SDE: System Design and Engineering

Lecture – 2

Introduction to

Foundation of Web, Network and HTTP

From Zero to Google: Architecting the Invisible Infrastructure

by

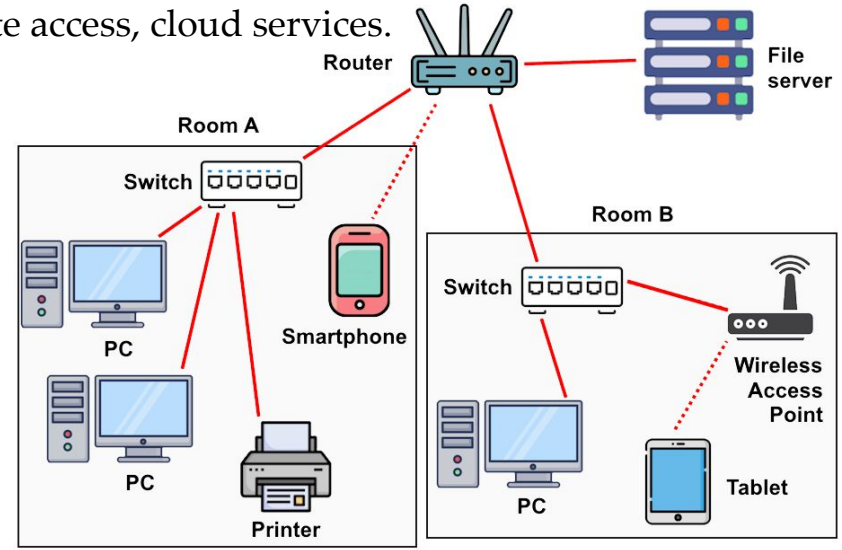
Aatiz Ghimire

Sections

- Foundations of Computer Networks
- DNS, Address Resolution & Routing
- Web Fundamentals
- HTTP Protocol & Mechanisms
- HTTPS & Transport Security
- Identity, Authentication, Authorization
- Real-Time & Asynchronous Web
- Secure Tunnels, VPN & Remote Shell
- Production Live Streaming Operations

Introduction to Computer Networks

- A computer network is a collection of interconnected devices that share resources and information.
- Devices include:
  - Computers, servers, printers, routers, switches, IoT devices.
- Networks enable:
  - Email, file sharing, internet browsing, remote access, cloud services.



Key Network Terminologies

Network: A system of connected devices enabling data exchange.

Node: Any device connected to a network.

Protocol: Rules for communication (e.g., TCP/IP, HTTP, FTP).

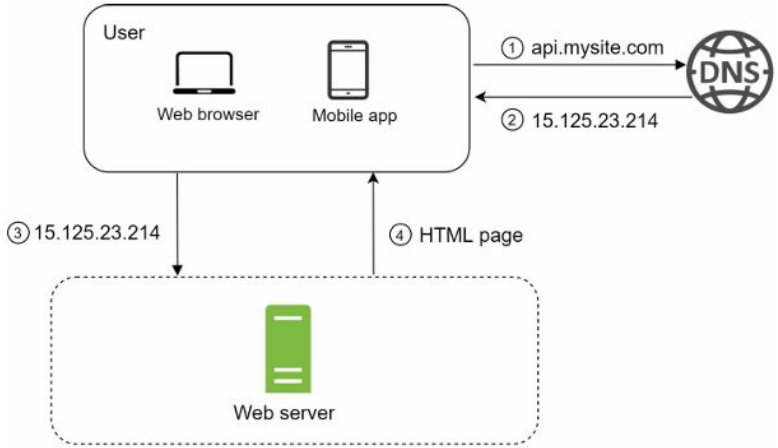
Topology: Layout of how devices are physically/logically connected (bus, star, ring, mesh, tree).

Service Provider Networks: Leased network capacity & functions (e.g., ISPs, mobile carriers).

IP Address and DNS

Fig: NIC Interface

- IP Address: Unique numeric identifier for each device given to NIC card based on MAC address.
  - Example: 192.168.1.1 (Private IP), 202.70.90.210 (Public IP)
  - In Terminal, ip a
- DNS (Domain Name System):
  - Translates domain names into IP addresses.
  - Example: www.google.com → 142.251.46.238



Interface name

IP

```
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 98:ee:cb:cd:17:80 brd ff:ff:ff:ff:ff:ff
    altname enx98eecbcd1780
    inet 10.80.0.10/24 brd 10.80.0.255 scope global noprefixroute enp2s0
       valid_lft forever preferred_lft forever
    inet6 fe80::332b:b244:9ac5:9702/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: wlp0s20f3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether ac:12:03:61:73:7b brd ff:ff:ff:ff:ff:ff
    altname wlxac120361737b
```

Types of Networks: Personal Area Network (PAN)

- Designed for short-range communication (within a few meters).
- Connects personal devices belonging to a single user.

Common Use Cases:

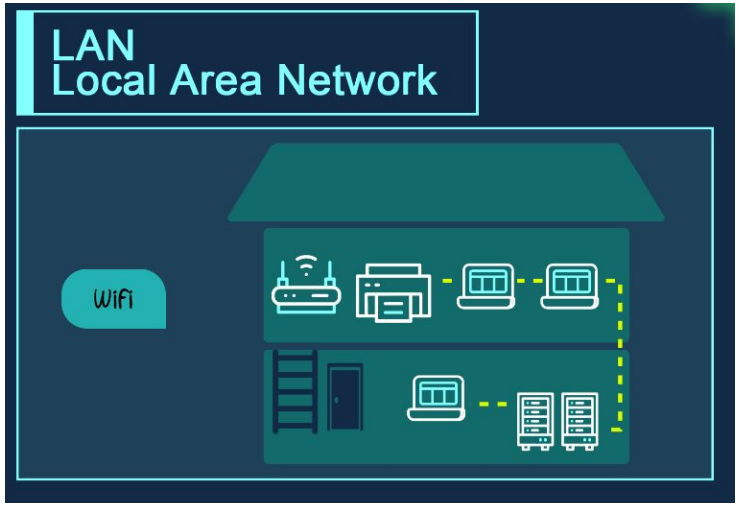
- Syncing smartphones, tablets, and wearables.
- Bluetooth-enabled hands-free communication.
- Data synchronization between computer and smartphone.

Types of Networks: Local Area Network (LAN)

- Covers a small, localized area (home, office, campus building).
- High-speed data exchange within the network.

Common Use Cases:

- Sharing printers, file servers, and resources.
- Internal communication and collaboration.
- Internet access for small businesses and homes.

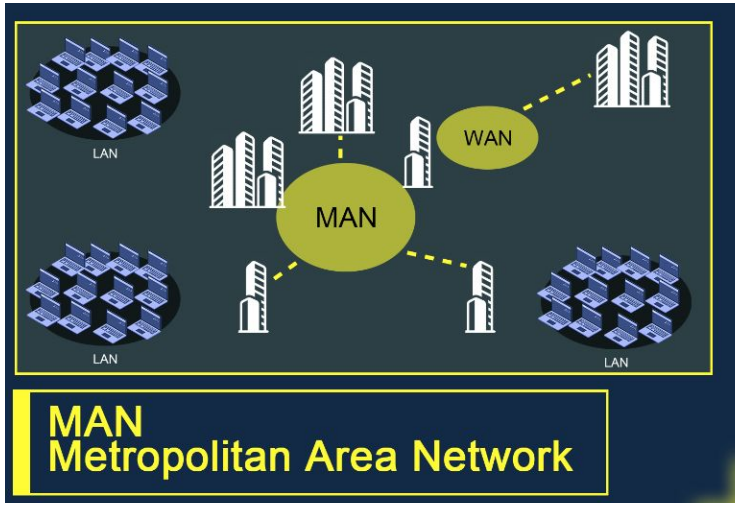


Types of Networks: Metropolitan Area Network (MAN)

- Covers an area larger than LAN but smaller than WAN (city or large campus).
- Often owned and managed by service providers or large institutions.

Common Use Cases:

- Connecting university campuses.
- City-wide high-speed internet.
- Linking municipal or government offices.

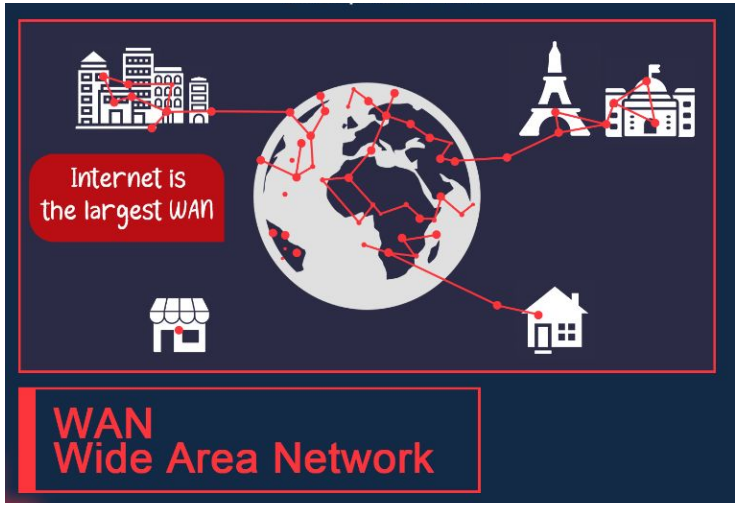


Types of Networks: Wide Area Network (WAN)

- Covers vast geographical areas (countries, continents).
- Interconnects multiple LANs and MANs.

Common Use Cases:

- Connecting multinational corporate branches.
- Supporting global communications.
- Providing remote access to centralized resources.
- **Example:** The Internet.

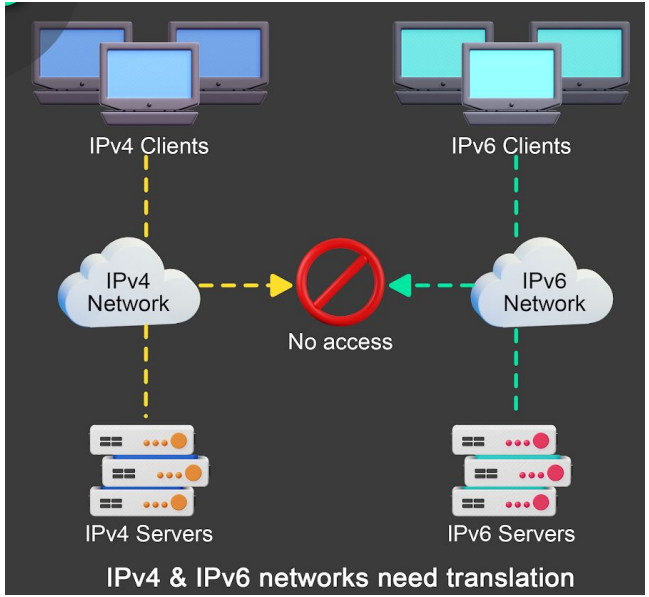


Network Addressing: IPv4 vs IPv6

Why IPv6 Was Introduced

The transition from IPv4 to IPv6 is primarily driven by two needs:

- **Address Exhaustion:** IPv4's 32-bit limit allows only ~**4.3 billion** unique IP addresses, which is insufficient for today's internet-connected world.
- **Efficiency and Scalability:** IPv6 simplifies network configuration and improves packet processing efficiency.



Network Addressing: IPv4 vs IPv6

Feature	IPv4	IPv6
Bit Length	32 bits	128 bits
Representation	Decimal, dot-separated (e.g., 192.168.0.12)	Hexadecimal, colon-separated (e.g., 2001:0db8:85a3::8a2e:0370:7334)
Address Capacity	~4.3 billion	~3.4×10^{38} (340 undecillion)
Address Space Usage	Nearly exhausted	Effectively unlimited for future needs

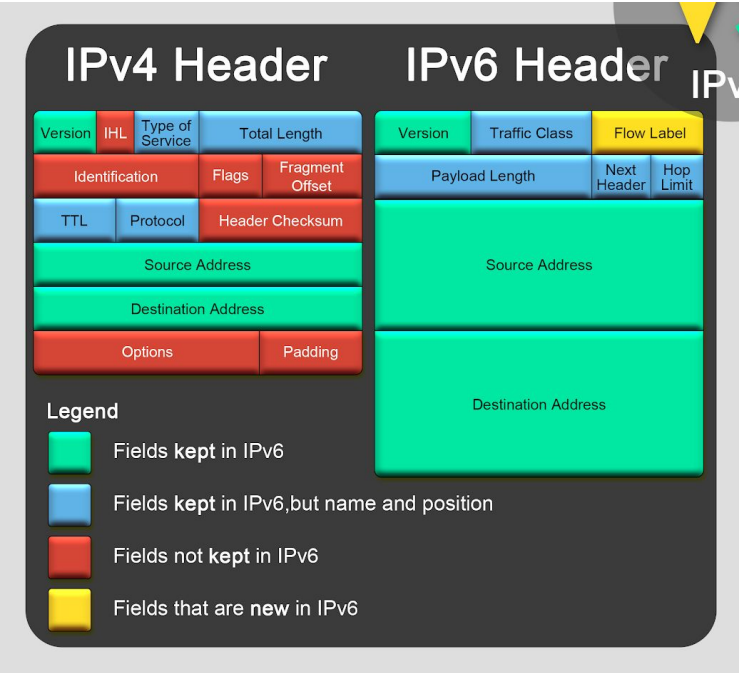
Network Addressing: IPv4 vs IPv6

IPv4 Header:

- Complex structure with multiple mandatory fields.
- Includes header length, identification, TTL, fragmentation info, and checksum.
- Variable-length header increases processing time.

IPv6 Header:

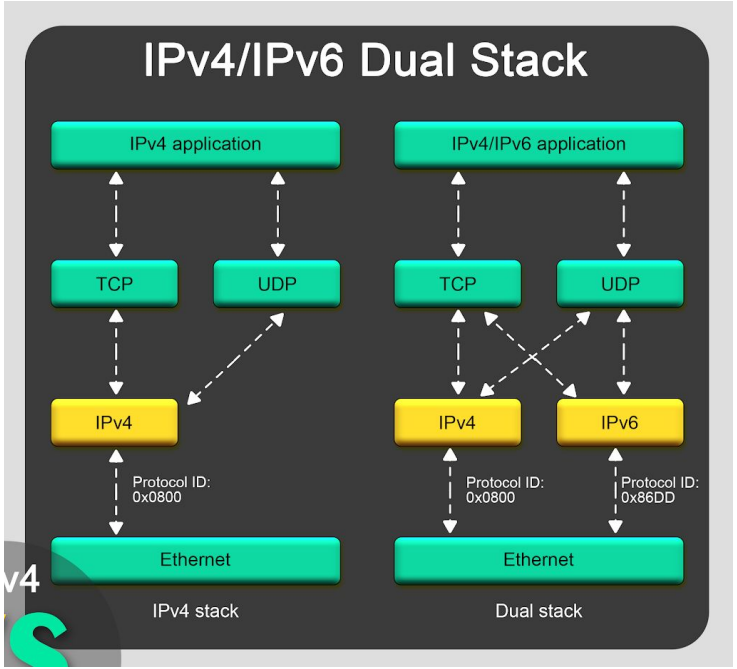
- Fixed size (40 bytes) and simplified.
- Removes rarely used fields; uses **extension headers** instead.
- Improves routing efficiency and reduces overhead.



Network Addressing: IPv4 vs IPv6

Coexistence Mechanism: Dual Stack

- **Dual Stack** enables devices and networks to operate IPv4 and IPv6 simultaneously.
- Ensures **interoperability** during the global transition phase.
- Supported by modern operating systems, routers, and ISPs.

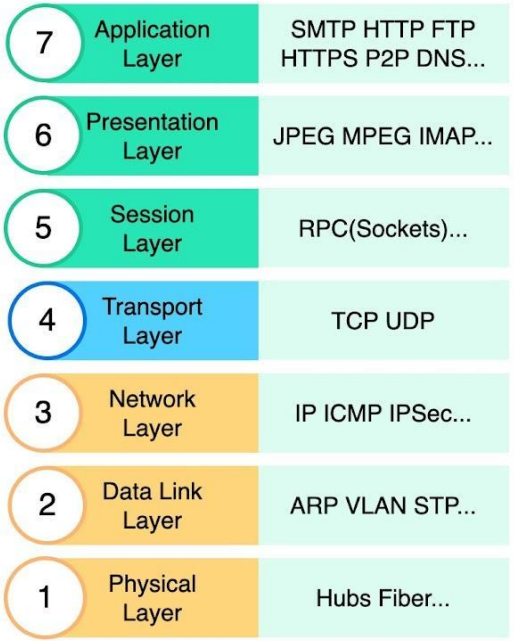


OSI Model

- **What is the OSI Model?**

  OSI stands for **Open Systems Interconnection.**

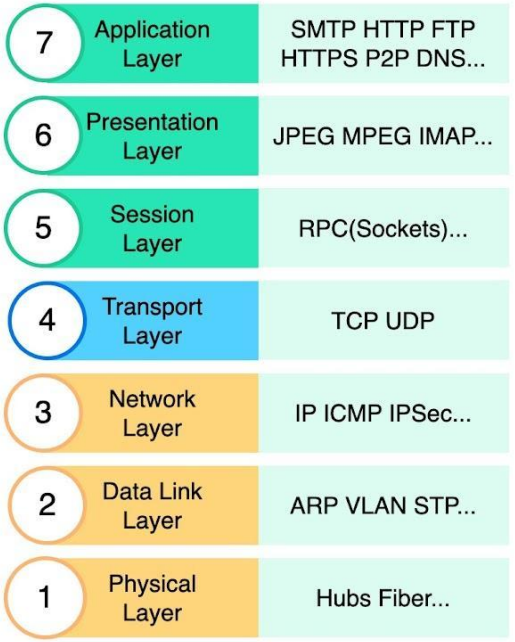
- It is a conceptual framework that standardizes how data is transmitted between devices in a network.
- Divides communication into seven layers, each with specific functions.



OSI Model

Why Do We Need Layers?

- Layers provide **modularity** and **abstraction**.
- Each layer handles **a specific task** and communicates with the layers directly above and below.
- This model:
  - Simplifies troubleshooting
  - Enhances interoperability
  - Allows for technology upgrades without overhauling the entire stack

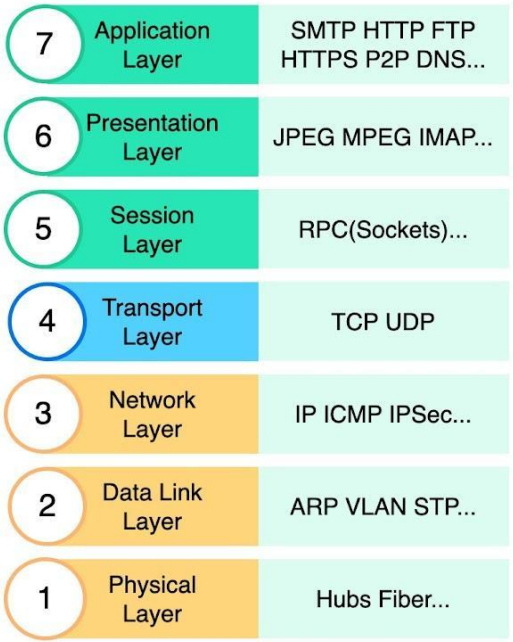


OSI Model

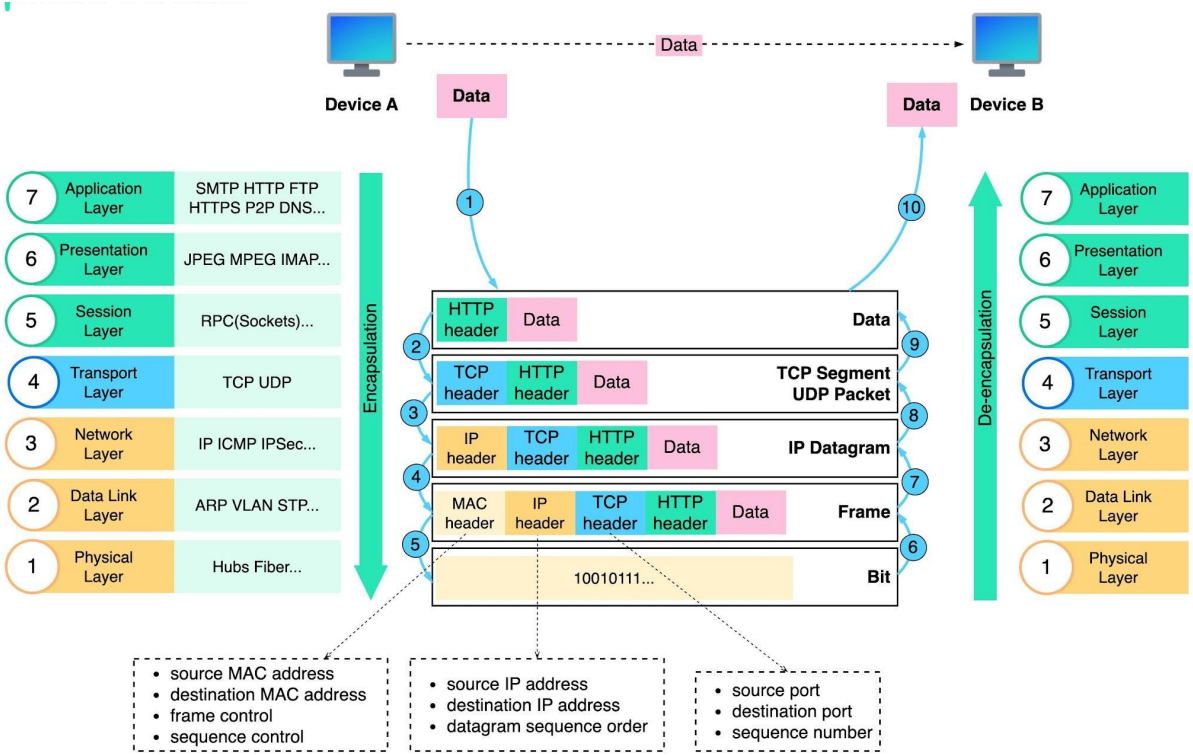
Data Encapsulation: Layer-by-Layer Breakdown

Device A (Sender Side)

1. **Application Layer (Layer 7):**
   Adds HTTP header → forms an HTTP message.
2. **Transport Layer (Layer 4):**
   Adds TCP/UDP header (e.g., source/destination ports, sequence numbers) → forms a segment.
3. **Network Layer (Layer 3):**
   Adds IP header (source and destination IP addresses) → forms a packet.
4. **Data Link Layer (Layer 2):**
   Adds MAC header (source/destination MAC addresses) → forms a frame.
5. **Physical Layer (Layer 1):**
   Converts the frame into binary bits (0s and 1s) and transmits it as electrical, optical, or radio signals.



OSI Model

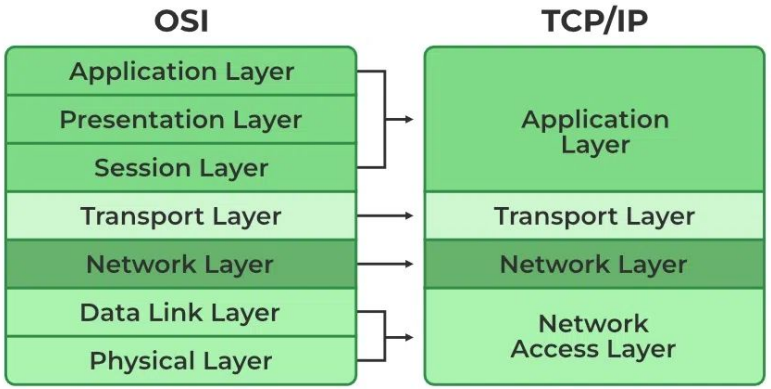


Data De-Encapsulation: Device B (Receiver Side)

- Receives binary stream via **Physical Layer**.
- Each layer **removes its corresponding header** and passes the remaining data up the stack.
- Eventually, the application (e.g., web browser) receives the original HTTP message.

TCP/IP

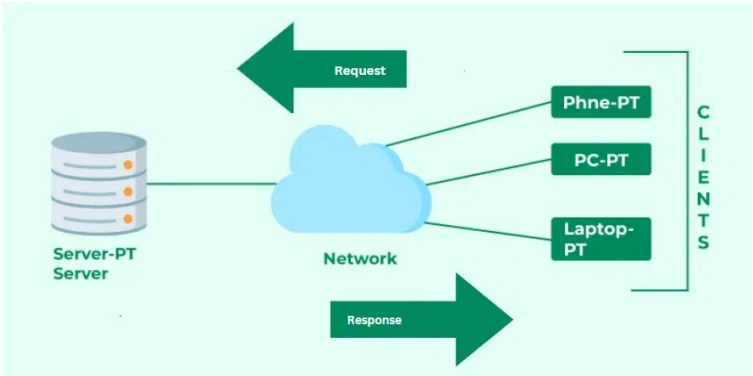
- **TCP/IP is practical and protocol-oriented**, designed to solve real-world communication problems.
- **OSI is theoretical and service-oriented**, offering a modular way to think about how data moves through a network.
- **TCP/IP combines functions** of OSI's Application, Presentation, and Session layers into one **Application Layer**.
- **TCP/IP's Internet Layer** maps closely to **OSI's Network Layer**, but focuses on **IP-based routing**.



Communication Architectures

Client-Server Architecture

- **Centralized model**: One or more servers provide resources or services to multiple client devices.
- Clients initiate requests; servers respond.
- Most common model for **web applications, databases, and email systems**.



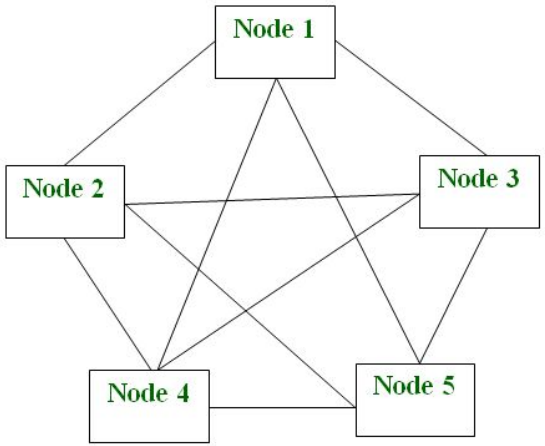
Examples:

- Web browser (client) ↔ Web server (HTTP)
- Email client (Outlook) ↔ Mail server (SMTP, IMAP)

Communication Architectures

Peer-to-Peer (P2P) Architecture

- **Decentralized model**: All nodes (peers) act as both clients and servers.
- Each peer can share, download, or relay data directly with others.
- Used in **file sharing, blockchain, VoIP, and decentralized apps (dApps)**.



P2P Architecture

Examples:

- BitTorrent file sharing
- Bitcoin or Ethereum blockchain networks

Communication Paradigms

How devices send/receive data across networks?

Mode	Description	Example Protocols
Unicast	One-to-one communication	HTTP, FTP, SMTP
Broadcast	One-to-all within local segment	ARP, DHCP (Discover), NTP
Multicast	One-to-selected-many	IPTV, Video Conferencing
Anycast	One-to-nearest node in a group	DNS, CDN Routing

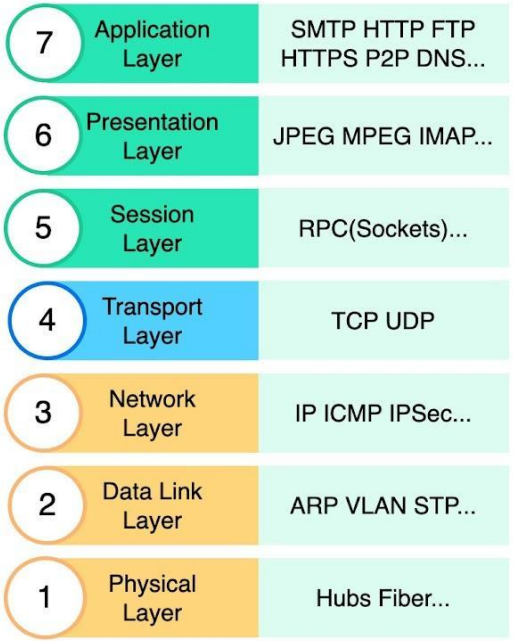
Use real-world analogies like a person whispering (unicast), giving a speech (broadcast), talking to a team (multicast), or speaking to the closest staff (anycast).

Protocols Across the Stack

Present protocols as per OSI/TCP-IP layers:

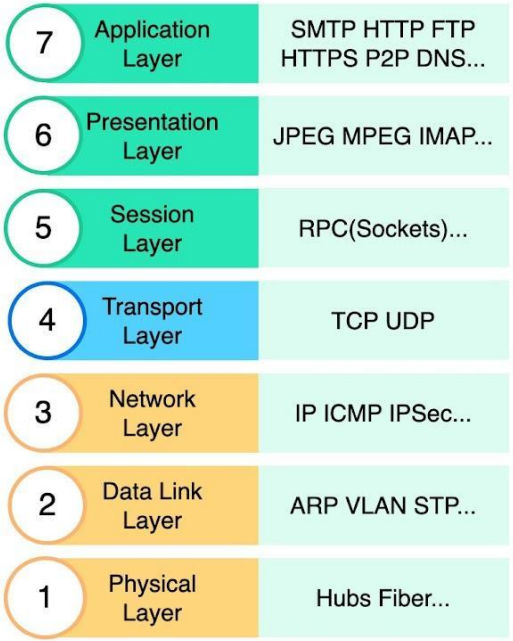
- **Transport Layer**
  - **TCP**: Reliable, ordered delivery
  - **UDP**: Faster, connectionless (used for streaming)

- **Application Layer**
  - HTTP, HTTPS – web traffic
  - FTP – file transfer
  - DNS – domain name resolution
  - SMTP/IMAP/POP3 – email services
  - DHCP – dynamic IP configuration



Protocols Across the Stack

- **Network Layer**
  - **IPv4, IPv6** – addressing
  - **ICMP** – diagnostics (ping)
  - **IPsec** – secure IP communication
- **Link Layer**
  - Ethernet, ARP – MAC resolution



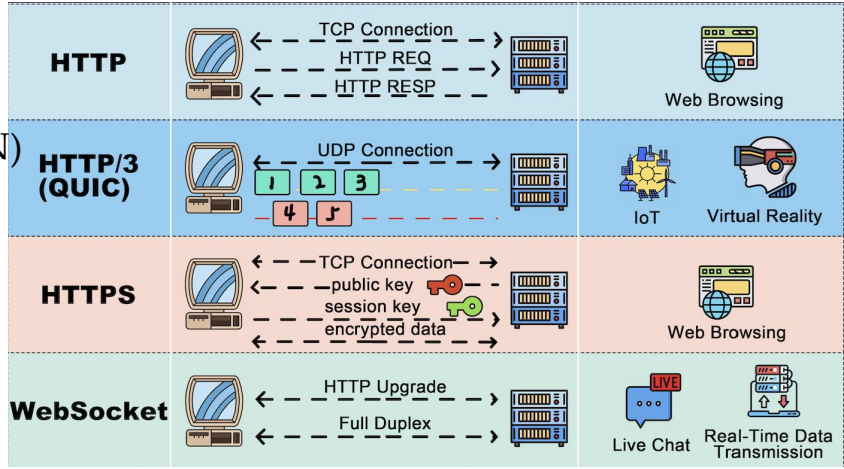
Common Network Ports

- **Port Concepts**
  - 16-bit logical endpoints used with TCP/UDP
  - **Well-known ports (0–1023)** are reserved for standard services.

Service	Protocol	Port	Usage
FTP	TCP	21	File Transfer Protocol
SSH	TCP	22	Secure Remote Login
Telnet	TCP	23	Remote Terminal (legacy)
SMTP	TCP	25	Sending Email
DNS	UDP/TCP	53	Domain Resolution
DHCP Server	UDP	67	Dynamic Host Configuration
HTTP	TCP	80	Web Access
HTTPS	TCP	443	Secure Web
RDP	TCP	3389	Remote Desktop Protocol
MySQL	TCP	3306	Database
PostgreSQL	TCP	5432	Database

Common Internet Protocols

Network protocols are standardized rules that define how data is formatted, transmitted, and interpreted between devices on a network.



1. HTTP – HyperText Transfer Protocol

- **Purpose**: Transfer web resources (HTML, CSS, JSON)
- **Model**: Client-Server
- **Transport Layer**: TCP
- **Use Case**: Web browsing, REST APIs

2. HTTPS – Secure HTTP

- **Purpose**: Secure version of HTTP using encryption
- **Security Layer**: TLS/SSL over TCP
- **Use Case**: Online banking, login systems, e-commerce

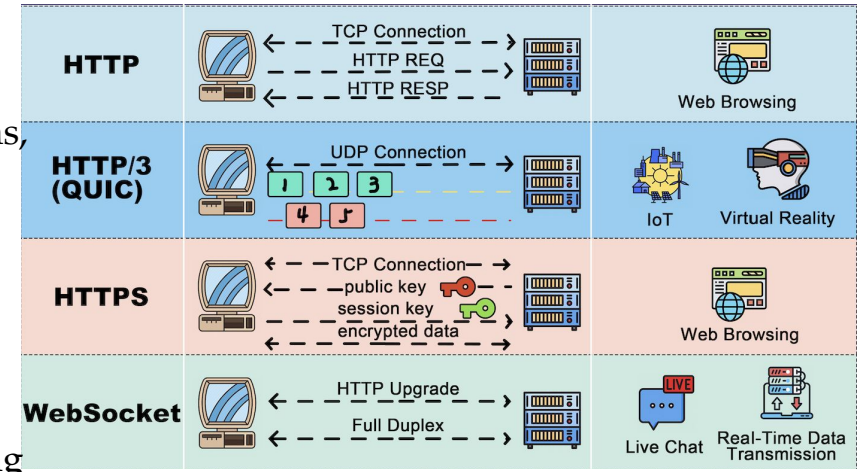
Common Internet Protocols

3. HTTP/3 – Next-Gen Web Protocol

- **Built On**: QUIC (UDP-based)
- **Advantages**: Faster connection setup, multiplexed streams
  Use Case: Mobile-first and high-latency applications, VR/AR

4. WebSocket

- **Purpose**: Real-time, bidirectional communication
- **Connection Type**: Persistent over TCP
- **Use Case**: Chat apps, online games, financial trading dashboards



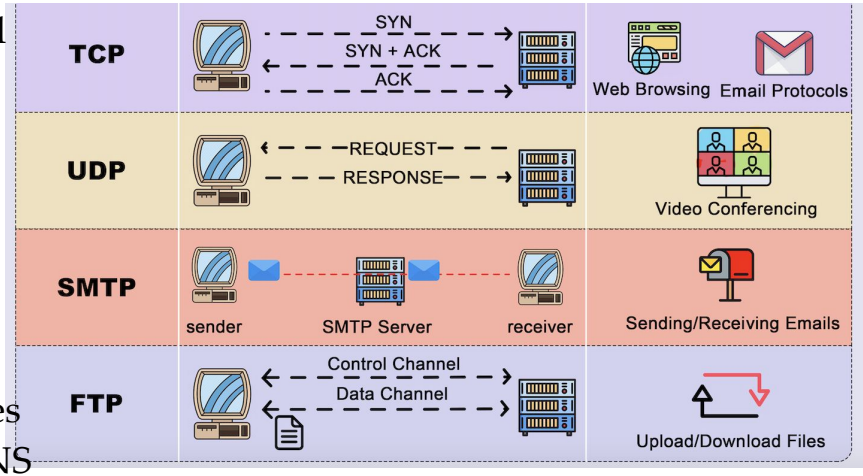
Common Internet Protocols

5. TCP – Transmission Control Protocol

- **Type**: Connection-oriented
- **Features**: Reliable delivery, error checking, ordered packets
- **Use Case**: Email, file transfer, web browsing

6. UDP – User Datagram Protocol

- **Type**: Connectionless
- **Features**: Lightweight, fast, no reliability guarantees
- **Use Case**: Live video/audio streaming, gaming, DNS



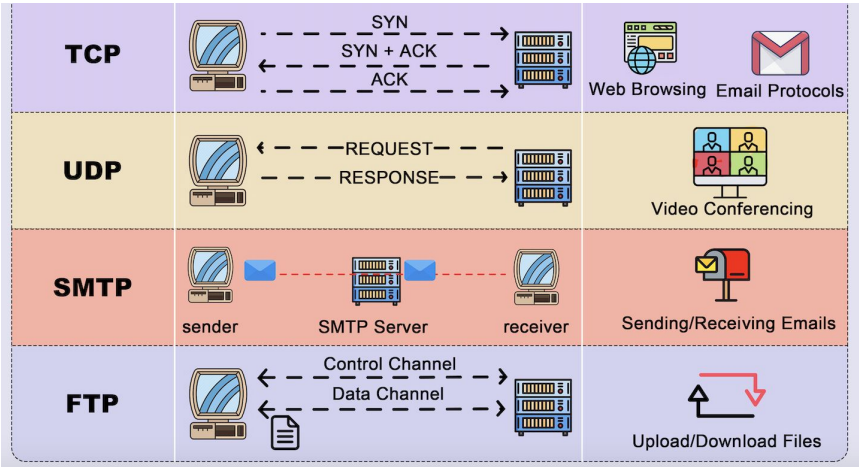
Common Internet Protocols

7. SMTP – Simple Mail Transfer Protocol

- **Purpose**: Send and route email between mail servers
- **Transport Layer**: TCP (typically port 25)
- **Use Case**: Outbound email delivery

8. FTP – File Transfer Protocol

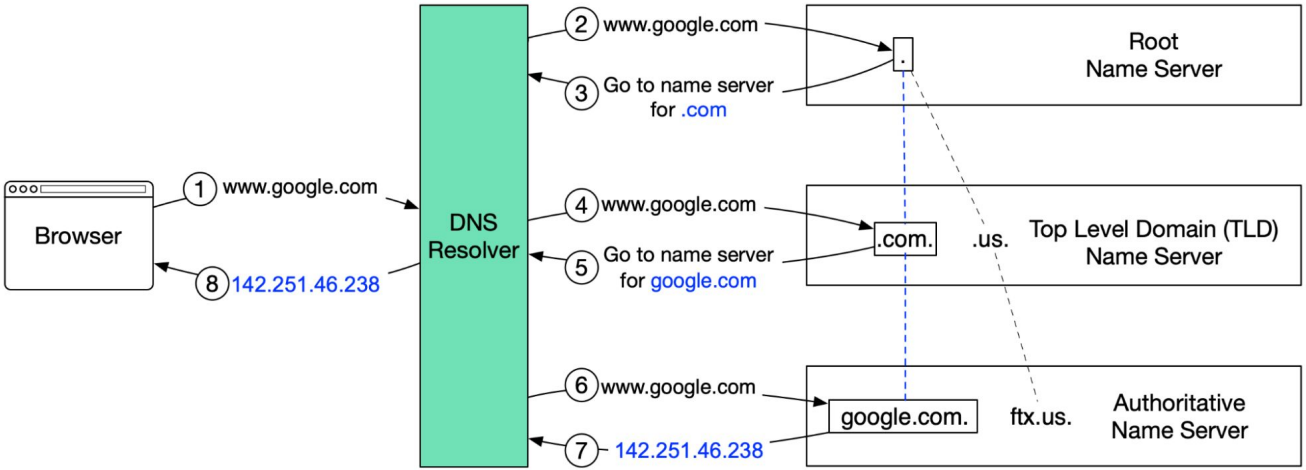
- **Purpose**: Transfer files between client and server
- **Connection**: Dual-channel (Control + Data)
- **Transport Layer**: TCP
- **Use Case**: File uploads, web hosting



DNS Architecture and Name Resolution

What is DNS?

The **Domain Name System (DNS)** functions as the Internet's address book. It translates **human-friendly domain names** (e.g., google.com) into **machine-readable IP addresses** (e.g., 142.251.46.238). This process allows users to access websites without remembering complex numerical addresses.



DNS Architecture and Name Resolution

How DNS is Organized

To ensure scalability and reliability, DNS uses a **hierarchical tree structure** with multiple tiers of servers:

1. **Root Name Servers (".")**
   - Store references to all Top-Level Domain (TLD) name servers.
   - There are **13 logical root servers** distributed globally.

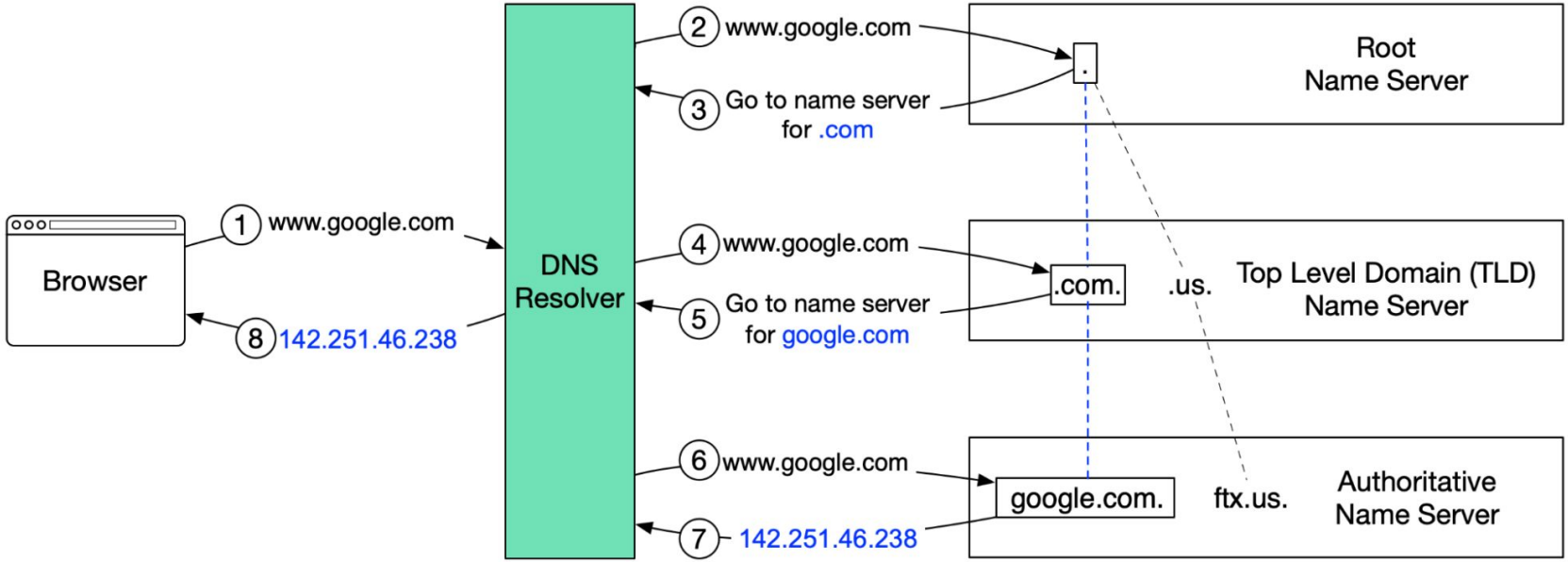
2. **TLD Name Servers**
   - Maintain information for each Top-Level Domain, such as:
     - Generic TLDs (`.com`, `.org`)
     - Country-code TLDs (`.us`, `.uk`)
     - Specialized TLDs (`.test`)
   - Direct queries to the appropriate authoritative servers.

DNS Architecture and Name Resolution

3. Authoritative Name Servers

- Hold the definitive records for a domain.
- Provide the final answer to DNS queries.
- Managed by domain registrars (e.g., GoDaddy, Namecheap).

How DNS Lookup Works



How DNS Lookup Works

When you enter google.com in your browser:

1. The browser contacts the **local DNS resolver**.
2. The resolver queries a **Root Name Server**.
3. The Root Server replies with the address of the relevant **TLD Name Server** (.com).
4. The resolver queries the .com TLD server.
5. The TLD server returns the address of Google's **Authoritative Name Server**.
6. The resolver queries the authoritative server for google.com.
7. The authoritative server responds with the **IP address** (142.251.46.238).
8. The resolver passes this IP back to the browser.
9. The browser connects to the website using the resolved IP.

Note: A typical DNS lookup completes in **20–120 milliseconds**.

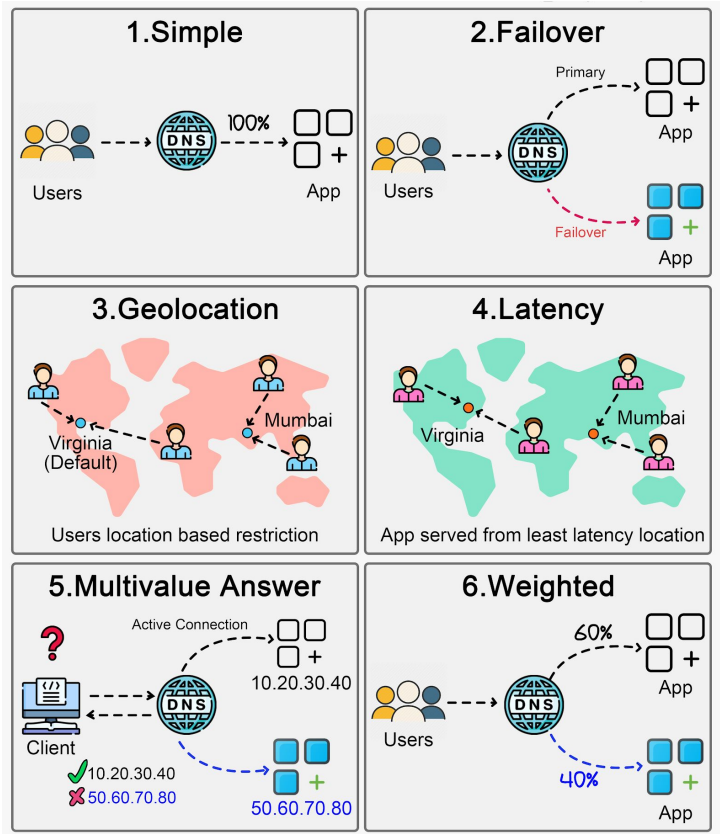
Common DNS Record Types

Record Type	Description
A (Address) Record	Maps a domain to an IPv4 address.
AAAA Record	Maps a domain to an IPv6 address.
CNAME (Canonical Name)	Aliases one domain to another (e.g., blog.example.com → example.com).
PTR (Pointer) Record	Provides reverse DNS lookups (IP → domain).
MX (Mail Exchange)	Directs email to the correct mail server.
NS (Name Server)	Specifies authoritative name servers for the domain.
SRV (Service) Record	Defines host and port for specific services (e.g., VoIP).
TXT (Text) Record	Contains human-readable text, such as verification keys and SPF data.

DNS Routing & Load Distribution Policies

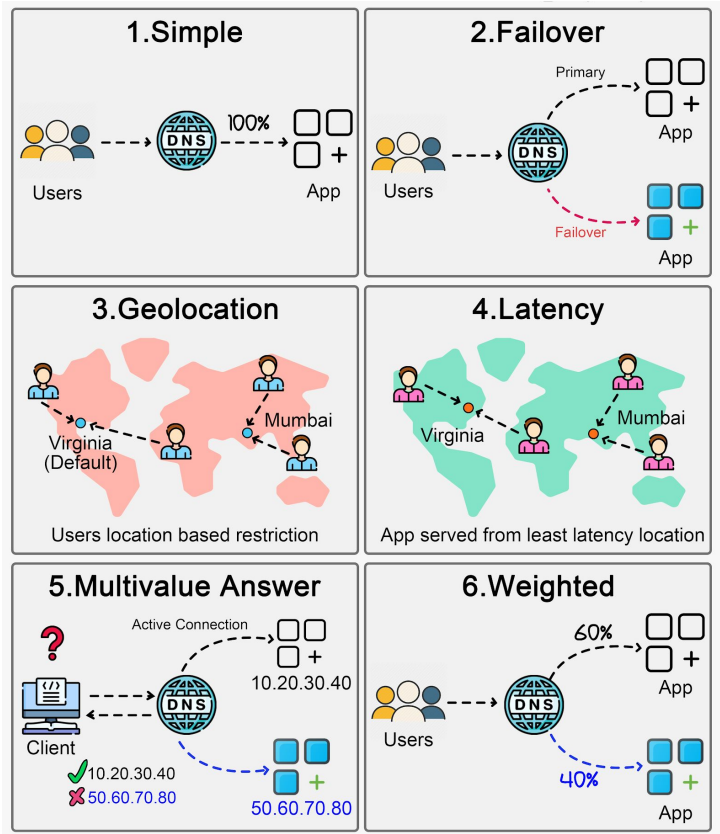
- **Common Types of Routing Policies**

  - **Simple:** Directs all traffic to a single endpoint based on a standard DNS query without any special conditions or requirements.
  - **Failover:** Routes traffic to a primary endpoint but automatically switches to a secondary endpoint if the primary is unavailable.
  - **Geolocation:** Distributes traffic based on the geographic location of the requester, aiming to provide localized content or services.



DNS Routing & Load Distribution Policies

- **Latency:** Directs traffic to the endpoint that provides the lowest latency for the requester, enhancing user experience with faster response times.
- **Multivalue Answer:** Responds to DNS queries with multiple IP addresses, allowing the client to select an endpoint. However, it should not be considered a replacement for a load balancer.
- **Weighted Routing Policy:** Distributes traffic across multiple endpoints with assigned weights, allowing for proportional traffic distribution based on these weights.



NAT and Address Translation

Network Address Translation (NAT) is a technique that enables multiple devices on a private network to **share a single public IP address** when accessing the Internet.

NAT is a foundational technology that has allowed the Internet to scale despite IPv4 address limitations.

- **How Does NAT Work?**

1. **Private Device Request**
   - A device (e.g., laptop or phone) sends an outbound request.
   - The request includes its **private IP address** (e.g., 192.168.1.10).

2. **Address Translation**
   - The router running NAT **replaces** the private IP with its **public IP address**.
   - It also records this translation in a **NAT table** to track which internal device initiated the connection.

NAT and Address Translation

3. Internet Communication

- The modified request, now showing the router's public IP, is sent to the destination server on the Internet.

4. Response Handling

- The server replies to the public IP address of the router.
- The router consults the NAT table to identify which private IP should receive the response.

5. Delivery to Device

- The router **rewrites the destination address back** to the original private IP and forwards the response to the correct device.

NAT and Address Translation

NAT has several important uses:

- **Conserves public IP addresses:** Without NAT, IPv4 addresses would have been depleted much faster, severely limiting the growth of the Internet.
- **Allows sharing:** It allows sharing a single public IP address across multiple devices.
- **Acts as a basic firewall:** NAT acts as a basic firewall that hides internal IP addresses.
- **Eases network management:** NAT also makes it easy to manage large networks.

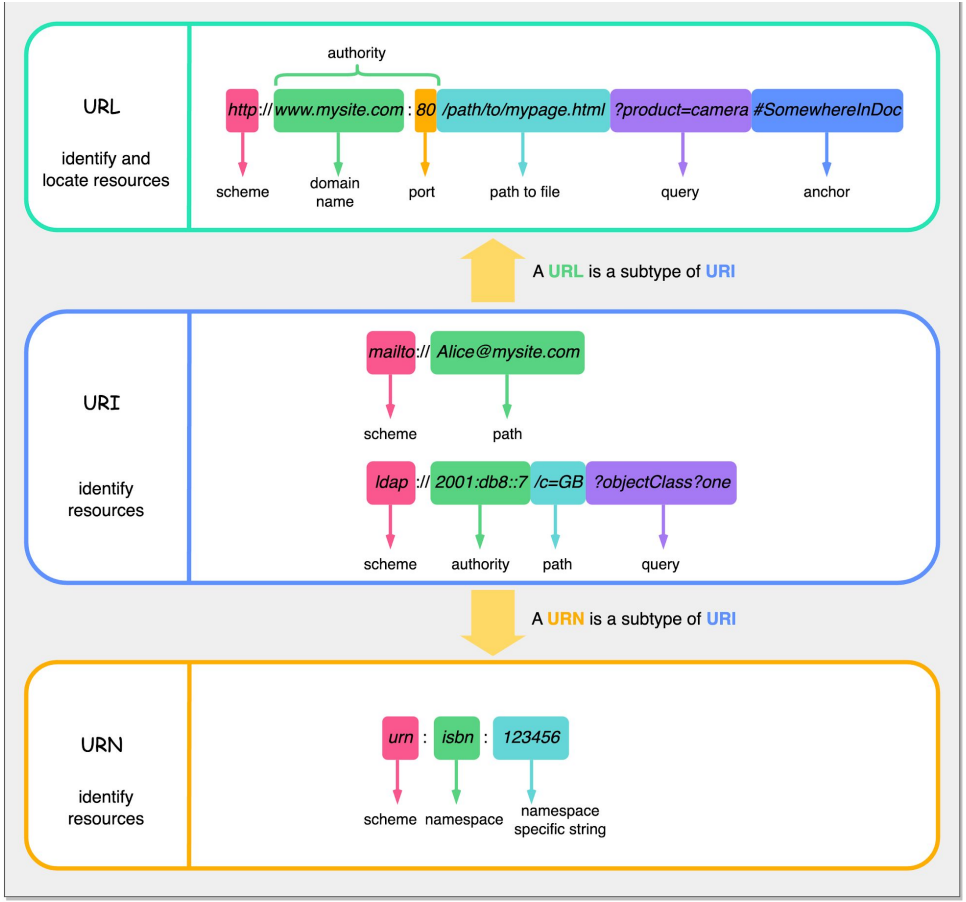
URLs, URIs, and Resource Identifiers

A **Uniform Resource Identifier (URI)** is a generic term used to identify any resource on the Internet.

A URI can either **locate** or **name** a resource.

Types of URIs:

- **URL** (*Uniform Resource Locator*): Locates the resource.
- **URN** (*Uniform Resource Name*): Names the resource without specifying its location.



HTTP Request Methods

HTTP defines a set of methods (sometimes called "verbs") that specify the intended action on a resource. Here are the **top 9 methods you should know:**

- HTTP GET
  This retrieves a resource from the server. It is idempotent. Multiple identical requests return the same result.
- HTTP PUT
  This updates or Creates a resource. It is idempotent. Multiple identical requests will update the same resource.
- HTTP POST
  This is used to create new resources. It is not idempotent, making two identical POST will duplicate the resource creation.

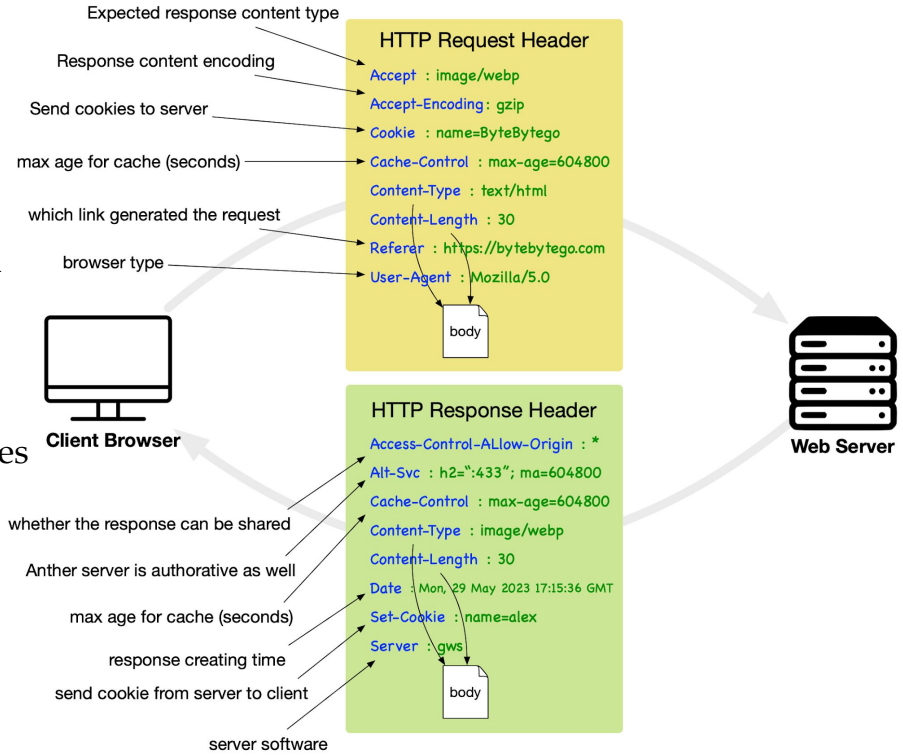
HTTP Request Methods

- HTTP DELETE
  This is used to delete a resource. It is idempotent. Multiple identical requests will delete the same resource.
- HTTP PATCH
  The PATCH method applies partial modifications to a resource.
- HTTP HEAD
  The HEAD method asks for a response identical to a GET request but without the response body.
- HTTP CONNECT
  The CONNECT method establishes a tunnel to the server identified by the target resource.
- HTTP OPTIONS
  This describes the communication options for the target resource.
- HTTP TRACE
  This performs a message loop-back test along the path to the target resource.

HTTP Headers

HTTP headers are metadata sent along with requests and responses.
They inform the server and client about:

- **Content Type:** What kind of data is being transmitted (Content-Type: application/json).
- **Authentication:** Credentials for access control (Authorization: Bearer <token>).
- **Caching:** Instructions for storing and reusing responses (Cache-Control).
- **Cookies:** State management (Set-Cookie).



Why headers matter:
They enable REST APIs to communicate effectively by standardizing how data and context are exchanged.

HTTP Cookies

HTTP is **stateless**, meaning each request is independent and has no memory of previous interactions. **Cookies** solve this by storing small pieces of stateful information on the client.

🔹 How Cookies Work

- When you log in, the server sends a Set-Cookie header.
- The browser stores the cookie and includes it in every subsequent request.
- Cookies act as **session identifiers** or **preference trackers**.

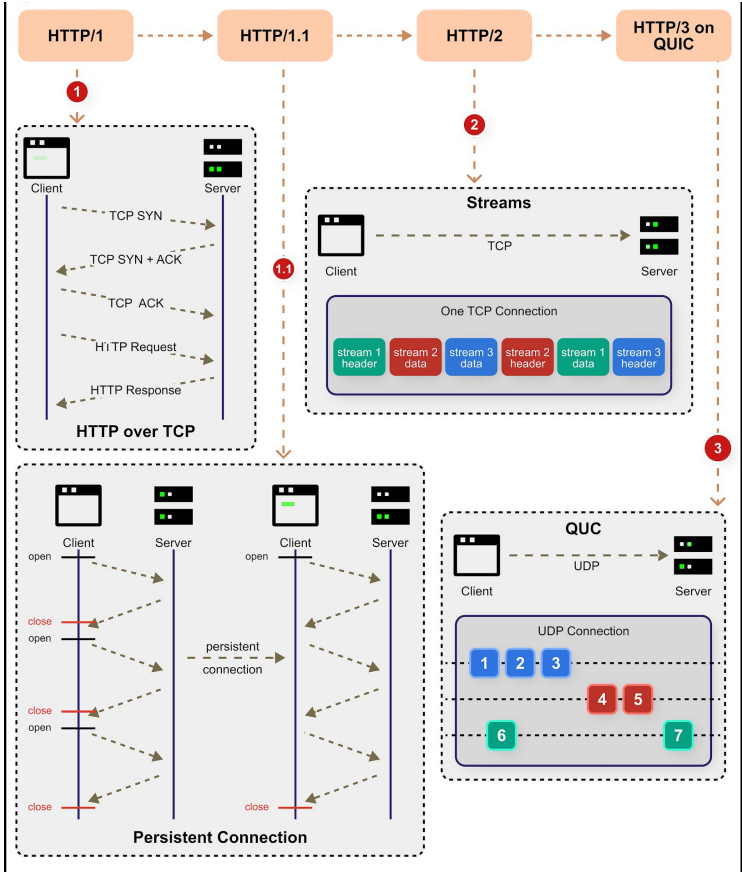
Key Cookie Attributes

- **Name & Value:** The data being stored.
- **Domain:** Scope of the cookie.
- **Path:** URL path where the cookie is valid.
- **Secure:** Sent only over HTTPS.
- **HttpOnly:** Inaccessible to JavaScript (mitigates XSS).
- **SameSite:** Controls cross-site sending.

HTTP Evolution

HTTP 1 started in 1996 followed by HTTP 1.1 the very next year. In 2015, HTTP 2 came about and in 2019 we got HTTP 3. With each iteration, the protocol has evolved in new and interesting ways.

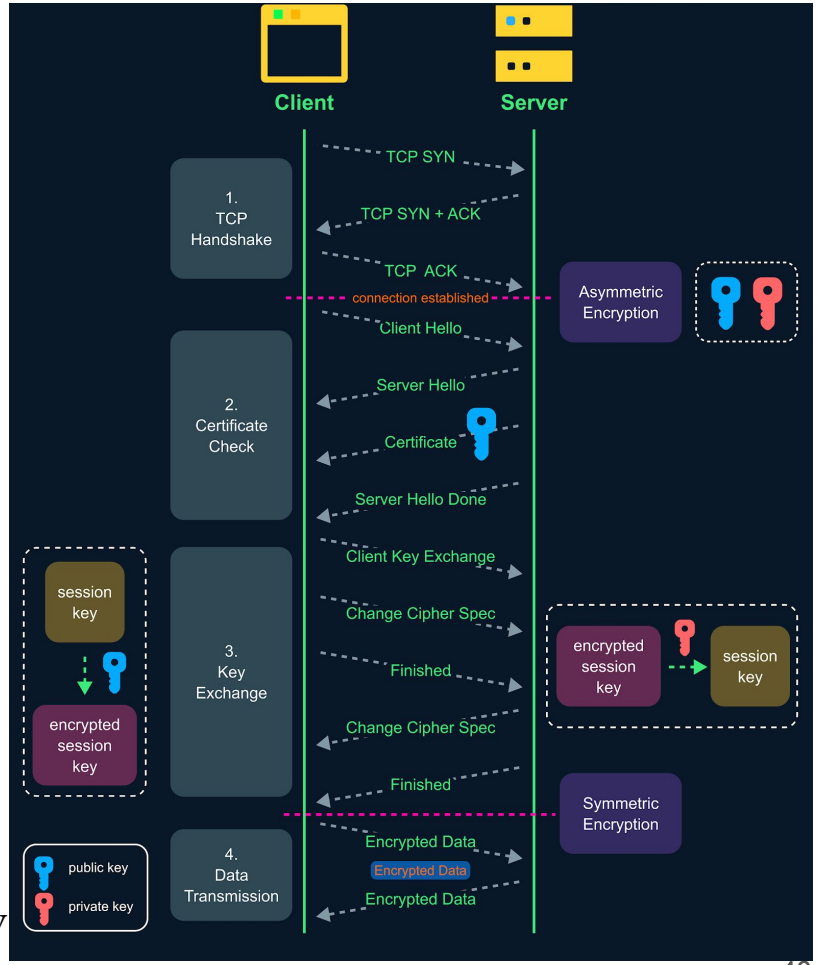
- HTTP 1 (and its sub-versions) introduced features like persistent connections, pipelining, and the concept of headers. The protocol was built on top of TCP and provided a reliable way of communication over the World Wide Web. It is still used despite being over 25 years old.
- HTTP 2 brought new features such as multiplexing, stream prioritization, server push, and HPACK compression. However, it still used TCP as the underlying protocol.
- HTTP 3 uses Google's QUIC, which is built on top of UDP. In other words, HTTP 3 has moved away from TCP.



HTTPS & SSL/TLS Fundamentals

- **What is HTTPS?**
  - Secure version of HTTP.
  - Encrypts data using **TLS (Transport Layer Security)**.
  - Prevents eavesdropping and tampering.
- **Encryption Process (Steps)**

1. **TCP Connection**
   - Client and server create a TCP connection.
2. **TLS Handshake**
   - **Client Hello:** Lists supported TLS versions and ciphers.
   - **Server Hello:** Chooses cipher and sends SSL certificate.
   - **Certificate Validation:** Client verifies server's identity



HTTPS & SSL/TLS Fundamentals

3. Session Key Exchange

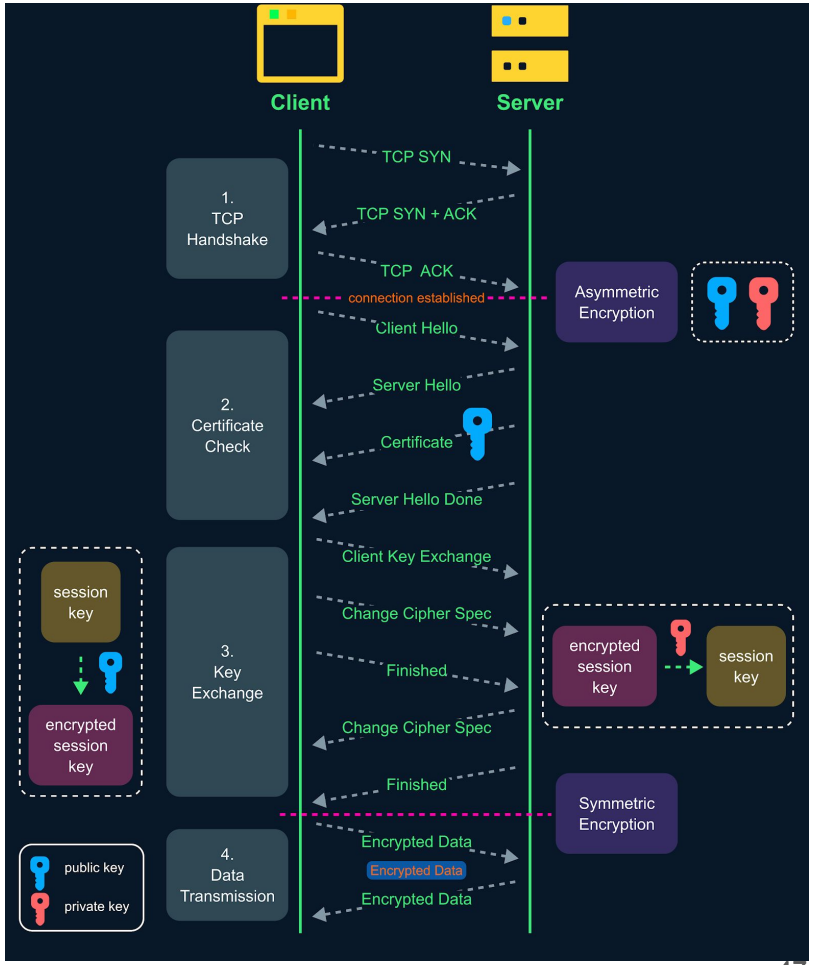
- Client generates a **session key**.
- Encrypts the key with the server's **public key**.
- Server decrypts using its **private key**.

4. Secure Data Transfer

- Both use the **same symmetric key** to encrypt/decrypt all data.

Why Use Symmetric Encryption?

- **Security:** Public keys are only for initial exchange.
- **Performance:** Symmetric encryption is faster for ongoing data.



HTTPS & SSL/TLS Fundamentals

HTTPS

- Encrypts your data to prevent eavesdropping and tampering.
- Uses **digital certificates** to verify website identity.
- Creates a secure, trusted connection.

SSL/TLS Handshake

- Cryptographic process that:
  - Negotiates encryption settings.
  - Exchanges keys securely.
  - Confirms server authenticity.

Secure Data Transmission

- Data travels through an **encrypted tunnel**.
- Protects information from cyber threats during transfer.

HTML's Role

- Defines how web pages are structured and linked.
- **Hypertext** means text containing embedded links to other resources.

How HTTPS Can Be Intercepted

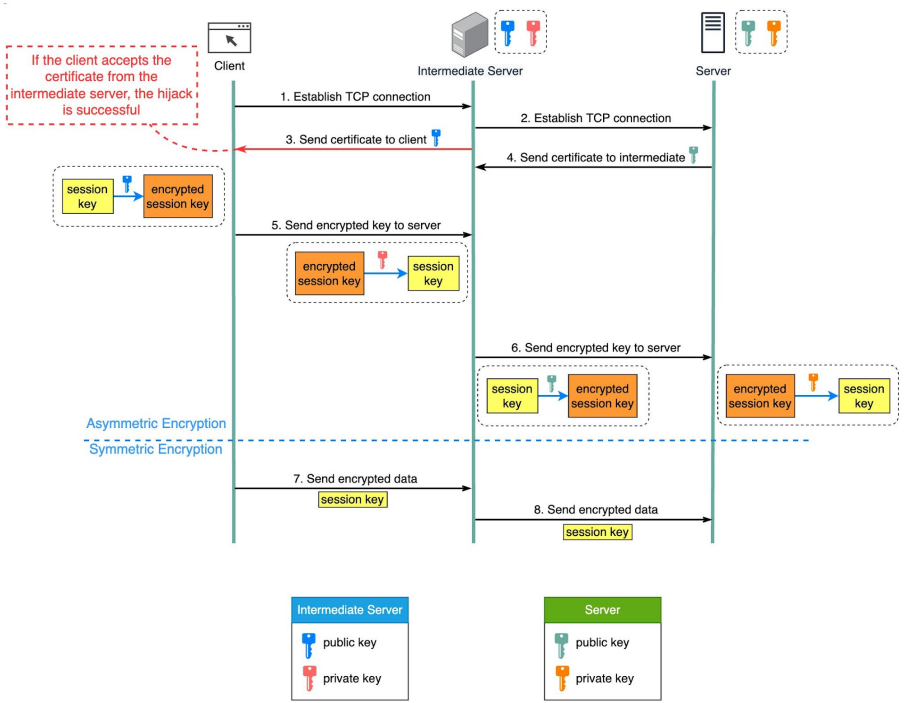
Why can tools like Fiddler capture HTTPS traffic?

- They act as a **trusted "man in the middle"**.
- This requires the interceptor's **root certificate installed in your system's trust store**.

How Packet Interception Works

1. Client Connection

- Client tries to connect to the server.
- Connection is routed to the **intermediate server** instead.



How HTTPS Can Be Intercepted

2. **Dual Connections**

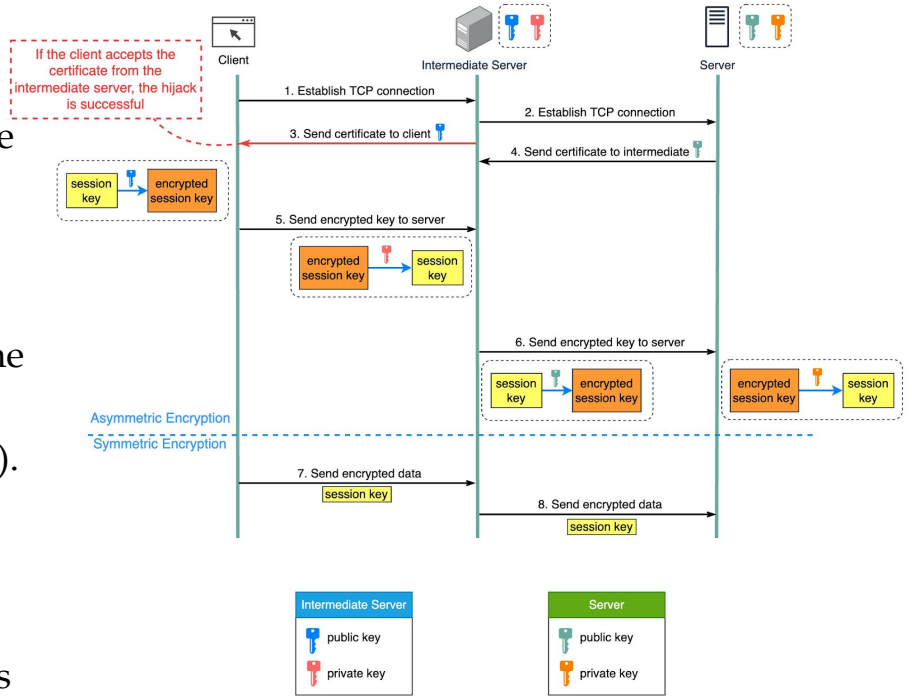
- The intermediate server connects separately to the real server.

3. **Certificate Exchange**

- Intermediate server sends **its own certificate** to the client.
- Client trusts it (since the root certificate is trusted).

4. **Server Certificate Validation**

- Intermediate server also validates the real server's certificate.



How HTTPS Can Be Intercepted

5. **Session Key Exchange**

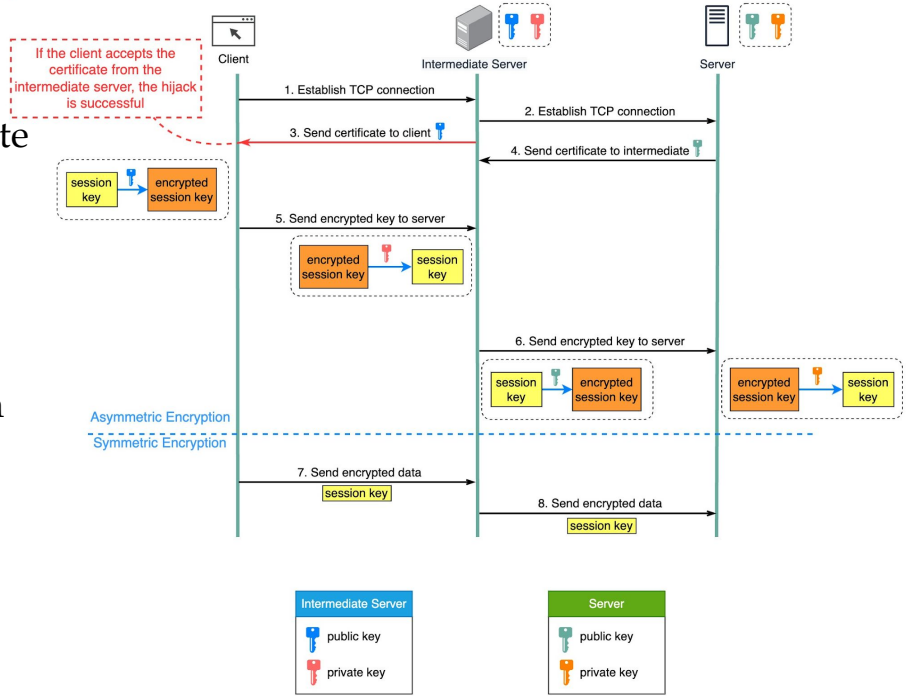
- Client encrypts a session key with the intermediate server's public key.
- Intermediate server decrypts it.

6. **Re-Encryption to Server**

- Intermediate server encrypts the session key with the real server's public key and forwards it.

7. **Transparent Relay**

- Client and server believe they have a secure connection.
- Intermediate server **sees and decrypts all data in the middle**.



PKI & Certificate Authorities

Public Key Infrastructure (PKI) is the system that:

- Manages **digital certificates**
- Enables **secure communication and authentication**

PKI uses **asymmetric encryption** (public/private keys).

Core Components of PKI

1. **Certificate Authority (CA)**

- A trusted organization that:
  - Issues digital certificates
  - Verifies identity of certificate holders

2. **Registration Authority (RA)**

- Verifies user or organization details before certificates are issued

3. **Certificate Repository**

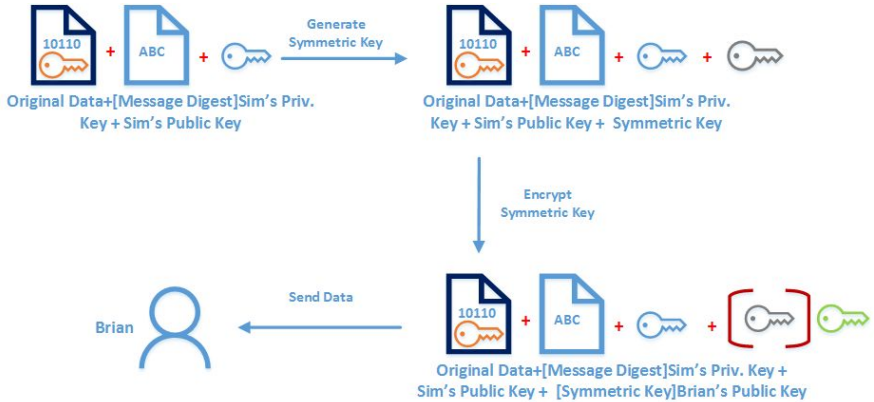
- Stores issued certificates and revocation lists

4. **Certificate Revocation List (CRL)**

- Lists revoked or invalid certificates

How PKI Works

1. The CA **verifies your identity**.
2. The CA **issues a certificate** linking your identity to your public key.
3. When someone connects to you:
    - They check your certificate.
    - They trust it because they trust the CA that signed it.
4. Encryption uses your **public key** and your **private key** remains secret.



Web Authentication Methods

Session

- Server **stores login state**.
- Browser gets a **session ID cookie**.
- Simple but **doesn't work well across devices**.

Token

- Server encodes identity into a **token**.
- Client stores the token and sends it with each request.
- No server-side session storage needed.
- Tokens require **secure handling and encryption**.

JWT (JSON Web Token)

- A **standardized token format**.
- Includes a **digital signature** for trust.
- Self-contained: all data is inside the token.
- Popular in REST APIs.

SSO (Single Sign-On)

- Central authentication service.
- **One login works across multiple websites**.
- Improves user experience and reduces password fatigue.

Web Authentication Methods

OAuth2

- Lets one site **access your data on another site** without sharing passwords.
- Common for “Login with Google/Facebook”.

QR Code Login

- Encodes a **random token in a QR code**.
- Scan the code to log in without typing credentials.
- Often used in mobile apps.

Real-Time Communication Methods

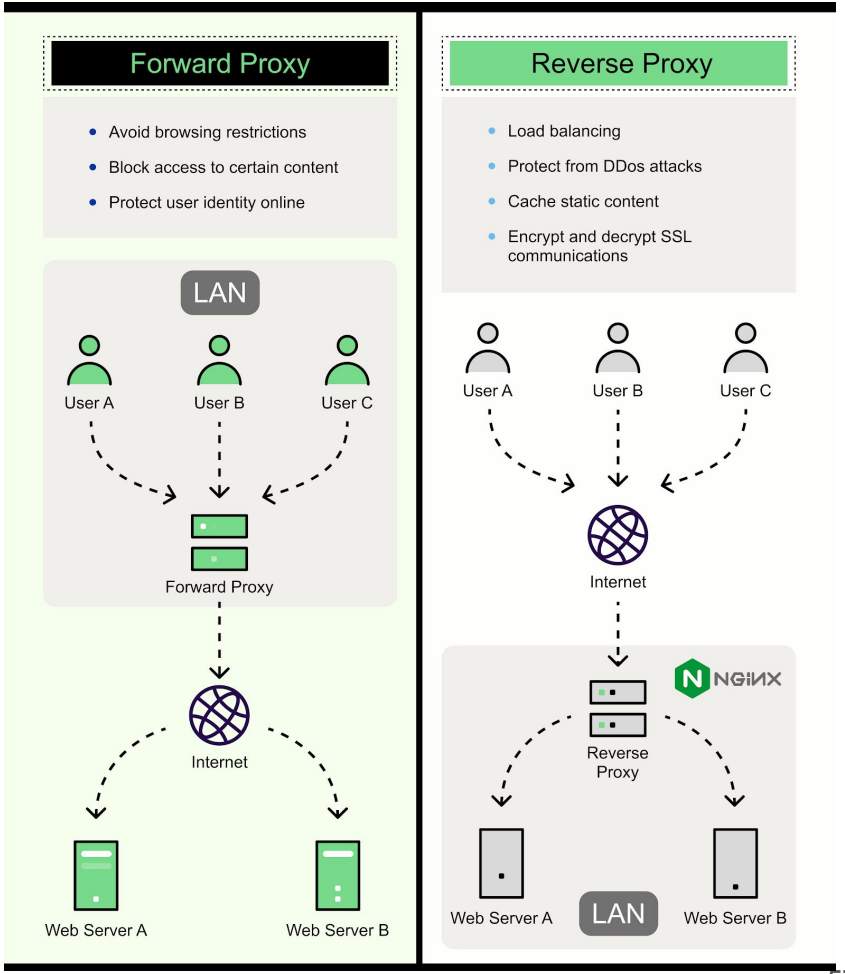
An HTTP server cannot automatically initiate a connection to a browser. As a result, the web browser is the initiator. What should we do next to get real-time updates from the HTTP server?

Both the web browser and the HTTP server could be responsible for this task.

- **Web browsers do the heavy lifting:** short polling or long polling. With short polling, the browser will retry until it gets the latest data. With long polling, the HTTP server doesn't return results until new data has arrived.
- **HTTP server and web browser cooperate:** WebSocket or SSE (server-sent event). In both cases, the HTTP server could directly send the latest data to the browser after the connection is established. The difference is that SSE is unidirectional, so the browser cannot send a new request to the server, while WebSocket is fully-duplex, so the browser can keep sending new requests.

Proxy vs Reverse Proxy

- **Forward Proxy**
  - **Position:** Between users and the internet.
  - **Purpose:** Acts on behalf of **clients**.
  - **Uses:**
    - Protect client devices.
    - Bypass browsing restrictions.
    - Block access to certain websites.
- **Reverse Proxy**
  - **Position:** Between clients and backend servers.
  - **Purpose:** Acts on behalf of **servers**.
  - **Uses:**
    - Protect backend servers.
    - Load balancing across multiple servers.
    - Cache static content for faster delivery.
    - Handle SSL encryption/decryption.



Virtual Private Network

A VPN, or **Virtual Private Network**, is a technology that creates a secure, encrypted connection over a less secure network, such as the public internet. The primary purpose of a VPN is to provide privacy and security to data and communications.

A VPN acts as a tunnel through which the encrypted data goes from one location to another. Any external party cannot see the data transferring.A VPN works in 4 steps:

- Step 1 - Establish a secure tunnel between our device and the VPN server.
- Step 2 - Encrypt the data transmitted.
- Step 3 - Mask our IP address, so it appears as if our internet activity is coming from the VPN server.
- Step 4 - Our internet traffic is routed through the VPN server.

SSH (Secure Shell)

- **What is SSH?**

  - A protocol for **secure remote connections** over untrusted networks.
  - Encrypts all traffic and supports authentication and data transfer.

- **Versions**

  - **SSH-1:** Original version (obsolete).
  - **SSH-2:** Modern standard (more secure), defined by IETF.

- **Three Main Layers**

  - **Transport Layer:** Handles **encryption**, integrity, and data protection.
  - **Authentication Layer:** Verifies **client identity** (e.g., password, keys).
  - **Connection Layer:** Creates multiple **logical channels** over one secure connection.

How Live Streaming Works

- **What is Live Streaming?**
  - Sending video in **real-time** over the internet.
  - Latency usually only a few seconds.
  - Used by YouTube Live, TikTok, Twitch.
- **Steps Behind the Scenes**

1. **Capture**

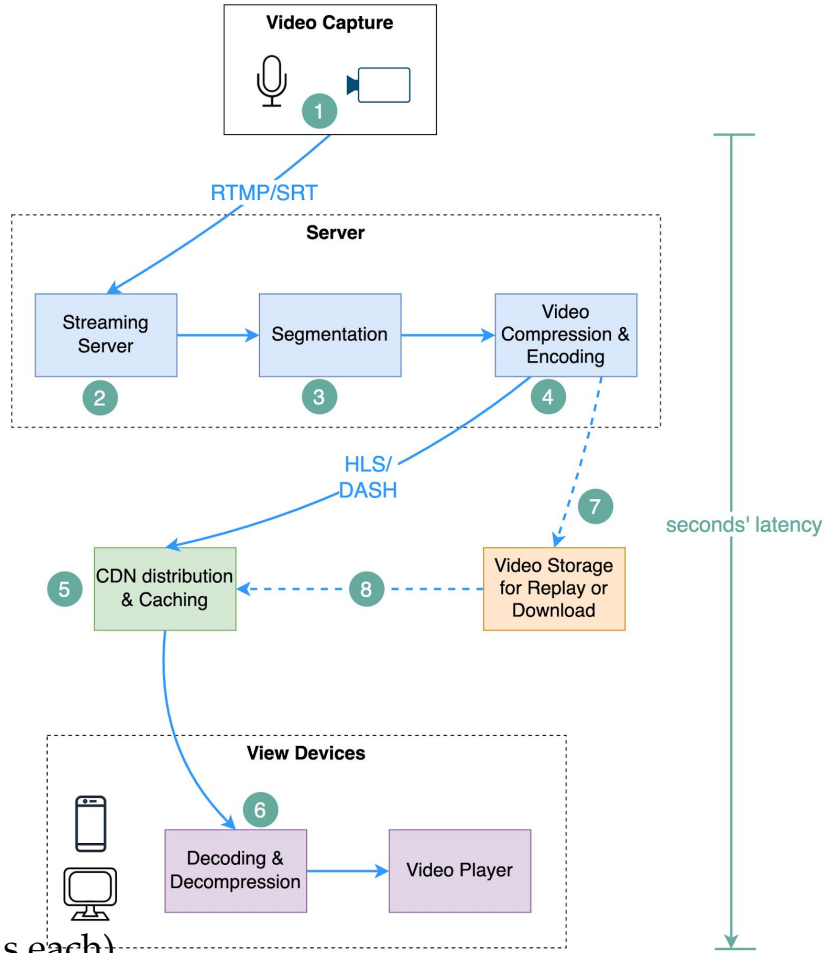
   Microphone and camera record raw video/audio.

2. **Compression & Encoding**

   Video compressed (e.g., H.264) to reduce size.

3. **Segmentation**

   Encoded video split into **small chunks** (few seconds each).



How Live Streaming Works

4. **Adaptive Bitrate Streaming**

- Multiple quality versions created to handle different devices and connections.

5. **Delivery to CDN**

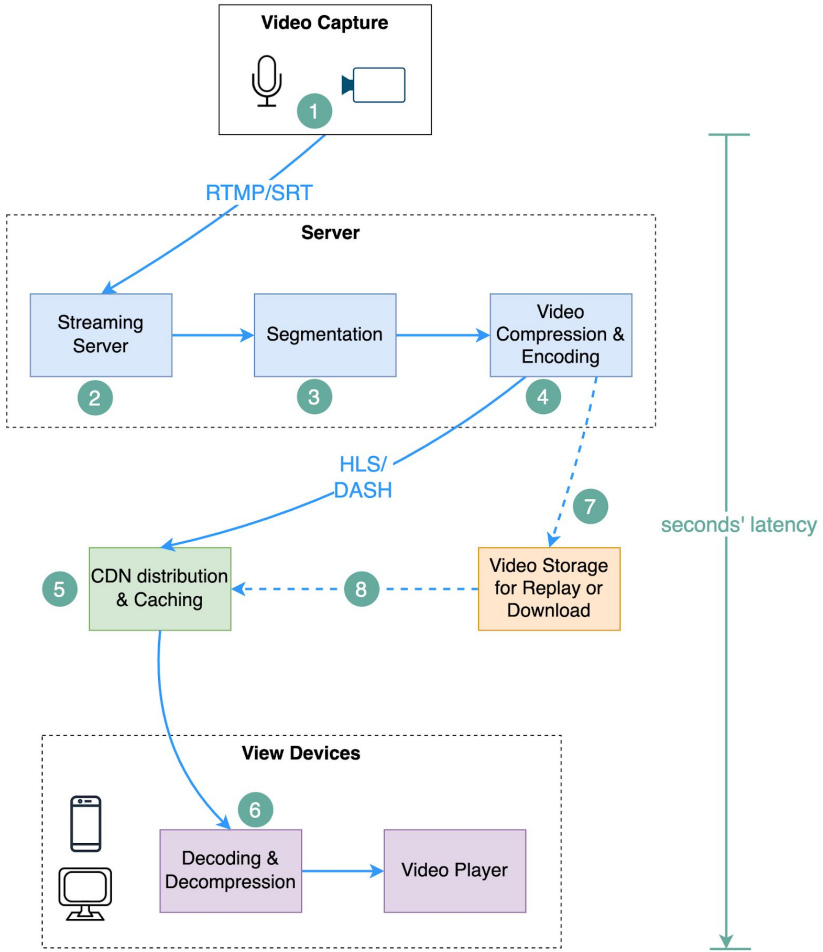
- Chunks sent to **CDN edge servers** close to viewers.

6. **Playback**

- Viewers' devices **download, decode, and play** the video.

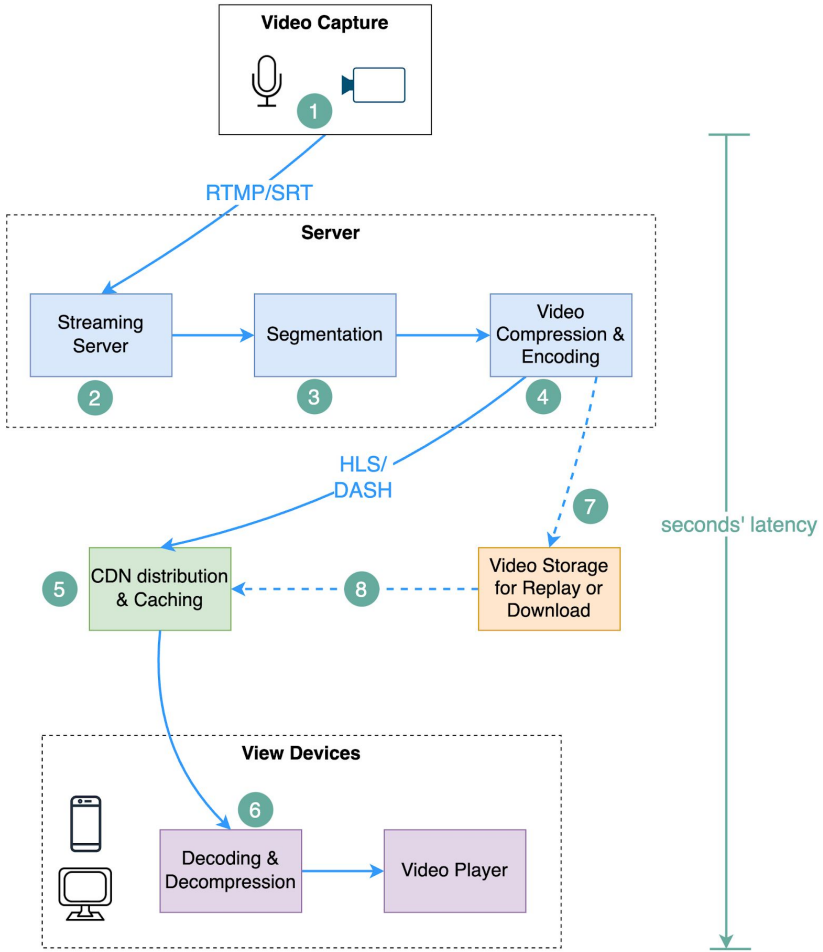
7. **Storage (Optional)**

- Content saved for replay later.



How Live Streaming Works

- **Standard Protocols**
  - **RTMP:** Sends live video to streaming servers.
  - **HLS:** HTTP Live Streaming (used on Apple devices).
  - **DASH:** Adaptive streaming over HTTP (not supported on Apple).



Thank you!
Any Questions?