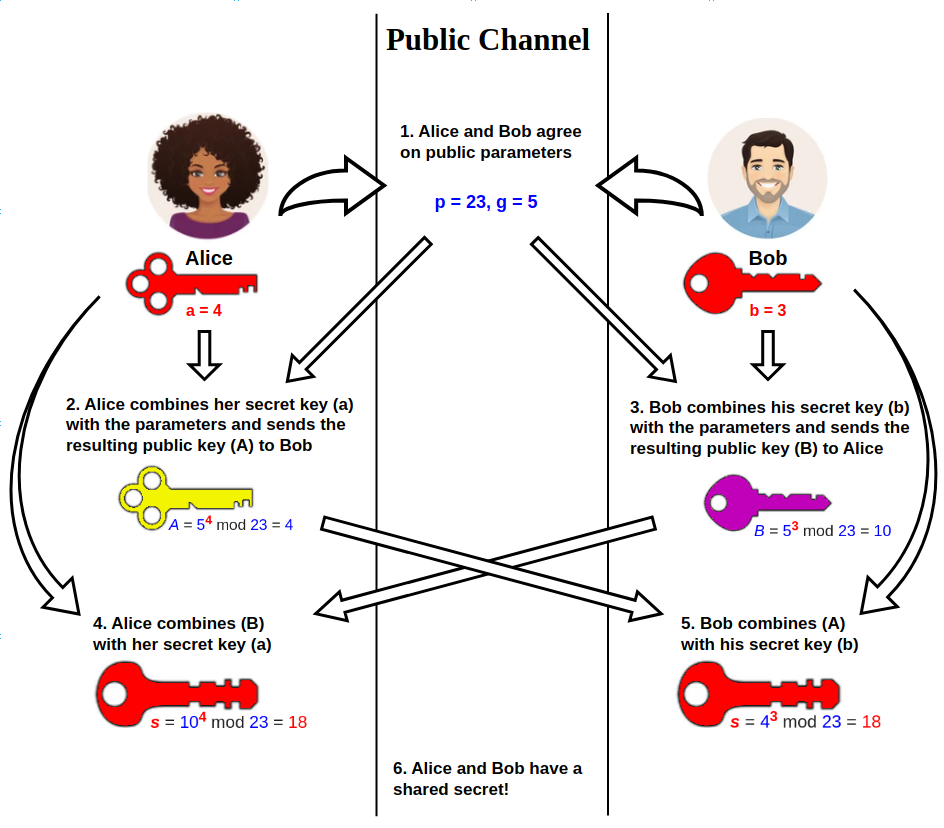
Common idea:

* Public information is exchanged
* Secret key is derived independently
* Attacker cannot compute the key

Used in:

* HTTPS
* VPNs
* Secure Wi-Fi (WPA)





**Industrial Networking & Real-Time Communication**

**Industrial Networking**

Used in:

* Factories
* Power plants
* Automation systems

Examples:

* PLCs
* Sensors
* Actuators

Key requirements:

* High reliability
* Deterministic timing
* Low latency
* Fault tolerance

**Real-Time Communication**

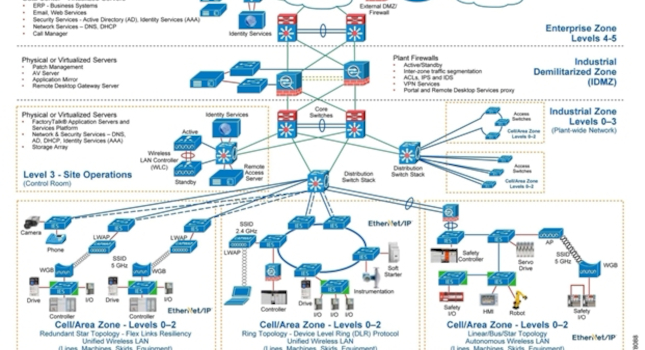
A **real-time network** guarantees that data is delivered **within a strict time limit**.

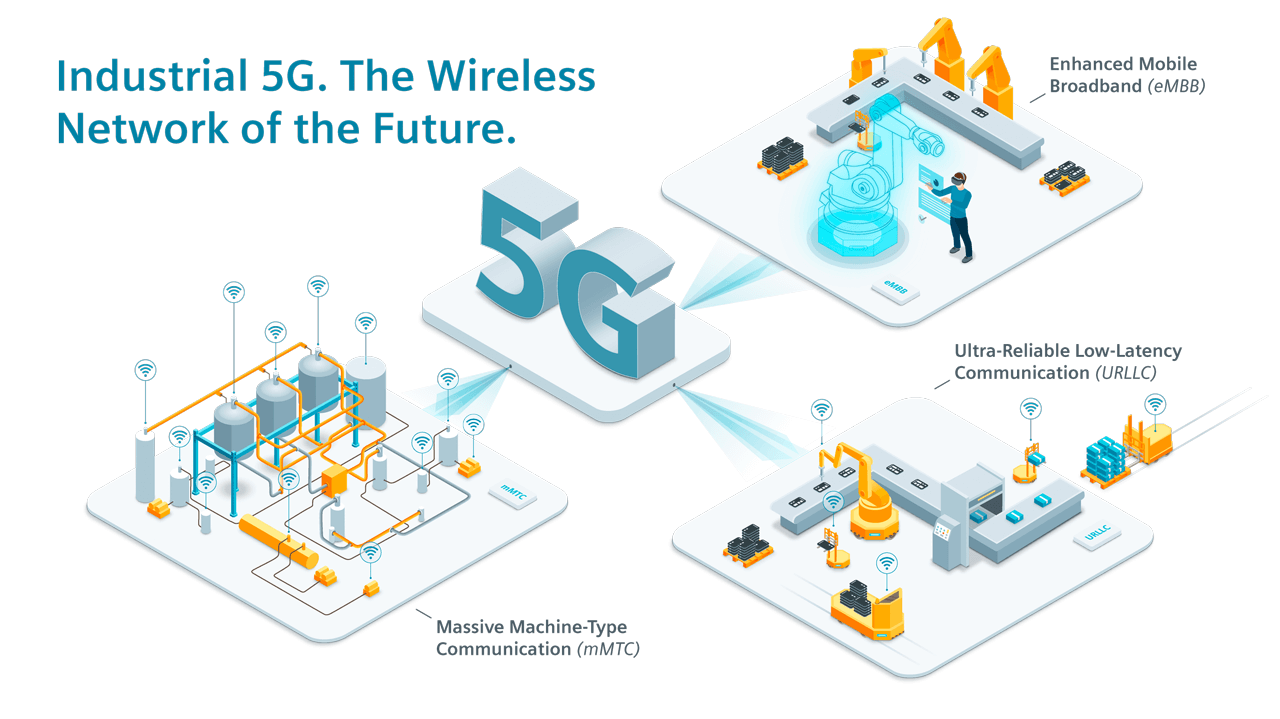
Types:

* **Hard real-time** – missing deadline = failure
* **Soft real-time** – delay degrades performance

Used in:

* Robotics
* Motion control
* Safety systems





**One-Page Exam Summary**

* Wireless networks offer mobility but suffer from interference and collisions.
* The **hidden terminal problem** causes collisions in Wi-Fi.
* **CSMA/CA** reduces collisions using sensing and backoff.
* **Cryptography and key exchange** secure data and communication.
* **Industrial and real-time networks** prioritize reliability and guaranteed timing over throughput.