**Week 1:**

**Network:** A system of “links” that interconnect “nodes” to move “information” between nodes

**Characteristics of the Internet:**

1. **Global Connectivity:** The internet connects people, devices and systems worldwide, enabling instant communication and data sharing regardless of geographic location

2. **Decentralisation:** The internet is not controlled by a single authority, allowing resilience and flexibility in its operations

3. **Interoperability:** The internet enables different devices, OS, and applications to communicate seamlessly through standard protocols like TCP/IP, HTTP and DNS

4. **Scalability:** The internet can expand to accommodate more users, devices, and services without significant structural changes

5. **Accessibility:** Users can access a limitless array of information, services and tools

6. **Multimedia Support:** The internet facilitates the transfer and consumption of diverse types of media, including text, images, audio and video

7. **Hypertext and Hyperlinks:** The web (a significant part of the internet) is built on hypertext and hyperlinks, enabling users to navigate through interconnected content easily

8. **Real-time Communications:** The internet supports real-time interactions through tools like instant messaging, video conferencing and live streaming

9. **User-driven Content:** Platforms like social media, blogs and forums have transformed users from passive consumers to active content creators

10. **Ubiquity and** **Mobility:** With wireless networks and mobile devices, the internet has become accessible anytime and anywhere, integrating seamlessly into daily life

11. **Anonymity and Privacy Challenges:** The internet allows for a degree of anonymity, but also brings challenges like cybersecurity threats, privacy content and the spread of misinformation

**Federated System:**

- The internet ties together different networks (>20,000 ISP (internet service provider) networks)

- Tied together by **IP (Internet Protocol):** A single common interface between users and the network and between networks

- A single, common interface is great for interoperability but tricky for business.

This is because:

* Ease of interoperability is the internet’s most important goal
* Practical realities of incentives, economic and real-world trust, drive topology, route selection and service evolution

**Diversity and Dynamic Range:**

- Communication Latency: Nanoseconds to seconds (109)

- Bandwidth: 100bits/second to 400 Gigabits/seconds (109)

- Packet Loss: 0 – 90%

- Technology: Optical, Wireless, Satellite, Copper

- **Endpoint Devices:** From sensors and cell phones to data centres and supercomputers

- **Applications:** Social networking, file transfers, skype, live TV, gaming, remote medicine, backup and IM

- **Users:** The governing, governed, operators, malicious, naïve, savvy, embarrassed, paranoid, addicted, cheap

**The Internet:**

- **Internet:** Network of networks (interconnected IPS)

- **Protocols** are everywhere.

* Control sending, and receiving of messages
* (e.g., HTTP (Web), streaming videos, Skype, TCP, IP, Wi-Fi, 4G, Ethernet)

- **Internet Standards:**

* RFC: Request for Comments
* IETF: Internet Engineering Task Force

- **Infrastructure:** Provides services to application

* Web, streaming, multimedia, teleconferencing, email, games, e-commerce, social media, interconnected appliances

- **Programming interface to Distributed Applications:**

* “Hooks” allows sending/receiving apps to “connect” to use internet transport service
* Provides service options, analogous to postal service

**Protocols:**

- Protocols define the format, order of messages sent and received among network entities and action taken on message transmission, receipt

- **Network Protocols:** All communication activity on the Internet is governed by protocols

**Network Edge:**

- Network Edge:

* Hosts: clients and servers
* Servers often in data centres

- Access networks, physical media:

* Wired, wireless communication links

- Network Core:

* Interconnected routers
* Network of networks

**Access Networks and Physical Media:**

How do you connect end systems and edge routers?

* Residential access nets
* Institutional access networks (school company)
* Mobile access networks (WiFi, 4G/5G)

**Access Networks: Cable-based access**

- Frequency Division Multiplexing (FDM): Different channels transmitted in different frequency bands

- HFC (Hybrid Fiber Coax): Asymmetric – up to 40 Mbps – 1.2 Gbps downstream transmission rate, 30-100 Mbps upstream rate

- Network of cable, and fibre attaches homes to IPS router

- homes share access network to the cable headend

**Access Networks: Digital Subscriber Line (DSL):**

- Use existing telephone line to central office DSLAM

- Data over DSL phone line goes to the internet

- Voice over DSL phone line goes to telephone net

- 24-52 Mbps dedicated downstream transmission rate

- 3.5-16 Mbps dedicated upstream transmission rate

**Access Networks: Home Networks:**

**Wireless Access Networks:**

- Shared wireless access network connects end system to routers (via access points)

Wireless Local Area Networks (WLANs):

* Typically, within or around building (~100ft)
* 802.11b/g/n (WiFi): 11,54,450 Mbps transmission rate

Wide Area Cellular Access Networks:

* Provided by mobile, cellular network operators (10 km)
* 10 Mbps
* 4G cellular networks (5G coming soon)

**Access Networks: Enterprise Networks:**

- Companies, universities, etc.

- Mix of wired, wireless link technologies, connecting a mix of switches and routers

- Ethernet: Wired access at 100Mbps, 1Gbps, 10Gbps

- WiFi: Wireless access point at 11, 54, 450 Mbps

**Hosts: Sends packets of data:**

* Takes application message
* Breaks into smaller chunks known as **packets,** of length L bits
* Transmits packet into access network at transmission rate R
  + Link transmission rate (aka link capacity (aka link bandwidth))

A black text on a white background

AI-generated content may be incorrect.

**Links: Physical media:**

* Bit: Propagates between transmitters/receivers’ pairs
* Physical link: What lies between transmitters and receiver
* Guided media: Signals propagate in solid media (copper, fibre, coax)
* Unguided media: Signals propagate freely (e.g., radio)
* Twisted Pairs (TP): Two insulated copper wires
  + Category 5: 100Mbps, 1Gbps ethernet
  + Category 6: 10Gbps ethernet
* Coaxial Cable: Two concentric copper conductors
  + Bidirectional
  + Broadband
    - Multiple frequency channels on cable
    - 100Mbps per channel
* Fiber Optic cable: Glass fibre carrying light pulses, each pulse a bit
  + High-speed operation:
    - High-speed point-to-point transmission (10-100 Gbps)
  + Low error rate:
    - Repeaters spaced far apart
    - Immune to electromagnetic noise

**Links: Physical media:**

* Wireless radio:
  + Signals carried in the electromagnetic spectrum
  + No physical “wire”
  + Broadcast and “half duplex” (send to receiver)
  + Propagation environment effects:
    - Reflection
    - Obstruction by objects
    - Interference
* Radio Link Types
  + Terrestrial microwave
    - Up to 45 Mbps channels
  + Wireless LAN (WiFi)
    - Up to100 Mbps
  + Wide Area (e.g., cellular)
    - 4G cellular: ~10Mbps
  + Satellite
    - Up to 45 Mbps per channel
    - 270 msec end-end delay
    - Geosynchronous versus low Earth orbit

**Network core: Packet/Circuit switching, internet structure:**

**Network Core:**

- Mesh of interconnected routers

- **Packet Switching:** Host breaks application layer messages into packets

- Forwards packets from one router to the next, across links on the path from source to destination

- Each packet is transmitted at full link capacity

**Packet Switching: Store and Forward:**

**- Transmission Delay:** Takes L/R seconds to transmit (push-out) L-bit packet into the link at R bps

- **Store and Forward:** The entire packet must arrive at the router before it can be transmitted to the next link

- **End-end delay:** 2L/R (above), assuming zero propagation delay

**Packet Switching: Queueing delay, loss:**

Packet queuing and loss: If the arrival rate (in bps) to the link exceeds the transmission rate (bps) of the link for a period of time:

- Packet will queue, waiting to be transmitted on the output link

- Packets can be dropped (lost) if memory (buffer) In router fills up

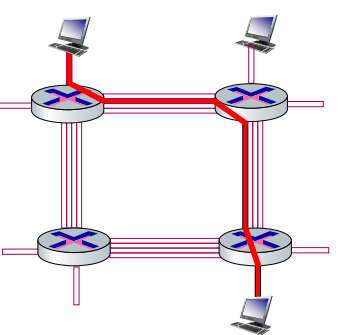
**2 Key Network Core Functions:**- **Forwarding:** Local action – Move arriving packets from the router’s input link to the appropriate router output link

**Routing:** Global Action – Determine source-destination paths taken by packets

* Routing algorithms

**Circuit Switching:**

- End-to-end resources allocated to and reserved for “call” between source and destination

- (In diagram) Each link has 4 circuits. (Call gets 2nd circuit in top link and 1st circuit in right link)

- Dedicated resources: no sharing (circuit-like performance)

- Circuit segment idle if not used by call (no sharing)

- Commonly used in traditional telephone networks

**Circuit Switching: FDM and TDM:**

**Frequency Division Multiplexing (FDM):**

- Optical, electromagnetic frequencies divided into (narrow) frequency bands

- Each call allocated its band, and can transmit at a max rate of that narrow band

**Time Division Multiplexing (TDM):**

- Time divided into slots

- Each call allocated periodic slot(s) can transmit at a maximum rate of (wider) frequency band, but only during its time slot(s)

A colorful lines with text

AI-generated content may be incorrect.

**Packet Switching vs. Circuit Switching:**

|  |  |  |
| --- | --- | --- |
| Feature | Packet Switching (Used in Internet) | Circuit Switching (Used in Phones) |
| Efficiency | High (shares resources) | Low (reserved fixed resources) |
| Scalability | High | Limited |
| Best for | Internet, Data communications | Voice calls |

Packet Switching:

- Great for “Bursty” data - sometimes had data to send, but at other times not:

- Resource sharing

- Simpler, no call setup

- Excessive congestion possible – packet delay and loss due to buffer overflow

- Protocols needed for reliable data transfer, congestion control

**Internet Structure: A “network of networks”:**

- Hosts connect to the internet via access Internet Service Providers (ISPs)

- Residential, enterprise (company, university, commercial) ISPs

- Access ISPs in turn must be interconnected

- so that any two hosts can send packets to each other

- The resulting network of networks is very complex

- Evolution was driven by economics and national policies

Connecting each access ISP directly doesn’t scale: O(N2) connections

**Internet Structure:**

- Hierarchy of ISP (Internet Service Provider)

- Tier 1 ISPs: Backbone of the internet (e.g., Verizon, AT&T, Sprint, Cable and Wireless)

- Tier 2 ISPs: Connect to Tier 1ISPs possible to even tier-2 ISPs, may peer privately and smaller (often regional)

- Tier 3 ISPs: Local ISPs serving home and businesses

- Internet Exchange Points (IXPs): Facilitates efficient traffic routing

**Scaling**

With N ISPs, each ISP needs to connect to (N – 1) other ISPs

- Total Connections = (N \* (N-1)) / 2

This is because:

- Each ISP must establish a direct link with every other ISP

- As N (the number of ISPs) increases, the number of required connections grows quadratically

**Why Doesn’t it scale:**

A white background with black text

AI-generated content may be incorrect.

**Scaling Challenges:**

**- Infrastructure Costs:** Establishing physical or virtual links between each pair of ISPs involves significant infrastructure, energy and maintenance costs, which increase quadratically

- **Management Complexity:** Each ISP would need routing and coordination for N-1 connections, leading to complex configurations and operational challenges

To address the scalability issue, the internet uses a hierarchical structure:

1. **Core ISPs (Tier 1 ISPs):** A few high-capacity, large-scale ISPs form the backbone of the internet and interconnect directly
2. **Peering Agreements:** ISPs use peering at Internet Exchange Points (IXPs) to share traffic with multiple ISPs simultaneously, reducing the need for direct links.
3. **Transits ISPs:** Smaller ISPs use larger ISPs to route traffic indirectly to other ISPs

**-** This hierarchical and indirect routing system reduces the number of direct connections and scales efficiently as the internet grows

**Asynchronous Operation:**

A screenshot of a white background

AI-generated content may be incorrect.- Fundamental Constraint: Speed of light

Thus, communication feedback is always dated

**Prone to failure:**

- To send a message, all components along a path must function correctly

- Software, wireless access points, firewall, links, network interface cards, switches…

- Including human operations



Plus:

- Scale: Lots of components

- Asynchrony: It takes a long time to hear (bad) news

- Federation (internet): Hard to identify fault or assign blame

**Cumulative Reliability:**

1. **System Overview:**
   1. The system has 50 components
   2. Each component works correctly 99% of the time (or has a failure rate of 1-0.99 = 0.01)
2. **System Reliability:**
   1. For the system to work, all 50 components must function correctly simultaneously
   2. The probability of a single component working correctly is P(correct) = 0.99
   3. Assuming the components are independent (the performance of one does not affect others), the probability that all 50 components work is: P(system works) = (P(correct))5 = 0.99
3. **Calculations:** 
   1. 0.995 is approximately 0.395 or 39.5%
   2. This means there is only a 39.5% chance that the system works perfectly
4. **Chance of Failure:**
   1. The probability the system fails is:

P(system fails = 1-P(system works) = 1-0.395 = 0.605

So, there’s a 60.5% chance the system will fail due to at least one component failing

**Interpretation:**

Even though each component is highly reliable (99%), the reliability of the entire system drops significantly because the failure of any single component leads to a system failure

**Week 2:**

A diagram of different types of internet

AI-generated content may be incorrect.**Network Layering:**

- Means breaking up the sending of messages into separate components and activities

- Each component handles a different part of the communication

- Referred as the Transmission Control Protocol/Internet Protocol (TCP/IP) model

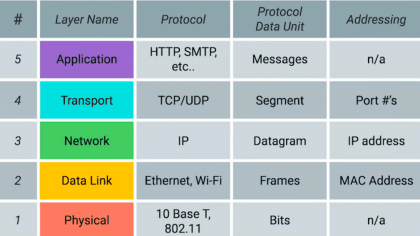
**TCP/IP Model:**

**- Application Layer:** Encodes/decodes the message in a form that is understood by the sender and the recipient

**- Transport Layer:** Breaks down the message into small chucks (packets). Each packet is given a packet number and the total number of packets. The recipient uses this information to assemble the packets together in the correct order. It also allows the recipient to see if there are any missing packets

**- Network Layer:** Adds the sender’s and that of the recipient. The network then knows where to send the message, and where it came from

**- Link Layer:** Enables the transfer of packets between nodes on a network, and between one network and another

**Why Layering?**

- Layering allows standards to be developed, but also the be adapted to new and overtime

- E.g., Different software packages (applications) may use the same transport, network and link layers but have their own application layer

- The way the program encodes the message changes – the rest of the communication methods remains the same

**Application Layer:**

- Principle of the network application:

- Web and HTTP

- Email, SMTP, IMAP

- The domain name system (DNS)

- P2P applications

- Video streaming and content distribution networks

- Socket programming with UDP and TCP

**Creating a network app:**

- Write a program that:

- Runs on (different) end systems

- Communicates over the network

- e.g., web server software communicates with the browser software

- No need to write software for network-core devices:

- Network-core devices don’t run user applications

- Application on-end systems allow for rapid app development. Propagation

**Client-Server paradigm:**

**Server:**

* Always-on host
* Permanent IP address
* Often in data centres, for scaling

**Clients:**

* Contact, communicate with server
* Maybe intermittently connected
* May have dynamic IP addresses
* Does not communicate directly with each other
  + E.g., HTTP, IMAP, FTP

**Peer-to-peer architecture:**

* No always-on server
* Arbitrary end system directly communicates
* Peers request service from other peers, provide service in return to other peers
  + Self-scalability: new peers bring new service capacity, as well as new service demands
* Peers are intermittently connected and change IP addresses
  + Complex management
  + E.g., P2P file sharing

**Processes communicating:**

**Process:** Program running with a host

* Within the same hosts, two processes communicate using inter-process communication
* Processes in different hosts communicate by exchanging messages

- Client Process: Process that initiates communication

- Server Process: Process that waits to be contacted

**Sockets:**

- Process sends/receives messages to/from its socket

- Socket analogous to door

- Sending process shoves messages outdoor

- Sending process relies on transport infrastructure on the other side door to deliver the message to the socket at the receiving process

- Two sockets involved: one on each side

**Addressing Process:**

**-** To receive messages, processes must have an **identifier**

- Host devices have a unique 32-bit IP address

- **Identifier** includes both IP address and port numbers associated with the process on the host

- to send HTTP message to the web server:

- IP address: 128.119.245.12

- Port number: 80

**Application Layer Protocol Defines:**

**-** Types of messages exchanged: e.g., request, response

- Message Syntax: What fields in messages & how fields are delineated

- Message Semantics: Messages of information in fields

- Rules: For when and how processes send & respond to messages

- Open Protocols:

- Define in RFCs, everyone has access to the protocol definition

- Allow for interoperability

- e.g., HTTP, SMTP

- Proprietary Protocols: e.g., Skype

**Internet Transport Protocol:**

**What Transport service does an app need?**

- **Data Integrity:** Some apps (e.g. file transfer) require 100% reliable data transfer, whereas other apps (e.g. audio) can tolerate some loss

- **Timing:** Some apps (e.g. internet telephony, interactive games) require low delay to be effective

- **Throughput:** Some apps (e.g. multimedia) require a minimum amount of throughput to be effective, whereas other apps (elastic apps) make use of whatever throughput they get

A screenshot of a computer

AI-generated content may be incorrect.- **Security:** Encryption, data integrity…

**TCP Services:**

- Reliable Transport: Between sending and receiving process

- Flow Control: Sender won’t overwhelm the receiver

- Congestion Control: Throttle sender when network overloaded

- Does not Provide: Timing, minimum throughput guarantee, security

- Connection Oriented: Setup required between client and server processes

**UDP Services:**

**-** Unreliable Data Transfer: Between sending and receiving process

- Does not Provide: Reliability, flow control, congestion control, timing throughput, guarantee, security or connection setup

A close-up of a computer code

AI-generated content may be incorrect.

**Web & HTTP:**

- Webs pages consist of objects, each can be stored on different Web servers

- Objects can be HTML files, JPEG images, Java applets, audio files

- Web pages consist of base HTML files which include several referenced objects, each addressable by URL

**HTTP Overview:**

- Hypertext Transfer Protocol

- Web application layer protocol

- Client/server modal:

- Client: Browser that requests, receives (using HTTP protocol) and displays Web objects

- Server: Web server sends (using HTTP protocol) objects in response to requests

**HTTP uses TCP:**

- Clients initiate TCP connection (creates sockets) to the server (port 80)

- Server accepts TCP connection from client

HTTP messages (application layer protocol messages) are exchanged between the browser (HTTP client) and Web server (HTTP server)

- TCP connection closed

**HTTP is stateless:**

- Server maintains no information about past client requests

**Two Types:**

**Non-persistent HTTP:**

1. TCP connection opened
2. At most one object sent over a TCP connection
3. TCP connection closed

**Persistent HTTP:**

1. TCP connection opened to a server
2. Multiple objects can be sent over a single TCP connection between the client, and the server
3. TCP connection closed

**Week 3:**

**Transport Layer Overview**

The Transport Layer provides logical communication between application processes running on different hosts. It ensures that messages are delivered correctly and efficiently.

**Key Services Provided:**

* **Multiplexing/Demultiplexing** – Handling multiple communication flows between applications.
* **Reliable Data Transfer** – Ensuring correct, ordered delivery (TCP).
* **Flow Control** – Preventing overwhelming the receiver.
* **Congestion Control** – Managing traffic to avoid network overload.

**Transport Layer Protocols**

There are **two main Internet transport protocols:**

|  |  |  |
| --- | --- | --- |
| **Protocol** | **Features** | **Use Cases** |
| TCP (Transmission Control Protocol) | Reliable, in-order delivery, error-checking, flow & congestion control, connection setup required. | Web browsing, file transfer, email. |
| UDP (User Datagram Protocol) | Unreliable, connectionless, no error correction, faster but less secure. | Streaming, online gaming, VoIP. |

**Key Differences Between TCP and UDP:**

* TCP is connection-oriented, while UDP is connectionless.
* TCP ensures reliability by retransmitting lost packets, UDP does not.
* TCP provides congestion control, UDP does not.

**Multiplexing and Demultiplexing**

**Multiplexing:**

* Multiple application processes share the same transport layer.
* Each data packet is assigned a port number to ensure it reaches the correct application.

**Demultiplexing:**

* The receiver identifies the correct destination **socket** using port numbers.
* Uses source and destination IP addresses and port numbers.

**UDP (User Datagram Protocol)**

**Features:**

* Simple, connectionless, and fast.
* No error-checking or retransmission, meaning packets can be lost or received out of order.
* Small header size, reducing overhead.
* Used for real-time applications where speed is more important than reliability.

**UDP Header Format:**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Source Port | Identifies the sender’s application. |
| Destination Port | Identifies the receiver’s application. |
| Length | Size of the UDP segment. |
| Checksum | Used for error detection. |

**Use Cases for UDP:**

* Streaming multimedia (video/audio).
* DNS (Domain Name System).
* Online gaming.
* VoIP (Voice over IP).

**Reliable Data Transfer (RDT) Protocols**

**Challenges in Reliable Data Transfer:**

* **Bit Errors** – Data may be altered during transmission.
* **Packet Loss** – Some packets may not reach the receiver.
* **Packet Reordering** – Packets may arrive in the wrong order.

**Reliable Data Transfer Protocols:**

|  |  |
| --- | --- |
| **Protocol** | **Key Features** |
| RDT 1.0 | Works over a perfectly reliable channel. No error detection or correction needed. |
| RDT 2.0 | Introduces error detection (checksums) and negative acknowledgments (NAK) for corrupted packets. |
| RDT 2.1 | Handles corrupted acknowledgments (ACKs) by adding sequence numbers. |
| RDT 2.2 | Removes NAK, relying only on ACKs with sequence numbers. |
| RDT 3.0 | Adds timeouts and retransmissions to handle packet loss. |

**Stop-and-Wait Protocol (RDT 3.0):**

* The sender sends one packet at a time and waits for an acknowledgment (ACK) before sending the next.
* If no ACK is received within a set time, the sender retransmits the packet.
* Disadvantage: Low efficiency because the sender is idle while waiting for ACKs.

**Pipelined Transport Protocols:**

To improve efficiency, pipelining allows multiple packets to be sent before waiting for ACKs. Two main approaches:

|  |  |
| --- | --- |
| **Protocol** | **Key Features** |
| Go-Back-N (GBN) | Sender can send up to N packets without waiting. If an error occurs, all packets from the error onwards are retransmitted. |
| Selective Repeat (SR) | The receiver stores out-of-order packets and only requests retransmission for lost/corrupt packets, improving efficiency. |

**Go-Back-N vs. Selective Repeat:**

* **Go-Back-N** retransmits multiple packets after an error, while **Selective Repeat** retransmits only the specific lost packet.
* **Selective Repeat** requires more buffer space at the receiver but is more efficient.

**Transport Layer Summary:**

* The transport layer provides end-to-end communication between applications.
* TCP is reliable, connection-oriented, and ensures ordered delivery.
* UDP is fast, connectionless, and allows data to be lost or unordered.
* Multiplexing/demultiplexing ensures packets are sent to the correct application.
* Reliable data transfer protocols (RDT) ensure error-free communication using checksums, ACKs, sequence numbers, and retransmissions.
* Pipelined protocols like Go-Back-N and Selective Repeat improve efficiency compared to Stop-and-Wait.

**Week 4:**

**rdt3.0 in Action**

* Handles packet loss, acknowledgement loss, and retransmissions.
* Uses timeouts to retransmit lost packets.

**Performance of rdt3.0 (Stop-and-Wait Protocol)**

* Inefficient for high-speed networks due to long waiting times.
* Sender must wait for an acknowledgment before sending the next packet.

**Pipelining (Improved Efficiency over Stop-and-Wait)**

* Multiple packets can be in transit at once, increasing efficiency.
* Requires buffering and sequencing at both sender and receiver.

**Go-Back-N Protocol**

* Sender maintains a window of up to N unacknowledged packets.
* If a packet is lost, all subsequent packets are retransmitted.

**Selective Repeat Protocol**

* Receiver buffers out-of-order packets and only requests retransmission for missing packets.
* More efficient than Go-Back-N but requires more complex buffering and tracking.

**Transmission Control Protocol (TCP)**

**TCP Overview**

* Reliable, connection-oriented transport protocol.
* Uses cumulative acknowledgements and congestion/flow control.
* Bi-directional communication with full-duplex support.

**TCP Segment Structure**

* Contains sequence numbers, acknowledgement numbers, and flow control fields.
* Sequence numbers ensure ordered delivery of data.

**TCP Round-Trip Time (RTT) & Timeout**

* Uses EstimatedRTT and Timeout Interval for efficient retransmission.
* Exponential Weighted Moving Average (EWMA) smooths RTT variations.

**TCP Flow Control**

* Prevents sender from overwhelming the receiver.
* Receiver advertises available buffer space (rwnd - receive window).
* Sender limits unacknowledged data to avoid buffer overflow.

**TCP Connection Management**

* Three-way handshake to establish a connection (SYN, SYN-ACK, ACK).
* Graceful termination using FIN-ACK exchange.

**Congestion Control in TCP**

**Congestion Overview**

* Congestion occurs when too much traffic is sent too quickly, leading to packet loss and delays.
* Different from flow control, which prevents overwhelming a single receiver.

**Causes of Congestion**

1. Router queue overflow → Packet loss.
2. High traffic load → Increased delays.
3. Retransmissions due to loss → Wasted bandwidth.

**Congestion Control Approaches**

* End-to-end control (TCP): Infers congestion from packet loss or delay.
* Network-assisted control: Routers provide feedback to senders (e.g., Explicit Congestion Notification (ECN)).

**TCP Congestion Control Strategies**

* Additive Increase, Multiplicative Decrease (AIMD):
  + Increase window size until loss occurs.
  + Reduce window size by half when congestion is detected.
* **Slow Start**:
  + Exponentially increases sending rate at the start of a connection.
  + Switches to AIMD once congestion is detected.
* **Fast Retransmit & Fast Recovery**:
  + If three duplicate ACKs are received, resend the lost packet immediately.
  + Avoids waiting for a timeout.

**TCP Variants & Evolution**

**TCP CUBIC (Linux Default TCP)**

* Optimized for high-speed networks.
* Probes for available bandwidth more efficiently than AIMD.

**Delay-Based Congestion Control**

* Uses RTT measurements to detect congestion before packet loss occurs.
* Reduces sending rate proactively to avoid excessive delays.

**Explicit Congestion Notification (ECN)**

* Routers mark packets instead of dropping them when congestion is detected.
* Receiver notifies sender, allowing rate adjustment before loss occurs.

**TCP Fairness**

* If multiple TCP flows share a bottleneck link, they should divide bandwidth fairly.
* UDP-based applications (like streaming) do not follow TCP congestion control, leading to unfair bandwidth usage.

**QUIC (Quick UDP Internet Connections)**

* New transport protocol built on UDP, improving performance over TCP.
* Used in HTTP/3 for faster web browsing and reduced latency.
* Features:
  + Multiplexing → No head-of-line blocking.
  + Built-in encryption (TLS-like security).
  + Faster connection establishment than TCP (one handshake instead of two).

**Summary**

* TCP ensures reliable, ordered, and congestion-controlled communication.
* Congestion control strategies like AIMD, slow start, and fast retransmit optimize network usage.
* UDP provides fast, unreliable transport, useful for streaming and real-time applications.
* QUIC improves on TCP’s limitations by reducing latency and enhancing security.

**Week 5:**

**Internet Layer Overview**

* The Internet Layer is responsible for routing data packets across networks using IP addresses.
* It sits between the Transport Layer and the Network Access Layer in the TCP/IP model.
* Ensures logical, flexible communication across different networks.

**Internet Layer Core Concepts**

* **Routing**: Determines the optimal path for packets using routing tables and protocols (e.g., OSPF, BGP).
* **Addressing**: Assigns IP addresses (IPv4/IPv6) for device identification.
* **Packetization**: Encapsulates transport layer segments into IP packets (datagrams).
* **Fragmentation & Reassembly**: Handles large packets that exceed network limits.

**IP Addressing and Hierarchical Structure**

* IPv4 addresses are 32-bit, divided into four octets (e.g., 192.168.1.1).
* IPv6 addresses are 128-bit, providing a larger address space.
* IP address classes:
  + **Class A**: Large networks (default mask: /8).
  + **Class B**: Medium networks (default mask: /16).
  + **Class C**: Small networks (default mask: /24).
  + **Class D**: Multicast.
  + **Class E**: Experimental.
* **Private IP Addresses**: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 (used for local networks with NAT).
* **CIDR (Classless Interdomain Routing)**: Uses subnet masks of arbitrary length (e.g., 200.23.16.0/23).

**DHCP and Address Management**

* DHCP dynamically assigns IP addresses to hosts when they connect to a network.
* DHCP process:
  1. **DHCP Discover** – Client broadcasts a request for an IP address.
  2. **DHCP Offer** – Server responds with an available IP address.
  3. **DHCP Request** – Client requests to use the offered address.
  4. **DHCP Acknowledgment** – Server confirms and assigns the address.
* DHCP can also provide the default gateway, DNS server, and subnet mask.

**NAT and Subnets**

* **Network Address Translation (NAT)** allows multiple devices in a local network to share a single public IP address.
* **Advantages of NAT**:
  + Reduces the need for public IPv4 addresses.
  + Provides security by hiding internal IP addresses.
  + Allows internal address changes without affecting external communication.
* **Subnets**: A subnet is a group of devices that can communicate without going through a router.
* **Subnet Mask**: Determines the network and host portions of an IP address (e.g., /24 for 255.255.255.0).

**Routing and Forwarding**

* **Routing**: Determines the best path for packets to travel from source to destination.
* **Forwarding**: Moves packets from one router interface to another based on routing decisions.
* **Static Routing**: Manually configured paths, predictable but not scalable.
* **Dynamic Routing**: Routes adjust automatically based on network changes, using protocols like RIP, OSPF, and BGP.

**Routing Protocols**

* **RIP (Routing Information Protocol)**:
  + Distance-vector protocol.
  + Uses hop count as the metric (max 15 hops).
  + Simple, but not suitable for large networks.
* **OSPF (Open Shortest Path First)**:
  + Link-state protocol using Dijkstra's algorithm.
  + Fast convergence and supports large networks.
  + Uses hierarchical area-based routing.
* **BGP (Border Gateway Protocol)**:
  + Used for routing between autonomous systems on the Internet.
  + Path-vector protocol with policy-based routing.

**Advanced Routing Concepts**

* **Distance Vector Routing**: Routers share hop count information with neighbours.
* **Dijkstra’s Algorithm**: Used in link-state routing to find the shortest path in a network.

**IPv6 and Network Evolution**

* IPv6 was introduced to address IPv4 exhaustion and includes 128-bit addressing.
* **Key differences between IPv4 and IPv6**:
  + No checksum (faster processing).
  + No fragmentation (handled by endpoints).
  + Built-in support for mobility and security.
* **IPv6 Tunnelling**: Used to send IPv6 packets over IPv4 networks.
* **IPv6 Deployment**: Adoption has been slow, but major companies (e.g., Google) report increasing usage.

**Week 6:**

**Link Layer Overview**

* Responsible for transferring data between physically connected nodes.
* Uses frames to encapsulate network-layer datagrams.
* Provides error detection, flow control, and addressing mechanisms.

**Link Layer Services**

* **Framing and Link Access**: Encapsulates network-layer data in frames.
* **Reliable Delivery**: Ensures data integrity (not always used).
* **Flow Control**: Manages data pacing between sender and receiver.
* **Error Detection & Correction**: Detects and corrects transmission errors.
* **Half-Duplex vs. Full-Duplex**: Defines whether devices can transmit simultaneously.

**Where is the Link Layer Implemented?**

* Implemented in network interface cards (NICs) or chips.
* Operates through a combination of hardware, software, and firmware.

**Error Detection & Correction**

* **Parity Checking**: Uses a single-bit or two-dimensional parity to detect errors.
* **Checksum (UDP, TCP, IP)**: Computes a sum of data bits for verification.
* **Cyclic Redundancy Check (CRC)**: More powerful error detection method using a generator polynomial.

**Multiple Access Protocols**

* **Used in shared broadcast channels** to avoid collisions.
* **Three main types**:
  + **Channel Partitioning** (TDMA, FDMA) – Divides the channel into time/frequency slots.
  + **Random Access** (ALOHA, CSMA/CD) – Allows transmission whenever data is ready, with collision handling.
  + **Taking Turns** (Polling, Token Passing) – Nodes take turns transmitting data.

**Ethernet & MAC Addressing**

* **MAC Address (Media Access Control)**: A 48-bit unique identifier assigned to NICs.
* Used for local addressing within a subnet (unlike IP addresses).
* **Address Resolution Protocol (ARP)**: Translates IP addresses to MAC addresses for communication.

**Switching & LAN Technologies**

* Ethernet is the dominant wired LAN technology, supporting speeds from 10 Mbps to 400 Gbps.
* Switches store and forward Ethernet frames, learning MAC addresses dynamically.
* **Switches vs. Routers**:
  + Switches operate at the link layer (layer 2) and forward based on MAC addresses.
  + Routers operate at the network layer (layer 3) and forward based on IP addresses.

**Synthesis: A Day in the Life of a Web Request**

1. Device connects to the network via DHCP – Obtains an IP address, default gateway, and DNS server.
2. DNS query sent to resolve a domain name to an IP address.
3. TCP connection established with the web server using a three-way handshake.
4. HTTP request is sent, and the web page is returned.

**Week 7:**

**Operating System Overview**

* An operating system (OS) controls the execution of application programs and acts as an interface between hardware and software.
* Manages CPU, memory, storage, and I/O devices.
* Provides convenience, efficiency, and evolution for system users.

**Basic Elements of a Computer System**

* **Processor (CPU)**: Controls computer operations and processes data.
* **Main Memory**: Stores programs and data; volatile and fast.
* **I/O Modules**: Handle input/output operations between the system and external devices.
* **System Bus**: Connects the processor, memory, and I/O modules.

**Instruction Execution & Interrupts**

* A program consists of instructions stored in memory.
* Instruction cycle: The processor fetches an instruction, then executes it.
* Interrupts allow the CPU to handle external events while running programs.
  + Types of Interrupts: Program (e.g., division by zero), Timer, I/O, and Hardware failures.

**Memory Hierarchy**

* **Primary Memory (RAM)**: Fast but volatile.
* **Cache Memory**: Small, fast memory that stores frequently accessed data.
* **Secondary Memory (HDD/SSD)**: Non-volatile storage for long-term data.
* **Memory Management Techniques**: Paging and Virtual Memory.

**Cache Memory**

* Increases processing speed by storing frequently accessed data.
* **Cache levels**: L1 (fastest, smallest), L2, and L3.
* Block Size & Mapping Functions affect cache efficiency.

**Input/Output (I/O) Techniques**

* **Programmed I/O**: Processor waits for I/O completion, slowing performance.
* **Interrupt-Driven I/O**: Processor continues tasks until I/O signals completion.
* **Direct Memory Access (DMA)**: Transfers data without CPU involvement, improving efficiency.

**Symmetric Multiprocessing (SMP) & Multicore Systems**

* **SMP**: Multiple processors share memory and perform tasks simultaneously.
* **Multicore Processors**: Multiple processing cores on a single chip for better performance and efficiency.

**Operating System Services**

* **Process Management**: Manages running applications and execution order.
* **Memory Management**: Allocates and protects memory.
* **File System Management**: Controls access to files.
* **Security & Protection**: Prevents unauthorized access.

**Process Management**

* A process is a running program with its own memory and execution context.
* OS schedules processes, ensuring fair resource allocation and avoiding conflicts.
* Causes of Errors in Processes: Deadlocks, race conditions, improper synchronization.

**Virtual Memory & Paging**

* **Virtual Memory**: Allows programs to use more memory than physically available.
* **Paging**: Divides memory into fixed-size pages, reducing fragmentation.

**Security & Fault Tolerance**

* **Information Security**: Protects data integrity, confidentiality, and system availability.
* **Fault Tolerance**: Ensures system operation despite hardware/software failures.
* **Redundancy Methods**:
  + **Spatial (physical)**: Extra components for backup.
  + **Temporal**: Repeating operations when errors occur.
  + **Information**: Using error-correcting codes.

**Week 8:**

**Introduction to Operating Systems**

* An OS is software that manages hardware and software resources.
* Functions include process management, memory management, file system management, device management, and user interface.

**Windows Operating System:**

**Introduction to Windows**

* Developed by Microsoft, first released in 1985.
* The most widely used desktop OS.
* Focuses on user-friendliness and broad hardware compatibility.

**Evolution of Windows Versions**

* Key versions: Windows 95, XP, 7, 8, 10, 11.
* Introduced features like Start Menu, Aero UI, Metro UI, and Fluent Design.

**Windows Architecture**

* **Kernel Mode**: Direct hardware access for critical operations.
* **User Mode**: Restricted access for applications and user processes.
* **Win32 API**: Interface for software development.

**Windows File System**

* **NTFS**: Default, supports security features.
* **FAT32**: Older, compatible but lacks security.
* **exFAT**: Optimized for flash drives.

**Windows Security Features**

* **Windows Defender**: Built-in antivirus.
* **User Account Control (UAC)**: Prevents unauthorised changes.
* **Windows Firewall**: Protects network communication.

**Pros & Cons of Windows**  
Pros: Wide compatibility, user-friendly, large software support.  
Cons: Vulnerable to malware, resource-intensive, licensing costs.

**macOS:**

**Introduction to macOS**

* Developed by Apple, Unix-based OS.
* Known for stability, security, and integration with Apple hardware.

**macOS User Interface**

* **Finder**: File management.
* **Dock**: Quick access to applications.
* **Spotlight**: System-wide search.
* **Menu Bar**: Application-specific menus.

**macOS File System**

* **HFS+**: Older file system.
* **APFS**: Modern file system, optimised for SSDs.

**macOS Security Features**

* **Gatekeeper**: Prevents unverified app installations.
* **FileVault**: Full-disk encryption.
* **T2/M series security chip**: Secure boot, encrypted storage.

**Apple Ecosystem**

* **iCloud**: Cloud storage and device synchronisation.
* **Continuity**: Handoff, Universal Clipboard, AirDrop.

**Pros & Cons of macOS**  
Pros: User-friendly, secure, seamless Apple ecosystem.  
Cons: Limited hardware compatibility, expensive, fewer gaming options.

**Unix:**

**Introduction to Unix**

* Developed in AT&T Bell Labs (1970s).
* A multi-user, multi-tasking OS.
* Influenced modern OSs like Linux and macOS.

**Unix File System**

* **Hierarchical structure** with root (/) and subdirectories.
* **File permissions**: Read, write, execute (rwx).
* **Ownership**: User and group-based access control.

**Unix Security Features**

* **Root privileges**: Superuser access for system administration.
* **User authentication**: Passwords, SSH keys.
* **Principle of least privilege**: Restricting user access.

**Unix Variants**

* **BSD** (FreeBSD, OpenBSD, NetBSD).
* **Solaris**: Enterprise-focused Unix OS.
* **AIX**: Developed by IBM.

**Pros & Cons of Unix**  
Pros: Stable, secure, powerful command-line tools, portable.  
Cons: Steep learning curve, limited user-friendly GUI.

**Linux:**

**Introduction to Linux**

* Open-source, Unix-like OS developed by Linus Torvalds (1991).
* Known for flexibility, security, and customizability.
* Popular distributions (distros): Ubuntu, Fedora, Debian, Arch Linux.

**Linux File System**

* **ext4**: Default for many distros, reliable.
* **XFS**: High-performance, good for large files.
* **Btrfs**: Advanced features like snapshots and error detection.
* **ZFS**: High storage capacities, robust data integrity.

**Linux Security Features**

* **SELinux**: Mandatory access control.
* **iptables/firewall**: Firewall management.
* **sudo privileges**: Controlled root access.

**Linux for Servers**

* Used for web servers, database servers, and cloud computing.
* Examples: Apache, Nginx, MySQL, PostgreSQL.

**Real-World Uses of Linux**

* **Embedded systems** (routers, smart devices).
* **IoT** (smart home devices, automation).
* **Supercomputing** (high-performance computing clusters).

**Pros & Cons of Linux**  
Pros: Secure, stable, open-source, highly customisable.  
Cons: Learning curve, occasional hardware compatibility issues.

**Android:**

**Introduction to Android**

* Linux-based mobile OS developed by Google.
* Most widely used mobile OS.
* Supports touchscreen devices, tablets, and smart TVs.

**Android Architecture**

* **Linux Kernel**: Handles hardware interactions.
* **Hardware Abstraction Layer (HAL)**: Connects software to hardware.
* **Android Runtime (ART)**: Executes applications.

**Android App Development**

* Uses Java and Kotlin as programming languages.
* Android Studio: Official development environment.

**Android Security Features**

* **Google Play Protect**: Malware scanning.
* **App sandboxing**: Isolates apps for security.
* **Encryption**: Protects stored data.

**Pros & Cons of Android**  
Pros: Customizable, wide range of devices, open source.  
Cons: Device fragmentation, security vulnerabilities, manufacturer bloatware.

**Debate on Windows vs macOS:**

**Key Discussion Points**

* **Cost**: Windows offers a range of prices; macOS is premium-priced.
* **Software Compatibility**: Windows supports more applications.
* **Security**: macOS is generally more secure due to Unix-based architecture.
* **User Experience**: macOS offers a streamlined interface; Windows provides customisation.
* **Support**: Windows has broader hardware support; macOS is optimised for Apple devices.