COMPUTER NETWORKS

Why Computer Networks (CN)

Computer networks connect independent computing devices so they can **exchange data and share resources** (files, services, compute, sensors). Networks enable distributed applications (web, email, messaging, cloud services), coordinate remote devices, and allow systems to scale and be resilient. For anyone building hardware or software that communicates, networking is the foundation for interoperability, performance, and security.

Common applications (examples)

- **Web browser** — a client that fetches resources from web servers using HTTP/HTTPS. It performs DNS lookups, opens TCP (or QUIC) connections, and renders content.
- **Email** — involves multiple protocols: SMTP for sending, POP/IMAP for receiving/reading; mail clients and servers exchange messages across the network reliably.
- **Instant messaging / VoIP (WhatsApp, Skype)** — real-time messaging and media use transport-layer services (TCP or UDP), signaling protocols, and often encryption and NAT traversal techniques to deliver messages and media streams.
- **Remote login / Remote desktop (SSH, RDP, Telnet)** — remote-control protocols that provide interactive shells or graphical desktop streams; they rely on authentication, encryption, and the transport layer for reliability and latency management.

Types of background/networked tasks

Examples of tasks that run "behind the scenes" and rely on networks:

- **Name resolution** (DNS queries)
- **Software updates** and package downloads
- **Background synchronization** (cloud file sync, email push)
- **Telemetry and monitoring** (device health, metrics)
- **Push notifications**
- **Content delivery** (CDNs fetching media segments)
- **Authentication and authorization checks** to identity providers

These tasks are often automatic, use minimal local UI, and depend on robust, fault-tolerant networking.

Networked devices (endpoints)

Any device with a network interface can be a host:

- **Laptops and desktops** (full TCP/IP stacks, rich apps)
- **Smartphones and tablets** (cellular + Wi-Fi + Bluetooth)
- **Smartwatches and IoT devices** (constrained resources, often sleep cycles)
- **Embedded systems** (sensors, gateways, industrial controllers)

Each device exposes network interfaces, drivers, an IP address (or other identifiers), and typically one or more application services.

Why learn CN to build networked devices

To design and build devices that communicate reliably and securely you must understand:

- **Network stacks** (how packets travel from app to wire)
- **Sockets APIs** and session management
- **Protocol behaviors** (latency sensitivity, retransmission, ordering)
- **Addressing, NAT, and routing** (how your device is reached)
- **Security** (TLS, key storage, secure boot)
- **Performance & power tradeoffs** (bandwidth vs. energy)
  Without these, software/hardware will be hard to interoperate, debug, secure, and scale.

Wireshark (why use it)

Wireshark is a packet-capture and analysis tool. Use it to:

- Capture frames/packets on an interface
- Apply capture/display filters (BPF)
- Inspect headers and payloads for each protocol layer
- Follow TCP streams or reassemble fragmented traffic
- Troubleshoot performance, protocol bugs, misconfigurations, or security issues
  It makes the abstract “stack” visible so you can learn real protocol interactions.

OSI reference model — purpose & benefits

OSI (Open Systems Interconnection) is a conceptual 7-layer model that **modularizes networking functions** to aid design and education. It's not a protocol stack you run in practice, but it helps reason about responsibilities and interfaces between layers. The seven layers (top → bottom): Application, Presentation, Session, Transport, Network, Data Link, Physical.

Responsibilities of each OSI layer (concise)

1. **Application** — End-user services and application protocols (e.g., HTTP, SMTP, FTP). Interfaces to user programs.
2. **Presentation** — Data representation and translation (character encoding, serialization), compression, and encryption formatting.
3. **Session** — Dialog control: setting up, managing and tearing down sessions; checkpointing and reconnection logic in some systems.
4. **Transport** — End-to-end communication, reliability, flow and congestion control, multiplexing via ports (e.g., TCP, UDP).
5. **Network** — Logical addressing and routing between networks (e.g., IPv4/IPv6), fragmentation, and path selection.
6. **Data Link** — Framing, MAC addressing, local delivery on a single link or LAN segment, error detection (CRC), and link-level flow control.
7. **Physical** — Electrical/optical/radio signals, bit encoding, connectors and the physical medium properties.

TCP/IP (Internet) model — practical stack

The Internet uses a **practical 4-layer model** that maps OSI concepts into fewer layers:

- **Application layer** — Application protocols (HTTP, SMTP, DNS, SSH).
- **Transport layer** — TCP (reliable, ordered) and UDP (datagram, low overhead).
- **Internet layer** — IP routing and addressing (IPv4/IPv6); packets are forwarded across networks.
- **Link (Network Interface) layer** — Frames and physical transmission on an interface (Ethernet, Wi-Fi).

Units of data by layer: **Application data → Transport segment (TCP) / datagram (UDP) → IP packet → Link frame → bits on the wire**.

Transport layer — detailed responsibilities

- **Segmentation & reassembly** — large application messages are split into segments and reassembled at receiver.
- **Ports & multiplexing** — transport ports (e.g., 80, 443) allow multiple services on one host.
- **Reliability (TCP)** — three-way handshake (SYN, SYN-ACK, ACK), sequence numbers, ACKs, retransmissions, and ordered delivery.
- **Flow control** — sender limits data in flight to what receiver can handle (sliding window).
- **Congestion control** — algorithms to avoid overloading the network (slow start, congestion avoidance, AIMD principles).
- **UDP characteristics** — connectionless, no ordering/repair; used where low latency or application-level reliability is preferred (e.g., live audio/video).

Network (Internet) layer — core ideas

- **Logical addressing** — IP addresses (IPv4/IPv6) identify hosts and subnets.
- **Routing** — routers forward packets using routing tables; routing protocols (link-state, distance-vector, BGP for inter-domain) determine paths.
- **Subnetting & CIDR** — divide address space into networks; masks and prefix lengths identify network vs host bits.
- **NAT (Network Address Translation)** — translates private IPs to public IPs at network edge.
- **Fragmentation** — when MTU limits are exceeded, packets may be fragmented and reassembled.

Data Link layer — practical details

- **MAC (Media Access Control) addresses** — hardware identifiers bound to interfaces used for local delivery.
- **Frames** — a data link packet includes destination MAC, source MAC, EtherType/length, payload, and FCS (CRC).
- **Switching vs bridging** — switches forward frames on a LAN by learning MAC-to-port mappings; bridges segment collision domains.
- **Media access control** — rules that decide who may transmit on a shared medium (e.g., CSMA/CD for classic Ethernet, CSMA/CA for Wi-Fi).

- **VLANs (802.1Q)** — logical separation of LANs at the switch level using tags.

Physical layer — signals & media

- **Bit encoding & signaling** — how 1s and 0s become electrical, optical, or radio signals (examples: Manchester encoding).
- **Guided media** — twisted-pair copper (Ethernet), coaxial cable, optical fiber. Tradeoffs: bandwidth, attenuation, cost, distance.
- **Unguided media** — wireless radio, microwave, infrared. Consider spectrum, interference, and attenuation.
- **Physical properties** — bandwidth (Hz), signal-to-noise ratio, attenuation, and how they constrain data rate (Shannon/Nyquist principles).

Media access & transmission modes

- **Media access** — methods like CSMA/CD (collision detection), CSMA/CA (collision avoidance), token passing, or switching eliminate collisions in modern networks.
- **Transmission modes**:
  - **Simplex** — one-way only (e.g., keyboard → computer).
  - **Half-duplex** — two-way, not simultaneous (e.g., walkie-talkie).
  - **Full-duplex** — simultaneous two-way (modern Ethernet over switches, duplex fiber).

Network types (scales)

- **PAN (Personal Area Network)** — very short-range personal devices (Bluetooth, NFC).
- **LAN (Local Area Network)** — devices in a building or campus connected by switches/routers (Ethernet).
- **WLAN (Wireless LAN)** — Wi-Fi networks providing LAN services wirelessly.
- **CAN (Campus Area Network)** — connects multiple LANs across a campus.
- **MAN (Metropolitan Area Network)** — covers a city or metropolitan area.
- **WAN (Wide Area Network)** — spans large geographic areas; the Internet is the largest WAN.

- **SAN (Storage Area Network)** — high-speed network that provides block-level storage access (iSCSI, Fibre Channel).

Topologies (physical/logical layouts)

- **Star** — devices connect to a central switch/hub (easy to manage; central point of failure for hubs).
- **Bus** — single shared medium; early Ethernet used bus; simple but collisions and scalability issues.
- **Ring** — devices connected in a closed loop (token ring historically).
- **Mesh** — multiple interconnections between nodes (redundant, high resilience).
- **Fully connected** — every node connects to every other (high redundancy, expensive).
- **Tree** — hierarchical combination of star networks (scalable for enterprise).

Key protocols (examples & roles)

- **DNS** — resolves domain names to IP addresses.
- **DHCP** — assigns IP addresses and network configuration dynamically.
- **ARP** — resolves IP addresses to MAC addresses on a LAN.
- **ICMP** — network diagnostics (ping, traceroute).
- **HTTP/HTTPS** — web resource transfer; HTTPS adds TLS encryption.
- **SMTP / POP / IMAP** — email transport and retrieval.
- **SSH / RDP** — secure remote shell and remote desktop access.
- **RTP / SIP** — media streaming and session initiation for voice/video.
- **TLS/SSL** — encryption and authentication for secure sessions.

Addressing and multiplexing

- **MAC address (Layer 2)** — unique hardware identifier for an interface.
- **IP address (Layer 3)** — logical, routable address for a host.
- **Port number (Layer 4)** — identifies specific application/service on a host.
- **Socket** — combination of IP + port used to identify an endpoint for a transport connection.

Error detection & correction

- **Checksum / CRC** — detect corrupted frames/packets; CRCs in data link frames detect bit errors.
- **ARQ (Automatic Repeat reQuest)** — retransmission strategy when errors are detected (used in TCP).
- **FEC (Forward Error Correction)** — sender adds redundant data so receiver can correct some errors without retransmission (used in wireless or streaming).

Troubleshooting & diagnostic tools (quick)

- **ping** — ICMP echo to verify reachability and basic latency.
- **traceroute** — discover the path and per-hop delay to a destination.
- **tcpdump / tshark** — CLI packet capture and analysis.
- **Wireshark** — GUI packet capture and deep protocol inspection.
- **nslookup / dig** — DNS queries and diagnosis.
- **iperf** — measure throughput between two endpoints.
- **netstat / ss** — socket and port state inspection on a host.

Security essentials

- **Encryption (TLS, IPsec)** — protect confidentiality and integrity of data in transit.
- **Authentication & Authorization** — verify identity (passwords, keys, OAuth) and enforce permitted actions.
- **Firewalls & ACLs** — control traffic flow at network boundaries.
- **VPNs & Tunnels** — secure remote access and site-to-site connectivity.
- **Secure coding & patching** — minimize attack surface and fix vulnerabilities.