## Chapter -1

# What's the Internet: "nuts and bolts" view-

Hosts are also sometimes referred to as end systems. These are the computing devices that are connected to the Internet, such as laptops, desktops, smartphones, tablets, servers, and other devices.

End systems run a variety of network applications, such as web browsers, email clients, file sharing software, and online gaming applications. These applications rely on the underlying network protocols and technologies to send and receive data packets over the Internet.

In addition to end systems, the Internet also includes a variety of intermediate devices, such as routers, switches, and hubs. These devices help to route data packets between different end systems, allowing for communication and data sharing across the global network.
—
Communication links are the physical connections that enable data to be transmitted between computing devices over the Internet. These links can take a variety of forms, including:

Fiber: Optical fibers are thin strands of glass or plastic that transmit data using light. Fiber optic cables are capable of transmitting data at very high speeds and are used for long-distance connections.

Copper: Copper cables are used for local connections and are commonly found in homes and offices. They are used for connecting devices to routers or switches, and for providing internet connectivity through DSL or cable modems.

Radio: Radio waves are used for wireless connections, such as Wi-Fi or cellular networks. These networks use radio waves to transmit data between devices, allowing for mobile connectivity and remote access.

Satellite: Satellite links are used for connecting remote locations, such as ships at sea or rural areas where other forms of connectivity are not available. Data is transmitted from the ground to a satellite in orbit, which then relays the data to another ground station.

The transmission rate of a communication link is commonly referred to as its bandwidth. Bandwidth refers to the amount of data that can be transmitted over a communication link in a given amount of time, usually measured in bits per second (bps), kilobits per second (kbps), or megabits per second (Mbps). Higher bandwidth links can transmit more data in less time, enabling faster download and upload speeds, and smoother streaming of audio and video content.

–

Packet switches that are used on the Internet: routers and switches.

Routers: Routers are devices that are used to forward data packets between networks. They examine the destination address of each packet and use this information to determine the best path to forward the packet towards its destination. Routers are used to connect different networks together, such as connecting a home network to the Internet.

Switches: Switches are devices that are used to forward data packets within a network. They examine the destination address of each packet and use this information to determine which device on the network the packet should be forwarded to. Switches are used to connect multiple devices within a local network, such as a home or office network.

## What's the Internet: a service view

The Internet is often described as a "network of networks" because it consists of many interconnected Internet Service Providers (ISPs) that exchange data with each other to provide global connectivity.

The communication between devices on the Internet is governed by a set of protocols that control the sending and receiving of messages. These protocols include the Transmission Control Protocol (TCP) and the Internet Protocol (IP), which are used to transmit data reliably over the Internet. Other protocols, such as the Hypertext Transfer Protocol (HTTP), are used to transfer web pages and other online content. Additionally, there are protocols for real-time communication, such as Skype, and for wireless communication, such as 802.11 (Wi-Fi).

**The standards for the Internet are developed and maintained by organizations such as the Internet Engineering Task Force (IETF)** and the World Wide Web Consortium (W3C). These organizations create and publish technical specifications and standards that ensure compatibility and interoperability between different devices and networks.

The IETF standard documents are called RFC.

—--

The infrastructure of the Internet provides a wide range of services to applications, including web services, Voice over IP (VoIP), email, online gaming, e-commerce, social networking, and many others.

**To enable these services, the infrastructure provides programming interfaces (APIs) that allow applications to send and receive data over the Internet. These APIs provide hooks that allow application programs to "connect" to the Internet and access the services that it provides.**

For example, web services are accessed through APIs such as the Representational State Transfer (REST) API or the Simple Object Access Protocol (SOAP) API. These APIs allow web applications to send and receive data over the Internet using standard protocols such as HTTP.

Similarly, VoIP services are accessed through APIs such as the Session Initiation Protocol (SIP) API or the Real-time Transport Protocol (RTP) API. These APIs allow voice and video data to be transmitted over the Internet in real time.

By providing these programming interfaces and hooks, the infrastructure of the Internet enables a wide range of applications and services to be developed and accessed by users around the world.

–

## What's a protocol?

Network protocols are the set of rules and guidelines that govern the communication between machines (devices) rather than humans in a computer network like the Internet. All communication activities on the Internet are governed by these protocols.

**Protocols define the format and order of messages (data) sent and received between different network entities and also specify the actions that need to be taken on message transmission and receipt.** The most commonly used protocols on the Internet are the Internet Protocol (IP) and the Transmission Control Protocol (TCP), which are used to transmit data reliably over the network.

Other protocols include the Hypertext Transfer Protocol (HTTP), which is used for transferring web pages and other online content; the Simple Mail Transfer Protocol (SMTP), which is used for sending and receiving email messages; and the File Transfer Protocol (FTP), which is used for transferring files between devices on the network.

Each protocol has a specific purpose and is designed to ensure that data is transmitted efficiently and reliably over the network. By following these protocols, devices on the network can communicate with each other and exchange data seamlessly.

## A closer look at network structure:

**The network structure can be divided into three main parts: the network edge, the access network, and the network core.**

At the network edge, there are hosts, which are the end systems such as clients and servers that are used by individuals or organizations to access the Internet. Servers are often located in data centers, which are large facilities that are specifically designed to host and manage large numbers of servers.

The access network is the part of the network that connects the hosts at the network edge to the rest of the Internet. It consists of physical media such as wired and wireless communication links that allow data to be transmitted between devices.

Finally, the network core is the part of the network that connects the different access networks together. It is made up of interconnected routers that route data packets from one network to another. The network core is often referred to as a "network of networks" because it consists of many interconnected ISPs (Internet Service Providers) that provide global connectivity.

By understanding the structure of the network, we can better understand how data is transmitted between devices on the Internet and how different parts of the network work together to provide global connectivity.

## Access networks and physical media

End systems are typically connected to the edge router through an access network. The type of access network used depends on the specific application, location, and available resources.

Residential access networks are used to connect homes to the Internet. These networks typically use technologies such as Digital Subscriber Line (DSL), cable modem, or fiber optic connections to provide high-speed Internet access. The bandwidth of these networks can vary widely depending on the specific technology used and the service plan selected by the user.

Institutional access networks are used to connect schools, universities, hospitals, and companies to the Internet. These networks are typically more robust than residential access networks and may include dedicated leased lines or higher-capacity fiber optic connections. The bandwidth of these networks can also vary depending on the specific technology used and the level of service provided.

Mobile access networks are used to connect mobile devices such as smartphones, tablets, and laptops to the Internet. These networks use wireless communication technologies such as cellular networks, Wi-Fi, or satellite connections. The bandwidth of these networks can also vary depending on the specific technology used and the location of the device.

It's important to consider the bandwidth of the access network when connecting end systems to the Internet, as this can impact the speed and reliability of the connection. It's also important to consider whether the network is shared or dedicated. Shared networks are typically less expensive but can become congested when multiple users are accessing the network simultaneously, while dedicated networks provide more consistent performance but can be more expensive

## Enterprise access networks (Ethernet)

**Enterprise access networks, also known as local area networks (LANs**), are typically used in companies, universities, and other organizations to provide high-speed connectivity between different devices within the same physical location. Ethernet is the most commonly used LAN technology, and it supports transmission rates of 10 Mbps, 100 Mbps, 1 Gbps, and even 10 Gbps.

In an Ethernet LAN, end systems such as computers, printers, and servers are typically connected to an Ethernet switch. The switch acts as a central hub, allowing the different devices on the network to communicate with each other by sending packets of data back and forth. The switch also helps to regulate the flow of data by only sending packets to the devices that are intended to receive them, which helps to improve the overall performance and efficiency of the network.

Ethernet LANs are often used in conjunction with other types of access networks, such as residential or mobile networks, to provide seamless connectivity between different locations and devices. By using a combination of different access networks, organizations can create a unified network infrastructure that provides high-speed connectivity and reliable data transmission for their employees and customers.

## Wireless access networks

Wireless access networks provide an alternative to wired access networks and are typically used in environments where it is difficult or impractical to run physical cables. In a wireless access network, end systems connect to the network via a base station, also known as an access point.

**Wireless LANs are typically used within a single building or other small area**, with a range of around 100 feet. The most common wireless LAN technology is 802.11b/g, also known as Wi-Fi, which provides transmission rates of 11 or 54 Mbps.

**In contrast, wide-area wireless access networks are provided by telecommunication operators and can cover much larger distances,** typically spanning several kilometers. These networks provide connectivity to mobile devices such as smartphones, tablets, and laptops and typically offer transmission rates between 1 and 10 Mbps. The most common wide-area wireless technologies are 3G and 4G, which are used by cellular networks to provide mobile data services.

Wireless access networks offer a number of advantages over wired networks, including greater flexibility, ease of installation, and scalability. However, they can also be more susceptible to interference and other environmental factors that can impact the quality and reliability of the

connection. As such, it is important to carefully consider the specific requirements of the application when selecting a wireless access network.

## Host: sends *packets* of data

The host sending function is responsible for breaking down application messages into smaller chunks known as packets and transmitting them over the network. This process typically involves the following steps:

**Segmentation**: **The sending host breaks down the application message into smaller chunks called packets.** Each packet typically contains a header and a payload.

**Packetization**: The packets are then encapsulated with additional information, such as the source and destination addresses, and other metadata required by the network protocols.

**Transmission**: The packets are transmitted into the access network at a rate that depends on the link transmission rate, also known as the link capacity or link bandwidth. The link transmission rate represents the maximum number of bits that can be transmitted over the link per second.

The link transmission rate is an important factor that determines the performance and capacity of the network. The link capacity is typically measured in bits per second (bps), and can vary depending on the type of network and the physical medium used to transmit the data. For example, a wired Ethernet network may have a link capacity of 1 Gbps or higher, while a wireless network may have a lower link capacity due to limitations in the available spectrum and other environmental factors.

$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

## Physical media-

In networking, **a bit refers to the smallest unit of digital information that can be transmitted between a transmitter and a receiver.** A bit can have one of two values, typically represented as 0 or 1, and it is the basic building block of all digital communications.

**The physical link, on the other hand, refers to the actual physical pathway that connects the transmitter and receiver**. This could be a wired connection, such as copper or fiber optic cable, or a wireless connection, such as radio waves or infrared.

In data transmission, **the bit propagates through the physical link, which is responsible for carrying the signal between the transmitter and receiver.**

physical media refers to the physical pathways that data travels through when it is transmitted between devices. **There are two main types of physical media: guided and unguided.**

**Guided media refers to transmission media that uses solid materia**l, such as copper wire, fiber optic cable, or coaxial cable, to transmit signals between devices. The signals are carried through the physical link, which is the actual path that connects the transmitter and receiver.

One example of guided media is twisted pair (TP) cabling, which consists of two insulated copper wires twisted together. TP cabling is commonly used in Ethernet networks and comes in different categories, such as Category 5 (Cat5) and Category 6 (Cat6), which support different data transmission speeds. For instance, Cat5 can support data rates of up to 100 Mbps and 1 Gbps Ethernet, while Cat6 can support data rates of up to 10 Gbps.

In contras**t, unguided media refers to transmission media that use wireless signals,** such as radio waves or infrared, to transmit data. These signals propagate freely through the air and are not contained by any solid material. Examples of unguided media include Wi-Fi, Bluetooth, and satellite communications.

## Physical media: coax, fiber

Coaxial cable consists of two concentric copper conductors, with the signal carried in the center conductor and a grounded shield around it. Coaxial cable is bidirectional, meaning that signals can be transmitted in both directions. It is used for broadband transmission, which means that multiple channels can be carried on a single cable. Coaxial cable is commonly used in cable television (CATV) networks, as well as in hybrid fiber-coax (HFC) networks that combine fiber optic and coaxial cable.

Fiber optic cable, on the other hand, uses glass fibers to transmit data as light pulses. Each pulse represents a bit of information. Fiber optic cable is known for its high-speed operation, with transmission rates in the tens to hundreds of gigabits per second (Gbps). It also has a low error rate due to repeaters being spaced far apart and its immunity to electromagnetic noise. Fiber optic cable is commonly used in long-distance and high-bandwidth applications, such as backbone networks and data centers.

## Physical media: radio

physical media can also refer to wireless transmissions, such as radio waves. Unlike wired media, radio waves do not require a physical "wire" to transmit data. Instead, data is carried in the electromagnetic spectrum.

Radio waves are bidirectional, meaning they can transmit signals in both directions. However, the propagation environment can have a significant impact on the quality and reliability of the

signal. Reflections and obstructions by objects, as well as interference from other sources, can all affect the quality of the signal.

There are different types of radio links used in networking. Local area network (LAN) radio links, such as Wi-Fi, typically have data rates ranging from 11 Mbps to 54 Mbps. Wide-area radio links, such as cellular networks, can provide data rates of a few Mbps for 3G cellular networks. Satellite radio links can offer even higher data rates, ranging from Kbps to 45 Mbps, but with a higher end-to-end delay of around 270 msec.

## The network core

The network core is a mesh of interconnected routers that work together to forward packets from one host to another. The core of the network typically uses a packet-switching approach, in which hosts break down their application-layer messages into packets and forward them from one router to the next, across links on the path from the source to the destination.

Packet switching allows multiple hosts to share the network resources and enables more efficient use of available bandwidth. Each packet is transmitted at the full link capacity, which means that multiple packets can be transmitted simultaneously over the same link. This allows the network to handle a large number of concurrent connections and high-volume traffic, while ensuring that each packet is delivered as quickly as possible.

In a packet-switched network, each packet is treated independently and is routed through the network based on its destination address. Routers use forwarding tables and routing protocols to determine the best path for each packet based on the current network conditions and the routing policies of the network. As the packet travels through the network, it may be forwarded through multiple routers and links before it reaches its destination. The network protocols and algorithms used to manage the packet-switched network are critical to ensuring that packets are delivered reliably and efficiently.

## Packet-switching: store-and-forward

In a packet-switched network, the store-and-forward technique is commonly used to forward packets from one router to the next. With store-and-forward, the entire packet must arrive at the router before it can be transmitted on the next link. This is because routers typically have finite buffers that can store a packet temporarily until it can be transmitted on the next link.

When a host sends a packet into the network, it takes L/R seconds to transmit the packet into the link at R bps. The end-to-end delay, or the time it takes for a packet to travel from the source to the destination, can be calculated as the sum of the transmission delay, propagation delay, and queuing delay. Assuming zero propagation delay and neglecting queuing delay, the end-to-end delay can be approximated as:

End-to-end delay = Transmission delay + Propagation delay
= L/R + 0

= 2L/R (since the packet must be transmitted twice, once from the source to the first router, and again from the last router to the destination)

## Packet Switching: queueing delay, loss

In a packet-switched network, if the arrival rate of packets to a link exceeds the transmission rate of the link for a period of time, the packets will queue up and wait to be transmitted on the link. This can result in queuing delay, which is the time a packet spends waiting in the queue to be transmitted on the link. The queuing delay can be significant if the router buffers become congested and packets have to wait in the queue for a long time.

If the router buffer becomes full and cannot hold any more packets, packets will be dropped or lost. This can happen if the arrival rate of packets is too high for the buffer to handle or if the buffer size is too small. Packet loss can result in degraded performance, increased retransmission, and lower quality of service for network applications. To reduce packet loss, routers can use various mechanisms such as flow control, congestion avoidance, and packet dropping policies based on the priority of packets.

## Two key network-core functions-

The network core performs two key functions: routing and forwarding.

Routing is the process of determining the path that packets should take from the source to the destination. Routing algorithms are used to calculate the best path for packets based on factors such as network topology, traffic load, and network policies. Different routing algorithms use different metrics to calculate the best path, such as shortest path, least cost, or highest bandwidth.

Forwarding is the process of moving packets from a router's input interface to the appropriate output interface based on the destination address of the packet. Each router has a forwarding table that maps destination addresses to output interfaces. When a packet arrives at a router, the router looks up the destination address in its forwarding table and forwards the packet to the appropriate output interface.

Together, routing and forwarding enable packets to be transmitted through the network core from the source to the destination.

## Alternative core: circuit switching

Circuit switching is an alternative to packet switching in which end-to-end resources are allocated and reserved for a "call" between the source and destination. In circuit switching, a dedicated path or circuit is established between the source and destination, and the circuit remains active for the duration of the call.

In circuit switching, the network resources are reserved for the duration of the call, which guarantees a certain level of performance and quality of service. However, the resources allocated to a circuit are not available to other calls, even if they are not being used, which can result in inefficient use of network resources.

Circuit switching is commonly used in traditional telephone networks, where a dedicated circuit is established for each phone call. In contrast, packet switching is used in modern computer networks, including the Internet, where packets are sent independently and may follow different paths through the network.

## Circuit switching: FDM versus TDM

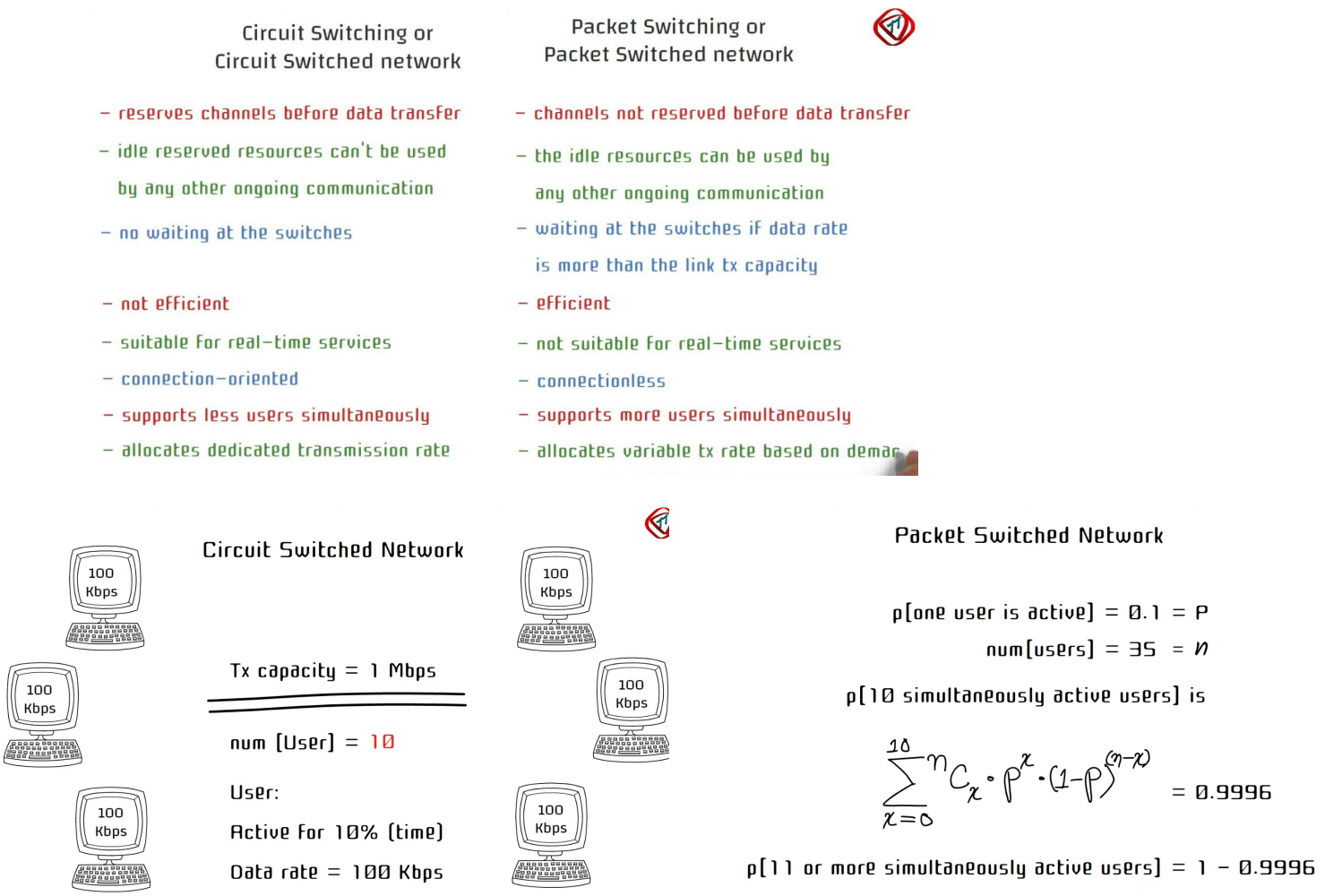
In circuit switching, there are two main approaches for sharing a link's bandwidth between multiple calls: Frequency Division Multiplexing (FDM) and Time Division Multiplexing (TDM).

In FDM, the available bandwidth of the link is divided into frequency bands, and each circuit is allocated a separate frequency band. Each circuit can then transmit data over its allocated frequency band simultaneously with other circuits. FDM is commonly used in analog telephone networks, where different voice signals are transmitted over separate frequency bands.

In TDM, the available bandwidth of the link is divided into time slots, and each circuit is allocated a separate time slot. Each circuit can then transmit data over its allocated time slot in a round-robin fashion, with each circuit taking turns to use the link. TDM is commonly used in digital telephone networks and is also used in some circuit-switched data networks.

Both FDM and TDM have their advantages and disadvantages. FDM is simple and efficient for analog signals, but it requires a guard band between the frequency bands to avoid interference. TDM is more efficient for digital signals, but it requires precise synchronization between the transmitting and receiving devices. In practice, TDM is more commonly used in modern networks, including the Internet, because it is more flexible and efficient for packet switching.

## Packet switching versus circuit switching




packet switching is a winner over circuit switching in many scenarios, but it also has some limitations.

Packet switching is great for bursty data because it allows for efficient use of network resources by sharing them among multiple users. It also does not require any call setup, which makes it simpler and more flexible than circuit switching.

However, packet switching can lead to excessive congestion, which can cause packet delay and loss. In addition, reliable data transfer and congestion control protocols are needed to ensure the quality of service.

Therefore, while packet switching is generally more efficient and flexible, it requires careful management to prevent congestion and ensure reliable data transfer.

*Providing circuit-like behavior in a packet-switched network is an ongoing research problem. One way to provide bandwidth guarantees is by using Quality of Service (QoS) mechanisms. QoS mechanisms can provide different priority levels for different types of traffic, and allocate resources accordingly to ensure*

*that high-priority traffic gets the required bandwidth and low-priority traffic does not impact the performance of high-priority traffic. Different QoS mechanisms include traffic shaping, admission control, and packet scheduling algorithms.*

*However, providing circuit-like behavior is still challenging in packet-switched networks because of the dynamic nature of traffic and the variability of network conditions. It is difficult to provide absolute guarantees of bandwidth and delay for each flow in a network, and congestion can still occur due to unexpected bursts of traffic. Therefore, providing circuit-like behavior in a packet-switched network remains an unsolved problem.*

One analogy for circuit switching can be reserving a table at a restaurant. When you make a reservation, the restaurant allocates a specific table for you at a specific time. The table is dedicated to you and is not shared with anyone else. Similarly, in circuit switching, dedicated resources are allocated for the duration of a call, and those resources are not shared with any other calls.

An analogy for packet switching can be ordering food from a food truck. When you place an order, the food truck prepares the food on demand and gives it to you when it's ready. There is no reservation or dedicated resource allocation involved. Similarly, in packet switching, data is transmitted on demand and the network resources are shared among all the packets.

## Internet structure: network of networks

End systems (such as computers, smartphones, and servers) connect to the Internet through access ISPs (Internet Service Providers). Access ISPs provide a range of services, such as DSL, cable, and fiber connections, to residential, company, and university customers.

Access ISPs interconnect with each other to form a backbone network. This backbone network is made up of high-capacity, long-distance communication links and routing devices such as routers and switches.

The backbone network is connected to regional ISPs, which provide Internet access to smaller cities and towns.

Regional ISPs are interconnected with each other and with national ISPs, which provide Internet access across entire countries.

National ISPs are interconnected with each other through international ISPs, which provide Internet access across different countries and continents.

The Internet is not owned or operated by a single entity or organization. Instead, it is a network of networks that is governed by a set of technical standards and protocols, as well as various national and international policies.

## Four sources of packet delay

Processing delay (dproc): The time it takes for a router to process a packet, typically measured in microseconds. This includes tasks such as checking the packet header and determining the outbound link.

Queueing delay (dqueue): The time a packet spends waiting in a router's output queue to be transmitted on the outbound link. This can be significant if the router is congested and there are many packets waiting to be transmitted.

Transmission delay (dtrans): The time it takes to push the packet's bits onto the link. This is determined by the packet length and the link's capacity, and is typically measured in microseconds.

Propagation delay (dprop): The time it takes for a bit to travel from one end of a link to the other. This is determined by the distance between the two routers and the speed of the link, and is typically measured in microseconds or milliseconds.

### $d_{trans}$: transmission delay:

- $L$: packet length (bits)
- $R$: link *bandwidth (bps)*
- $d_{trans} = L/R$

### $d_{prop}$: propagation delay:

- $d$: length of physical link
- $s$: propagation speed in medium ($\sim 2 \times 10^8$ m/sec)
- $d_{prop} = d/s$

## Queueing delay (revisited)

The queueing delay depends on the packet arrival rate, the link bandwidth, and the packet length. As the packet arrival rate increases, the average queueing delay also increases, because more packets are competing for the same link bandwidth and queue space. Similarly, as the packet length increases, the average queueing delay also increases, because longer packets occupy the queue for a longer time.

The ratio of the packet arrival rate to the link bandwidth, denoted by La/R, is also an important factor in determining the average queueing delay. **When La/R is small, the average queueing delay is small, because the packets are arriving at a rate that is lower than the link capacity. When La/R approaches 1, the average queueing delay can become large, because the packets are arriving at a rate that is close to or exceeding the link capacity. When La/R is greater than 1, the average queueing delay becomes infinite, because the arrival rate exceeds the link capacity and the queue will continue to grow indefinitely.**

## “Real” Internet delays and routes

traceroute program: provides delay measurement from source to router along end-end Internet path towards destination. For all i:

- ☐ sends three packets that will reach router i on path towards destination
- ☐ router i will return packets to sender
- ☐ sender times interval between transmission and reply.

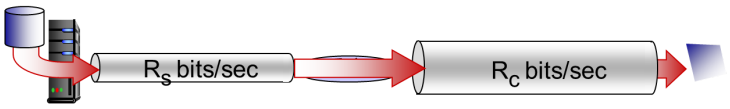
## Packet loss

. When a packet arrives at a full queue, it is dropped or lost. This can occur when the queue preceding a link has finite capacity and is overwhelmed with packets arriving at a rate faster than the rate at which packets are being transmitted on the link. The dropped packet may be retransmitted by the previous node, by the source end system, or not at all, depending on the specific protocol being used.

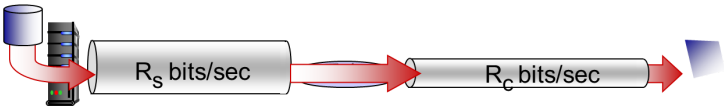
## Throughput

Throughput refers to the rate at which bits are transferred between sender and receiver, and it can be measured both instantaneously and over a longer period of time. Instantaneous throughput refers to the rate at a given point in time, while average throughput refers to the rate over a longer period of time, such as over the duration of a data transfer or communication session.

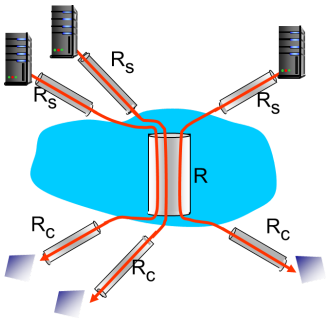
- $R_s < R_c$ What is average end-end throughput?




- $R_s > R_c$ What is average end-end throughput?




A bottleneck link is a link in a network where the available bandwidth or capacity is lower than the other links. It can cause delays and congestion in the network, as it limits the amount of data that can be transferred through it. Think of it like a narrow street in a busy city - the cars have to slow down or wait in a line to pass through it, which slows down the overall traffic flow. Similarly, data passing through a bottleneck link can experience delays or congestion, which can impact the performance of the network.



The per-connection end-to-end throughput refers to the rate at which data can be transferred between the source and destination for a single connection. The formula for calculating the per-connection end-to-end throughput is:

min(Rc, Rs, R/10)

where Rc is the receiver's capacity, Rs is the sender's capacity, and R is the capacity of the bottleneck link in the network.

In practice, either the receiver's or the sender's capacity is often the bottleneck for a single connection. This means that the throughput of the connection is limited by the capacity of either the sender or the receiver, rather than the capacity of the bottleneck link.

## Protocol “layers”

The most commonly referenced layered network architecture is the OSI (Open Systems Interconnection) model, which consists of seven layers:

1. Physical Layer: d**eals with the physical transmission of data** over the network medium, such as electrical or optical signals.
2. Data Link Layer: provides error-free transfer of data frames **between adjacent node**s and handles flow control and error detection and correction.
3. Network Layer: provides logical addressing and routing services to ensure that data is sent from **source to destination** via the best available path.
4. Transport Layer: provides end-to-end communication services, such as reliable data delivery and flow control, **between applications running on different hosts.**
5. Session Layer: est**ablishes, manages, and terminates connections between applications,** including authentication and encryption services.
6. Presentation Layer: **translates data into a format that the application layer can understand and manages data encryption and compression.**
7. Application Layer: pr**ovides network services to the end-user applications, such as email, file transfer, and web browsing.**

## Why layering?

1. **Structured approach:** Layering provides a systematic and structured approach to design complex systems. It allows for the identification and relationship of various components of the system and provides a clear understanding of how each component interacts with others.

2. **Modularization:** Layering allows for the modularization of complex systems. Each layer can be designed and implemented independently, making it easier to maintain, update, and replace individual layers without affecting the rest of the system.

3. **Abstraction**: Each layer provides an abstraction of the layer below it, allowing higher layers to operate without knowledge of the underlying details of lower layers. This makes it easier to design and implement protocols that can work across different types of networks and technologies.

4. **Standardization:** Layering facilitates standardization by providing a framework for designing and implementing protocols that are interoperable across different systems and vendors. This allows for greater compatibility and ease of integration between different systems.

While layering can bring many benefits, it is not without its drawbacks. Some criticisms of layering include:

1. **Overhead**: The addition of multiple layers and the need for each layer to communicate with the layer above and below it can introduce extra overhead and decrease performance.

2. **Rigidity**: Layering can make it difficult to introduce new functionality or adapt to changing network requirements because any changes made to one layer can impact other layers in the system.

3. **Inefficiency**: Layering can lead to inefficiencies when different layers perform similar or redundant tasks. For example, if both the transport and network layers perform error correction, this can waste resources and slow down the system.

## Internet protocol stack

1. Application Layer: This layer provides services to the end-user applications for accessing the network. It includes protocols such as HTTP, FTP, SMTP, etc.

2. Transport Layer: This layer provides end-to-end communication services for applications. It includes protocols such as TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

3. Network Layer: This layer handles the routing and forwarding of data across multiple networks. It includes protocols such as IP (Internet Protocol), ICMP (Internet Control Message Protocol), and routing protocols such as OSPF (Open Shortest Path First) and BGP (Border Gateway Protocol).

4. Link Layer: This layer provides data transfer services between neighboring network elements. It includes protocols such as Ethernet, WiFi (802.11), and PPP (Point-to-Point Protocol).

## ISO/OSI reference model

Ager layer gulao ase.
The presentation and session layers of the ISO/OSI reference model were designed to provide additional services to the application layer, such as data compression, encryption, and checkpointing. In contrast, the Internet protocol stack does not include these layers. While some of these services are not needed for many applications, some applications may require them.

For example, encryption is critical for secure communication between applications, such as online banking or e-commerce. Compression can be useful for reducing the size of large files before transmission. Checkpointing and recovery can be important for applications that need to resume data transfer after a network failure or interruption.If these services are needed, they must be implemented in the application layer itself, which can make the application more complex and potentially less efficient.

**—----------Slide 61-67 Read—----------**

# Chapter -2

## Creating a network app

When creating a network application, the focus is on writing programs that run on different end systems and communicate over the network. This allows for rapid app development and propagation since there is no need to write software for network-core devices, which do not run user applications. Instead, network-core devices provide the necessary infrastructure and services for end systems to communicate with each other. Therefore, the network application can be developed independently of the network infrastructure.

## Application architectures

In a client-server architecture, the application is structured such that there is a central server that provides services or resources to multiple clients. Clients typically initiate communication with the server to request a service or resource, and the server responds to those requests. Examples of client-server applications include web browsing, email, and file transfer.

In a peer-to-peer (P2P) architecture, the application is structured such that there are multiple peers (or nodes) that are connected to each other, and each peer can act as both a client and a server. Peers can request resources or services from other peers, and can also provide resources or services to other peers. P2P applications are often used for file sharing and streaming media, among other things.

There are also hybrid architectures that combine elements of both client-server and P2P architectures, such as distributed hash tables (DHTs) used in some peer-to-peer file sharing systems.

## Client-server architecture

In a client-server architecture, the server is a powerful and always-on host that provides services to multiple clients. The server has a permanent IP address and is usually located in a data center to provide scaling and high availability. Clients, on the other hand, are end systems that request services from the server. Clients may be intermittently connected, have dynamic IP addresses, and do not communicate directly with each other.

Examples of client-server architecture include web applications where a client's browser requests web pages from a server, email clients that retrieve messages from a mail server, and file transfer applications that download files from a file server.

## P2P architecture

P2P (peer-to-peer) architecture is a distributed network architecture where all participating nodes, called peers, communicate directly with each other, without the need for a central server or a hierarchy of servers. In a P2P system, each peer can act both as a client and as a server, providing and requesting services from other peers.

There is no always-on server in a P2P architecture. Instead, each peer is responsible for maintaining its own resources and providing services to other peers when requested. This makes P2P architecture very scalable, as new peers can easily join the network and provide additional resources and services.

However, the dynamic nature of peers in a P2P system, with intermittent connections and changing IP addresses, makes it more complex to manage compared to client-server architecture. Additionally, P2P systems often face challenges in terms of security and trust, as it can be difficult to ensure that peers are providing legitimate services and not maliciously attacking the network.

## Processes communicating

In a client-server architecture, the client process is the one that initiates communication by sending requests to the server process. The server process, on the other hand, waits to be contacted by the client process and responds to the requests it receives.

In a P2P architecture, each peer may act as both a client and a server. A peer may request services from other peers and provide services to other peers as well. This allows for a more distributed and decentralized approach to networking compared to the client-server architecture.

Processes within the same host can communicate through inter-process communication (IPC) mechanisms provided by the operating system, such as shared memory, message passing, and pipes. On the other hand, processes in different hosts communicate by exchanging messages over the network. This is typically done through socket programming, where processes use the Internet Protocol (IP) and transport layer protocols such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) to establish connections and exchange data.

## Sockets

A socket is an endpoint of a two-way communication link between two processes running over a network. A process binds itself to a socket to establish a network connection with another process. Once a socket is established, a process can send or receive data to/from the socketA socket is the interface between the application layer and the transport layer within a host. It is also referred to as the Application Programming Interface (API) between the application and the network, since the socket is the programming interface with which network applications are built.

To use an analogy, think of a socket as a mailbox with an address. The mailbox can send and receive letters to and from other mailboxes with their own addresses. Similarly, a socket can send and receive data to and from other sockets with their own IP addresses and port numbers. The only control that the application developer has on the transport-layer side is (1) the choice of transport protocol and (2) perhaps the ability to fix a few transport-layer parameters such as maximum buffer and maximum segment sizes

*Ekta host arekta host r sathe jogajog korte chaile ek host  application layer theke socket r moddhe data pathai de . socket baki kaj transportlayer r sathe buijha nei.abr jar kase pathano house se socket theke data buijha nei.*

## Addressing processes

The combination of IP address and port number is called a socket address, which uniquely identifies a process on a host in the network.To enable communication between processes, each process must have a unique identifier. In the Internet, this identifier consists of the IP address of the host on which the process is running, and a port number associated with the process.

IP addresses are unique 32-bit identifiers assigned to each device connected to the Internet. Ports are 16-bit numbers used by transport layer protocols (e.g., TCP and UDP) to identify the process on the host that is sending or receiving data.

For example, when a web browser sends an HTTP request to a web server running on host gaia.cs.umass.edu, the destination address is the IP address of gaia.cs.umass.edu (which is 128.119.245.12), and the destination port number is 80 (which is the standard port number for HTTP). The browser may also be assigned a temporary port number by the transport layer protocol to enable two-way communication with the server.

The combination of the IP address and the port number allows for end-to-end addressing of the process on the host, enabling messages to be sent to and received from the correct process.

## App-layer protocol defines

An application-layer protocol defines t**he types of messages exchanged, the message syntax, message semantics, and rules for when and how processes send and respond to messages.**

The types of messages exchanged refer to the specific requests and responses that are expected between communicating processes. For example, a client may send a request for a web page, and the server will respond with the requested page.

**The message syntax defines the structure of the messages exchanged.** This includes the fields in the message and how they are delimited. For example, in HTTP, a request message consists of a request line, headers, and an optional message body.

**The message semantics define the meaning of the information in the fields**. For example, in an email message, the "To" field specifies the recipient of the message.

**Finally, the protocol specifies the rules for when and how processes send and respond to messages**. For example, in HTTP, the client sends a request message to the server and waits for a response before sending another request.

Proprietary protocols are developed by a particular company or organization and are not openly available for others to use or implement. They are often used for commercial or strategic reasons to gain a competitive advantage. On the other hand, open protocols are defined in RFCs (Request for Comments) and are openly available for anyone to use and implement. This allows for interoperability between different systems and encourages innovation and competition. Examples of open protocols include HTTP and SMTP.

## What transport service does an app need?

some applications may require 100% reliable data transfer, while others can tolerate some loss. Therefore, the transport service needed by an app depends on the application requirements. For example, a file transfer application would need a reliable transport service that ensures all the data is delivered without errors or loss, while a real-time audio streaming application may tolerate some loss or delay, but requires timely delivery of the packets to maintain the quality of the audio

**Timing is another factor that determines the type of transport service an application needs**. Some applications, such as internet telephony and interactive games, require low delay to be effective. In other words, there should be minimal delay in the delivery of data from one

end system to another. This is important to ensure that real-time communication is smooth and responsive. On the other hand, there are other applications that can tolerate some delay, such as email or file transfer.

Some applications such as video streaming require a minimum amount of throughput to function properly, while others such as email or file transfer can make use of any available throughput.

Some applications (e.g., online banking, e-commerce) require security services such as encryption and data integrity to protect sensitive information from being intercepted or tampered with during transmission.

## Internet transport protocols services

**TCP (Transmission Control Protocol)** is a connection-oriented transport protocol in the Internet Protocol suite. It provides reliable, ordered, and error-checked delivery of data between applications running on hosts communicating over an IP network.

TCP provides several services to ensure reliable transport of data between the sending and receiving processes. It uses a sequence number mechanism to ensure that all segments are received in the correct order and to detect missing segments. TCP also provides acknowledgment and retransmission mechanisms to ensure that all data is successfully transmitted and received without errors.

TCP also provides flow control and congestion control mechanisms to prevent the sender from overwhelming the receiver or the network. Flow control is achieved through the use of a sliding window mechanism, where the receiver advertises the amount of data it can receive, and the sender adjusts its transmission rate accordingly. Congestion control is achieved through the use of congestion avoidance algorithms, where the sender throttles its transmission rate when it detects network congestion.

TCP is a connection-oriented protocol, which means that a connection must be established between the sender and receiver before any data can be transmitted. This connection setup involves a three-way handshake, where the sender and receiver exchange control messages to synchronize their sequence numbers and agree on other parameters. Once the connection is established, data can be transmitted in both directions.

TCP does not provide timing guarantees, minimum throughput guarantees, or security features such as encryption or data integrity. These services must be provided by other layers in the protocol stack or by the application layer protocols that run on top of TCP.

**UDP (User Datagram Protocol)** is a transport protocol that provides an unreliable data transfer service between sending and receiving processes. It does not provide many of the services that TCP offers, such as reliability, flow control, congestion control, timing, throughput

guarantee, security, or connection setup. Instead, UDP provides a simple, connectionless service that is often used for applications that require low overhead and do not require reliable delivery, such as online gaming, streaming media, and DNS (Domain Name System) queries. Because UDP does not provide any reliability or flow control mechanisms, the receiving process may receive duplicate packets, out-of-order packets, or packets that have been lost in transit. Applications that use UDP must implement their own mechanisms for handling these issues, if necessary.

## Securing TCP

When data is transmitted over TCP or UDP, it is sent in cleartext form which means it can be intercepted and read by anyone who has access to the network. This can be a security risk, especially for sensitive information such as passwords.

TCP and UDP do not provide encryption or security mechanisms to protect the data being transmitted. As a result, if passwords or other sensitive data are transmitted over a TCP or UDP connection, they are vulnerable to interception and theft.To address this issue, additional security measures can be added on top of TCP or UDP.

SSL (Secure Sockets Layer) is a protocol that provides secure communication between client and server over the internet. It is designed to provide encryption, data integrity, and end-point authentication. SSL works by establishing an encrypted connection between a client and a server, which ensures that any data transmitted between them is protected from eavesdropping, tampering, and forgery.

1. SSL uses a combination of public-key and symmetric-key encryption to secure the communication between the client and server.
2. SSL also provides data integrity by adding a message digest to each message sent between the client and server. This ensures that any tampering with the data can be detected by either party.
3. End-point authentication is also provided by SSL. The server's digital certificate is used to verify its identity, and the client can also be authenticated by the server if required.

*SSL (Secure Sockets Layer) is a protocol that operates at the application layer of the TCP/IP protocol stack. Applications can make use of SSL libraries (such as OpenSSL) to establish secure communication over TCP connections. The SSL libraries handle the encryption and decryption of data, as well as the authentication of the endpoints.*

*The SSL socket API provides a way for applications to use SSL/TLS encryption to secure their communication over TCP. When an application wants to use SSL, it first creates a TCP socket and then initiates an SSL handshake with the server. During the handshake, the two endpoints negotiate encryption algorithms, exchange certificates for end-point authentication, and establish a session key that is used to encrypt and decrypt the data.*

## *Web and HTTP->slide porlei hbe*

## HTTP overview

HTTP (Hypertext Transfer Protocol) is an application-layer protocol that is primarily used for transmitting data over the World Wide Web. It follows a client-server model where the client, typically a web browser, requests for web objects such as HTML pages, images, videos, etc., from the server. The server, which is typically a web server, receives the request and responds with the requested objects using HTTP.

HTTP is a stateless protocol, which means that each request-response pair is independent of the others, and the server does not maintain any information about the previous requests made by the client. To overcome this limitation, cookies are often used to maintain state information between the client and the server.

HTTP uses TCP as its underlying transport protocol, and it typically operates over port 80 for non-secure communication and port 443 for secure communication using HTTPS (HTTP Secure). HTTP messages consist of a request message sent by the client to the server, and a response message sent by the server to the client.

HTTP is an extensible protocol that supports various features such as caching, authentication, and compression, and it is widely used for various applications such as web browsing, web services, and IoT (Internet of Things) devices.

## HTTP overview (continued)

1. Client initiates TCP connection.
2. server accepts TCP connection from client.
3. HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
4. TCP connection closed

protocols that maintain “state” are complex!

1. past history (state) must be maintained
2. if server/client crashes, their views of “state” may be inconsistent, must be reconciled

## HTTP connections

HTTP connections can be either persistent or non-persistent. In a non-persistent connection, a separate TCP connection is established between the client and the server for each request/response pair. After each response is received, the connection is closed.

In a persistent connection, the client and server keep the connection open after the first request/response pair, allowing multiple requests and responses to be exchanged on the same connection. This reduces the overhead of establishing new connections for each request, improving performance.

## —Read Slide 22-27 as it is—

Slide 24: RTT (Round-Trip Time) is the time it takes for a packet to travel from the sender to the receiver and back again. It is a measure of the latency or delay in a network, and it is often used as a key metric for evaluating network performance.

Slide 25:

**non-persistent HTTP issues:**

1. requires 2 RTTs per object
2. OS overhead for each TCP connection
3. Non-persistent HTTP has a performance issue known as the "head-of-line blocking" problem. This is because browsers open multiple parallel TCP connections to fetch referenced objects, but each connection can only request one object at a time. If one object is slow to download, it holds up the rest of the objects requested over that connection, which can significantly increase the overall response time. This problem can be mitigated by using persistent HTTP, which allows multiple objects to be fetched over a single TCP connection.

**Persistent HTTP** allows for multiple requests and responses to be sent over the same TCP connection, eliminating the overhead of establishing a new connection for each request/response. This can significantly reduce the latency and improve the performance of web applications. In persistent HTTP, the server leaves the connection open after sending the initial response, and subsequent requests from the client are sent over the same connection. This can be done indefinitely or until a timeout or maximum number of requests is reached.

## Uploading form input

*Post method-> web page r form e use kora hoi.secured.Entity body te iput pathano hoi.*

The POST method is used to upload data from a form input to a server. The form input data is included in the entity body of the HTTP request message. The HTTP request message includes a Content-Type header that indicates the type of data being sent in the entity body (e.g., text, image, video, etc.) and a Content-Length header that specifies the length of the entity body in bytes.

*Url method-> secured na . Url sathe data append kore pathai dea hoi.*

## Method types

**HTTP/1.0:**

1. The GET method is used to request an object.
2. The POST method is used to upload data to the server.
3. The HEAD method in HTTP/1.0 is used to request the server to leave the requested object out of the response, which is useful to obtain metadata about the object without transferring the entire object itself.

**HTTP/1.1**

1. PUT->uploads file in entity body to path specified in URL field

2. DELETE->deletes file specified in the URL field

**—Read Slide 30,31 as it is—**

## User-server state: cookies

Cookies are a mechanism for maintaining state between a user and a server. A cookie is a small piece of data that a server sends to a client, and the client stores it and sends it back to the server on subsequent requests. The server can use the cookie to identify the user and maintain information about the user's session.

Cookies are sent in the header of an HTTP response from the server to the client, and the client sends the cookie back in the header of subsequent HTTP requests to the same server. The server can then use the information in the cookie to retrieve the user's state information and provide personalized responses.

Cookies can be used for various purposes, including session management, user tracking, and personalization. However, they can also be used for tracking users across multiple websites, which raises privacy concerns. To address these concerns, modern browsers provide controls for users to manage cookies, such as blocking third-party cookies or deleting cookies after a certain period.

**—Read Slide 34,34 as it is—**

## Web caches (proxy server)

**A web cache, also known as a proxy server, is a server that sits between a client (e.g., a web browser) and a web serve**r. When a client requests a web object, the request goes to the web cache instead of the origin server. If the requested object is in the cache, the web cache returns the object to the client. If the object is not in the cache, the web cache forwards the request to the origin server, retrieves the object, and returns it to the client.

Web caches are used to reduce response time and network traffic. By storing frequently accessed objects, web caches can reduce the number of requests that need to be forwarded to the origin server, which can improve response time. Additionally, web caches can reduce network traffic by serving objects that are already in the cache, which can reduce the amount of data that needs to be transferred over the network.

Web caches can be either private or public. Private caches are typically used by individual users or organizations, while public caches are typically used by Internet service providers or other large organizations to serve multiple clients.

**—Read Slide 35 as it is—**

## FTP: the file transfer protocol

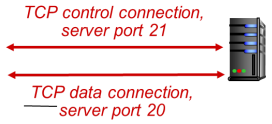
FTP uses a client-server architecture. The client software is typically a web browser or a dedicated FTP client, while the server software is typically an FTP server running on a remote machine. The client connects to the server using a TCP connection on port 21 by default.

1. FTP client contacts FTP server at port 21, using TCP
2. client authorized over control connection
3. client browses remote directory, sends commands over **control connection**
4. when server receives file transfer command, server opens 2nd TCP data connection (for file) to client
5. after transferring one file, server closes data connection

**->Important**

1. Server opens another TCP data connection to transfer another file
2. FTP server maintains “state”: current directory, earlier authentication
3. In FTP**, the control connection is considered "out of band"** because it is used only for sending control information, such as commands and replies between the client and the server. The actual file data transfer is performed using a separate data connection.
4. **control connection r data connection r tcp port alada.**

**—Read Slide 40 as it is—**




## Electronic mail

It involves three major components:-

1. **User agents**: User agents are the programs that allow users to send and receive email messages. They include web-based email clients such as Gmail, Yahoo! Mail, and Outlook, as well as standalone email programs such as Microsoft Outlook, Mozilla Thunderbird, and Apple Mail.

2. **Mail servers**: Mail servers are responsible for storing and forwarding email messages. There are two types of mail servers: incoming mail servers (also known as mail delivery agents) and outgoing mail servers (also known as mail transfer agents). Incoming mail servers receive email messages from other servers and store them in the recipient's mailbox until they are retrieved by the user agent. Outgoing mail servers send email messages from the user agent to the recipient's mail server.

3. **Simple Mail Transfer Protocol (SMTP):** SMTP is the protocol used to transfer email messages between mail servers. When a user sends an email message, their user agent communicates with their outgoing mail server using SMTP to transmit the message to the recipient's mail server. The recipient's mail server then stores the message until it is retrieved by the recipient's user agent.

## Electronic mail: mail servers

mail servers:

1. mailbox
2. message queue

3. SMTP protocol between mail servers to send email messages
    a. client: sending mail server
    b. “server”: receiving mail server

## Electronic Mail: SMTP [RFC 2821]

SMTP (Simple Mail Transfer Protocol) is the primary protocol used to send and receive email messages between mail servers. It operates on top of TCP and listens on port 25. SMTP uses a client-server model where the client is responsible for initiating a connection to the server.

SMTP is a text-based protocol where commands and responses are sent as plain text messages. SMTP defines a set of commands that are used to transfer email messages between servers.

- ☐ uses TCP to reliably transfer email message from client to server, port 25
- ☐ direct transfer: sending server to receiving server
- ☐ three phases of transfer
- ☐ command/response interaction (like HTTP, FTP)
- ☐ messages must be in 7-bit ASCII

The SMTP protocol involves three phases of transfer:

Handshaking (greeting): In this phase, the client establishes a TCP connection with the server, and they exchange greeting messages to identify themselves and negotiate the parameters of the connection.

Transfer of messages: In this phase, the client sends one or more email messages to the server, which in turn stores them in the appropriate mailbox or queue for delivery. Each message consists of a header and a body, separated by a blank line.

Closure: In this phase, the client and server terminate the TCP connection, which indicates the end of the SMTP transaction. If there are more messages to be sent, the client can establish a new connection and repeat the process.

**—Read Slide 45,46 as it is—**

## SMTP: final words

SMTP uses persistent connections: Once the client connects to the server, it stays connected until all the messages are sent. This allows for faster transmission of multiple messages since the overhead of setting up and tearing down the connection is avoided.

SMTP requires the message (header and body) to be in 7-bit ASCII: This is because the Internet was originally designed to transmit only 7-bit ASCII characters, and SMTP still adheres to this limitation. To represent non-ASCII characters, various encoding schemes such as MIME (Multipurpose Internet Mail Extensions) are used.

SMTP server uses CRLF.CRLF to determine the end of the message: The message is terminated by a special sequence of characters, consisting of two carriage return (CR) and line feed (LF) pairs (CRLF.CRLF). This signals the end of the message, and the server can then process and deliver the message to the recipient's mailbox.
—->
The client (web browser) initiates the request for a resource (object) from the server by sending an HTTP request message, which is a pull-based model. The server responds by sending the requested object as an HTTP response message.
In contrast, in SMTP, the email message is pushed to the receiver's mailbox by the sender's mail server. The receiver's mail client then pulls the message from the mailbox to read it. So, SMTP is a push-based model.

. In HTTP, each object (such as an image or a web page) is encapsulated in its own response message, whereas in SMTP, multiple objects can be sent in a single multipart message. The multipart message contains a boundary string that separates each object in the message. This allows for efficient transmission of multiple related objects, such as images and text, in a single email message.

**—Read Slide 48,49 as it is—**

## POP3 (more) and IMAP

POP3 is a simple protocol that typically downloads all messages from the server to the client computer and deletes them from the server. It has two modes of operation. In the "download-and-delete" mode, the messages are deleted from the server once they are downloaded by the client. In the "download-and-keep" mode, the messages are not deleted from the server and can be accessed by different clients. However, this mode requires more storage space on the server. but there is no synchronization of message state between the client and server.POP3 (Post Office Protocol version 3) is a stateless protocol, meaning that it does not maintain any information about the previous sessions.

IMAP is a more feature-rich protocol that allows for the manipulation of stored messages on the server, such as organizing messages into folders and searching for specific messages. With IMAP, messages are stored on the server, and the client can download only the message headers or specific message parts, rather than the entire message. IMAP also supports synchronization of message state between the client and server, so that actions taken on one device are reflected on other devices accessing the same account.IMAP also maintains user state across sessions, which means that the server remembers the names of folders and mappings between message IDs and folder names, even if the user logs in from a different device or client. This makes it easier for users to manage their email messages and folders, even when using different devices or clients to access their email.

## DNS: domain name system

The Domain Name System (DNS) is a hierarchical and distributed naming system that maps domain names to IP addresses. It is used to translate user-friendly domain names, such as www.example.com, into the numerical IP addresses, such as 93.184.216.34, that computers use to identify each other on the internet.

DNS is based on a client-server model, where a client (such as a web browser) sends a query to a DNS server, asking for the IP address associated with a particular domain name. The DNS server then responds with the IP address, allowing the client to establish a connection with the requested server.

DNS operates on **a hierarchical and decentralized structure,distributed database** consisting of multiple levels of servers. At the top of the hierarchy are the root servers, which store information about top-level domains such as .com, .org, and .edu. Below the root servers are the authoritative name servers, which store information about individual domains and their associated IP addresses. Finally, there are the local DNS resolvers, which are typically provided by internet service providers and cache DNS information to improve performance.

## DNS: services, structure

DNS (Domain Name System) provides a distributed database lookup service for translating human-readable domain names into IP addresses that computers can understand. DNS is a critical component of the Internet infrastructure and is responsible for resolving billions of queries every day.

The services provided by DNS include:

1. Name to IP address resolution: DNS translates human-readable domain names, such as www.example.com, into IP addresses, such as 192.0.2.1.

2. IP address to name resolution: DNS can also translate IP addresses into domain names, which can be useful in various network troubleshooting scenarios.

3. Load balancing: DNS can be used to distribute traffic across multiple servers, by returning different IP addresses for the same domain name, in a technique known as round-robin DNS

.

*why not centralize DNS?*

- single point of failure
- traffic volume
- distant centralized database
- maintenance

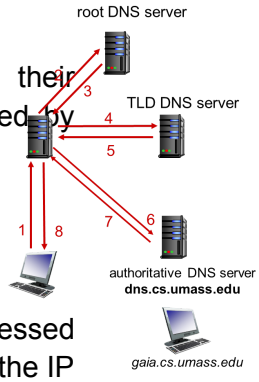
**—Read Slide 54-56 as it is—**

## Local DNS name server

A local DNS name server, also called a DNS resolver, is a server that is responsible for resolving domain name queries from clients within a local network. When a client sends a DNS query to the local DNS resolver, the resolver checks its cache to see if it already has the IP address associated with the requested domain name. If the address is not in the cache, the resolver queries other DNS servers to find the answer.

The local DNS resolver is usually configured by the network administrator, and it can forward queries to other DNS servers if it cannot find the answer in its cache. The resolver can also cache the answers it receives from other DNS servers, which can improve the performance of the DNS system and reduce network traffic.

Many Internet service providers (ISPs) provide their own local DNS resolvers for their customers, but it is also possible to use public DNS resolvers, such as those provided by Google, Cloudflare, or OpenDNS.




### DNS name resolution example

*Iterated query:*

1. The user's computer first checks its local DNS cache to see if it has recently accessed "www.example.com" and if the corresponding IP address is stored in the cache. If the IP address is found in the cache, the name resolution process is complete.

2. If the IP address is not found in the cache, the user's computer sends a DNS query message to its configured local DNS server. The query message includes the name "www.example.com".

3. The local DNS server first checks its cache to see if it has recently resolved "www.example.com" and if the corresponding IP address is stored in the cache. If the IP address is found in the cache, the local DNS server returns the IP address to the user's computer.

4. If the IP address is not found in its cache, the local DNS server contacts the root DNS server and asks for the IP address of the ".com" top-level domain.

5. The root DNS server responds to the local DNS server with the IP address of the ".com" top-level domain.

6. The local DNS server then contacts the ".com" top-level domain DNS server and asks for the IP address of "example.com".

7. The ".com" top-level domain DNS server responds to the local DNS server with the IP address of the "example.com" domain.

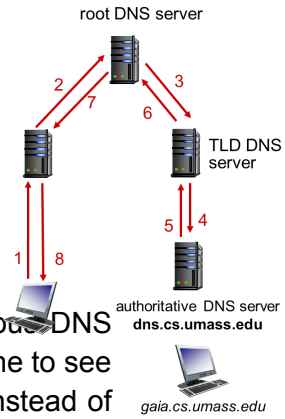
8. Finally, the local DNS server contacts the  authoritative DNS server and asks for the IP address of "www.example.com".

9. The  authoritative DNS server responds to the local DNS server with the IP address of "www.example.com".

10. The local DNS server returns the IP address to the user's computer, which can then use it to establish a connection to the "www.example.com" website.

## recursive query:

Write the recursive query in previous style.




## DNS: caching, updating records

DNS caching is the mechanism used by DNS servers to store the results of previous DNS queries. When a DNS server receives a query for a domain name, it first checks its cache to see if it has a recent record of that domain name. If it does, it returns the cached record instead of querying other DNS servers. This caching mechanism helps reduce the DNS query traffic and improves the DNS lookup performance.

DNS records have a time-to-live (TTL) value associated with them, which specifies the amount of time a record can be cached before it is considered stale and needs to be updated. When the TTL of a record expires, the DNS server needs to perform a new query to update its cache. This ensures that the cached records are accurate and up to date.

DNS servers can also perform zone transfers to update their local databases with the latest DNS records from authoritative name servers. Zone transfers are typically used in primary-secondary DNS server configurations, where the primary DNS server is responsible for maintaining the DNS zone data and the secondary servers periodically retrieve the updated data through zone transfers.

## DNS records

RR format: **`(name, value, type, ttl)`**

1. A record (Address Record): This record maps a hostname to an IPv4 address.

2. CNAME record (Canonical Name Record): This record is used to map one hostname to another. For example, you can map "www.example.com" to "example.com".

3. MX record (Mail Exchange Record): This record specifies the mail server responsible for accepting email messages on behalf of a domain.

4. NS record (Name Server Record): This record specifies the name servers responsible for a domain.

# Chapter 3
## Transport services and protocols

Transport layer is responsible for providing end-to-end communication between applications running on different hosts. It shields the upper layers from the underlying network details and provides a reliable communication channel by ensuring that data packets are delivered without errors, in sequence and without duplication.

The most common transport protocols used on the Internet are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

TCP is a reliable, connection-oriented protocol that provides a stream-oriented service for transporting data between applications. It ensures that data is transmitted without errors, in sequence, and without duplication, by using mechanisms such as flow control, congestion control, and error detection.

UDP, on the other hand, is a connectionless, unreliable protocol that provides a datagram-oriented service for transporting data between applications. It does not guarantee that data is delivered or that it arrives in order, and it does not provide any error detection or correction mechanisms.

Transport layer protocols provide services such as **flow control, error control, congestion control, and multiplexing/demultiplexing** to the applications running on top of them. The

choice of transport protocol depends on the specific requirements of the application and the characteristics of the network being used.
->
Transport protocols provide logical communication between application processes running on different hosts. They run in end systems and are responsible for breaking down application messages into segments, passing them to the network layer for transmission over the network, and reassembling them into messages at the receiving end to be passed to the application layer. Transport protocols also provide services such as reliable message delivery, flow control, and congestion control. There are different transport protocols available to applications, but on the Internet, the two most common ones are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

## Transport vs. network layer

T**he network layer is responsible for providing logical communication between hosts,** i.e., it is responsible for delivering packets from one host to another. **On the other hand, the transport layer is responsible for providing logical communication between processes running o**n different hosts, i.e., it is responsible for delivering messages from one process to another.

**The transport layer receives messages from the application layer, breaks them into segments, and sends them to the network layer. At the receiving end, the transport layer receives segments from the network layer, reassembles them into messages, and passes them to the application layer**. The network layer is responsible for routing packets between hosts using IP addresses, whereas the transport layer uses port numbers to identify the processes running on the hosts

12 kids in Ann's house sending letters to 12 kids in Bill's house:
**hosts = houses**
**processes = kids**
**app messages = letters in envelopes**
**transport protocol = Ann and Bill who demux to in-house siblings**
**network-layer protocol = postal service**

To expand on this analogy:

- Each house represents a host, which could be a computer or other device on the network.
- The kids in each house represent the processes running on each host that want to communicate with each other.
- The letters in envelopes represent the application messages being sent between the processes.

- Ann and Bill act as the transport protocol, which breaks up the application messages into segments and sends them across the network to the corresponding processes in the other house.
- The postal service represents the network-layer protocol, which routes the segments through the network to the correct destination based on the addresses on the envelopes.

## Internet transport-layer protocols

The two most commonly used transport-layer protocols on the Internet are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

**TCP is a connection-oriented, reliable protocol**. **It provides reliable data transfer between applications running on different hosts, using a mechanism called flow control and congestion control. TCP segments are retransmitted if they are lost or corrupted during transmission, and the receiver acknowledges the receipt of every segment to ensure reliable delivery.**

UDP, on the other hand, is a connectionless, unreliable protocol. It provides a way to send and receive datagrams, which ar**e independent, self-contained pac**kets of data that may arrive out of order or be lost during transmission. UDP is often used for applications that require low-latency communication, such as online gaming, video streaming, and VoIP.

->
The Internet transport-layer protocols, namely TCP and UDP**, do not provide any guarantees on delay or bandwidth. Instead, they provide best-effort service, meaning that they try their best to deliver data in a timely and efficient manner, but they do not make any promises about the quality or speed of delivery**. This is because the underlying network infrastructure is shared and dynamic, and congestion and other factors can affect the performance of data transmission.

## Multiplexing/demultiplexing

*Msg Segment E Break Kora / Abr Join Lagano*
*multiplexing at sender*
*demultiplexing at receiver:*

Multiplexing is the process of combining multiple data streams into a single stream, which can be transmitted over a single communication link. Demultiplexing is the reverse process of separating the multiplexed data stream into individual streams at the receiving end.

In the context of transport-layer protocols, multiplexing refers to the ability to combine data from multiple application-layer processes into a single message, which can be transmitted over the network. At the receiving end, the transport-layer protocol must be able to separate the

multiplexed message back into the individual messages for delivery to the correct application-layer process.

For example, in TCP/IP, the transport layer uses source and destination port numbers to multiplex and demultiplex messages. Each application-layer process on a host is assigned a unique port number, and when a message is sent, the transport-layer protocol adds the source and destination port numbers to the message header. The receiving transport-layer protocol uses the destination port number to demultiplex the message and deliver it to the correct application-layer process.

## How demultiplexing works

Demultiplexing is the process of delivering incoming transport-layer segments to the correct socket. It is done using a combination of t**he destination IP address, destination port number, and the protocol field in the IP datagram.**

Based on the protocol number, the **IP datagram is handed to the appropriate transport-layer protocol module.** For example, if the protocol field indicates TCP, the IP datagram is passed to the TCP module.

Next, **the transport-layer protocol examines the destination port number in the segment header to identify the socket to which the segment should be delivered.** The transport layer maintains a demultiplexing table that maps each socket to its associated connection. Based on the destination port number, the transport layer uses this table to identify the socket to which the segment should be delivered. Finally, the segment is delivered to the appropriate socket for processing by the receiving application.

## Connectionless demultiplexing

In connectionless demultiplexing, the transport layer of the receiving host examines the destination address (IP address and port number) in **the header of each arriving segment to determine the socket to which the segment should be delivered. Each socket has a unique destination address associated with it, which allows the transport layer to identify the receiving process. The transport layer then directs the segment to the appropriate socket so that the receiving process can extract the data from the segment and pass it up to the application layer.**

This process is called demultiplexing because it involves separating the incoming segments and directing them to the correct socket on the receiving host. **It is called connectionless because there is no prior setup required between the sending and receiving hosts before data can be sent. Each segment is treated independently, and the receiving host determines where to deliver it based on the destination address contained in the segment header**.

**Connectionless demultiplexing is used by the User Datagram Protocol (UDP).**

**—Read Slide 11 as it is—**

## Connection-oriented demux

*Jotogula connection totogula socket.*

Each TCP socket is identified by a unique 4-tuple consisting of the source IP address, source port number, destination IP address, and destination port number. This 4-tuple is also known as a socket or an endpoint. A server host may support many simultaneous TCP sockets, and each socket provides a virtual circuit service to its application-layer protocol. When a segment arrives at a server host, the transport layer uses the destination port number to demultiplex the segment to the appropriate socket.

In non-persistent HTTP, also known as HTTP/1.0, **a new TCP connection is established for each HTTP request/response transaction. Therefore, a web server will have different sockets for each connecting client, as each client request will result in a new TCP connection and a new socket.** This approach can be inefficient for websites with a high volume of requests, as it requires a significant amount of time and resources to establish new connections for each request.

It is called connection-oriented demultiplexing because the transport layer protocol, in this case TCP, establishes a connection between the sender and receiver before any data is transmitted.

**Connection-oriented demux is used by the TCP.**

**—Read Slide 13,14 as it is—**

*In connection-oriented demux, the application server and threaded server both can handle multiple simultaneous client requests by creating separate threads to handle each request. However, the main difference between the two is in their purpose and design.*

*An application server is a software framework that provides a platform for developing, deploying, and running complex business applications. It often includes features like transaction management, security, and data access that are needed to build enterprise-level applications. In the context of connection-oriented demux, an application server may use multiple threads to handle client requests, but its main focus is on providing a high-level application platform rather than low-level network services.*

*On the other hand, a threaded server is designed specifically for handling network connections and providing low-level network services. It typically uses a single listening socket to accept incoming connections and creates a new thread to handle each client request. This approach is often used for simple network services like web servers, file servers, or chat servers, where the focus is on providing fast and efficient network communication rather than complex business logic.*

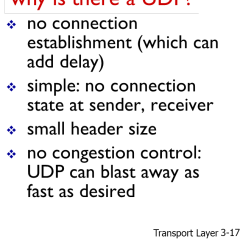
## *UDP: User Datagram Protocol [RFC 768]*

*Slide porlei hbe.*

UDP (User Datagram Protocol) is a transport protocol in the Internet protocol suite that provides a connectionless, unreliable communication service between processes running on end systems. I**t is often referred to as a "no frills" or "bare bones" protocol because it does not provide the same level of reliability and congestion control as TCP.**

One of the key features of UDP is that it is connectionless, meaning that there is no need to establish a connection before sending data. Instead, each packet of data is sent independently and may take a different path through the network. This makes UDP faster and more efficient than TCP in some situations, but also means that packets may be lost or arrive out of order.

Another important feature of UDP is that it does not provide any flow control or congestion control mechanisms. This means that UDP packets can be sent at any rate, regardless of network conditions or the ability of the receiver to handle the traffic. As a result, UDP can be more susceptible to network congestion and may cause performance problems in some situations.
->
UDP does not provide any reliable data transfer mechanism by default. However, it is possible to implement a reliable data transfer protocol over UDP in the application layer. One such protocol is the User Datagram Protocol with Reliable Delivery (UDPR).UDPR adds reliability to UDP by implementing mechanisms such as **error detection and correction, sequence numbering, acknowledgments, and retransmissions.**




## UDP: segment header

**—Read Slide 18,19 as it is—**

## Principles of reliable data transfer

Here are some principles of reliable data transfer:-

1. Acknowledgments
2. Checksum
3. Sequence numbers
4. Retransmission
5. Flow control
6. Error detection and correction

**—Read Slide 20-84 as it is—**

## rdt1.0: reliable transfer over a reliable channel

The rdt1.0 protocol is designed to provide reliable data transfer over a reliable channel, which means that the underlying communication channel is assumed to be error-free. In this protocol, the sender divides the data into packets, adds a sequence number to each packet, and sends the packets to the receiver.

The receiver acknowledges the receipt of each packet by sending an acknowledgment message to the sender. If the sender does not receive an acknowledgment message within a specified time period, it assumes that the packet was lost and retransmits the packet.

The rdt1.0 protocol provides reliable data transfer over a reliable channel by ensuring that all packets are delivered to the receiver in the correct order, without any loss or corruption. However, this protocol does not provide any error detection or correction mechanism to handle errors that may occur during transmission.

One limitation of the rdt1.0 protocol is that it assumes that the underlying channel is completely reliable, which is not a realistic assumption in most real-world scenarios. In practice, data communication channels are often subject to errors and other forms of interference that can lead to packet loss, corruption, or delay.

Therefore, more advanced protocols, such as rdt2.0, rdt2.1, and rdt3.0, have been developed to address these limitations and provide more robust and reliable data transfer over unreliable channels.

## rdt2.0: channel with bit errors

The rdt2.0 protocol is an improvement over rdt1.0 that is designed to provide reliable data transfer over a channel that may experience bit errors. This protocol uses checksums to detect errors in the received packets.

In rdt2.0, the sender divides the data into packets, adds a sequence number and checksum to each packet, and sends the packets to the receiver. The receiver checks the checksum of each packet to detect errors. If a packet is received with an incorrect checksum, the receiver discards the packet and sends a negative acknowledgment (NAK) message to the sender. The sender then retransmits the packet.

If a packet is received with a correct checksum, the receiver sends a positive acknowledgment (ACK) message to the sender, indicating that the packet was received successfully. If the sender does not receive an ACK message within a specified time period, it assumes that the packet was lost or corrupted and retransmits the packet.

The rdt2.0 protocol provides reliable data transfer over a channel with bit errors by detecting and correcting errors in the received packets. However, this protocol does not provide any mechanism to handle packet loss or delay, which can occur in a real-world communication channel. Therefore, further improvements such as rdt2.1 and rdt3.0 were introduced to handle these issues.

**new mechanisms in rdt2.0 (beyond rdt1.0):**
**error detection**
**feedback: control msgs (ACK,NAK) from receiver to sender**

## rdt3.0: channels with errors *and* loss

The rdt3.0 protocol is an improvement over rdt2.0 that is designed to provide reliable data transfer over a channel that may experience both errors and packet loss. In this protocol, the sender uses a timeout mechanism and selective repeat technique to handle lost packets.

In rdt3.0, the sender divides the data into packets, adds a sequence number and checksum to each packet, and sends the packets to the receiver. The receiver checks the checksum of each packet to detect errors. If a packet is received with an incorrect checksum, the receiver discards the packet and sends a negative acknowledgment (NAK) message to the sender. The sender then retransmits the packet.

If a packet is received with a correct checksum, the receiver sends a positive acknowledgment (ACK) message to the sender, indicating that the packet was received successfully. If the sender does not receive an ACK message within a specified time period, it assumes that the packet was lost and retransmits the packet.

To handle lost packets, the sender also maintains a timer for each packet it sends. If the sender does not receive an ACK message for a packet before the timer expires, it assumes that the packet was lost and retransmits the packet.

In addition, rdt3.0 uses the selective repeat technique to handle out-of-order packets. When a receiver receives a packet out of order, it buffers the packet until all previous packets have been received. Then, the receiver sends an ACK message for the out-of-order packet and delivers all the buffered packets in the correct order to the application layer.

The rdt3.0 protocol provides reliable data transfer over a channel that may experience both errors and packet loss by detecting and correcting errors and handling lost packets using a timeout mechanism and selective repeat technique. This protocol is more robust and reliable than rdt2.0 and rdt1.0, making it suitable for use in real-world communication channels.

## TCP: Overview  RFCs: 793,1122,1323, 2018, 2581

**Full duplex data refers** to a type of data transmission where communication can occur in both directions simultaneously. This means that data can flow in both directions in the same connection at the same time. Full duplex communication is commonly used in networking technologies such as Ethernet and TCP/IP.

In full duplex communication, each end of the connection can send and receive data independently without interference from the other end. This enables faster and more efficient communication, as data can be transmitted in both directions without having to wait for a response from the other end.

**The Maximum Segment Size (MSS)** is the largest amount of data that can be transmitted in a single TCP segment. It is typically determined by the Maximum Transmission Unit (MTU) of the underlying network protocol. MSS is negotiated during the TCP three-way handshake and is typically set to the maximum value that can be transmitted without fragmentation.

In full duplex communication, each end of the connection can transmit segments of data up to the negotiated MSS size, in both directions simultaneously. This allows for efficient use of network resources and can improve the overall performance of the communication. However, it is important to ensure that the amount of data being transmitted in both directions does not exceed the available bandwidth of the network, as this can lead to congestion and decreased performance.

## TCP seq. numbers, ACKs

In TCP (Transmission Control Protocol), sequence numbers and acknowledgements (ACKs) are used to provide reliable, ordered delivery of data between hosts on the Internet.

When a TCP connection is established between two hosts, each host generates an initial sequence number. Sequence numbers are used to identify individual segments of data that are transmitted over the connection. The sequence number for the first segment of data is typically a random number generated by the sending host, while subsequent sequence numbers are incremented by the number of bytes in the previous segment.

When a host receives a segment of data, it sends an acknowledgement message back to the sender to confirm receipt of the data. The acknowledgement message includes the next expected sequence number, indicating that the receiver is ready to receive the next segment of data. If a segment of data is lost or not received by the receiver, the receiver will not send an acknowledgement message for that segment, and the sender will retransmit the data.

In addition to acknowledging receipt of data, acknowledgements can also be used to provide flow control and congestion control. For example, a receiver can use the ACK message to indicate that its buffer is full and cannot receive any more data, or to inform the sender that the network is congested and that it should slow down the rate at which it is transmitting data.

## TCP round trip time, timeout

TCP Round Trip Time (RTT) is the time it takes for a data packet to travel from a sender to a receiver and back again. In TCP, RTT is measured by sending a packet and waiting for an acknowledgement (ACK) packet to be received back from the receiver. The time it takes for the ACK to be received is called the SampleRTT. TCP uses SampleRTT to dynamically adjust its transmission behavior to improve performance and reliability.

TCP Timeout is a mechanism used by TCP to detect and recover from lost packets. If a sender does not receive an ACK packet within a certain amount of time (called the Timeout value), it assumes that the packet was lost and retransmits the packet. The Timeout value is dynamically adjusted based on the SampleRTT. If the SampleRTT is large, the Timeout value will be increased to allow for longer delays, and if the SampleRTT is small, the Timeout value will be decreased to allow for faster retransmissions.
—>

When measuring SampleRTT, retransmissions are typically ignored because they do not represent a successful round trip. The SampleRTT should only be measured for packets that are successfully transmitted and acknowledged. If a packet is retransmitted, the SampleRTT for the retransmitted packet will be different from the original packet and could skew the timeout calculation.
The formula calculates DevRTT as a weighted average of the previous DevRTT and the absolute difference between the SampleRTT and the EstimatedRTT. The smoothing factor β determines the weight given to the previous DevRTT value versus the new deviation value calculated from the difference between the SampleRTT and the EstimatedRTT.

## TCP reliable data transfer

TCP (Transmission Control Protocol) creates a reliable data transfer (rdt) service on top of IP's (Internet Protocol) unreliable service by adding various features to ensure reliable delivery of data.

**One of these features is pipelining of segments, where multiple segments are sent by the sender before waiting for an acknowledgment from the receiver.** This allows for more efficient use of network bandwidth and can improve throughput.

**TCP also uses cumulative acknowledgments (ACKs), where the receiver acknowledges receipt of a segment by sending an ACK for the next expected sequence number.** This allows TCP to ensure that all segments are received in order and can be reassembled correctly.

To handle lost or delayed segments, TCP uses a single retransmission timer for each segment. If an ACK is not received for a particular segment within a certain amount of time, the sender retransmits the segment.

## TCP sender events:

Here are the main events performed by a TCP sender:

Data transmission: The sender first receives data from the application layer and segments the data into TCP segments. Each segment is assigned a sequence number, and the sender sends the segments to the receiver over the network.

Segment retransmission: If a segment is not acknowledged by the receiver within a specified time, the sender retransmits the segment. This is done using a single retransmission timer for each segment.

Congestion control: TCP sender monitors network congestion and dynamically adjusts the amount of data it sends based on the congestion level. It uses algorithms such as slow start, congestion avoidance, and fast retransmit to adjust its transmission behavior.

Acknowledgment processing: The sender processes acknowledgments received from the receiver. If an ACK is received for a particular segment, the sender updates its variables such as the expected sequence number, congestion window, and round trip time (RTT).

Window management: TCP sender maintains a sender window, which is the number of unacknowledged segments that can be sent. The sender adjusts the window size based on the receiver's advertised window and the congestion level of the network.

Timeout management: If a segment is not acknowledged within a certain time (determined by the retransmission timer), the sender retransmits the segment. The sender dynamically adjusts the timeout value based on the measured round trip time (RTT) and deviation of RTT.

## TCP fast retransmit

TCP (Transmission Control Protocol) fast retransmit is a mechanism used by the sender to quickly retransmit lost packets without waiting for a timeout to occur. The fast retransmit algorithm is triggered when the sender receives duplicate ACKs for the same segment.

When the sender receives duplicate ACKs, it assumes that the receiver has received all the packets up to the missing packet and is requesting the retransmission of the missing packet. Instead of waiting for the retransmission timer to expire, the sender immediately retransmits the missing packet. This can help reduce the time it takes to recover from packet loss and improve network performance.

Here's how the fast retransmit algorithm works in TCP:

1. The sender sends a segment to the receiver.
2. If the sender receives three duplicate ACKs for the same segment, it assumes that the segment has been lost and triggers the fast retransmit algorithm
3. The sender immediately retransmits the missing segment without waiting for the retransmission timer to expire
4. After the missing segment is retransmitted, the sender resumes normal transmission.
5. If the sender receives duplicate ACKs for the same segment again, it assumes that multiple packets have been lost and enters the fast recovery mode.

Fast retransmit can help improve the reliability of TCP by quickly recovering from lost packets and minimizing the impact of packet loss on network performance. However, it can also lead to unnecessary retransmissions if the duplicate ACKs are caused by out-of-order packets instead of lost packets. Therefore, TCP uses a combination of fast retransmit and other algorithms such as timeout-based retransmission to ensure reliable data transmission.

—>

**Yes, TCP (Transmission Control Protocol) uses duplicate ACKs to detect lost segments and trigger the fast retransmit algorithm.**

When the sender sends multiple segments back-to-back, it expects to receive an acknowledgment (ACK) for each segment. However, if a segment is lost in transit, the receiver will not send an ACK for that segment. Instead, the receiver will send an ACK for the next segment it receives. This means that the sender will not receive an ACK for the lost segment, and it will assume that the segment is still in transit.

To detect the lost segment, the receiver sends a duplicate ACK for the missing segment. The sender can use these duplicate ACKs to identify which segment is lost and trigger the fast retransmit algorithm to quickly retransmit the missing segment. If the sender receives three duplicate ACKs for the same segment, it assumes that the segment has been lost and triggers the fast retransmit algorithm.

Therefore, if a segment is lost, there will likely be many duplicate ACKs for the previous segments, as the receiver is acknowledging the segments it has received and requesting retransmission of the lost segment. TCP can use these duplicate ACKs to detect and recover from lost segments, ensuring reliable data transmission over the network.

## TCP flow control

TCP (Transmission Control Protocol) flow control is a mechanism used to manage the rate of data transmission between a sender and receiver. It ensures that the receiver can handle the data being transmitted by the sender and prevents the sender from overwhelming the receiver with too much data.

TCP flow control works by using a sliding window mechanism. The sender maintains a send window, which is the range of sequence numbers of the segments it can send. The receiver maintains a receive window, which is the range of sequence numbers of the segments it can receive.

When the sender sends a segment, it waits for an acknowledgment (ACK) from the receiver before sending the next segment. The receiver sends ACKs to the sender to acknowledge the receipt of the segments. If the receiver's buffer is full, it sends a window update to the sender, indicating the size of the receive window.

The sender adjusts the size of the send window based on the size of the receive window reported by the receiver. **If the size of the receive window is smaller than the size of the send window, the sender reduces the size of the send window to prevent the receiver's buffer from overflowing. If the size of the receive window is larger than the size of the send window, the sender increases the size of the send window to utilize the available bandwidth.**

TCP flow control ensures that the receiver can handle the rate of data transmission by controlling the amount of data sent by the sender. It prevents the sender from sending too much data too quickly, which can result in packet loss, retransmissions, and network congestion. It

also ensures efficient use of network resources by allowing the sender to adjust its transmission rate based on the receiver's capacity.

**The receiver advertises its free buffer space by including a receive window (rwnd) value in the TCP header of the receiver-to-sender segments. The rwnd value indicates the amount of buffer space available in the receiver's buffer, and it tells the sender how much data it can send before the receiver's buffer becomes full.**

The receiver's buffer size is set using socket options, and the typical default size is 4096 bytes. However, many operating systems can auto-adjust the receive buffer size based on network conditions and application needs.

**When the sender receives a segment, it checks the rwnd value in the TCP header to determine the amount of available buffer space in the receiver's buffer. If the sender's send window exceeds the size of the receiver's receive window, the sender will stop transmitting data until it receives a new acknowledgment with an updated rwnd value.**

**The receiver can use the rwnd value to control the rate of data transmission by adjusting the size of its buffer. If the receiver's buffer becomes full, it will advertise a smaller rwnd value to the sender, which will cause the sender to slow down its transmission rate. If the receiver's buffer has more free space, it will advertise a larger rwnd value to the sender, allowing it to send more data.**

## Connection Management

TCP (Transmission Control Protocol) connection management involves establishing, maintaining, and terminating a connection between two hosts. It is responsible for ensuring that data is transmitted reliably between the sender and receiver and that the connection is closed correctly when communication is complete.

TCP connection management involves three-way handshake to establish a connection, data transmission during the connection, and four-way handshake to terminate a connection.

Here is a brief overview of each step:

**Establishing a Connection: T**he three-way handshake process is used to establish a connection between the sender and receiver. The sender sends a SYN (synchronize) segment to the receiver to initiate the connection. The receiver responds with a SYN-ACK (synchronize-acknowledgment) segment, which acknowledges the SYN segment and sends a

SYN segment to the sender. The sender responds with an ACK (acknowledgment) segment, which acknowledges the SYN segment from the receiver. At this point, the connection is established, and data transmission can begin.

**Data Transmission:** During the connection, data is transmitted between the sender and receiver using TCP segments. TCP ensures that the data is transmitted reliably by using mechanisms such as flow control, error detection, and retransmission of lost or corrupted segments.

**Termination of Connection:** When communication is complete, the four-way handshake process is used to terminate the connection. The sender sends a FIN (finish) segment to the receiver, indicating that it has no more data to transmit. The receiver responds with an ACK segment, acknowledging the FIN segment. The receiver then sends its own FIN segment to the sender, indicating that it has no more data to transmit. The sender responds with an ACK segment, acknowledging the FIN segment from the receiver. At this point, the connection is terminated, and no more data can be transmitted.