✅ 1. Networking Fundamentals

🔹 What is a Network?

✅ Definition:

A **computer network** is a collection of interconnected devices (nodes) that communicate and share resources (e.g., data, printers, internet) with each other using predefined communication protocols over transmission media.

✅ Purpose:

- Resource sharing
- Communication (email, chat, video calls)
- Remote access and administration
- Centralized data management

🔹 Types of Networks

✅ 1. LAN (Local Area Network)

- Small geographical area (home, office)
- High speed
- Examples: Ethernet, Wi-Fi in a building

✅ 2. MAN (Metropolitan Area Network)

- Covers city-sized area
- Higher cost, medium speed
- Example: Cable TV network in a city

✅ 3. WAN (Wide Area Network)

- Covers country or continent
- Uses leased telephone lines or satellites
- Example: The Internet

✅ 4. PAN (Personal Area Network)

- Short range (~10 meters)
- Used for personal devices
- Example: Bluetooth, USB tethering

🔹 Network Topologies

✅ Definition:

Topology refers to the physical or logical arrangement of network devices.

✅ Types:

1. **Star Topology**
   - All nodes connect to a central hub
   - Easy to manage; hub failure leads to total failure
2. **Bus Topology**
   - All nodes connected to a single backbone
   - Cheap but suffers from collisions
3. **Ring Topology**
   - Devices connected in a circular fashion
   - Token passing used; break in one node affects the whole
4. **Mesh Topology**
   - Every node connected to every other node
   - Expensive but reliable
5. **Hybrid Topology**
   - Combination of two or more topologies

✅ Diagram:

```
 Star Topology:
     [PC1]
       |
[PC2]--HUB--[PC3]
       |
     [PC4]

Ring Topology:
[PC1] -- [PC2] -- [PC3]
  |                 |
 [PC5]-------------[PC4]
```

🔷 OSI Model – 7 Layers

✅ Definition:

The **OSI (Open Systems Interconnection)** model standardizes the functions of a telecommunication system into **seven layers**, helping different networks communicate reliably.

✅ Layers:

Layer	Name	Functions
7	Application	User interface, services like HTTP, SMTP
6	Presentation	Data formatting, encryption, compression
5	Session	Establish/manage/end communication sessions
4	Transport	Reliable delivery (TCP/UDP), segmentation
3	Network	Routing, IP addressing
2	Data Link	MAC addressing, error detection
1	Physical	Transmission media, bits, signals

✅ Diagram (Layer Stack):

```
+---------------------+  ← Layer 7: Application (HTTP, FTP)
| Application Layer   |
+---------------------+  ← Layer 6: Presentation (SSL, JPEG)
| Presentation Layer  |
+---------------------+  ← Layer 5: Session (API calls)
| Session Layer       |
+---------------------+  ← Layer 4: Transport (TCP/UDP)
| Transport Layer     |
+---------------------+  ← Layer 3: Network (IP, routers)
| Network Layer       |
+---------------------+  ← Layer 2: Data Link (Ethernet, MAC)
| Data Link Layer     |
+---------------------+  ← Layer 1: Physical (cables, radio)
| Physical Layer      |
+---------------------+
```

🔹 Encapsulation and Decapsulation

✅ Concept:

When data travels **down the OSI model** from sender → receiver, it undergoes:

- **Encapsulation:** Each layer adds its own header (and possibly trailer)
- **Decapsulation:** On the receiver side, each layer removes its respective header

✅ Example:

```
Sender:
Application Data
↓
[Transport Header] + Application Data → Segment
↓
[Network Header] + Segment → Packet
↓
[Data Link Header/Trailer] + Packet → Frame
↓
Bits → Physical Transmission

Receiver:
```

```
Bits
→ Frame → Packet → Segment → Data
```

🔹 Sample Code: Socket Connection (Application Layer)

```cpp
// Simple TCP Client in C++
#include <iostream>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>

int main() {
    int sock = socket(AF_INET, SOCK_STREAM, 0);

    sockaddr_in server;
    server.sin_family = AF_INET;
    server.sin_port = htons(8080);
    inet_pton(AF_INET, "127.0.0.1", &server.sin_addr);

    connect(sock, (struct sockaddr*)&server, sizeof(server));
    send(sock, "Hello Server!", 13, 0);

    char buffer[1024] = {0};
    read(sock, buffer, 1024);
    std::cout << "Server Reply: " << buffer << std::endl;

    close(sock);
    return 0;
}
```

✅ Sample Output:

```
Server Reply: Welcome Client!
```

🔹 Gantt Chart Example

> Not applicable for this CN section. Gantt charts are more relevant for **CPU Scheduling**.

🔹 Real-World Scenario Q&A

Q1: What happens when you type `www.google.com` in your browser?

Answer:

1. Browser checks cache
2. DNS resolution (`www.google.com → IP`)
3. TCP 3-way handshake (SYN, SYN-ACK, ACK)
4. TLS handshake (if HTTPS)
5. HTTP request sent to server
6. Server responds with HTML
7. Browser renders page

Q2: How does OSI help in troubleshooting?

Answer: OSI lets you isolate issues:

- Layer 1: Broken cable?
- Layer 3: Wrong IP or routing?
- Layer 7: Browser not sending request?

Q3: Why is TCP more reliable than UDP?

Answer:

- TCP ensures delivery via acknowledgements
- Retransmission of lost packets
- Ordered delivery
- Congestion control

✅ TCP/IP Model & Protocols – Complete Notes

🔹 TCP/IP Model – Comparison with OSI

✅ Definition:

The **TCP/IP Model** (a.k.a. DoD model) is a **4-layer framework** developed by the Department of Defense, which defines how data should be transmitted across networks. It's the foundation of the modern Internet.

✅ Layers in TCP/IP:

Layer (TCP/IP)	Equivalent OSI Layer(s)	Key Responsibilities
4. Application	OSI Layers 7, 6, 5	Protocols like HTTP, DNS, FTP
3. Transport	OSI Layer 4	TCP, UDP – port addressing, reliability
2. Internet	OSI Layer 3	IP addressing, routing
1. Network Access	OSI Layers 2 and 1	MAC addressing, framing, transmission

---

✅ TCP/IP vs OSI – Tabular Comparison

Feature	OSI Model (7 Layers)	TCP/IP Model (4 Layers)
Developed By	ISO (International Standard Org)	DoD (US Department of Defense)
No. of Layers	7	4
Layer Names	Application, Presentation, Session, Transport, Network, Data Link, Physical	Application, Transport, Internet, Network Access
Protocol Support	Theoretical	Practical, Internet-focused
Standardization	More descriptive and layered	More protocol-specific
Used In Practice?	Rarely in full	Yes, TCP/IP is the Internet backbone

---

✅ Diagram: OSI vs TCP/IP Stack

```
 +----------------------+     +----------------------+
| 7. Application       |     | 4. Application       |
| 6. Presentation      |     |                      |
| 5. Session           |     |                      |
+----------------------+     +----------------------+
| 4. Transport         |     | 3. Transport         |
+----------------------+     +----------------------+
| 3. Network           |     | 2. Internet          |
+----------------------+     +----------------------+
| 2. Data Link         |     | 1. Network Access    |
| 1. Physical          |     |                      |
+----------------------+     +----------------------+
```

🔷 Network Protocols Overview

✅ What is a Protocol?

A **network protocol** is a set of **rules and standards** that define how devices communicate over a network.

✅ Categories of Protocols:

📦 Application Layer Protocols:

- **HTTP / HTTPS** – Web communication
- **FTP / SFTP** – File transfer
- **SMTP / POP3 / IMAP** – Email transfer
- **DNS** – Resolving domain names
- **DHCP** – Dynamic IP configuration
- **Telnet / SSH** – Remote login

🚚 Transport Layer Protocols:

- **TCP** – Reliable, connection-oriented
- **UDP** – Fast, connectionless

🛰️ **Internet Layer Protocols:**

- **IP (IPv4 / IPv6)** – Logical addressing
- **ICMP** – Diagnostics (Ping)
- **ARP / RARP** – MAC/IP resolution

🔌 **Network Access Layer:**

- **Ethernet** – LAN communication
- **PPP / HDLC** – Point-to-point links
- **MAC / LLC** – Data link sublayers

✅ Sample Output: Using `ping` and `traceroute` to observe ICMP

```
$ ping www.google.com
PING www.google.com (142.250.77.36): 56 data bytes
64 bytes from 142.250.77.36: icmp_seq=0 ttl=114 time=18.34 ms
```

```
$ traceroute www.google.com
1  192.168.0.1 (local router)
2  10.180.0.1 (ISP gateway)
3  142.250.77.36 (Google server)
```

🔹 Real-World Scenario Q&A

Q1: Why is TCP used for file transfer but not video streaming?

Answer:
TCP ensures **ordered and reliable delivery**, which is needed for files.
UDP is better for streaming due to **low latency** and tolerance for packet loss.

Q2: How does DNS work in the Application Layer?

Answer:

1. Query sent to local resolver
2. Resolver queries root → TLD → authoritative server
3. IP of domain returned to browser

Q3: Why are protocols layered?

Answer:

- Modularity: Each layer handles its function independently
- Reusability: TCP can support HTTP, FTP, etc.
- Easier debugging and development

Q4: What happens if ARP fails?

Answer:
The host cannot resolve IP to MAC address → **data link layer cannot transmit frame** → packet dropped.

Q5: Why is HTTPS more secure than HTTP?

Answer:

- HTTPS = HTTP + TLS/SSL
- Data is encrypted → prevents MITM, sniffing

✅ 2. Physical Layer – In-depth Notes

🔹 Analog vs Digital Signals

✅ Definition:

- **Analog Signal:** Continuous in time and value. Can have infinite variations within a range.
- **Digital Signal:** Discrete in time and value. Uses binary (0 and 1).

✅ Characteristics:

Feature	Analog	Digital
Nature	Continuous	Discrete
Representation	Sine waves	Square waves
Examples	Sound waves, radio signals	Computer data, Ethernet signals
Susceptibility	More prone to noise	Less susceptible to noise

✅ Diagrams:

```
 Analog Signal:
        ~~~~~~~~                    ~~~~~~~
    ~           ~~~~~~~~~~~~~             ~~~~~
~                                             ~~~

Digital Signal:

|    |    |      |     |       |
|____|    |______|     |_______|
```

🔹 Bit Rate vs Baud Rate

✅ Definitions:

- **Bit Rate (bps):** Number of bits transmitted per second.
- **Baud Rate:** Number of signal changes (symbols) per second.

✅ Key Relationship:

```
Bit Rate = Baud Rate × log₂(L)
Where L = Number of distinct signal levels.
```

- If each symbol carries 1 bit (binary encoding):
  `Bit Rate = Baud Rate`
- If each symbol carries multiple bits (e.g., QAM):
  `Bit Rate > Baud Rate`

✅ Example:

If a signal has 4 voltage levels, each level represents **2 bits** ($\log_2 4 = 2$).
If baud rate = 1000 baud, then:
`Bit Rate = 1000 × 2 = 2000 bps`

✅ Real-World Analogy:

- **Baud Rate:** How many times you blink your eye per second.
- **Bit Rate:** How much information is communicated by each blink.

✅ Interview Q&A:

Q1: Why is baud rate lower than bit rate in modern systems?
A: Because we encode multiple bits in one signal level (multi-level modulation like QAM, PSK).

Q2: Is it possible to have a high bit rate with low baud rate?
A: Yes, by increasing signal levels (e.g., using 16-QAM, 64-QAM).

🔹 Nyquist & Shannon's Theorems

✅ 1. Nyquist Theorem (Noiseless Channel)

Formula:

```
Max Bit Rate = 2 × Bandwidth × log₂(L)
```

- L = Number of discrete signal levels

- Bandwidth in Hz

Implication:
Higher signal levels → Higher data rate
But limited by hardware and signal distinguishability

✅ 2. Shannon Capacity (Noisy Channel)

Formula:

```
C = B × log₂(1 + S/N)
```

Where:

- C = Channel Capacity (bps)
- B = Bandwidth (Hz)
- S/N = Signal-to-noise ratio (unitless)

✅ Example:

If:

- Bandwidth = 3 kHz
- SNR = 30 dB = 1000 (10^(30/10))

Then:

```
C = 3000 × log₂(1 + 1000)
  ≈ 3000 × 9.97
  ≈ 29910 bps
```

✅ Diagrams:

```
 Shannon Curve (SNR vs Capacity):
Capacity ↑ logarithmically as SNR ↑

Nyquist Graph:
Bit Rate ↑ linearly with log₂(L) and bandwidth
```

✅ Real-World Q&A:

Q1: What limits real-world transmission speed?
A: Noise, hardware constraints, and channel capacity.

Q2: Why can't we increase L (levels) infinitely?
A: High L makes signals harder to distinguish, especially in noisy environments.

Q3: What's more realistic: Nyquist or Shannon?
A: Shannon – it accounts for noise, which is always present.

✅ Physical Layer – Transmission Media, Multiplexing, Switching

---

🔹 Transmission Media

✅ Definition:

Transmission media refers to the physical pathway through which data travels from source to destination.

📦 Categories:

1. Guided Media (Wired)

Data travels through a physical medium (cables).

a. Twisted Pair Cable

- Two insulated copper wires twisted together.
- Types: UTP (Unshielded), STP (Shielded)
- Speed: Up to 1 Gbps (Cat5e), 10 Gbps (Cat6a)
- Usage: Ethernet, telephone lines

b. Coaxial Cable

- Central copper conductor with insulation, metallic shield, and outer cover.
- Higher noise immunity than twisted pair.
- Usage: Cable TV, legacy Ethernet

c. Fiber Optic Cable

- Transmits light instead of electrical signals.
- Extremely high bandwidth and long-distance support.
- Types: Single-mode, Multi-mode
- Usage: Backbone networks, ISPs, submarine cables

---

2. Unguided Media (Wireless)

Data travels through air or vacuum as electromagnetic waves.

a. Radio Waves

- Omni-directional
- Range: Few meters to kilometers
- Usage: FM radio, walkie-talkies, Wi-Fi

b. Microwaves

- Unidirectional
- Requires line-of-sight
- Usage: Satellite communication, mobile phones

c. Infrared

- Short-range and directional
- Blocked by walls
- Usage: Remote controls, short-range device links

✅ Comparative Table:

Medium	Guided/Unguided	Bandwidth	Cost	Use Case
Twisted Pair	Guided	Low to Medium	Low	LAN, phones
Coaxial	Guided	Medium	Medium	Cable TV
Fiber Optic	Guided	Very High	High	Long-distance ISP
Radio	Unguided	Low to Medium	Low	Broadcasting
Microwave	Unguided	High	Medium	Satellites
Infrared	Unguided	Low	Very Low	Remote Controls

🔹 Multiplexing

✅ Definition:

Multiplexing is a technique to combine multiple signals and transmit them over a single communication channel.

🧩 Types of Multiplexing:

1. FDM (Frequency Division Multiplexing)

- Different signals occupy different frequency bands.
- Example: Radio/TV channels
- Analogy: Multiple people speaking at different pitches.

2. TDM (Time Division Multiplexing)

- Each signal occupies the entire bandwidth but at different time slots.
- Types: Synchronous TDM, Statistical TDM
- Example: Traditional digital telephony
- Analogy: Single microphone used by speakers one after another.

3. WDM (Wavelength Division Multiplexing)

- Fiber-optic version of FDM using different light wavelengths.

- Dense WDM (DWDM) supports up to 160+ channels
- Example: Fiber-optic internet backbones

✅ Diagram (Conceptual):

```
TDM:
| A | B | C | A | B | C | A | ...

FDM:
<--A--><--B--><--C-->  (separate frequencies)

WDM:
[ λ1 | λ2 | λ3 | λ4 ] over a single fiber
```

🔹 Switching Techniques

✅ Definition:

Switching is the method used to direct data from source to destination through a network.

🔀 Types of Switching:

1. Circuit Switching

- A dedicated path is established before data transfer.
- Resources reserved throughout the session.
- Example: Traditional telephone systems
- Pros: Consistent performance
- Cons: Inefficient resource usage if idle

2. Packet Switching

- Data is split into packets, sent independently.
- No dedicated path required.
- Packets may take different routes and reorder at destination.
- Example: Internet, VoIP
- Pros: Efficient, scalable
- Cons: Delay, packet loss possible

3. Message Switching (Store and Forward)

- Whole message stored at intermediate node before forwarding.
- No need for a dedicated path.
- High delay but reliable.
- Mostly historical or used in email-style services

✅ Comparison Table:

Switching Type	Path Setup	Delay	Resource Usage	Suitable For
Circuit Switching	Required	Low (after setup)	Fixed	Voice calls
Packet Switching	Not Needed	Variable	Dynamic	Internet data
Message Switching	Not Needed	High	Dynamic	Store-and-forward apps

---

🧠 Real-world Q&A:

Q1: Why is packet switching preferred in the Internet?
A: Because it's scalable, utilizes bandwidth efficiently, and adapts to network conditions.

Q2: Can you combine TDM and FDM?
A: Yes, modern systems use hybrid multiplexing (e.g., 4G uses OFDM – Orthogonal FDM with TDM).

Q3: Why is fiber preferred for backbone networks?
A: High bandwidth, low signal loss, and immunity to electromagnetic interference.

✅ Physical Layer – Modulation Techniques (ASK, FSK, PSK)

🔹 What is Modulation?

✅ Definition:

Modulation is the process of **converting digital or analog data into a transmittable signal** by modifying a carrier wave's properties (amplitude, frequency, or phase).

Used primarily in:

- Wireless communication
- Radio broadcasting
- Optical transmission

🔹 Why Modulate?

- To match the frequency range of transmission media
- To transmit over long distances
- To allow **multiplexing** (multiple signals on the same channel)
- To improve signal quality and reduce interference

🔹 Basic Terminology:

- **Carrier Wave:** A high-frequency wave that carries the actual data
- **Baseband Signal:** The original data signal
- **Modulated Signal:** Final signal after combining carrier + data

---

🔹 Digital Modulation Techniques

✅ 1. Amplitude Shift Keying (ASK)

📌 Concept:

- The **amplitude** of the carrier wave is varied to represent binary data.
- Binary 1 = High amplitude, Binary 0 = Low or no amplitude

🧠 Characteristics:

- Simple, easy to implement
- Very susceptible to noise (since noise affects amplitude)

📊 Diagram:

```
 Bitstream:    1    0    1    1    0
ASK Signal:
                 |   |     |   ||   |
```

📦 Use Case:

- Optical fiber systems
- RFID tags

✅ 2. Frequency Shift Keying (FSK)

📌 Concept:

- The **frequency** of the carrier wave is changed to represent binary data.
- Binary 1 = High frequency, Binary 0 = Low frequency

🧠 Characteristics:

- Less affected by amplitude noise
- Requires more bandwidth than ASK

📊 Diagram:

```
 Bitstream:    1    0    1    0
FSK Signal:  ~~~~~  ~~~  ~~~~~  ~~~~
```

📦 **Use Case:**

- Bluetooth
- Radio transmission
- Caller ID systems

✅ 3. Phase Shift Keying (PSK)

📌 **Concept:**

- The **phase** of the carrier wave is altered to represent binary data.
- Binary 1 = Phase shift (e.g., 180°), Binary 0 = No shift

🧠 **Characteristics:**

- Robust against noise
- Efficient bandwidth usage

📊 **Diagram:**

```
 Bitstream:    1     0     1
PSK Signal:  ϕ----  ----  ϕ----
           (Phase flip → 180°)
```

📦 **Use Case:**

- Wi-Fi (BPSK, QPSK)
- Satellite communication
- RFID readers

🔹 Comparison Table

Technique	Modifies	Noise Resistance	Bandwidth Usage	Complexity	Use Cases
ASK	Amplitude	Low	Low	Simple	Optical comm., RFID
FSK	Frequency	Medium	High	Medium	Bluetooth, radio
PSK	Phase	High	Low	Complex	Wi-Fi, satellite, RFID

🔹 Advanced Forms (Mention Only)

- **QPSK (Quadrature PSK)**: 2 bits per symbol (0°, 90°, 180°, 270°)
- **QAM (Quadrature Amplitude Modulation)**: Combines ASK + PSK
- **BPSK (Binary PSK)**: Simplest form of PSK
- **OFDM (Orthogonal Frequency Division Multiplexing)**: Used in 4G/5G

🔹 Real-World Q&A

Q1: Why is PSK better than ASK?

- Because PSK is more **noise-resistant**, making it suitable for wireless systems.

Q2: Where is FSK commonly used?

- In **Bluetooth**, due to its simplicity and moderate robustness.

Q3: Which modulation is used in Wi-Fi?

- **BPSK**, **QPSK**, and **QAM**, depending on speed and signal strength.

Q4: Can modulation be used over fiber optics?

- Yes, typically **ASK or QAM-based optical modulation** techniques are used.

✅ 3. Data Link Layer – Framing & Error Detection

🔹 Framing

✅ Definition:

Framing is the process of converting a stream of bits into distinguishable units (frames) so that the receiver can interpret boundaries and data correctly.

🧩 Techniques of Framing:

1. **Character Count**

- A field in the header specifies the number of characters in the frame.
- ❌ Issue: If count gets corrupted, entire frame alignment is lost.

```
[05][Data1][Data2][Data3][Data4][Data5]
```

---

2. **Byte Stuffing (Character-Oriented)**

- Special **flag byte** indicates frame boundaries (e.g., `FLAG = 01111110`)
- If data contains flag byte, insert **escape (ESC)** character before it.

```
 Original Data:     FLAG, A, B, ESC, FLAG
Stuffed Frame:     FLAG, A, B, ESC, ESC, ESC, FLAG, FLAG
```

---

3. **Bit Stuffing (Bit-Oriented)**

- After 5 consecutive `1`s in data, insert a `0` to avoid confusion with `flag = 01111110`.

```
 Sender Data:       01111110 → Data: 0111110
Transmitted:       011111010

Receiver removes the extra 0 after every 5 ones.
```

✅ Real-world Q&A:

Q1: Why is bit stuffing preferred in HDLC?

- Because HDLC is bit-oriented and flag-based, making bit stuffing a natural choice.

Q2: What if a frame delimiter appears in the actual data?

- Use byte/bit stuffing to escape such sequences and preserve frame boundaries.

🔹 Error Detection Techniques

✅ 1. Parity Bits

📌 Concept:

Add an extra bit to make the total number of 1s **even (even parity)** or **odd (odd parity)**.

🧪 Example:

- Data: `1011001` → Even parity = `10110011`
- Single-bit error detection only

✅ 2. Checksum

📌 Concept:

- Break data into equal segments (e.g., 8 or 16 bits)
- Add all segments, take **1's complement** of the sum
- Send it as checksum

✅ Example (8-bit):

```
 Data1 = 01010101
Data2 = 01110010
Sum   = 11000111
Checksum = ~(11000111) = 00111000
```

At receiver:

```
Add Data1 + Data2 + Checksum → Should be all 1s (i.e., 11111111)
```

✅ 3. CRC (Cyclic Redundancy Check)

📌 Concept:

- Treat data as a **binary polynomial**
- Divide it by a **generator polynomial (G(x))**
- Append **remainder** (CRC bits) to the data
- Receiver divides the whole message by same G(x); if remainder is 0 → no error

🔢 Example:

Data = `1101011011`
Generator (G) = `10011` (5-bit → CRC = 4 bits)

Perform binary division, obtain 4-bit remainder, append to data.

```
Transmitted: 1101011011 + CRC
```

✅ Comparison Table

Technique	Detects	Correction?	Overhead	Use Case
Parity Bit	1-bit errors	No	Low	Simple hardware links
Checksum	Burst errors	No	Medium	IP, UDP, TCP
CRC	Multiple bits	No	High	Ethernet, HDLC

✅ Real-world Q&A:

Q1: Why isn't checksum enough in Ethernet?

- CRC is more robust for detecting burst and complex errors; hence used in Ethernet.

Q2: Can parity detect 2-bit errors?

- ❌ No, it only works for **odd number** of bit errors.

Q3: What happens if CRC detects an error?

- Frame is discarded; **upper layers (Transport)** handle retransmission (e.g., via TCP).

Q4: Is error detection same as error correction?

- No:
  - **Detection:** Know that error occurred
  - **Correction:** Fix the error (e.g., Hamming code)

✅ 3. Data Link Layer – Error Correction, Flow Control, ARQ Protocols

---

🔷 Error Correction

✅ Hamming Code

📌 Concept:

Hamming Code is a forward error correction technique that can detect and correct **single-bit errors** using redundant parity bits placed at power-of-two positions.

🧠 How It Works:

For `k` data bits, we add `r` parity bits such that:

```
2^r ≥ k + r + 1
```

✅ Example (4-bit data)

Let's encode `1011` using Hamming(7,4):

Step 1: Positions (1-based indexing)

```
 Positions: 1 2 3 4 5 6 7
Bits:      P P D P D D D
           1 2 3 4 5 6 7
```

Let data bits = `D = 1 0 1 1` → Positions 3, 5, 6, 7

Insert:

```
 3 → 1
5 → 0
6 → 1
7 → 1
```

Step 2: Calculate parity bits

- **P1 (bit 1)** → covers bits 1,3,5,7 → 1 ⊕ 1 ⊕ 0 ⊕ 1 = **0**
- **P2 (bit 2)** → covers bits 2,3,6,7 → 2 ⊕ 1 ⊕ 1 ⊕ 1 = **1**
- **P4 (bit 4)** → covers bits 4,5,6,7 → 0 ⊕ 0 ⊕ 1 ⊕ 1 = **0**

Final encoded word:

```
 Bits: [0 1 1 0 0 1 1]
Index: 1 2 3 4 5 6 7
```

✅ Error Detection & Correction:

If receiver receives: `0 1 1 0 1 1 1`
Check parity bits → calculate syndrome → get error position = `5`
→ Flip bit 5 to correct.

✅ Use Cases:

- RAM memory protection
- Satellite communications
- Reliable low-latency systems

🔹 Flow Control

✅ Definition:

Flow control ensures that the sender does not overwhelm the receiver by sending data too fast.

✅ 1. Stop-and-Wait Protocol

📌 Concept:

- Sender sends **one frame** and waits for **ACK** before sending the next.
- Simple but inefficient (low bandwidth utilization).

```
 S → Frame1 → R
S ←  ACK1  ← R
S → Frame2 → R
```

🧠 Problem:

- Idle time due to round-trip delay.

✅ 2. Sliding Window Protocol

📌 Concept:

- Allows multiple frames to be sent before waiting for ACK.
- Sender has a **window size (N)** of outstanding unacknowledged frames.
- Receiver may also buffer out-of-order frames.

🔁 Terms:

- **Window size**: Number of unACKed frames allowed in transit.
- **Sequence numbers**: Wrap-around using modulo arithmetic.

```
 Window [0,1,2,3] → Send Frame0 → Frame3 → Wait
ACKs shift window forward
```

✅ Comparison:

Protocol	Efficiency	Complexity	Use Case
Stop-and-Wait	Low	Simple	Low-latency channels
Sliding Window	High	Moderate	High-speed networks

🔹 ARQ (Automatic Repeat reQuest) Protocols

✅ 1. Stop-and-Wait ARQ

- Sender sends one frame and waits.
- If **timeout**, retransmit.
- Detects error via checksum/CRC.

```
 S → Frame1 → R
S ←  ACK1   ← R
(no ACK?) → retransmit
```

✅ 2. Go-Back-N ARQ

- Sender can send `N` frames before needing ACK.
- If a frame is lost, **all subsequent frames are retransmitted**.

Example:

Frame 3 is lost → receiver discards frames 4,5
→ Sender retransmits 3,4,5

✅ 3. Selective Repeat ARQ

- Sender retransmits **only the lost frame**
- Receiver **buffers** out-of-order frames

Example:

Frame 3 lost, but 4,5 received
→ Receiver stores 4,5
→ Requests retransmission of 3 only

✅ Comparison Table:

ARQ Type	Window Size	Retransmission	Buffer at Receiver	Efficiency
Stop-and-Wait	1	1 frame	None	Low
Go-Back-N	N	From error frame	Discards future	Medium
Selective Repeat	N	Only lost frames	Buffers out-of-order	High

---

✅ Real-world Q&A:

Q1: Why is sliding window more efficient than stop-and-wait?
Because it keeps the link busy by allowing multiple outstanding frames.

Q2: Why is Selective Repeat rarely used in hardware?
Because it requires **more buffer** and **complex out-of-order reassembly**.

Q3: What happens if ACK is lost?
Sender waits until timeout, then **retransmits** the frame.

Q4: Is ARQ used in TCP?
Yes, **TCP uses a variant of sliding window with cumulative ACKs and retransmissions**.

✅ 3. Data Link Layer – MAC Protocols, Ethernet, LAN Types

🔹 MAC (Media Access Control)

✅ Definition:

MAC is a sublayer of the Data Link Layer responsible for **controlling how devices access and transmit on a shared medium** to avoid collisions.

🔹 ALOHA Protocols

✅ 1. Pure ALOHA

📌 **Concept:**

- Devices transmit **anytime** they have data.
- If collision occurs → wait for random time and retransmit.

❌ **Drawback:**

- High chance of collision due to random access.

🧠 **Efficiency:**

- Max throughput: **18.4%**

✅ 2. Slotted ALOHA

📌 **Concept:**

- Time is divided into equal **slots**.
- Devices can **only transmit at the beginning of a time slot**.

✅ **Benefit:**

- Reduces chances of collision by synchronizing transmissions.

🧠 **Efficiency:**

- Max throughput: **36.8%**

✅ Comparison Table:

Protocol	Time Division	Collision Window	Efficiency
Pure ALOHA	No	Full frame time	~18%
Slotted ALOHA	Yes	Half frame time	~37%

🔹 CSMA (Carrier Sense Multiple Access)

✅ Concept:

Listen before transmitting (i.e., **sense the channel**).

✅ 1. CSMA/CD (Collision Detection)

- Used in **wired Ethernet**
- **Listen** before transmitting
- If collision occurs, **stop transmission** and send a jamming signal
- Wait for a **random backoff time**, then retry

⚠️ **Works only in half-duplex wired medium.**

✅ 2. CSMA/CA (Collision Avoidance)

- Used in **Wi-Fi / wireless networks**
- Cannot detect collisions (broadcast medium), so it **tries to avoid them**

- Uses **ACK-based confirmation** and **RTS/CTS** (Request to Send / Clear to Send)

✅ Comparison Table:

Protocol	Used In	Detect or Avoid?	Medium Type	Collision Handling
CSMA/CD	Ethernet	Detect	Wired	Stop + Backoff
CSMA/CA	Wi-Fi	Avoid	Wireless	RTS/CTS + ACK

🔹 Ethernet (IEEE 802.3)

✅ Definition:

Ethernet is the most widely used LAN technology. Defined under **IEEE 802.3**, it uses **CSMA/CD** and is a **frame-based** protocol.

✅ Frame Format:

```
+--------+--------+---------+------------+----------+
| Preamble | Dest MAC | Src MAC | Type/Length | Data     |
+--------+--------+---------+------------+----------+
```

- Preamble: 7 bytes of 10101010 for synchronization
- MAC addresses: 6 bytes each
- Payload: 46–1500 bytes
- CRC: 4-byte checksum

✅ Characteristics:

- Speeds: 10 Mbps, 100 Mbps (Fast Ethernet), 1 Gbps, 10 Gbps
- Topology: Star (logically bus)
- Cabling: Twisted pair or fiber

✅ Collision Domain:

- In hubs: single collision domain
- In switches: separate domains per port

🔹 LAN Types

✅ 1. Wired LANs

- **Uses Ethernet (IEEE 802.3)**

- Cables: Twisted pair, coaxial, fiber
- Devices: Switches, NICs
- Reliable and faster

✅ 2. Wireless LANs (Wi-Fi)

- **Standard: IEEE 802.11 (a/b/g/n/ac/ax)**
- Medium: Radio waves
- Uses **CSMA/CA**
- Requires **Access Point (AP)**

✅ IEEE 802.11 Variants:

Standard	Frequency	Max Speed	Notes
802.11a	5 GHz	54 Mbps	Shorter range
802.11b	2.4 GHz	11 Mbps	Longer range
802.11g	2.4 GHz	54 Mbps	Backward compatible
802.11n	2.4/5 GHz	600 Mbps	MIMO support
802.11ac	5 GHz	1+ Gbps	Beamforming
802.11ax	2.4/5 GHz	10 Gbps	OFDMA, MU-MIMO

✅ Real-world Q&A:

Q1: Why doesn't CSMA/CD work in Wi-Fi?

- Because **collision detection requires simultaneous transmit + listen**, which is impractical in wireless due to echo and signal fading.

Q2: Why is Slotted ALOHA more efficient than Pure ALOHA?

- Slotted ALOHA **reduces collision window by 50%** using time slots.

Q3: What's the role of RTS/CTS in CSMA/CA?

- Helps **avoid hidden terminal problem** by reserving the channel before data transmission.

Q4: What causes hidden terminal and exposed terminal problems?

- Wireless medium's inability to sense all nodes → **hidden** nodes cause collision, **exposed** nodes unnecessarily defer transmission.

Q5: Is Ethernet full-duplex?

- Modern Ethernet is **full-duplex**, hence **CSMA/CD is no longer used** with switches.

✅ 4. Network Layer – IP Addressing & Concepts

🔹 IP Addressing

✅ Definition:

An **IP address** is a **32-bit (IPv4)** or **128-bit (IPv6)** logical identifier for a device on a network. It uniquely identifies the **source and destination** for data transmission.

🔹 IP Address Classes (IPv4)

IPv4 uses **32-bit addressing**, represented as 4 decimal octets (e.g., `192.168.1.1`).

✅ Address Classes:

Class	Starting Bits	Range	Default Subnet Mask	Usage
A	0xxxxxxx	1.0.0.0 – 126.255.255.255	255.0.0.0	Large networks
B	10xxxxxx	128.0.0.0 – 191.255.255.255	255.255.0.0	Medium networks
C	110xxxxx	192.0.0.0 – 223.255.255.255	255.255.255.0	Small networks
D	1110xxxx	224.0.0.0 – 239.255.255.255	N/A	Multicast
E	1111xxxx	240.0.0.0 – 255.255.255.254	N/A	Research (Reserved)

> Note: `127.x.x.x` is reserved for loopback.

🔹 Public vs Private IPs

✅ Private IP Ranges:

Class	Private Range
A	10.0.0.0 – 10.255.255.255
B	172.16.0.0 – 172.31.255.255

Class	Private Range
C	192.168.0.0 – 192.168.255.255

- **Private IPs**: Used within LAN; not routable on the internet.
- **Public IPs**: Globally unique; routable on the internet.

🔹 Subnetting

✅ Definition:

Subnetting is dividing a large network into smaller **sub-networks (subnets)** by borrowing bits from the host portion of an IP address.

✅ Subnet Mask:

- Defines which part of IP is **network** and which is **host**.
- Example:
  - IP: `192.168.1.10`
  - Subnet Mask: `255.255.255.0` (24 bits → /24)
  - Network Address: `192.168.1.0`
  - Broadcast Address: `192.168.1.255`

✅ Calculation:

- To create `n` subnets, borrow `x` bits:

```
2^x ≥ n
```

- Number of Hosts per Subnet:

```
2^h - 2 (excluding network & broadcast)
```

✅ Example:

Divide `192.168.1.0/24` into 4 subnets:

- 4 = 2^2 → Borrow 2 bits
- New subnet mask = /26 = 255.255.255.192

Subnets:

- 192.168.1.0 → 192.168.1.63
- 192.168.1.64 → 192.168.1.127
- 192.168.1.128 → 192.168.1.191
- 192.168.1.192 → 192.168.1.255

🔹 Supernetting

✅ Definition:

Supernetting (route aggregation) is combining multiple **smaller networks** into a **larger one** to reduce routing table size.

- Opposite of subnetting
- Common in ISP-level routing

✅ Example:

Combine:

- 192.168.4.0/24
- 192.168.5.0/24
- 192.168.6.0/24
- 192.168.7.0/24

→ Supernet: 192.168.4.0/22

🔹 CIDR (Classless Inter-Domain Routing)

✅ Concept:

- Introduced to **overcome classful limitations** and enable efficient IP allocation.
- IP addresses written as `IP/prefix-length`
  - Example: `192.168.1.0/24`
- Removes class boundaries
- Allows flexible subnetting

✅ Benefits:

- Reduces IP wastage
- Supports route summarization
- Used in modern routers and ISPs

🔹 VLSM (Variable Length Subnet Mask)

✅ Concept:

- Allows **different subnet masks** for different subnets of the same network.
- Maximizes IP utilization.

✅ Example:

Given: `192.168.10.0/24`, allocate:

- 100 hosts → /25 → `192.168.10.0 – 192.168.10.127`
- 50 hosts → /26 → `192.168.10.128 – 192.168.10.191`
- 25 hosts → /27 → `192.168.10.192 – 192.168.10.223`

🔹 Real-world Q&A:

Q1: Why is subnetting useful?

- It improves **network performance**, **security**, and **address organization**.

Q2: What's the difference between CIDR and VLSM?

- CIDR is for **external routing and aggregation**, VLSM is for **internal subnet design**.

Q3: Why can't public IPs be reused?

- They're globally routable and must be unique to avoid collisions.

Q4: What happens if the subnet mask is misconfigured?

- Devices may not communicate with their intended subnet or gateway.

Q5: Why subtract 2 from host count?

- One address is **network**, one is **broadcast** (not assignable to hosts).

✅ 4. Network Layer – Binary Subnetting, IP Fragmentation, IPv4 vs IPv6

🔹 Binary Math for Subnetting

✅ Goal:

Determine network, broadcast, and host ranges using **bitwise operations** on the IP and subnet mask.

✅ Key Binary Concepts:

- IP and Subnet Mask are **32-bit binary values**
- **Network Address** = IP AND Subnet Mask
- **Broadcast Address** = Network Address OR Inverted Subnet Mask
- Total Hosts = 2^H - 2 (H = number of host bits)

✅ Example:

Given:

- IP = `192.168.1.130`
- Subnet Mask = `255.255.255.192` = `/26`

Convert to binary:

```
 IP:        11000000.10101000.00000001.10000010
Subnet:     11111111.11111111.11111111.11000000
Network:    11000000.10101000.00000001.10000000 = 192.168.1.128
Broadcast:  11000000.10101000.00000001.10111111 = 192.168.1.191
```

✅ Result:

- Network: `192.168.1.128`
- Broadcast: `192.168.1.191`
- Range: `192.168.1.129` – `192.168.1.190`
- Hosts: `2^(32-26) - 2 = 62`

✅ Binary Tips:

Decimal	Binary
192	11000000
224	11100000
240	11110000
248	11111000
252	11111100
254	11111110
255	11111111

🔹 IP Fragmentation and Reassembly

✅ Why Fragment?

- Networks may have different **MTU (Maximum Transmission Unit)** sizes.
- If a datagram exceeds MTU, it's **fragmented** into smaller packets.

✅ Fragmentation Fields:

Field	Description
ID	Unique identifier for each datagram

Field	Description
Offset	Position of fragment in original packet
MF (More Fragments)	1 if more fragments follow

✅ Fragmentation Example:

Original IP Packet: 4000 bytes
MTU = 1500 bytes
→ IP Header = 20 bytes
→ Max data per fragment = 1480 bytes

Fragments:

- Frag1: Offset = 0, Length = 1480, MF=1
- Frag2: Offset = 1480/8 = 185, Length = 1480, MF=1
- Frag3: Offset = 2960/8 = 370, Length = 1040, MF=0

✅ Reassembly:

- Performed at **destination only**.
- Fragments reassembled based on **ID, offset**, and **MF** flag.
- If any fragment is missing → entire packet is discarded.

✅ Real-world Notes:

- Fragmentation increases overhead.
- Often avoided using **Path MTU Discovery**.

🔹 IPv4 vs IPv6

✅ Address Format:

Feature	IPv4	IPv6
Address Length	32 bits	128 bits
Format	Decimal dotted (e.g. 192.168.1.1)	Hexadecimal colon (e.g. 2001:0db8::1)
Address Space	~4.3 billion	3.4 × 10^38
Header Size	20 bytes	40 bytes

✅ Feature Comparison:

Feature	IPv4	IPv6
NAT Support	Required (due to IP shortage)	Not needed (huge address space)
Broadcast	Supported	Not used (uses multicast instead)
Security	Optional (IPSec)	Built-in IPSec
Fragmentation	Routers & Host	Only Host
Configuration	Manual / DHCP	Auto-config (SLAAC) + DHCPv6
Mobility & QoS	Limited	Built-in support
Packet Routing	Less efficient	Simplified header & routing

✅ IPv6 Address Example:

```
 Full:      2001:0db8:0000:0000:0000:ff00:0042:8329
Shortened:  2001:db8::ff00:42:8329
```

✅ Transition Techniques:

- **Dual Stack:** Devices run both IPv4 and IPv6
- **Tunneling:** IPv6 over IPv4 (6to4, ISATAP)
- **Translation:** NAT64

✅ Real-world Q&A:

Q1: Why is IPv6 adoption slow?

- Legacy infrastructure, cost of transition, compatibility issues.

Q2: Can IPv6 solve NAT problems?

- Yes, because it provides **enough public IPs** for every device.

Q3: Why was fragmentation moved from routers to hosts in IPv6?

- To reduce **router processing overhead** and improve performance.

Q4: Why doesn't IPv6 use broadcast?

- **Multicast** and **anycast** are more efficient and scalable.

✅ 4. Network Layer – Routing Algorithms & Protocols

🔹 Routing

✅ Definition:

Routing is the process of selecting the best path for a packet to travel from source to destination across interconnected networks.

Routers use **routing tables** and **algorithms** to make decisions.

🔹 Static vs Dynamic Routing

✅ Static Routing

- Manually configured by the administrator
- Routes remain fixed unless manually changed
- No overhead due to route recalculations

✅ Pros:

- Simple to implement
- Secure and predictable

❌ Cons:

- No fault tolerance
- Not scalable in large networks

✅ Dynamic Routing

- Routers **exchange information** using routing protocols
- Routing tables are updated automatically
- Supports fault tolerance and scalability

✅ Pros:

- Adapts to network failures
- Easier to manage in large networks

❌ Cons:

- Consumes CPU, memory, and bandwidth
- Potential for routing loops and convergence delays

🔹 Distance Vector Routing (RIP)

✅ Protocol: **RIP (Routing Information Protocol)**

- Type: Distance Vector

- Metric: **Hop Count**
- Max Hop Count: 15 (16 = unreachable)
- Update Interval: Every 30 seconds
- Uses **Bellman-Ford algorithm**

✅ How it Works:

- Each router sends its **entire routing table** to its neighbors.
- Neighbors update their tables by **adding one hop** to each entry.

✅ Example:

Router A:

```
 Dest  | Hops
------|-----
Net1  | 0
Net2  | 1
```

Sends table to B → B updates Net1 to 1 hop, Net2 to 2 hops.

✅ Problems:

- **Count-to-Infinity Problem**
- **Routing Loops**

✅ Solutions:

- Split Horizon
- Route Poisoning
- Hold-down timers

🔹 Link State Routing (OSPF)

✅ Protocol: **OSPF (Open Shortest Path First)**

- Type: Link State
- Metric: **Cost (Bandwidth)**
- Algorithm: **Dijkstra's Shortest Path First**
- Updates: Only when topology changes
- Divides network into **areas** (e.g., Area 0 – backbone)

✅ How it Works:

1. Each router discovers its neighbors.
2. Sends **Link State Advertisements (LSAs)**.
3. Builds a complete **topology map**.

4. Runs Dijkstra's algorithm to compute shortest paths.

✅ Benefits:

- Faster convergence
- More scalable than RIP
- Loop-free by design

✅ OSPF Packet Types:

Type	Purpose
Hello	Discover neighbors
DB Description	Summarize database
LSR	Request missing LSAs
LSU	Update LSAs
LSAck	Acknowledge receipt

🔹 Path Vector Routing (BGP)

✅ Protocol: **BGP (Border Gateway Protocol)**

- Type: Path Vector
- Used for **inter-domain routing** (between ASes)
- Backbone of the **Internet**
- Metric: **AS Path (list of autonomous systems)**

✅ How it Works:

- Each BGP router advertises **entire path (AS numbers)** to reach a destination.
- Loop prevention by checking if its own AS appears in the path.

✅ Key Concepts:

Term	Description
AS (Autonomous System)	A group of IP networks under one admin
IBGP	Internal BGP (within AS)
EBGP	External BGP (between ASes)
Policy-Based	Admin can apply filters & preferences

✅ BGP Attributes:

- AS_PATH
- NEXT_HOP
- LOCAL_PREF
- MED (Multi Exit Discriminator)

✅ BGP vs OSPF vs RIP

Feature	RIP	OSPF	BGP
Type	Distance Vector	Link State	Path Vector
Metric	Hop Count	Cost (bandwidth)	AS Path
Max Hop	15	No Limit	No Limit
Usage	Small LANs	Enterprise Networks	Internet Routing
Algorithm	Bellman-Ford	Dijkstra	Path vector
Update Type	Periodic	Triggered	Triggered

🔹 Real-world Q&A:

Q1: Why is RIP not suitable for large networks?

- Limited hop count (15), slow convergence, and no support for complex topologies.

Q2: Why is OSPF preferred in enterprise networks?

- Fast convergence, support for hierarchy (areas), and efficient routing.

Q3: Why does the Internet use BGP?

- BGP supports **policy-based routing**, scalability, and **AS-level control** over routing.

Q4: Can a router run OSPF and BGP together?

- Yes. Enterprise routers often use **OSPF internally** and **BGP externally**.

Q5: What prevents loops in BGP?

- **AS-PATH checking** – a router discards routes containing its own AS number.

✅ 4. Network Layer – Routing Algorithms & Address Resolution

🔹 Routing Algorithms

✅ 1. Dijkstra's Algorithm (Link State Routing)

📌 **Concept:**

- Used in **Link State Routing** (e.g., OSPF)
- Computes the **shortest path** from a source node to all other nodes in the network using **weights/costs**.

✅ Steps:

1. Initialize distances from source to all nodes as ∞, except source = 0
2. Mark all nodes as unvisited
3. Pick unvisited node with smallest tentative distance
4. Update distance for neighbors
5. Repeat until all nodes are visited

✅ Example:

Graph:

```
 A --1-- B --2-- C
|       |
4       3
|       |
D ----- E
```

- Source: A
- Shortest paths: A-B (1), A-D (4), A-E (4), A-C (3)

✅ Complexity:

- $O(V^2)$ for simple
- $O((V+E)\log V)$ with min-priority queue (Dijkstra + Heap)

✅ Used in:

- OSPF
- IS-IS

✅ 2. Bellman-Ford Algorithm (Distance Vector Routing)

📌 **Concept:**

- Used in **Distance Vector Routing** (e.g., RIP)
- Each node knows **only distance to neighbors**
- Periodically exchanges distance vectors

✅ Steps:

1. Initialize all distances to ∞, source = 0
2. Repeat V-1 times:
    - For each edge (u,v), update:

    ```
    if dist[u] + weight(u,v) < dist[v] then update
    ```

✅ Example:

Node A learns from B that it can reach C in 2 hops
→ A updates its route to C as `dist[B] + 1`

✅ Complexity:

- O(V × E)

✅ Problems:

- **Slow convergence**
- **Routing loops**
- **Count-to-Infinity**

🔹 Count-to-Infinity Problem

✅ Concept:

- In DVR, a bad route can keep increasing its cost indefinitely due to **slow propagation of "bad news"**

✅ Example:

- A → B → C
- C goes down
- B doesn't know yet, tells A it can still reach C
- A updates route through B
- B updates through A... loop continues

✅ Solutions:

✅ Split Horizon

- Don’t advertise a route **back** to the router from which it was learned.

```
If A learned route to C from B → A does NOT tell B about it.
```

✅ Poison Reverse

- Advertise route **with infinite metric** back to the sender to indicate it's no longer valid.

```
A tells B: "My distance to C = ∞"
```

🔹 Address Resolution

✅ ARP (Address Resolution Protocol)

📌 Purpose:

- Resolve **IP address** → **MAC address** in a local network.

✅ Working:

1. Host wants to send packet to IP `192.168.1.5`
2. Sends **ARP Request**: "Who has 192.168.1.5?"
3. Target replies with its MAC address
4. Host caches the mapping

✅ Packet Format:

- Sender MAC, Sender IP
- Target IP, Target MAC (unknown in request)

✅ ARP Table:

Stored in OS for quick lookup

```
192.168.1.5 → 00:14:22:01:23:45
```

✅ Types:

- **Gratuitous ARP**: Sent to update neighbors (e.g., after IP change)
- **Proxy ARP**: Router replies on behalf of other devices

✅ Security Issues:

- ARP spoofing → Man-in-the-middle attacks

✅ Command:

```
$ arp -a
```

✅ RARP (Reverse ARP)

📌 **Purpose:**

- Resolve **MAC address** → **IP address**
- Used by **diskless machines** during boot

✅ Working:

1. Device knows only its MAC
2. Sends RARP request: "This is my MAC, what is my IP?"
3. RARP server replies with IP

✅ Obsolete:

- Replaced by **BOOTP** and **DHCP**

🔹 Real-world Q&A

Q1: Why does RIP suffer from slow convergence?

- Uses periodic updates + no full topology knowledge → delay in detecting failures

Q2: What's the benefit of Dijkstra over Bellman-Ford?

- Dijkstra has **faster convergence** and is **loop-free**

Q3: Why is ARP needed if IP already identifies a host?

- IP is logical; for data link layer delivery, **MAC is required**

Q4: Is RARP used today?

- No, it's deprecated in favor of **BOOTP** and **DHCP**, which are more flexible

Q5: Why does split horizon prevent loops?

- It stops incorrect reverse advertisements, reducing the chance of misinformed updates

🔹 ICMP (Internet Control Message Protocol)

✅ Purpose:

ICMP is used for **diagnostics and error reporting** in IP networks.

- Defined in RFC 792
- Works alongside IP (not TCP/UDP)
- Used by tools like `ping` and `traceroute`

✅ Common ICMP Message Types:

Type	Name	Use Case
0	Echo Reply	Ping response
3	Destination Unreachable	No route, port unreachable, etc.
5	Redirect	Suggest alternate gateway
8	Echo Request	Ping request
11	Time Exceeded	Used in Traceroute

✅ Ping

📌 How it Works:

1. Sends ICMP Echo Request
2. Awaits ICMP Echo Reply
3. Measures **round-trip time (RTT)**

```
$ ping google.com

PING google.com (142.250.183.206): 56 data bytes
64 bytes from 142.250.183.206: icmp_seq=0 ttl=117 time=22.5 ms
```

✅ Traceroute

📌 How it Works:

- Sends packets with **increasing TTL** values
- Each hop returns **ICMP Time Exceeded**
- Reveals the **path** a packet takes

```
$ traceroute google.com

1  router.local (192.168.1.1)  1.2 ms
2  isp.gateway (10.0.0.1)      10.5 ms
3  ...
```

🔹 NAT (Network Address Translation)

✅ Purpose:

NAT allows **multiple devices** on a private network to share **one public IP** when accessing the internet.

- Defined in RFC 3022
- Operates at the **router boundary** between LAN and WAN

✅ Types of NAT:

Type	Description
Static NAT	One-to-one mapping between private and public IP
Dynamic NAT	Maps private IPs to any available public IP
PAT (Port Address Translation)	Many-to-one using ports

✅ Example:

Internal Device: `192.168.1.10:4321`
NAT Router maps to: `203.0.113.5:12345`
External server replies to NAT's public IP + port → NAT forwards to original host.

✅ Benefits:

- Conserves IPv4 addresses
- Adds basic security by **masking internal IPs**

✅ Limitations:

- Breaks end-to-end transparency
- Some protocols (like VoIP, FTP) need NAT traversal (e.g., STUN, TURN)

🔹 DHCP (Dynamic Host Configuration Protocol)

✅ Purpose:

DHCP automatically assigns **IP addresses and configuration parameters** to devices on a network.

- Defined in RFC 2131
- Replaces older BOOTP
- Uses **UDP (Ports 67 for server, 68 for client)**

✅ DHCP Process (DORA):

Step	Description
Discover	Client broadcasts DHCPDISCOVER
Offer	Server replies with DHCPOFFER
Request	Client sends DHCPREQUEST
Ack	Server sends DHCPACK

✅ What DHCP Assigns:

- IP address
- Subnet mask
- Gateway
- DNS servers
- Lease time

✅ Example:

```
$ ipconfig /all
IPv4 Address. . . . . . . . . . . : 192.168.1.20
DHCP Server . . . . . . . . . . . : 192.168.1.1
Lease Obtained. . . . . . . . . . : Monday, July 22
```

🔹 BOOTP (Bootstrap Protocol)

✅ Purpose:

- Older protocol used to assign **IP address and boot file** to diskless clients.

✅ Differences: BOOTP vs DHCP

Feature	BOOTP	DHCP
Static/Dynamic	Mostly static	Fully dynamic
Lease Time	No lease, permanent	Has lease expiration
Extensions	Not extensible	Supports extensions (options)
Popularity	Legacy (obsolete)	Widely used

✅ Use Cases:

- BOOTP: Legacy devices, embedded systems
- DHCP: Modern LAN/Wi-Fi setups

🔹 Real-world Q&A

Q1: Why does NAT break peer-to-peer apps?

- NAT hides internal IPs → direct incoming connections are blocked unless port forwarding is used.

Q2: Can ICMP be blocked by firewalls?

- Yes, often **ping/traceroute** fail due to ICMP filtering.

Q3: What happens if two devices have the same DHCP IP?

- A conflict occurs → one may get disconnected or auto-assigned a link-local IP (169.x.x.x)

Q4: Can a device have static IP with DHCP running?

- Yes, but IP must be outside DHCP range or conflicts may occur.

Q5: Why does traceroute show `* * *` sometimes?

- Routers may be configured to **not respond to ICMP TTL expired** messages.

✅ 5. Transport Layer – Core Concepts

🔹 Multiplexing and Demultiplexing

✅ Definition:

- **Multiplexing**: Combining data from multiple application processes at the sender and sending them through a single transport layer connection.
- **Demultiplexing**: Delivering received segments to the correct receiving application process at the receiver.

✅ Real Example:

- You have a web browser (HTTP) and a mail client (SMTP) running simultaneously.
- Both use the **same IP address**, but different **port numbers**.
- The transport layer uses port numbers to ensure correct delivery.

✅ Key Point:

- **Transport Layer** uses **port numbers** to **identify different applications** (processes).

🔹 Process-to-Process Communication

✅ Concept:

Unlike the network layer (which does **host-to-host** delivery), the transport layer provides **process-to-process** delivery (application-level communication).

✅ Analogy:

- IP = Apartment address (host)
- Port = Room number (process)

✅ Example:

A client sends HTTP request:

- IP = `172.217.167.142` (google.com)
- Port = 80 (HTTP)

The server replies:

- To client IP + random ephemeral port (e.g., 49152)

🔹 Ports and Sockets

✅ Ports

- **Port = 16-bit number** used to identify a specific process/application on a host
- Ranges:
  - **0–1023**: Well-known ports (HTTP=80, FTP=21, DNS=53)
  - **1024–49151**: Registered ports
  - **49152–65535**: Ephemeral/private ports

✅ Sockets

- A **socket** is an endpoint of a two-way communication link between two programs running on a network.
- Identified by pair

✅ Example:

Client Socket: `192.168.1.10:49152`
Server Socket: `172.217.167.142:80`

Connection: from Client Socket → Server Socket

✅ Real-world Q&A:

Q1: Why are ephemeral ports used?

- To allow multiple simultaneous client connections without conflict.

Q2: What happens if two apps try to use the same port?

- OS prevents it; port already in use error occurs.

🔹 UDP vs TCP

✅ UDP – User Datagram Protocol

Feature	Description
Connection Type	Connectionless
Reliability	No guarantees (no ACKs, retransmission)
Header Size	8 bytes
Speed	Fast
Use Cases	DNS, VoIP, video streaming, gaming

✅ TCP – Transmission Control Protocol

Feature	Description
Connection Type	Connection-oriented
Reliability	Guaranteed (ACKs, retransmissions, ordering)
Header Size	20–60 bytes
Flow Control	Sliding window
Use Cases	HTTP, FTP, SMTP, SSH

✅ TCP Features:

- **Three-way Handshake**: SYN → SYN-ACK → ACK
- **Reliable Transmission**: Lost packets are retransmitted
- **Ordered Delivery**: Segments are sequenced
- **Flow Control**: Receiver window size
- **Congestion Control**: Avoids overwhelming network

✅ UDP Features:

- **No connection establishment**
- **No sequencing or retransmission**
- Lightweight and faster

✅ Comparison Table:

Feature	TCP	UDP
Connection	Yes (3-way handshake)	No
Reliability	Reliable, ordered	Unreliable, unordered
Overhead	Higher	Low
Speed	Slower	Faster
Use Cases	Web, File Transfer	Streaming, DNS, Games
Congestion Control	Yes	No

✅ Real-world Q&A:

Q1: Why is TCP used for web traffic?

- Reliable and ordered delivery ensures correct rendering of web pages.

Q2: Can video calls use TCP?

- They can, but usually use UDP for lower latency despite potential loss.

Q3: What happens if a UDP packet is lost?

- It's discarded. No retransmission by the protocol.

Q4: Is UDP always faster than TCP?

- Yes in setup and per-packet overhead, but **can be slower** if packet loss requires app-level retransmission.

✅ 5. Transport Layer – TCP Internals

🔹 TCP – Transmission Control Protocol

✅ Overview:

TCP is a **connection-oriented**, **reliable**, **full-duplex**, and **byte-stream-based** transport protocol. It ensures:

- Reliable delivery
- Ordered delivery
- Congestion control
- Flow control

🔹 3-Way Handshake (Connection Establishment)

✅ Purpose:

Establishes a reliable connection between client and server by **synchronizing sequence numbers** and acknowledging readiness.

✅ Steps:

1. **SYN**: Client → Server
   Sends a TCP segment with `SYN=1`, `seq=x`
2. **SYN-ACK**: Server → Client
   Responds with `SYN=1`, `ACK=1`, `seq=y`, `ack=x+1`
3. **ACK**: Client → Server
   Sends `ACK=1`, `seq=x+1`, `ack=y+1`

✅ Diagram:

```
Client                                   Server
  | ---------- SYN (seq=x) --------> |
  | <------- SYN+ACK (seq=y,ack=x+1)|
  | ----------- ACK (ack=y+1) -----> |
  |----------- CONNECTION ESTABLISHED -----------|
```

🔹 4-Way Termination (Connection Teardown)

✅ Purpose:

Gracefully closes the TCP connection in both directions.

✅ Steps:

1. **FIN from Client**: Client → Server
   Requests to terminate (half-close)
2. **ACK from Server**: Server → Client
   Acknowledges client's FIN
3. **FIN from Server**: Server → Client
   Server now ready to close
4. **ACK from Client**: Client → Server
   Final ACK to complete termination

✅ Diagram:

```
Client                                Server
  | ----------- FIN -------------> |
  | <------------ ACK ------------- |
  | <------------ FIN ------------- |
  | ----------- ACK -------------> |
  |--------- CONNECTION CLOSED ------------|
```

✅ TIME_WAIT State:

- After sending final ACK, the client enters **TIME_WAIT** state for 2×MSL (Maximum Segment Lifetime) to ensure no delayed packets are misinterpreted.

🔹 TCP Header Fields

✅ Size: 20–60 bytes (without options: 20 bytes)

Field	Size	Description
Source Port	16 bits	Sender's port number
Destination Port	16 bits	Receiver's port number
Sequence Number	32 bits	Byte offset of the first byte in this segment
Acknowledgment Number	32 bits	Next byte expected by the receiver
Data Offset	4 bits	Header length
Reserved	3 bits	Reserved for future use
Flags	9 bits	Control bits (SYN, ACK, FIN, RST, PSH, URG)
Window Size	16 bits	Receiver's available buffer size
Checksum	16 bits	Error checking
Urgent Pointer	16 bits	Used if URG flag is set
Options (Optional)	Variable	E.g., for MSS, SACK, timestamps

✅ Key TCP Flags:

Flag	Description
SYN	Synchronize sequence numbers (start)
ACK	Acknowledgment field valid
FIN	Finish (terminate connection)
RST	Reset connection

Flag	Description
PSH	Push buffered data to application
URG	Urgent pointer field significant

✅ Example TCP Segment:

```
 Source Port: 12345
Dest Port: 80
Seq: 1000
Ack: 0
Flags: SYN
Window: 65535
```

This is the first segment in a 3-way handshake from client.

🔹 Real-world Q&A:

Q1: Why is 4 steps needed for termination, not 3 like handshake?

- Because TCP is **full-duplex**, each side must independently close its direction of communication.

Q2: What happens during TIME_WAIT?

- Prevents duplicate delayed packets from reappearing in a new connection.

Q3: What is MSS in TCP options?

- Maximum Segment Size – negotiated during handshake to avoid fragmentation.

Q4: Why use sequence numbers in TCP?

- To ensure **ordered delivery** and **detect packet loss or duplication**.

Q5: What's the purpose of window size?

- Supports **flow control** by informing sender of receiver's buffer availability.

✅ 5. Transport Layer – TCP Internals (Part 2)

🔹 Sequence and Acknowledgment Numbers

✅ Sequence Number

- Specifies the **byte number** of the first byte in the current segment.

- Ensures **ordered delivery** and **tracking of lost packets**.

✅ Acknowledgment Number

- Specifies the **next expected byte** from the sender.
- Implies: "I've received all bytes up to `ack-1`"

✅ Example:

If a client sends a segment with:

- `SEQ = 1000` and `LEN = 500`

Then the receiver sends:

- `ACK = 1500` (expecting next byte)

✅ Use Case in 3-Way Handshake:

- Client sends SYN with `SEQ = x`
- Server responds with `SEQ = y`, `ACK = x+1`
- Client replies with `ACK = y+1`

🔹 Congestion Control

TCP uses **congestion control** to avoid overwhelming the network. It adjusts the **congestion window (cwnd)** based on network feedback.

✅ 1. Slow Start

- Starts with small `cwnd` (e.g., 1 MSS)
- For every ACK received, **cwnd doubles** (exponential growth)
- Continues until **threshold** (ssthresh) is reached

✅ 2. Congestion Avoidance (AIMD)

- After reaching `ssthresh`, growth becomes **linear**
  - `cwnd = cwnd + MSS` per RTT
- Balances performance and congestion safety

✅ 3. Fast Retransmit

- On receiving **3 duplicate ACKs**, sender assumes packet loss
- Retransmits lost segment **without waiting for timeout**

✅ 4. Fast Recovery

- After fast retransmit, instead of going to slow start:
  - **ssthresh = cwnd / 2**
  - **cwnd = ssthresh**
  - Resume from congestion avoidance phase

✅ State Diagram:

```
[Slow Start] → (cwnd ≥ ssthresh) → [Congestion Avoidance]
      ↑                               ↓
     Loss (Timeout)          3 Duplicate ACKs
      ↑                               ↓
   Reset cwnd                     [Fast Retransmit + Recovery]
```

✅ Real-world Q&A:

Q: Why not always use large `cwnd`?

- Large window can overload routers → **packet drops**, reduced throughput

---

🔹 Flow Control using Sliding Window

✅ Purpose:

Ensure sender does not **overwhelm receiver's buffer**

✅ Mechanism:

- Receiver advertises a **window size (rwnd)** in each ACK
- Sender can only send `min(cwnd, rwnd)` worth of unACKed data

✅ Example:

```
rwnd = 4096 bytes
cwnd = 3000 bytes

→ Sender can send 3000 bytes without ACK
```

✅ Window Update:

Receiver increases window as it processes more data, allowing sender to resume transmission.

✅ Silly Window Syndrome:

- Caused by sending very small segments due to small advertised windows.
- Solved by:
    - **Nagle's Algorithm**
    - Delayed ACKs
    - Avoiding sending until window has sufficient space

🔹 RTT Estimation & Karn's Algorithm

✅ RTT (Round Trip Time):

- Time between sending a segment and receiving ACK
- Used to set **Retransmission Timeout (RTO)**

✅ Estimation Formula (RFC 6298):

```
 EstimatedRTT = (1 - α) * EstimatedRTT + α * SampleRTT
DevRTT = (1 - β) * DevRTT + β * |SampleRTT - EstimatedRTT|
RTO = EstimatedRTT + 4 * DevRTT

Typical values: α = 0.125, β = 0.25
```

✅ Karn's Algorithm

- **Problem:** Can't accurately estimate RTT for retransmitted segments (ACK might be for first or second copy)

- **Solution:**

    - **Ignore** RTT samples for retransmitted segments
    - **Double RTO** on timeout (exponential backoff)

✅ Real-world Q&A

Q1: Why is ACK number always `SEQ+LEN`?

- Because TCP is byte-oriented; ACK represents **next byte expected**

Q2: What happens if RTT is underestimated?

- Premature retransmissions → **network congestion**

Q3: Why use exponential backoff for RTO?

- To avoid flooding the network with retries when congestion is severe

Q4: Can sender send unlimited data if no congestion?

- No, limited by both **rwnd (receiver buffer)** and **cwnd (network capacity)**

✅ 5. Transport Layer – UDP, Nagle's Algorithm, Silly Window Syndrome

---

🔹 UDP – User Datagram Protocol

✅ Overview:

UDP is a **connectionless**, **unreliable**, and **lightweight** transport protocol defined in **RFC 768**.

✅ Key Characteristics:

Feature	Description
Connection	**Connectionless** (no handshake)
Reliability	No delivery guarantee
Ordering	No sequencing or reordering
Flow/Congestion Control	None
Header Size	**8 bytes**
Speed	Very fast and low latency

✅ UDP Header Format (8 bytes total):

Field	Size
Source Port	16 bits
Destination Port	16 bits
Length	16 bits
Checksum	16 bits

✅ UDP Applications:

✅ DNS (Domain Name System)

- Quick query-response model
- Typically uses UDP port **53**
- Retries using TCP only if needed (e.g., large response)

✅ VoIP (Voice over IP)

- Tolerates minor loss, prefers **low latency**
- Dropped packets are better than delayed packets

✅ **Video Streaming / Gaming**

- Real-time requirements
- Uses **application-layer logic** to handle packet loss, buffering

---

✅ Pros of UDP:

- Minimal overhead
- Works well for **broadcasting/multicasting**
- Excellent for time-sensitive applications

✅ Cons:

- No guarantee of delivery
- No flow or congestion control
- Requires extra application-layer logic for reliability (if needed)

🔹 Nagle's Algorithm

✅ Purpose:

To **reduce the number of small packets** (a.k.a. "tinygrams") sent over the network and improve **network efficiency**.

- Defined in **RFC 896**
- Used in **TCP only**

✅ How It Works:

- Send the **first segment** immediately
- Then, **buffer** small segments **until previous ACK is received**
- Coalesces small messages into larger ones

✅ Example:

Typing characters in Telnet:

- Without Nagle: One TCP segment per keystroke
- With Nagle: Combines multiple keystrokes into one packet

✅ When to Disable:

- In real-time apps like gaming or live chat
- Use TCP_NODELAY flag to disable Nagle

✅ Q&A:

Q: What happens if both sender and receiver use delayed ACK + Nagle?
A: Can cause deadlock or high latency (known as **"ACK delay + Nagle interaction problem"**)

🔹 Silly Window Syndrome (SWS)

✅ Problem:

Occurs when sender or receiver transmits **very small segments** due to **small window size**, leading to **inefficient bandwidth usage**.

✅ Causes:

- **Sender-side SWS:** Sender keeps sending 1-byte packets
- **Receiver-side SWS:** Receiver advertises window size in very small increments

---

✅ Example:

- App generates 1 byte every few milliseconds
- TCP sends each byte as its own packet → waste of header space

---

✅ Solutions:

✅ At Sender:

- Use **Nagle's Algorithm**: Buffer small data until ACK is received

✅ At Receiver:

- Use **receiver-side flow control policy**:
    - Don't advertise small window sizes
    - Wait for a threshold before updating window

✅ App Layer:

- Accumulate data before writing to socket

✅ Comparison: Nagle vs SWS

Feature	Nagle's Algorithm	Silly Window Syndrome
Side Affected	Sender	Both sender & receiver
Purpose	Reduce small segment transmission	Prevent inefficient small windows

Feature	Nagle's Algorithm	Silly Window Syndrome
Solution For	TCP overhead	Poor flow control

🔹 Real-world Q&A

Q1: Why isn't UDP used for file transfers?

- No reliability or sequencing → unsuitable for lossless delivery

Q2: Can Nagle's Algorithm improve performance in HTTP?

- Yes, for bulk transfers; but in HTTP/2 or latency-critical cases, it's usually disabled

Q3: What does `TCP_NODELAY` do?

- Disables Nagle's Algorithm; sends packets immediately

Q4: Can SWS occur even with large bandwidth?

- Yes, if app sends/receives in tiny chunks, SWS still wastes network resources

---

✅ 6. Application Layer – DNS & HTTP/HTTPS

🔹 DNS (Domain Name System)

✅ Purpose:

DNS translates **human-readable domain names** (e.g., google.com) into **IP addresses** (e.g., 142.250.68.14).

✅ How It Works:

1. **Browser cache** check
2. **OS cache**
3. **DNS Resolver** query
4. Resolver contacts:
    - **Root DNS server**
    - **TLD server** (e.g., `.com`)
    - **Authoritative DNS server**
5. Final IP is returned and cached

✅ DNS Record Types:

Type	Description
A	Maps hostname to IPv4 address
AAAA	Maps hostname to IPv6 address
CNAME	Canonical name (alias)
MX	Mail server
NS	Authoritative name server
PTR	Reverse DNS
TXT	Human-readable text

✅ Example Query:

```
$ nslookup google.com
Name:    google.com
Address: 142.250.183.238
```

✅ DNS Port:

- UDP **53** (TCP for large queries or zone transfers)

✅ DNS Issues:

- **Spoofing/poisoning**
- **Delay in recursive lookups**

🔹 HTTP & HTTPS

✅ HTTP – HyperText Transfer Protocol

- Application-layer **stateless protocol**
- Runs over **TCP port 80**

✅ HTTPS – HTTP Secure

- Uses **TLS/SSL encryption** over TCP
- Runs on **port 443**
- Ensures:
  - **Confidentiality** (encryption)
  - **Integrity** (MAC)
  - **Authentication** (certificates)

🔹 HTTP Methods

Method	Purpose
GET	Retrieve data (idempotent)
POST	Submit data (non-idempotent)
PUT	Update/replace resource
DELETE	Delete a resource
HEAD	Same as GET but without body
OPTIONS	Query supported operations
PATCH	Partial update to resource

✅ Example:

```
 GET /index.html HTTP/1.1
Host: www.example.com
```

🔹 Persistent vs Non-Persistent Connections

✅ Non-Persistent (HTTP/1.0):

- **1 TCP connection per object**
- Inefficient for modern web (many resources per page)

✅ Persistent (HTTP/1.1+):

- **One TCP connection reused** for multiple requests/responses
- Uses `Connection: keep-alive`

🔹 HTTP Versions

✅ HTTP/1.1

- Default persistent connections
- Pipelining supported (but rarely used)
- **Head-of-Line (HOL) blocking**

✅ HTTP/2

- **Binary framing layer**
- **Multiplexed streams** on a single TCP connection
- Header compression (HPACK)
- Still vulnerable to **HOL blocking** at TCP level

✅ HTTP/3

- Uses **QUIC** instead of TCP (based on UDP)
- Removes HOL blocking entirely
- Faster connection setup with **0-RTT handshakes**
- Built-in encryption (TLS 1.3)

✅ Comparison Table:

Feature	HTTP/1.1	HTTP/2	HTTP/3
Transport Layer	TCP	TCP	**QUIC (UDP)**
Multiplexing	No	Yes	Yes
HOL Blocking	Yes	Yes (at TCP)	**No**
Encryption	Optional (via TLS)	Mandatory (TLS)	Mandatory (TLS 1.3)
Performance	Moderate	High	**Very High**

🔹 Real-world Q&A

Q1: Why is DNS critical to the internet?

- Without DNS, users must remember IPs → impractical and unscalable.

Q2: How does HTTP/2 solve HTTP/1.1's problems?

- Enables multiplexing, reducing latency caused by multiple connections.

Q3: Why does HTTP/3 use QUIC?

- To eliminate TCP's HOL blocking and enable faster encrypted handshakes over UDP.

Q4: Why is POST not idempotent?

- Because it can create or update resources multiple times if retried.

Q5: What happens if DNS fails?

- Browsers cannot resolve hostnames → websites fail to load.

✅ 6. Application Layer – Protocols & Encryption

🔹 FTP vs SFTP

✅ FTP (File Transfer Protocol)

- Transfers files over **TCP port 21**
- **Unencrypted**: Sends data, usernames, and passwords in plaintext
- Uses separate control and data channels (active/passive modes)

❌ **Drawbacks:**

- Vulnerable to **MITM attacks**, sniffing

✅ SFTP (SSH File Transfer Protocol)

- Runs over **SSH (port 22)**
- Fully **encrypted** file transfer protocol
- Supports authentication, directory listing, file permission changes

---

✅ Comparison:

Feature	FTP	SFTP
Port	21	22
Encryption	None	End-to-end (via SSH)
Security	Weak	Strong
Auth Method	Plaintext login	SSH keys or credentials

🔹 Email Protocols – SMTP, POP3, IMAP

✅ SMTP (Simple Mail Transfer Protocol)

- **Used for sending** emails (client to server, server to server)
- TCP **port 25**, **587** (with authentication)
- Push protocol

✅ POP3 (Post Office Protocol v3)

- Used to **retrieve emails**
- Downloads and **deletes from server**
- TCP **port 110**
- Simple, one-device usage

✅ IMAP (Internet Message Access Protocol)

- Accesses email **without deleting** from server
- Allows multiple device sync
- TCP **port 143**, or **993** with SSL
- Complex and modern

✅ Email Flow Diagram:

```
[Client] --SMTP--> [Mail Server] --SMTP--> [Recipient Mail Server]
                                      ↓
                    (via POP3/IMAP)
                    [Recipient Client]
```

✅ Comparison Table:

Protocol	Role	Port	Deletes Email	Multi-device	Encryption
SMTP	Send	25/587	N/A	Yes	Optional
POP3	Receive	110	Yes	No	Optional
IMAP	Receive	143/993	No	Yes	Recommended

🔹 Telnet vs SSH

✅ Telnet

- Terminal emulation protocol for **remote login**
- TCP **port 23**
- **No encryption**, insecure

✅ SSH (Secure Shell)

- Encrypted remote shell over TCP **port 22**
- Uses **public-key cryptography**
- Can forward X11, port tunneling, and secure file transfers (SFTP, SCP)

✅ Telnet vs SSH:

Feature	Telnet	SSH
Port	23	22
Encryption	None	End-to-end

Feature	Telnet	SSH
Authentication	Basic login	Public key / password
Usage	Obsolete	Widely used

🔹 DHCP (App Layer View)

✅ DHCP (Dynamic Host Configuration Protocol)

- Assigns **IP address**, **subnet mask**, **default gateway**, **DNS server**, etc.
- Uses UDP ports: **67 (server)**, **68 (client)**
- Located at **Application Layer**, but configures **Network Layer parameters**

✅ Lifecycle (DORA):

1. **DHCPDISCOVER**: Client → broadcast
2. **DHCPOFFER**: Server → proposed config
3. **DHCPREQUEST**: Client accepts offer
4. **DHCPACK**: Server confirms

✅ Lease:

- IP address is temporary (renewable)
- DHCP server manages lease duration and reuse

✅ Benefits:

- Plug-and-play network config
- Centralized IP management

🔹 SSL/TLS – Basics of Encryption

✅ SSL/TLS Overview:

- **SSL (Secure Sockets Layer)** → obsolete
- **TLS (Transport Layer Security)** → modern encryption protocol
- Works between **Transport and Application layers**

✅ Goals:

- **Confidentiality** – Encrypt data
- **Integrity** – Prevent tampering (MAC)
- **Authentication** – Via digital certificates

✅ TLS Handshake (Simplified):

1. **Client Hello**: Sends supported ciphers, TLS version
2. **Server Hello**: Sends certificate and chosen cipher
3. **Key Exchange**: Via RSA/DH/DHE/ECDHE
4. **Session Keys** established and encrypted communication begins

---

✅ Protocol Usage:

Protocol	Encrypted Version
HTTP	HTTPS
SMTP	SMTPS
FTP	FTPS / SFTP
Telnet	SSH

✅ Real-world Q&A

Q1: Why is FTP insecure?

- Sends credentials and data in plaintext

Q2: Why use IMAP over POP3?

- IMAP supports **multi-device sync** and doesn't delete server copy

Q3: What does SSH secure besides login?

- File transfer (SFTP/SCP), port forwarding, remote shell

Q4: Why is DHCP at the application layer?

- It configures IP and routing but uses application-level protocols and format

Q5: Why is TLS preferred over SSL?

- SSL has known vulnerabilities; TLS is **secure, faster, and modern**

✅ 7. Congestion Control & Quality of Service (QoS)

🔹 Congestion vs Flow Control

✅ Congestion Control

- A **network-wide** mechanism to prevent **too much traffic** from degrading performance.
- Managed at **transport/network layer**
- Aims to prevent **buffer overflows**, **packet drops**, and **increased latency** in routers/switches.

✅ Flow Control

- A **sender-receiver mechanism** to ensure the **sender does not overwhelm the receiver's buffer**.
- Managed at **transport layer (e.g., TCP sliding window)**
- Deals with **end-to-end communication**, not intermediate devices.

✅ Comparison Table:

Feature	Flow Control	Congestion Control
Scope	End-to-end	Entire network
Trigger	Receiver's buffer size	Network congestion (e.g., router queue)
Protocol Layer	Transport	Transport/Network
Examples	TCP sliding window	TCP congestion window, ECN

🔹 Leaky Bucket Algorithm

✅ Purpose:

Provides a **constant rate output** regardless of bursty input. It smoothens traffic and prevents congestion.

✅ Working:

- Uses a **bucket (queue)** to store incoming packets.
- Packets are **removed at a fixed rate**.
- If bucket overflows → **packets are dropped**.

✅ Analogy:

- Imagine pouring water (packets) into a bucket with a hole.
- Water leaks at constant rate.
- If poured too fast → overflow (packet drop).

✅ Diagram (Conceptual):

```
[Incoming Packets] ---> [Bucket] ---> [Fixed Rate Outflow]
                           |
```

```
                     ↓
                Drop if full
```

✅ Pros:

- Simple and effective in **traffic shaping**
- Enforces **rate limiting**

✅ Cons:

- Does **not allow bursts**, even when network is idle

🔹 Token Bucket Algorithm

✅ Purpose:

Allows **controlled bursts** while maintaining average rate. Common in **modern traffic shaping** and **QoS systems**.

✅ Working:

- Tokens are added to a **bucket at fixed rate**
- Each packet requires a token to be transmitted
- If enough tokens → send burst
- If not enough → wait (or drop)

✅ Analogy:

- Bucket holds tokens (permissions to send)
- If tokens exist, data can go through
- Tokens accumulate when idle → burst allowed later

✅ Diagram (Conceptual):

```
 [Token Generator] --> [Token Bucket]
       |                     |
[Packets Arrive] --------> [Send if token available]
                               ↓
                         Wait or drop if no token
```

✅ Pros:

- **Supports bursts**
- Enforces average rate over time
- More flexible than leaky bucket

✅ Comparison Table:

Feature	Leaky Bucket	Token Bucket
Output Rate	Constant	Variable (bursty allowed)
Token Concept	No	Yes
Drop Policy	Drop when bucket full	Wait or drop if no token
Use Case	Traffic smoothing	Traffic shaping with bursts

🔹 Real-world Q&A

Q1: Why do we need congestion control in TCP?

- To avoid overloading the network, which causes packet loss, retransmission, and delay.

Q2: Why is token bucket better for real-time applications?

- Allows bursts (e.g., video/audio packets) while maintaining rate limits.

Q3: Can you use both leaky and token bucket together?

- Yes. Token bucket for shaping + leaky bucket for smoothing output.

Q4: Which one is better for strict bandwidth limiting?

- **Leaky bucket** – ensures steady output rate.

Q5: Does flow control prevent congestion?

- **No**, it only ensures the receiver is not overloaded, not the network.

✅ 7. Congestion Control & Quality of Service (QoS) – Part 2

🔹 QoS Metrics

Quality of Service (QoS) defines the overall performance of a network, particularly in terms of predictable delivery and performance for specific traffic types.

✅ 1. Bandwidth

- Maximum rate of data transfer over a network path.
- Measured in **bps (bits per second)**.
- Higher bandwidth = more simultaneous traffic.

✅ 2. Delay (Latency)

- Time taken for a packet to travel from source to destination.
- Includes:
  - **Processing delay**
  - **Queueing delay**
  - **Transmission delay**
  - **Propagation delay**

✅ 3. Jitter

- Variation in delay of received packets.
- Critical in **real-time applications** like VoIP, video conferencing.
- High jitter = choppy audio/video.

✅ 4. Packet Loss

- Packets dropped due to:
  - Congestion
  - Buffer overflow
  - Corruption or timeout
- Leads to retransmissions, reduced throughput

✅ Summary Table:

Metric	Unit	Affects	Importance In
Bandwidth	Mbps/Gbps	Speed	All traffic
Delay	ms	Responsiveness	Gaming, VoIP
Jitter	ms	Smooth playback	VoIP, video streaming
Packet Loss	%	Reliability	TCP (retransmission), UDP (drop)

🔹 Bufferbloat

✅ Definition:

Bufferbloat is the excessive delay caused by **large network buffers** holding too many packets during congestion.

✅ Cause:

- Buffers in routers/switches are too deep.
- Instead of dropping packets, they queue them, causing **high latency**.

✅ Effects:

- Increased ping times
- Lag in interactive apps (e.g., Zoom, gaming)
- TCP sees no packet loss → doesn't trigger congestion control

---

✅ Detection:

```
$ ping -f google.com
→ Watch for increasing delay under load
```

✅ Solution:

- **Smaller buffers**
- **Active Queue Management (AQM)**: e.g., RED, CoDel

🔹 RED (Random Early Detection)

✅ Purpose:

Preemptively drops packets to **signal congestion** before buffers overflow.

✅ How RED Works:

1. Monitors **average queue size**
2. If below minimum threshold → accept all packets
3. If between min & max → **drop packets probabilistically**
4. If above max threshold → **drop all packets**

✅ RED vs Tail Drop:

Feature	Tail Drop	RED
Drop Timing	Only when buffer full	Before buffer overflows
Drop Behavior	Sudden	Gradual, probabilistic
TCP Reaction	All flows affected	Spreads loss across flows

✅ Benefits:

- Reduces global synchronization of TCP flows
- Prevents bufferbloat
- Encourages early congestion control

🔹 Explicit Congestion Notification (ECN)

✅ Purpose:

Allows routers to **signal congestion without dropping packets**.

✅ How It Works:

1. Routers mark packets with **ECN bits** in IP header if congestion is detected.
2. Receiver echoes ECN mark back to sender.
3. Sender reduces congestion window as if a packet loss occurred.

✅ ECN Bits in IP Header:

ECN Bits	Meaning
00	Not ECN Capable
10	ECN Capable
11	Congestion Experienced (CE)

✅ Requirements:

- Both sender and receiver must **support ECN**
- Supported in **TCP/IP stack and routers**

✅ Benefits:

- Maintains throughput
- Avoids packet loss
- Works well with RED

🔹 Real-world Q&A

Q1: Why is jitter more critical than latency for VoIP?

- Even if packets are delayed, consistent delay is tolerable → **variation causes audio breaks**

Q2: Why is RED preferred over tail drop?

- Prevents **queue buildup** and global synchronization in TCP flows

Q3: What makes bufferbloat hard to detect?

- No packet drops → TCP doesn't slow down → only visible as latency increase

Q4: Why does ECN require end-to-end support?

- Sender must understand ECN flags and reduce congestion window accordingly

Q5: How do RED and ECN work together?

- RED detects congestion → ECN marks instead of dropping → TCP reduces rate smoothly

---

✅ 8. Switching & Routing Devices

🔹 Hub vs Switch vs Router vs Gateway vs Bridge vs Modem

✅ Hub

- **Layer**: Physical (Layer 1)
- **Function**: Broadcasts incoming signal to **all ports**
- No MAC address learning
- **No filtering** or collision handling
- Rarely used today

✅ Switch

- **Layer**: Data Link (Layer 2)
- **Function**: Learns **MAC addresses**, forwards frames only to the correct port
- Reduces collisions, increases efficiency
- Supports **full-duplex communication**

✅ Router

- **Layer**: Network (Layer 3)
- **Function**: Routes packets between **different networks**
- Uses **IP addresses**
- Performs **NAT**, filtering, path selection

✅ Gateway

- **Layer**: All layers (typically Layer 7)
- **Function**: Connects **two dissimilar networks** (e.g., VoIP-to-PSTN)
- Performs **protocol conversion**

✅ Bridge

- **Layer**: Data Link (Layer 2)
- **Function**: Connects **two LAN segments** and filters traffic using MAC
- Smarter than a hub but simpler than a switch

✅ Modem

- **Function**: Converts **digital** → **analog** and vice versa
- Used to connect to ISP over telephone or cable
- Modem = **MO**dulator + **DEM**odulator

✅ Comparison Table:

Device	OSI Layer	Works on	Use Case
Hub	1	Bits	Obsolete, basic signal forwarding
Switch	2	MAC Addr	LAN frame forwarding
Router	3	IP Addr	LAN-to-WAN or internet routing
Bridge	2	MAC Addr	Segment traffic within LAN
Gateway	All	Protocol	Protocol conversion, app-level
Modem	Physical	Signals	Internet access (DSL, Cable)

🔹 Layer 2 vs Layer 3 Switches

✅ Layer 2 Switch

- Operates at **Data Link layer**
- Uses **MAC address table**
- Forwards **frames within the same network**

✅ Layer 3 Switch

- Operates at **Network layer**
- Performs **routing between VLANs/subnets**
- Uses **IP routing table**
- Faster than routers (hardware-based switching)

✅ Comparison:

Feature	Layer 2 Switch	Layer 3 Switch
Layer	Data Link (L2)	Network (L3)
MAC Learning	Yes	Yes
IP Routing	No	Yes
Speed	Very fast	Fast (hardware routed)

Feature	Layer 2 Switch	Layer 3 Switch
Use Case	Intra-VLAN switching	Inter-VLAN routing

🔹 NAT Routers

✅ NAT (Network Address Translation) Router

- Translates **private IP** ↔ **public IP**
- Maintains mapping of **internal IP:port** → **external IP:port**
- Enables multiple devices to share a **single public IP**

✅ Types:

- **Static NAT**: One-to-one mapping
- **Dynamic NAT**: Uses a pool of public IPs
- **PAT (Port Address Translation)**: Many-to-one using ports

---

✅ NAT Table Example:

Private IP:Port	Public IP:Port
192.168.1.10:50234	203.0.113.5:40001
192.168.1.11:50235	203.0.113.5:40002

🔹 Load Balancers

✅ Purpose:

Distribute incoming network traffic **across multiple servers** to improve reliability, throughput, and availability.

✅ Types:

- **Layer 4 Load Balancer**: Operates at transport layer (TCP/UDP)
  - Uses IP + Port
  - Faster, limited visibility
- **Layer 7 Load Balancer**: Operates at application layer (HTTP/HTTPS)
  - Understands URL, headers, cookies
  - Enables routing based on content

✅ Load Balancing Algorithms:

Algorithm	Description
Round Robin	Distributes requests in rotation
Least Connections	Server with fewest active connections
IP Hash	Same client IP always hits same server

🔹 Firewalls (Stateful vs Stateless)

✅ Firewall

- A security system that **monitors and filters** network traffic based on rules.

✅ Stateless Firewall

- Filters packets based **only on headers** (e.g., IP, port, protocol)
- Does **not track connection state**
- Faster, but less secure

✅ Stateful Firewall

- Tracks **connection state** (e.g., part of TCP handshake?)
- Allows only **valid connections**
- Provides **better security** for modern apps

✅ Comparison:

Feature	Stateless Firewall	Stateful Firewall
Layer	Layer 3–4	Layer 3–7
Tracks Sessions?	No	Yes
Performance	High (lightweight)	Moderate
Security	Basic	Stronger
Use Case	Simple filtering (routers)	Enterprise-grade firewalls

---

🔹 Real-world Q&A

Q1: Why use a switch over a hub?

- A switch reduces collisions by forwarding only to the target port; a hub broadcasts to all.

Q2: Can routers perform switching?

- Yes, modern routers have **built-in switches** for LAN.

Q3: Why is a Layer 3 switch faster than a router?

- Layer 3 switches perform routing using **hardware (ASICs)**, not software.

Q4: What happens if NAT router crashes?

- All devices behind it lose internet access as NAT mappings are lost.

Q5: Why are stateful firewalls preferred?

- They understand connection context and prevent spoofed/malformed packet attacks.

✅ 9. Wireless & Mobile Networks

🔹 Mobile IP & Handoff

✅ Mobile IP

- Protocol that allows users to **move across networks** while maintaining **the same IP address**.
- Introduces 3 components:
  1. **Home Agent (HA)**: On home network
  2. **Foreign Agent (FA)**: On visited network
  3. **Care-of Address (CoA)**: Temporary address at new location

---

✅ Working:

1. Mobile device moves to a foreign network.
2. FA assigns a CoA and registers with HA.
3. HA tunnels packets to the CoA.
4. Replies are sent directly back to sender.

✅ Handoff (Handover)

- Process of transferring an active session (e.g., call, data) from one cell/tower to another.

Types:

- **Hard Handoff**: Break before make (used in GSM)
- **Soft Handoff**: Make before break (used in CDMA)

🔹 Wi-Fi Architecture (BSS, ESS)

✅ Basic Service Set (BSS)

- A group of devices communicating via **one Access Point (AP)**.
- Identified by **BSSID** (MAC of AP).

✅ Extended Service Set (ESS)

- Multiple BSSs interconnected via a **Distribution System (DS)** (usually Ethernet).
- Identified by a common **SSID** (network name).
- Enables **roaming** within a Wi-Fi network.

✅ Diagram:

```
[BSS1: AP1] ---+
               \
                [Router/Switch] -- Internet
               /
[BSS2: AP2] ---+

= ESS (Common SSID)
```

🔹 Hidden Terminal & Exposed Terminal

✅ Hidden Terminal Problem

- Nodes **A and C can't sense each other**, both send to B → collision.

```
A --- B --- C

A and C are hidden from each other
```

Solution: RTS/CTS handshake (used in CSMA/CA)

✅ Exposed Terminal Problem

- Node **B wants to send to A**, but senses **C is sending to D** and waits unnecessarily.

```
A --- B     C --- D

B is exposed to C's transmission
```

Solution: Allow transmission if **destinations are different**

🔹 Bluetooth, RFID, ZigBee

✅ Bluetooth

- Short-range wireless tech (10m)
- Based on **IEEE 802.15.1**
- Used for peripherals (headphones, mice)

- Topology: **Piconet**, **Scatternet**

✅ RFID (Radio Frequency Identification)

- Uses **radio waves** to identify and track tags
- Passive (no battery) or Active
- Used in inventory, tolls, IDs

✅ ZigBee

- Low-power, low-data-rate wireless standard
- Based on **IEEE 802.15.4**
- Used in **IoT, home automation, sensors**

🔹 Cellular Networks: 1G to 5G

Generation	Key Tech	Speed	Features
1G	Analog Voice	~2.4 Kbps	Analog calls only
2G	GSM, CDMA	~64 Kbps	SMS, digital voice
3G	UMTS, HSPA	~2 Mbps	Mobile Internet
4G	LTE, WiMAX	~100 Mbps	HD streaming, VoIP
5G	NR, mmWave	~10 Gbps	IoT, ultra-low latency, slicing

🔹 802.11 Wi-Fi Standards

Standard	Frequency	Max Speed	Notes
802.11a	5 GHz	54 Mbps	Shorter range, less interference
802.11b	2.4 GHz	11 Mbps	Longer range, more interference
802.11g	2.4 GHz	54 Mbps	Compatible with b
802.11n	2.4/5 GHz	600 Mbps	MIMO support
802.11ac	5 GHz	~1.3 Gbps	Beamforming, wider channels
802.11ax	2.4/5/6 GHz	~10 Gbps	OFDMA, MU-MIMO, Wi-Fi 6

✅ Real-world Q&A

Q1: Why do mobile IP packets experience triangular routing?

- Packets first go to HA → FA → MN, instead of directly to MN.

Q2: Why is RTS/CTS used in wireless?

- To mitigate **hidden terminal** issues by reserving the medium.

Q3: What's the benefit of 802.11ax over 802.11ac?

- Better concurrency, lower latency, **OFDMA**, supports dense environments.

Q4: How does ZigBee differ from Wi-Fi?

- ZigBee is **low power, low data rate**, Wi-Fi is **high throughput**.

Q5: What does "handoff" ensure in cellular networks?

- **Seamless connectivity** while moving between towers.

✅ 10. Network Security

🔹 Cryptography Basics

✅ Symmetric Encryption

- Same key used for both **encryption and decryption**
- Faster, used for **bulk data encryption**

🔒 **Examples:**

- AES (Advanced Encryption Standard)
- DES, 3DES
- RC4 (stream cipher)

✅ Asymmetric Encryption

- Uses a **public key** (encrypt) and a **private key** (decrypt)
- Slower, used for **key exchange and digital signatures**

🔐 **Examples:**

- RSA
- ECC (Elliptic Curve Cryptography)
- Diffie-Hellman (key exchange)

✅ Comparison Table:

Feature	Symmetric	Asymmetric
Key Type	Same key	Public / Private pair

Feature	Symmetric	Asymmetric
Speed	Faster	Slower
Use Case	Data encryption	Key exchange, identity
Example	AES	RSA, ECC

🔹 SSL/TLS & HTTPS

✅ SSL (Secure Sockets Layer)

- Legacy protocol (deprecated)
- Replaced by TLS

✅ TLS (Transport Layer Security)

- Provides **encryption, authentication, and integrity**
- Used in HTTPS, FTPS, SMTPS, etc.

✅ TLS Handshake Overview:

1. **Client Hello**: TLS version, cipher suites
2. **Server Hello**: Certificate, selected cipher
3. **Key Exchange** (DH/RSA)
4. **Session Keys** derived
5. **Encrypted communication begins**

✅ HTTPS

- HTTP over TLS
- Ensures **confidentiality** of web traffic
- Uses **port 443**
- Requires SSL/TLS certificate (e.g., from Let's Encrypt)

🔹 Firewalls, IDS, IPS

✅ Firewalls

- Monitor and **filter incoming/outgoing traffic** based on rules
- Types:
    - **Stateless**: Packet filters
    - **Stateful**: Connection-aware
    - **Application Firewall**: Layer 7 inspection

✅ IDS (Intrusion Detection System)

- **Detects** suspicious activity
- Does **not block** traffic
- Can be host-based (HIDS) or network-based (NIDS)

✅ IPS (Intrusion Prevention System)

- **Detects and blocks** malicious traffic
- Sits **inline** with network flow
- Can terminate or reroute suspicious traffic

✅ Comparison Table:

Feature	Firewall	IDS	IPS
Function	Filter traffic	Detect intrusion	Detect + Prevent
Inline?	Yes	No (passive)	Yes
Response	Allow/Deny	Alert only	Block/Drop

🔹 VPN (Virtual Private Network)

✅ Purpose:

- Creates an **encrypted tunnel** over the public internet.
- Allows secure access to private networks remotely.

✅ How VPN Works:

1. Client initiates VPN connection (via VPN software)
2. Tunnel is established (IPSec, SSL, L2TP)
3. All traffic is **encrypted** and routed via VPN server

✅ Protocols Used:

Protocol	Description
PPTP	Fast but insecure (legacy)
L2TP	Often paired with IPSec
IPSec	Secure at network layer
OpenVPN	Open-source, TLS-based
WireGuard	Lightweight, modern protocol

✅ Benefits:

- **Data confidentiality** on public Wi-Fi
- **Bypass geo-restrictions**
- **Mask IP address**
- Enables secure **remote work access**

🔹 Real-world Q&A

Q1: Why is asymmetric encryption slower than symmetric?

- It involves **more complex math operations** (modular exponentiation).

Q2: When does HTTPS use both symmetric and asymmetric encryption?

- TLS handshake uses **asymmetric** for key exchange, then **symmetric** for data.

Q3: Why is a firewall not sufficient on its own?

- It doesn't detect **zero-day or internal threats** → need IDS/IPS.

Q4: How does a VPN secure public Wi-Fi usage?

- Encrypts traffic between user and VPN server, protecting from eavesdroppers.

Q5: What's the difference between IDS and IPS in placement?

- IDS is **out-of-band (passive)**, IPS is **inline (active prevention)**.

✅ 10. Network Security (Part 2)

🔹 IPsec (Internet Protocol Security)

✅ Definition:

- A **suite of protocols** for securing IP communication via:
  - **Authentication**
  - **Integrity**
  - **Confidentiality**

✅ Protocol Modes:

1. **Transport Mode**: Encrypts only the **payload** of the IP packet.
2. **Tunnel Mode**: Encrypts the **entire IP packet** (used in VPNs).

✅ Core Protocols:

Protocol	Role
AH	Authentication Header – integrity + auth, no encryption
ESP	Encapsulating Security Payload – provides encryption + integrity
IKE	Internet Key Exchange – negotiates keys for IPsec sessions

---

✅ Use Cases:

- Site-to-site VPN
- Remote-access VPN
- Secure communication between routers/gateways

✅ Diagram:

```
[Sender] --[IPsec (ESP/AH)]--> [Receiver]
     ↓                            ↑
   IKE for key negotiation and SA establishment
```

🔹 DoS vs DDoS Attacks

✅ DoS (Denial of Service)

- Attacker floods a server with **requests** to exhaust resources.
- Targets **availability** of a service.

✅ DDoS (Distributed Denial of Service)

- Same as DoS but launched from **multiple compromised systems** (botnets).
- More powerful and **harder to mitigate**.

✅ Attack Types:

Type	Description
SYN Flood	Half-open TCP connections
UDP Flood	Overwhelms with UDP packets
HTTP Flood	High-level request overload
ICMP Flood	Ping of death, smurf attacks

✅ Mitigations:

- Rate limiting
- CAPTCHAs
- Traffic filtering
- DDoS protection services (e.g., Cloudflare)

🔹 Spoofing, Sniffing, MITM

✅ Spoofing

- Forging identity, typically:
  - **IP spoofing**: Fake source IP
  - **Email spoofing**
  - **MAC spoofing**

✅ Sniffing

- Capturing network packets using tools like:
  - **Wireshark**
  - **tcpdump**
- Can extract passwords, credentials in plaintext networks

✅ Man-in-the-Middle (MITM)

- Attacker intercepts communication between two parties.

Common techniques:

- ARP spoofing
- DNS spoofing
- SSL stripping

✅ Example:

```
 [Client] <---> [Attacker] <---> [Server]

Attacker reads/modifies data
```

✅ Mitigations:

- Encryption (TLS/HTTPS)
- VPNs
- ARP inspection
- DNSSEC

🔹 Authentication Protocols

✅ Kerberos

- **Ticket-based authentication system**
- Used in enterprise Windows domains

Components:

- **KDC** (Key Distribution Center)
- **TGT** (Ticket Granting Ticket)

Flow:

1. Login → KDC gives TGT
2. TGT → service ticket for app access

✅ OAuth (Open Authorization)

- Authorization framework for **third-party access** without sharing passwords.

Roles:

- **Resource Owner** (user)
- **Client** (3rd party app)
- **Authorization Server**
- **Resource Server**

Example:

- You log into a website using **Google/Facebook**

✅ Other Protocols:

Protocol	Purpose
SAML	SSO for web-based apps
OpenID	Federated identity protocol
RADIUS	Centralized auth for networks
LDAP	Directory-based authentication
TACACS+	Cisco protocol for AAA

🔹 Real-world Q&A

Q1: Why use IPsec instead of SSL?

- IPsec works at **network layer**, transparent to applications; SSL is application-layer.

Q2: How does a DDoS attack differ from a high-traffic day?

- DDoS is **malicious**, typically **uniform, unresponsive, and uncontrollable**.

Q3: How can sniffing be detected?

- Monitor for **promiscuous mode** NICs, unusual traffic, or use IDS.

Q4: Why is OAuth considered secure?

- Uses **access tokens** and scopes to limit what the third party can do.

Q5: Why is Kerberos better than password-based login?

- Uses **time-limited tickets** and avoids transmitting passwords directly.

✅ 11. Performance & Monitoring

🔹 Key Performance Metrics

Understanding the metrics below is essential for analyzing and optimizing network performance.

✅ 1. Bandwidth

- **Definition**: Maximum amount of data that can be transferred over a network path per unit time.
- **Measured in**: bps (bits per second), Kbps, Mbps, Gbps.
- **Indicates**: The *capacity* of the link.

✅ 2. Throughput

- **Definition**: Actual rate at which data is successfully transferred.
- Always **≤ Bandwidth**, affected by network congestion, retransmissions, and protocol overhead.

  Example:
  Link Bandwidth = 100 Mbps
  Measured Throughput = 70 Mbps (due to retransmissions, latency)

✅ 3. Latency (Delay)

- **Definition**: Time taken by a packet to travel from sender to receiver.
- Composed of:
  - **Transmission Delay**: Size / Bandwidth
  - **Propagation Delay**: Distance / Speed of signal
  - **Processing Delay**: Router processing time
  - **Queueing Delay**: Waiting in buffer

✅ 4. Jitter

- **Definition**: Variation in delay between packets arriving.
- High jitter = uneven playback in VoIP/video.

  Example:
  Packet 1 arrives in 10 ms, Packet 2 in 40 ms → Jitter = 30 ms

---

✅ 5. Packet Loss

- **Definition**: Percentage of packets lost or dropped during transmission.
- Causes: Congestion, faulty hardware, buffer overflow.
- High loss severely affects real-time protocols (UDP, VoIP).

✅ Summary Table:

Metric	Unit	Affects	Related Tools
Bandwidth	Mbps/Gbps	Capacity	`iperf`, SNMP
Throughput	Mbps	Real transfer rate	`iperf`, `netstat`
Latency	ms	Responsiveness	`ping`, `traceroute`
Jitter	ms	Real-time traffic	`jitterbug`, `Wireshark`
Packet Loss	%	Reliability	`ping`, `mtr`

🔹 MTU (Maximum Transmission Unit)

✅ Definition:

- Maximum size (in bytes) of a data packet that can be sent in a single frame without fragmentation.
- Common default MTU for Ethernet: **1500 bytes**

✅ Why It Matters:

- If a packet is larger than MTU → **fragmentation** occurs
- Too small MTU → more packets → overhead
- Too large MTU → risk of fragmentation or drop

✅ Tools:

```
# Discover MTU without fragmentation
ping -M do -s 1472 google.com
```

- 1472 + 28 (IP + ICMP header) = 1500

🔹 RTT (Round Trip Time)

✅ Definition:

- Time taken for a signal to travel from source → destination → back to source.
- Includes forward + reverse propagation + processing delays.

✅ Measured With:

```
ping google.com
```

Output: `64 bytes from ...: icmp_seq=1 ttl=56 time=22.1 ms`
→ RTT ≈ 22.1 ms

✅ Uses:

- Network latency measurement
- TCP congestion control (e.g., RTO estimation)
- CDN node selection

✅ RTT vs Latency:

Metric	Direction	Includes ACK?	Use Case
Latency	One-way (theoretical)	No	Delay analysis
RTT	Round trip	Yes	TCP timeout, diagnostics

🔹 Real-world Q&A

Q1: Why is throughput less than bandwidth?

- Due to **network overhead**, retransmissions, protocol inefficiencies.

Q2: How to reduce jitter in VoIP?

- Use **jitter buffers**, prioritize traffic (QoS), reduce hops.

Q3: What causes packet loss?

- Congestion, poor signal (wireless), faulty NICs, buffer overflows.

Q4: Why is MTU tuning important?

- To avoid **fragmentation**, which increases latency and loss.

Q5: What does high RTT indicate?

- Possible **long physical distance**, **congestion**, or **routing loops**.

✅ 11. Performance & Monitoring (Part 2)

🔹 QoS Metrics

Quality of Service (QoS) refers to a network's ability to provide guaranteed performance metrics to different types of traffic. It is crucial for ensuring reliable delivery, especially for real-time and priority-sensitive applications like VoIP, video conferencing, and gaming.

✅ Key QoS Metrics

Metric	Description
Bandwidth	Maximum data that can be transferred per unit time (Mbps, Gbps)
Latency	Time taken for a packet to travel from source to destination (ms)
Jitter	Variation in packet arrival time; affects real-time apps
Packet Loss	Percentage of packets that fail to reach the destination
Availability	Uptime percentage of the network over a given period
Error Rate	Number of corrupted packets in transmission

✅ Classification Techniques:

- **Differentiated Services (DiffServ)**: Uses DSCP bits to mark packets.
- **Integrated Services (IntServ)**: Resource reservation using RSVP protocol.
- **Traffic Shaping**: Regulating data flow (e.g., Token Bucket, Leaky Bucket).
- **Priority Queuing**: Queues with priority for specific traffic types.

✅ Use Case Mapping:

Application	Bandwidth	Latency	Jitter	Packet Loss
VoIP	Low	Very Low	Very Low	Very Low
Video Streaming	High	Medium	Low	Medium
File Transfer (FTP)	High	High	High	Low
Gaming	Medium	Very Low	Very Low	Very Low

🔹 Network Monitoring Tools

Monitoring tools help measure, analyze, and troubleshoot network performance issues.

✅ Wireshark

- **Type**: Packet sniffer and analyzer
- **Use Cases**:
    - Capture and inspect live packet data
    - Debug protocols (TCP handshakes, DNS queries)
    - Detect ARP spoofing, malformed packets

```
 # Sample Filters:
tcp.port == 80         # Capture HTTP traffic
ip.addr == 192.168.1.5 # Filter by IP address
```

✅ Netstat

- **Type**: CLI tool to display active network connections, routing tables
- **Use Cases**:
    - Check which ports are open
    - Identify listening services
    - Analyze connection states (e.g., TIME_WAIT, ESTABLISHED)

```
 netstat -an       # All connections and listening ports
netstat -s        # Per-protocol statistics
```

✅ Traceroute (Linux) / Tracert (Windows)

- **Type**: Path discovery tool
- **Use Cases**:
    - Identify path from source to destination
    - Detect routing loops, hops, and delays

```
 traceroute google.com      # Linux
tracert google.com         # Windows
```

Sample Output:

```
 1  192.168.1.1   1 ms
2  10.0.0.1     20 ms
3  142.251.42.14 35 ms
```

✅ Ping

- **Type**: Reachability and RTT tester
- **Use Cases**:
    - Test host availability
    - Measure round-trip time
    - Estimate packet loss

```
ping 8.8.8.8
```

Sample Output:

```
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=14.2 ms
```

✅ Summary Table:

Tool	Primary Use	Layer
Wireshark	Deep packet inspection	Layer 2–7
Netstat	View active sockets & connections	Layer 4
Traceroute	Path trace across routers	Layer 3
Ping	Basic connectivity and latency	Layer 3

🔹 Real-world Q&A

Q1: When would you prefer Wireshark over Netstat?

- When you need to **inspect packet contents**, protocol headers, and sequence numbers.

Q2: Why does Traceroute show * (stars)?

- Timeout or ICMP packets **blocked by a router**.

Q3: How is jitter measured in real networks?

- **Difference in delay** between consecutive packets (measured by tools or manually from logs).

Q4: What causes fluctuating RTT in ping?

- **Congestion**, **routing changes**, or **packet queuing**.

Q5: How can QoS help in video conferencing?

- Prioritizes video/audio packets, reduces jitter and latency using **traffic shaping and classification**.

✅ 12. Protocols Summary (Across OSI Layers)

This section provides a quick-reference summary of commonly used **network protocols** categorized by their respective **OSI layers**, along with their key functionalities and use cases.

◆ OSI Layer-wise Protocol Mapping

Layer	Representative Protocols	Functionality / Usage
Application	HTTP, FTP, DNS, SMTP, DHCP, POP3, IMAP	User-facing services: web browsing, file transfer, email, address assignment
Transport	TCP, UDP	Reliable/Unreliable data delivery, flow and congestion control
Network	IP, ICMP, IGMP, ARP, RARP	Routing, addressing, diagnostic and control messaging
Data Link	Ethernet, PPP, HDLC, Frame Relay	Framing, MAC addressing, reliable node-to-node delivery
Physical	NRZ, Manchester Encoding, DSL, USB	Physical transmission of raw bits through medium (cables, signals, modulation schemes)

◆ Layer-by-Layer Protocol Summary

✅ Application Layer

Protocol	Function
HTTP	HyperText Transfer Protocol (web access)
HTTPS	Secure HTTP using TLS
FTP	File Transfer Protocol
SMTP	Send mail from client to server
POP3	Retrieve mail (deletes from server)
IMAP	Retrieve mail (retains on server)
DNS	Domain name resolution
DHCP	IP address assignment
Telnet	Remote shell access (unencrypted)
SSH	Secure remote shell access

✅ Transport Layer

Protocol	Function
TCP	Reliable, connection-oriented transmission
UDP	Unreliable, connectionless transmission

✅ Network Layer

Protocol	Function
IP	Logical addressing and routing (IPv4/IPv6)
ICMP	Control messages (ping, unreachable)
IGMP	Multicast group management
ARP	IP to MAC address resolution
RARP	MAC to IP resolution (obsolete)

✅ Data Link Layer

Protocol	Function
Ethernet	LAN communication using MAC addresses
PPP	Point-to-Point communication
HDLC	High-level data link control (WANs)
Frame Relay	Packet-switched WAN protocol
MAC	Sub-layer handling addressing/collision

✅ Physical Layer

Technique	Function / Usage
NRZ	Non-Return-to-Zero (digital encoding)
Manchester	Clock synchronization + data encoding
DSL	Internet over telephone lines
FSK/ASK/PSK	Modulation techniques
USB	Universal Serial Bus (wired comm.)

🔹 Diagram: OSI Model with Protocols

```
| Layer        | Example Protocols                   |
|--------------|-------------------------------------|
| Application  | HTTP, FTP, DNS, SMTP                |
| Transport    | TCP, UDP                            |
| Network      | IP, ICMP, ARP                       |
| Data Link    | Ethernet, PPP                       |
| Physical     | NRZ, DSL, Manchester Encoding       |
```

🔹 Real-world Q&A

Q1: Why is TCP used in HTTP but not in video streaming?

- HTTP needs reliable delivery, video streaming (e.g., RTP/UDP) tolerates some loss but requires **low latency**.

Q2: Why do we need ARP in Ethernet LANs?

- To map **IP address** → **MAC address**, since Ethernet uses MAC for actual delivery.

Q3: What's the difference between PPP and Ethernet?

- **PPP** is for **point-to-point links** (e.g., serial), Ethernet is for **shared medium LANs**.

Q4: How is ICMP different from IP?

- ICMP is used to **diagnose** issues (e.g., unreachable host), while IP is for **routing packets**.

Q5: Can TCP run without IP?

- No. TCP is dependent on IP for **routing and addressing** – together they form the **TCP/IP stack**.

✅ 13. Cloud, CDN, and Modern Networking

🔹 CDNs (Content Delivery Networks)

✅ What is a CDN?

- A **geographically distributed** set of servers used to deliver web content **faster and reliably** to users.
- Reduces latency by **serving data from the nearest edge server** to the user.

✅ Key Features:

- Caching static content (HTML, JS, CSS, images, videos)
- Reducing origin server load
- DDoS protection
- TLS/SSL termination

✅ Popular CDNs:

- **Akamai**
- **Cloudflare**
- **Amazon CloudFront**
- **Fastly**

🔹 DNS Load Balancing

✅ What is it?

- Distributes client requests across **multiple servers** based on **DNS responses**.
- Helps achieve **high availability** and **load distribution**.

✅ Types:

- **Round-Robin DNS**
- **GeoDNS**: Routes based on client location
- **Weighted DNS**: Routes based on server capacity or priority

✅ Example:

```
 example.com resolves to:
192.0.2.1
192.0.2.2
192.0.2.3
```

🔹 SDN (Software Defined Networking)

✅ Definition:

- Networking architecture where **control plane is separated from data plane**.
- The network is **centrally programmable** via software.

✅ Architecture:

- **Application Layer**: Business logic
- **Control Layer**: SDN Controller (e.g., OpenDaylight)
- **Infrastructure Layer**: Routers, Switches

✅ Benefits:

- Centralized control
- Dynamic reconfiguration
- Better security and monitoring

🔹 NFV (Network Function Virtualization)

✅ Definition:

- Virtualizes **network services** like routing, firewall, NAT, load balancing, etc., on commodity hardware.

✅ Components:

- **VNFs**: Virtual Network Functions
- **NFVI**: Infrastructure (compute/storage/network)
- **MANO**: Management and orchestration

✅ Benefits:

- Reduces hardware costs
- Scalable and flexible deployment
- Faster provisioning

🔹 Overlay Networks (VPNs, Tunnels)

✅ Overlay Network:

- A **virtual network** built on top of an existing physical network.

✅ Examples:

- **VPN (Virtual Private Network)**
- **GRE Tunnels**
- **VXLAN**

✅ Use Cases:

- Secure private communication over public internet
- Isolated containers/networks (Kubernetes, SDN)

🔹 Cloud Networking Basics

✅ AWS (Amazon Web Services) – VPC

- **VPC (Virtual Private Cloud)**: A logically isolated section of AWS.
- Includes:
    - Subnets (Public/Private)
    - Route Tables
    - Internet Gateways
    - NAT Gateways
    - Security Groups & NACLs

✅ GCP (Google Cloud Platform) – Networking

- **VPC** spans **regions** (global)
- Uses **firewall rules**, **routes**, and **Cloud NAT**

- Supports **peering**, **interconnect**, and **load balancers**

🔹 Edge Computing vs Cloud Computing

Feature	Edge Computing	Cloud Computing
Location	Near data source (IoT, local)	Centralized data centers
Latency	Very low	Higher
Bandwidth usage	Lower (pre-processed locally)	Higher
Use Case	Real-time apps, IoT, AR/VR	Data storage, machine learning
Example	Autonomous cars, smart cameras	Web hosting, analytics

---

✅ Diagram:

```
[ IoT Device ] --> [ Edge Node ] --> [ Cloud Server ]
Real-time          Local Processing     Storage/Training
```

🔹 Proxy Servers & Reverse Proxies

✅ Proxy Server

- **Client-facing** intermediary that forwards client requests to the internet.
- Used for:
    - **Anonymity**
    - **Access control**
    - **Content filtering**

✅ Reverse Proxy

- **Server-facing** intermediary that receives requests on behalf of servers.
- Used for:
    - **Load balancing**
    - **TLS termination**
    - **Caching**
    - **Security (hiding internal services)**

✅ Tools:

- **Nginx**
- **HAProxy**
- **Squid**
- **Apache Traffic Server**

🔹 Real-world Q&A

Q1: How do CDNs improve page load time?

- By **caching content** near the user and serving it from **edge locations**, reducing RTT.

Q2: How does DNS Load Balancing differ from a hardware load balancer?

- DNS LB happens at **name resolution time**; hardware LB happens at **packet routing level**.

Q3: What's the advantage of SDN in modern data centers?

- Enables **dynamic control** over traffic, **automation**, and **network slicing**.

Q4: When should you use a reverse proxy?

- For **TLS offloading**, **load balancing**, **caching**, or **centralized access control**.

Q5: Why is edge computing needed despite cloud computing?

- For **low-latency** use cases and to **reduce bandwidth usage** by preprocessing locally.

✅ 14. Miscellaneous & Advanced Topics

🔹 BitTorrent / Peer-to-Peer Networking

✅ What is Peer-to-Peer (P2P)?

- A **decentralized network model** where each node (peer) acts as both **client and server**.
- Used in file sharing, distributed systems (e.g., blockchain).

✅ BitTorrent Protocol

- A popular **P2P file-sharing protocol**.
- Breaks files into chunks → Peers download chunks from multiple sources simultaneously.

✅ Key Components:

Term	Description
Torrent File	Metadata (file name, size, tracker URL, etc.)
Tracker	Server that coordinates peers
Seeder	Peer with full copy of file
Leecher	Peer still downloading
Swarm	All peers sharing a specific torrent

✅ Advantages:

- Fast download speeds via **parallelism**
- Scales well without central server load

🔹 Onion Routing (Tor Network)

✅ Definition:

- A technique for **anonymous communication** by encrypting messages in layers (like an onion).

✅ How It Works:

1. Client encrypts data in multiple layers
2. Each node decrypts **only one layer**, revealing the next hop
3. Final node sends decrypted message to destination

   Used by the **Tor (The Onion Router)** network.

✅ Diagram:

```
[Client] → [Node 1] → [Node 2] → [Node 3] → [Destination]
             ↓ decrypt ↓         ↓ decrypt ↓         ↓ decrypt
```

✅ Benefits:

- High anonymity and privacy
- IP and data source obscuration

✅ Use Cases:

- Censorship evasion
- Whistleblower communication

🔹 Socket Programming (Basics in C/C++)

✅ What is a Socket?

- An endpoint for **bidirectional communication** between devices.

✅ Basic Flow (TCP Server in C):

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
bind(sockfd, ...);
```

```
listen(sockfd, 5);
int clientfd = accept(sockfd, ...);
read(clientfd, buffer, sizeof(buffer));
write(clientfd, response, strlen(response));
```

✅ Common Socket Functions:

Function	Purpose
socket()	Create a new socket
bind()	Bind socket to IP + port
listen()	Mark socket as passive (server)
accept()	Accept incoming connection
connect()	Connect to a remote socket
send()/recv()	Data transmission

✅ Port Numbers:

Type	Port Range
Well-known	0–1023
Registered	1024–49151
Dynamic	49152–65535

🔹 Network Simulation Tools

✅ 1. ns-2 / ns-3

- **Network simulator** for academic and research purposes.
- Simulate packet-level behavior, wireless/mobile networks.

✅ 2. Cisco Packet Tracer

- **Graphical simulation tool** for learning networking (routers, switches, topologies).
- Used by **Cisco Networking Academy**.

✅ 3. Mininet

- Simulates **Software Defined Networks (SDNs)**.
- Creates virtual networks using Linux containers.

🔹 Network Layers in Linux

✅ Netfilter

- A framework inside the Linux kernel for **packet filtering**, **NAT**, and **packet mangling**.

✅ IP Tables

- A user-space utility to configure **Netfilter rules**.

```
 iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -j DROP
```

- Chains: **INPUT**, **OUTPUT**, **FORWARD**
- Tables: **filter**, **nat**, **mangle**

✅ Use Cases:

- Firewall configuration
- Port forwarding
- Blocking IPs or ports

🔹 IP Spoofing, DNS Poisoning

✅ IP Spoofing

- Forging source IP address in packet headers to:
  - Impersonate another device
  - Bypass IP-based filters
  - Launch DDoS attacks

✅ DNS Poisoning (DNS Spoofing)

- Attacker corrupts DNS cache to redirect users to **malicious sites**.

Example:

User types `google.com` → attacker redirects to `malicious.com` by poisoning DNS resolver cache.

✅ Mitigations:

Threat	**Defense Mechanisms**
IP Spoofing	Packet filtering, ingress/egress rules
DNS Poisoning	DNSSEC, query validation, cache TTL control

🔹 Real-world Q&A

Q1: Why is BitTorrent more scalable than traditional HTTP download?

- Because **peers share with each other**, reducing load on a central server.

Q2: What makes Tor different from a VPN?

- Tor uses **multi-hop layered encryption**, VPN encrypts only once and the provider can see your traffic.

Q3: What is the use of `bind()` in socket programming?

- Binds a socket to a **specific IP and port** for listening.

Q4: Why use iptables in a Linux server?

- To **control packet flow**, act as a firewall, and **set up NAT or port forwarding**.

Q5: How does DNS poisoning affect users?

- Redirects them to **fake/malicious websites**, compromising security and privacy.

✅ 14. Miscellaneous & Advanced Topics (Part 2)

🔹 Latency Optimization Techniques

Modern networks use a variety of techniques to **minimize latency** and improve user experience, especially for web and mobile applications.

✅ HTTP/2 Server Push

- Allows the server to **proactively send resources** (like CSS/JS) to the client before it's requested.
- Reduces latency for page load since resources are **preloaded**.

```
HTTP/2 PUSH: style.css, script.js
```

> ❗ Misuse can increase bandwidth and cache duplication if not handled carefully.

✅ TCP Fast Open (TFO)

- Enables **data transfer to begin during the handshake** instead of waiting for it to complete.

Traditional TCP:

1. SYN →
2. SYN-ACK ←
3. ACK →
4. THEN send data

TCP Fast Open:

- Data sent **with SYN**, reducing one RTT

```
sysctl -w net.ipv4.tcp_fastopen=3  # Enable client/server support
```

🔹 Head-of-Line (HOL) Blocking

✅ What is it?

- A performance issue where **one blocked packet stalls others** behind it, even if they are unrelated.

✅ Example:

In HTTP/1.1 with one TCP connection:

- If one large request blocks, **all others** queued behind it must wait.

✅ Mitigations:

Technology	How It Helps
HTTP/2	Multiplexes streams over one TCP
QUIC/HTTP3	Uses UDP → independent streams
Connection Pooling	Multiple TCP connections

🔹 Keep-Alive & Connection Pooling

✅ HTTP Keep-Alive

- Maintains a persistent TCP connection across multiple HTTP requests.

```
Connection: keep-alive
```

Avoids **TCP handshake overhead** for every request.

✅ Connection Pooling

- Reuses open connections from a **pool** instead of creating new ones.
- Improves performance for:
  - Database connections
  - REST API clients
  - Microservices communication

🔹 Port Forwarding & Tunneling

✅ Port Forwarding

- Redirects traffic from one port/IP to another.
- Common in NAT and remote access.

```
# Example using SSH
ssh -L 8080:internal.server:80 user@proxy.server
```

✅ Types:

Type	Description
Local Forward	Redirects local port to remote server
Remote Forward	Exposes local port on remote server
Dynamic	Acts like a SOCKS proxy

✅ Use Cases:

- SSH access to internal networks
- Accessing web apps from behind NAT
- Secure tunneling of insecure protocols

🔹 MPLS (Multiprotocol Label Switching)

✅ What is MPLS?

- A high-performance **packet forwarding** technology that uses **labels** instead of IP addresses for routing decisions.

✅ Key Concepts:

Term	Description
Label	Short fixed-length identifier assigned to packets
LSR (Router)	Label Switch Router: forwards based on label
LER	Edge router: assigns and removes labels

✅ Benefits:

- Fast forwarding decisions (no IP lookup)
- Supports **QoS** and **traffic engineering**
- More scalable than traditional IP routing

✅ Diagram:

```
[Client] → [LER] → [LSR] → [LSR] → [LER] → [Destination]
             ↓       ↓        ↓        ↓
          Label1   Label2   Label3   POP label
```

🔹 Real-world Q&A

Q1: Why is HTTP/2 better than HTTP/1.1 for latency?

- It **multiplexes requests** over a single connection and avoids HOL blocking.

Q2: How does TCP Fast Open reduce latency?

- Sends data **during** the TCP handshake, saving **one RTT**.

Q3: What's the problem with Head-of-Line blocking?

- It **delays unrelated requests**, reducing throughput.

Q4: When should you use connection pooling?

- In high-volume request environments (e.g., databases, APIs) to **avoid connection setup overhead**.

Q5: What are the advantages of MPLS over IP routing?

- **Faster forwarding**, **predictable performance**, and **support for traffic engineering**.

✅ 15. Common Interview Questions

This section includes frequently asked interview questions from networking rounds, with detailed explanations, step-by-step dry runs, real-world use cases, and diagrams where applicable.

🔹 1. TCP vs UDP (with Use Cases)

✅ TCP (Transmission Control Protocol)

Feature	TCP
Connection	Connection-oriented (requires handshake)
Reliability	Reliable (acknowledgements, retransmission)
Ordering	Guaranteed order of data delivery
Overhead	Higher (due to connection and control)
Speed	Slower than UDP
Use Cases	Web browsing (HTTP/HTTPS), Email (SMTP), File Transfer (FTP)

✅ UDP (User Datagram Protocol)

Feature	UDP
Connection	Connectionless (no handshake)
Reliability	Unreliable, no guarantee of delivery
Ordering	No ordering guarantees
Overhead	Low
Speed	Faster
Use Cases	Live Streaming, VoIP, DNS, Gaming

✅ Real-World Example:

Application	Protocol	Why?
WhatsApp voice call	UDP	Needs low-latency, tolerates minor loss
Gmail web app	TCP	Ensures message delivery and order

🔹 2. DNS Lookup Process (Step-by-Step)

✅ What Happens When You Type `www.google.com`?

1. **Browser Cache** → Check if domain is cached
2. **OS Cache** → Check local DNS cache (nscd, systemd-resolved)
3. **Router Cache** → Router's DNS table
4. **ISP DNS Resolver** → Forward query

5. **Root DNS Server** → Responds with TLD (e.g., .com) name server
6. **TLD Server** → Responds with authoritative name server for domain
7. **Authoritative Server** → Returns actual IP of `www.google.com`
8. **DNS Resolver** → Sends IP to client

✅ Diagram:

```
[User] → [OS Cache] → [ISP Resolver]
             ↘              ↘
            [Root DNS] → [TLD DNS] → [Authoritative DNS]
```

✅ Tool:

```
dig www.google.com
nslookup www.google.com
```

🔹 3. TCP 3-Way Handshake Dry Run (with SYN/ACK flags)

✅ Purpose:

- To establish a **reliable connection** between two hosts.

✅ Steps:

Step	Sender (Client)	Receiver (Server)	Flag
Step 1	Sends SYN (Seq=x)		SYN
Step 2		Sends SYN+ACK (Ack=x+1, Seq=y)	SYN+ACK
Step 3	Sends ACK (Ack=y+1)	Connection established	ACK

✅ Diagram:

```
Client → Server : SYN (Seq = x)
Client ← Server : SYN + ACK (Seq = y, Ack = x+1)
Client → Server : ACK (Ack = y+1)
```

✅ Code Simulation Snippet (in C):

```
// pseudo-code
send(SYN);
recv(SYN+ACK);
send(ACK);
```

🔹 4. IP Address vs MAC Address

Feature	IP Address	MAC Address
Full Form	Internet Protocol Address	Media Access Control Address
Layer	Network Layer (Layer 3)	Data Link Layer (Layer 2)
Uniqueness	Logical (can be changed)	Physical (burned-in by manufacturer)
Format	IPv4: 192.168.1.1	00:1A:2B:3C:4D:5E
Use Case	Routing over Internet	Local LAN delivery (Ethernet)

✅ IP-MAC Mapping:

- Done using **ARP** (Address Resolution Protocol)

🔹 5. Subnetting a Given IP (With Mask)

✅ Example:

```
IP Address: 192.168.10.0
Subnet Mask: 255.255.255.224 (/27)
```

✅ Steps:

1. Convert Mask:
   - /27 = 255.255.255.224 = 11111111.11111111.11111111.11100000
   - 2^5 = 32 IPs per subnet
2. Number of Subnets:
   - From a Class C block → 256 addresses
   - 256 / 32 = 8 subnets
3. Subnet Ranges:
   - 192.168.10.0 → 192.168.10.31
   - 192.168.10.32 → 192.168.10.63
   - ...
   - 192.168.10.224 → 192.168.10.255

✅ Diagram:

```
| Subnet | Range                   | Broadcast          | Usable Hosts |
|--------|-------------------------|--------------------|---------------|
| 1      | 192.168.10.0 - .31      | 192.168.10.31      | .1 - .30       |
```

```
| 2      | 192.168.10.32 - .63    | 192.168.10.63     | .33 - .62      |
...
```

✅ Real-World Q&A

Q1: Why is TCP preferred over UDP for HTTP?

- Ensures **reliable, ordered** delivery with congestion control.

Q2: What if DNS cache is poisoned?

- User may be redirected to a **malicious IP**.

Q3: How many usable IPs in a /30 subnet?

- 4 total IPs → 2 usable (excluding network + broadcast)

Q4: Can MAC address be changed?

- Yes, temporarily via software (`ifconfig` or `ip link set`) but not permanently in hardware.

Q5: What if TCP 3-way handshake fails?

- Connection is **not established**, likely due to firewall, port block, or packet loss.

✅ 15. Common Interview Questions (Part 2)

🔹 1. Explain NAT and Port Forwarding

✅ NAT (Network Address Translation)

- Translates **private IP addresses** to a **public IP** for internet communication.
- Saves IPv4 address space and provides **basic security**.

✅ Types of NAT:

Type	Description
Static NAT	One-to-one mapping
Dynamic NAT	Many-to-many (from a pool)
PAT (Port Address Translation)	Many-to-one, uses ports

Example:

```
192.168.1.5:12345 → 203.0.113.20:50001
```

✅ Port Forwarding

- Forwards a request from one IP/port to another.
- Commonly used to access internal services behind NAT.

```
ssh -L 8080:localhost:80 user@remote
```

Allows access to a local web server via remote SSH.

🔹 2. OSI vs TCP/IP Model

Feature	OSI Model (7 Layers)	TCP/IP Model (4 Layers)
Layers	Physical → Application	Network Access → Application
Conceptual	Theoretical reference model	Practical implementation model
Layers Split	Clear separation (Presentation, Session)	Merged into Application

---

✅ Mapping:

OSI Layer	TCP/IP Equivalent
Application	Application
Presentation	Application
Session	Application
Transport	Transport
Network	Internet
Data Link	Network Access
Physical	Network Access

🔹 3. HTTP vs HTTPS (with Certificates)

Feature	HTTP	HTTPS
Security	Insecure, plain text	Encrypted using TLS/SSL
Port	80	443
Encryption	None	TLS uses symmetric + asymmetric crypto
Certificates	None	Requires SSL certificates (X.509)

✅ HTTPS Handshake Steps:

1. **Client Hello**: Sends supported ciphers, TLS version
2. **Server Hello**: Sends certificate
3. **Certificate Verification**
4. **Key Exchange**: Using RSA/ECDHE
5. **Session Key Setup**
6. **Encrypted Data Transmission**

🔹 4. How Does a Browser Load a Webpage?

1. **URL parsing** → `https://example.com`
2. **DNS resolution** → Convert domain to IP
3. **TCP 3-way handshake** → Establish connection
4. **TLS handshake** (if HTTPS)
5. **Send HTTP GET request**
6. **Receive HTML response**
7. **Browser renders** page:
   - Parse HTML
   - Load CSS, JS, images
   - Execute JS
   - Construct DOM and render tree

✅ Diagram:

```
User Input → DNS Lookup → TCP Handshake → TLS → HTTP GET → HTML → Render Page
```

🔹 5. Difference: Firewall vs Proxy vs IDS

Component	Description	OSI Layer	Example Use Case
Firewall	Filters traffic based on rules	Network/Transport	Block port 22 traffic
Proxy	Intercepts requests/responses	Application	Content caching, anonymity
IDS	Monitors traffic for intrusion attempts	Network/Application	Detect SQL injection

---

✅ Bonus:

- **IPS** (Intrusion Prevention System): Blocks malicious packets (active defense)

🔹 6. MTU Impact on Packet Fragmentation

✅ MTU (Maximum Transmission Unit)

- Maximum packet size (in bytes) that a network layer can transmit without fragmentation.

  Ethernet MTU: **1500 bytes**

✅ If Packet > MTU:

- Packet is **fragmented** into smaller pieces
- Adds overhead due to fragmentation headers
- If `Don't Fragment (DF)` bit is set → packet is **dropped** and ICMP error is sent

---

✅ Impacts:

Issue	Effect
Fragmentation	Adds latency, processing overhead
Path MTU Discovery	Prevents fragmentation with optimal MTU
VPNs/Tunnels	Can reduce effective MTU (due to headers)

🔹 7. DNS Poisoning or Spoofing

✅ DNS Poisoning

- **Injecting false DNS records** into resolver cache
- Redirects users to malicious sites (e.g., phishing, malware)

✅ Example:

```
User → DNS Server → Malicious IP for google.com
```

✅ Techniques:

- Compromising resolver
- Cache poisoning with fake responses
- Man-in-the-middle during DNS query

✅ Protection:

Defense	Method
DNSSEC	Digital signatures for DNS records
Random TXID	Prevent predictable query IDs
Query Validation	Ensure response matches request

◆ Real-world Q&A

Q1: Why is NAT used at home routers?

- To allow multiple private IP devices to **share one public IP**.

Q2: Why is MTU important for VPNs?

- VPN headers reduce MTU → may lead to **fragmentation or dropped packets**.

Q3: How does DNS poisoning affect end-users?

- Redirects them to **fake or malicious websites**.

Q4: Why is HTTPS preferred over HTTP?

- It ensures **data integrity, confidentiality, and authentication** via TLS.

Q5: What role does a proxy play in a corporate network?

- Filters and logs employee traffic, **caches content**, and enhances **security**.
