COURSE NAME: *Computer Networking* | **PREPARED BY:** *Irfan Ferdous Siam* (Student Mentor) | **BATCH:** *231* | **DATE:** *12/07/2025*

What is the Internet?
The Internet is a global network that connects billions of computing and communication devices, such as computers, smartphones, and servers. It enables these devices to communicate and share information with each other using standardized protocols.

What is the Internet to You?
To me, the Internet is like a massive digital world that connects people, ideas, and information from every corner of the globe. It's not just a network—it's a platform for learning, communicating, working, and entertainment. Whether it's sending a message, streaming a video, attending online classes, or building a website, the Internet makes it all possible.

What is a Network?
A network is a system that connects two or more computing devices (like computers, smartphones, routers, etc.) to share resources, exchange data, and communicate with each other. These devices are linked using wired (like Ethernet) or wireless (like Wi-Fi) technologies.

In simple terms, a network allows devices to "talk" to each other—whether it's sharing a file, accessing the Internet, or printing from a shared printer.

What is a Host?
In a network, the connected devices—such as computers, smartphones, servers, and tablets—are called **hosts**. A host is any device that has a unique IP address and is capable of sending or receiving data over the network.

Each host plays a role in communication, such as requesting a web page, sending emails, or storing and serving files.

To connect and manage communication between host devices in a network, we use **networking devices** like **switches** and **routers**.

- A **switch** connects multiple host devices within the same local network (LAN). It helps them communicate by forwarding data only to the intended recipient device.
- A **router** connects different networks together—most commonly, your local network to the Internet. It decides the best path for data to travel from one network to another.

Together, switches and routers ensure that all connected devices can share information smoothly and efficiently.

Switches and Routers: The Backbone of the Internet
Switches and routers are often considered the **backbone of the Internet** because they are essential for directing and managing data traffic across networks.

- **Switches** keep local networks organized and efficient by managing data flow between devices within the same network.
- **Routers** handle communication between different networks, including connecting home or office networks to the broader Internet.

Without these devices, the Internet as we know it wouldn't function—they are the key to enabling global connectivity.

Communication Link
The **communication link** is the physical or wireless medium that connects one router to another, allowing data to travel across networks. These links form the paths over which data packets move from source to destination across the Internet.

Common types of communication links include:

- **Fiber-optic cables** – high-speed and long-distance transmission
- **Copper wires** – commonly used in traditional telephone lines (e.g., DSL)
- **Radio waves** – used in wireless communication (e.g., Wi-Fi, mobile networks)
- **Satellite links** – for remote or global coverage, especially in areas without cable infrastructure

These links work together to keep the Internet connected and functioning worldwide.

What is Bandwidth?
Bandwidth refers to the **maximum amount of data** that can be transmitted over a network connection in a given amount of time. It is usually measured in **bits per second (bps)**, such as Kbps (kilobits), Mbps (megabits), or Gbps (gigabits).

In simple terms, **bandwidth is like the width of a highway**—the wider it is, the more cars (data) can pass through at once.
Higher bandwidth means faster data transfer, smoother streaming, quicker downloads, and better overall network performance.

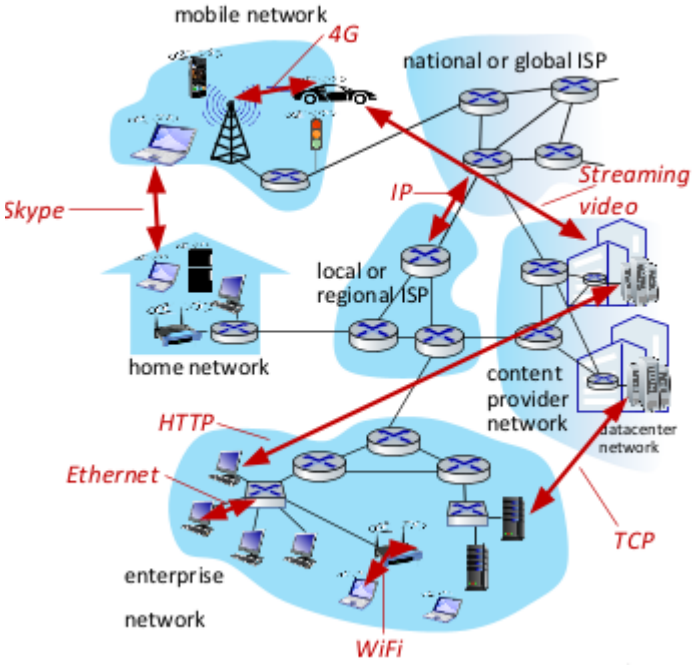
The **more bandwidth** a network has, the **higher the data rate** it can support.
This means more data can be sent or received in less time.

Think of it like this:
If bandwidth is the width of a road, then data rate is how many vehicles (data packets) can pass per second. A wider road (higher bandwidth) allows more cars (data) to travel at the same time, resulting in faster communication.

So, **increased bandwidth leads to increased data transfer speed**.

The **Internet** is often defined as a **"network of networks."**
It is a massive, interconnected system made up of **millions of smaller networks**—including personal, organizational, academic, and government networks—all linked together using standardized communication protocols.

This global structure allows devices from different parts of the world to communicate seamlessly, forming one unified system: the Internet.



This picture illustrates how different computer networks connect to share data. Here's a simple explanation:

- **Mobile network (4G)**: A car with a phone connects to a 4G tower, allowing internet access.
- **Home network**: Computers at home link through a router using Ethernet and HTTP protocols.
- **Enterprise network**: Office computers connect via Ethernet.
- **WiFi network**: Devices connect wirelessly using WiFi.
- **Local or regional ISP**: This internet service provider connects home, enterprise, and WiFi networks to the internet.
- **National or global ISP**: A larger ISP links the local ISP to the global internet.
- **Content provider network**: Services like streaming video send data through this network.
- **Datacenter network**: Large servers store and distribute data using TCP.
- **IP**: All networks use Internet Protocol (IP) to communicate.

The red arrows show data flowing between these networks, enabling communication from mobile devices to datacenters.

What are Edge Devices?
Edge devices are the devices that are located at the **edge of a network**, meaning they are the first point of contact between the user and the network. These devices collect, process, or transmit data to and from the network.

Examples of edge devices include:

- Smartphones
- Laptops
- Smart sensors
- Routers
- IoT devices (like smart thermostats or smartwatches)

Edge devices often interact with cloud systems or central servers but perform some tasks **locally** to reduce delay and save bandwidth. This concept is widely used in **edge computing** to improve speed and efficiency.

Applications of the Internet in Smart Systems
The Internet plays a key role in enabling various **smart management systems**, which help make our lives more efficient and connected. Some examples include:

- **Smart Utility Management**: Internet-connected devices monitor and control water, gas, and energy usage, helping to reduce waste and improve service efficiency.
- **Smart Electricity Management**: Systems like smart meters and grids track electricity usage in real-time, automate distribution, and reduce power outages.
- **Smart Environment Management**: Sensors connected to the Internet help monitor air quality, pollution levels, weather conditions, and natural disasters, allowing for better environmental protection and disaster response.

These smart solutions are part of the **Internet of Things (IoT)** revolution, where devices communicate and work intelligently through the Internet.

What is RFC?

RFC stands for **Request for Comments**. It is a **formal document** from the Internet Engineering Task Force (**IETF**) that describes the **standards, protocols, procedures, and technologies** used on the Internet.

- RFCs are used to **propose and define** how things like TCP, IP, HTTP, and many other Internet technologies should work.
- Each RFC is assigned a **unique number** and is **publicly available**.
- Some RFCs are **informational**, while others become **official Internet standards**.

👉 **Example**:

- **RFC 791** defines the **Internet Protocol (IP)**.
- **RFC 2616** defines **HTTP/1.1**.

Why is it important?
RFCs ensure that everyone around the world follows the **same rules**, allowing devices and systems to work together smoothly on the Internet.

What Does IETF Do?

The **Internet Engineering Task Force (IETF)** is the main organization responsible for the **development and standardization of Internet technologies and protocols**.

🔧 Key Responsibilities:

- **Developing Internet standards** (like TCP/IP, HTTP, DNS, etc.)
- Publishing **RFCs (Request for Comments)** to propose new ideas and protocols
- Ensuring **interoperability** across global Internet systems
- Promoting secure, scalable, and efficient Internet communication

🧠 How It Works:

- The IETF is made up of **volunteers**—engineers, researchers, and developers from around the world.
- It operates through **working groups** that focus on specific areas like routing, security, transport protocols, and more.
- Decisions are made through **open discussion and consensus**, not voting.

🌍 Why It Matters:

Thanks to the IETF, the Internet works **reliably and consistently** across different countries, networks, and devices.

The Evolution of the Internet

The **Internet was originally invented** as a tool for **data transfer** between researchers and military institutions (like in the ARPANET project). It was mainly used to **share information efficiently and securely**.

However, over time, the Internet has evolved into something much greater—
It is now the **backbone of a global digital industry**, powering:

- **E-commerce** (online shopping and business)
- **Entertainment** (YouTube, Netflix, gaming)
- **Communication** (email, social media, video calls)
- **Education** (online courses, virtual classrooms)

- **Finance** (online banking, cryptocurrency)
- **Healthcare, transportation, smart cities**, and much more

Today, **billions of dollars** and **millions of jobs** rely on the Internet, making it one of the most powerful and essential technologies in the world.

Why Do We Need Network Protocols?

To perform different operations on the Internet—such as browsing websites, sending emails, streaming videos, or downloading files—we need **network protocols**.

A **network protocol** is a set of **rules and standards** that define **how data is formatted, transmitted, and received** between devices on a network.

These protocols ensure that devices from different manufacturers and platforms can **communicate smoothly and reliably**.

🌐 Common Network Protocols:

- **HTTP/HTTPS** – for browsing web pages
- **TCP/IP** – for reliable data transmission across networks
- **FTP** – for transferring files
- **SMTP/IMAP** – for sending and receiving emails
- **DNS** – for translating domain names into IP addresses

In short, **network protocols are the language of the Internet**—without them, connected devices wouldn't understand each other.

In the **TCP/IP model**, every layer has its own **set of protocols and algorithms** that define how devices communicate over the Internet. Each layer is responsible for a specific part of the communication process, and together they ensure that data is transferred **efficiently and reliably** between devices.

Human Protocol for Communication
Human communication also follows a basic protocol. For example:

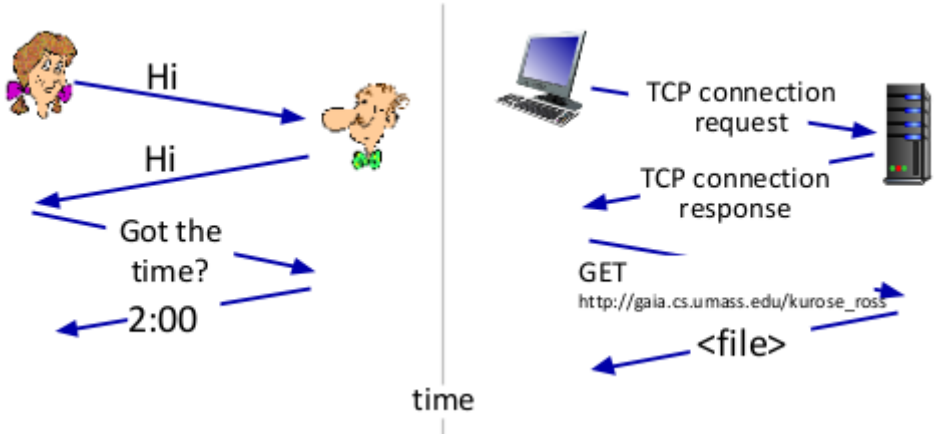
- We **start with a greeting** (e.g., "Hi")
- Then **express our intent** (e.g., "I have a question")
- Finally, we **ask or share information** (e.g., "What's the time?")

Just like in networking, these steps help make communication clear and organized.

Just like humans follow a set of steps to communicate clearly, **Internet protocols** are designed to help devices communicate in an organized and reliable way.

- **Human Protocol**: Greet → State intent → Share information
- **Internet Protocol**: Establish connection → Exchange data → End connection

These protocols ensure that data is **understood, delivered correctly**, and follows a proper sequence—just like a smooth conversation between two people.



When we exchange information over the Internet, we send different types of **messages** from one **host (device)** to another—such as web requests, emails, files, or video streams.

These messages follow a specific **format or structure**, which includes:

- **Source and destination addresses**
- **Type of data**
- **Sequence number**
- **Error-checking information**, etc.

To ensure the message:

- **Reaches the correct destination**
- **Is received completely and correctly**
- **Arrives on time and in order**

...we need a set of **rules and regulations**, which are known as **protocols**.

In short, **protocols define how messages should be structured, sent, received, and verified** across a network. They make sure that communication between hosts is reliable, accurate, and efficient.

Or What is a Protocol?

A **protocol** is a set of **rules and standards** that define **how data is exchanged** between devices in a network.

It ensures that both the sender and receiver understand the format, timing, and process of communication—so that data is transmitted **accurately and efficiently**.

What is Network Edge?

The **network edge** refers to the part of a network where **end devices (hosts)** like computers, smartphones, and IoT devices connect to the network. It's essentially the boundary between the user and the core network infrastructure.

At the network edge:

- Devices **generate or consume data**
- Data traffic **enters or leaves** the network
- Basic processing like **data filtering or security checks** can happen

Think of it as the "front door" where users and devices access the network before data travels deeper into the core network or the Internet.

The **network edge** includes not only end-user devices like computers and smartphones but also **servers, data centers, and cloud computing devices** that are located closer to the users rather than centralized deep inside the core network.

An **access network** is the **connection between a router (or other network gateway)** and the **end-user devices** such as computers, phones, or smart devices.

It is the part of the network that allows users to **access the Internet or other networks** through an Internet Service Provider (ISP).

The **Network Core** is the **central part of the Internet** that connects different networks together and is responsible for **high-speed, reliable data transmission** across long distances.

Unlike the **access network**, which connects individual users, the **network core** consists of **powerful routers and high-capacity communication links** that move data between regions, countries, and continents.

🔁 Key Functions:

- **Routing** data from source to destination
- **Forwarding** data packets quickly and efficiently
- Ensuring **fast, large-scale communication** between networks

🌐 Components of the Core:

- **Core Routers**
- **High-bandwidth fiber-optic links**
- **Backbone networks** owned by ISPs or telecom companies

🧠 Example:

When you send an email from Bangladesh to the USA, it travels through many routers and high-speed links in the **network core** before reaching its destination.

Network Core: The Internet Backbone

The **network core** is made up of a system of **interconnected routers** that form the **backbone of the Internet**. These routers are responsible for **forwarding data packets** across long distances—from one corner of the world to another.

The **routers in the network core** are connected using **very high-speed communication links**, most commonly **optical fiber cables**.

When a host wants to send data over the Internet, it follows a series of steps to break down the data and transmit it efficiently through the network.

Host Sending Function

1. **Takes Application Message**
   - The host receives a message from the application layer (e.g., a web request, email content, file data). Or The host takes the message generated by the application (e.g., email, web data).
2. **Breaks Message into Smaller Chunks**

   - The message is divided into smaller pieces called **packets**, each of fixed size **L bits**.
3. **Transmits Packets into the Access Network**
   - These packets are sent into the network through the **access link** (e.g., Ethernet, Wi-Fi).
4. **Transmission Rate (R)**
   - Packets are transmitted at the **link transmission rate R**, also known as:
     - **Link Capacity**
     - **Link Bandwidth**
     - Typically measured in **bits per second (bps)**

What is Data Rate?

Data rate refers to the **amount of data transmitted per second** over a communication link or network.

It is usually measured in **bits per second (bps)** and can also be expressed in:

- **Kbps** (kilobits per second)
- **Mbps** (megabits per second)
- **Gbps** (gigabits per second)
- A **packet** is a **small unit of data** that is sent across a network.
- When a large message (like a file, video, or webpage) needs to be sent from one device to another, it is **broken down into smaller chunks**, and each chunk is called a **packet**.

Packet size determines how the total data is divided, not how much data we need to send.

🔁 Example:

- If you need to send **10,000 bytes** of data and your **packet size** is **1,000 bytes**, then you'll need to send **10 packets**.
- So, **total data size ÷ packet size = number of packets to send**

How a Data Packet Travels Through the Network

A **data packet** is made up of **binary bits (0s and 1s)**—the actual digital information that computers understand.

1. The packet is created by the **sending host**, containing both **data** and **control information** (in the header).
2. These **bits (0s and 1s)** are **converted into electrical signals, light pulses, or radio waves** depending on the type of transmission medium (cable, fiber, or wireless).
3. The packet is then **inserted into the communication link** (e.g., Ethernet cable or Wi-Fi).
4. As it travels across the network, the packet **passes through multiple routers**.
   - Each router **reads the destination IP address** and **forwards** the packet toward the next best path.

5. Finally, the packet **reaches the destination host**, where it is **reassembled** (if it was split) and processed.

What is a Segment?

A **segment** is a unit of data created at the **transport layer** of the TCP/IP model.

When an application sends data (like a message or file), the **transport layer** (usually using **TCP** or **UDP**) breaks this data into smaller parts called **segments** before passing them to the network layer for delivery.

📜 A Segment Contains:

1. **Payload** – a piece of the actual message
2. **Header** – transport layer information, including:
    - **Source and destination port numbers**
    - **Sequence number** (for ordering)
    - **Checksum** (for error detection)

🚀 Difference from Packet:

- **Segment** → created at **transport layer**
- **Packet** → created at **network layer** (includes segment + IP header)

Maximum Transmission Unit (MTU) and Data Size

In many networks, especially Ethernet, the **maximum size of data that can be sent in a single packet** (called the **Maximum Transmission Unit** or **MTU**) is typically **1500 bytes**.

This means:

- A host can send up to **1500 bytes of data** in one packet without needing to split it further.
- If the data is larger than 1500 bytes, it must be **broken into smaller packets or segments**.

What is Packet Transmission Delay?

Packet transmission delay is the time it takes to **push all the bits of a packet onto the communication link**.

🔑 Key Points:

- It depends on two main factors:
  - **Packet size (L)** in bits
  - **Transmission rate (R)** of the link in bits per second (bps)

- **Formula:**

$$\text{Transmission Delay} = \frac{L}{R}$$

Where:

  - L = packet length in bits
  - R = link transmission rate in bps
- This delay **starts once the packet begins transmission** and finishes when the last bit is sent onto the link.

🧠 Example:

If a packet is **1,500 bytes** (= 12,000 bits) and the link speed is **1 Mbps (1,000,000 bps)**, then:

$$\text{Transmission Delay} = \frac{12,000}{1,000,000} = 0.012 \text{ seconds} = 12 \text{ milliseconds}$$

Transmission Delay

Transmission delay is the time taken by a network device (like a network card) to **push all the bits of a data packet onto the communication link (cable or wireless medium)**.

Transmission delay is the amount of time required to **send all the bits of a packet onto the network link** from the sender's device.

What is Propagation Delay?

Propagation delay is the time it takes for a signal (bit) to travel from the sender to the receiver **through the physical medium** (like a cable or wireless channel).

🔑 Key Points:

- It depends on:
  - **Distance (d)** between sender and receiver
  - **Propagation speed (s)** of the signal in the medium (usually close to the speed of light for fiber or copper cables)

- **Formula:**

$$\text{Propagation Delay} = \frac{d}{s}$$

Example:

- For a fiber optic cable where the signal speed is roughly 2×10^8 meters per second, and the distance is 1,000 km (1,000,000 meters):

$$\text{Propagation Delay} = \frac{1,000,000}{2 \times 10^8} = 0.005 \text{ seconds} = 5 \text{ milliseconds}$$

Difference from Transmission Delay:

- **Transmission delay** is how long it takes to put all bits on the wire.
- **Propagation delay** is how long it takes for the bits to travel through the wire.

Bit and Transmission Media

Bit

- A **bit** is the **smallest unit of data** in networking (0 or 1).
- It **propagates (travels)** from the **transmitter to the receiver** across a physical medium.

Physical Link

- The **physical link** is the **actual path** or medium through which bits are transmitted.
- It lies **between the sender (transmitter)** and the **receiver**.
- This can be a **wire, fiber optic cable, or air** (for wireless).

⚙️ Types of Transmission Media

1. Guided Media

- Bits travel through a **solid physical medium**.
- Examples:
  - **Copper cables** (e.g., Ethernet)
  - **Fiber optic cables** (very high speed and long distance)
  - **Coaxial cables** (used in cable TV and broadband)

2. Unguided Media

- Bits travel **freely through air or space**.
- Examples:
  - **Radio waves**
  - **Microwaves**
  - **Infrared**
- Used in technologies like **Wi-Fi, Bluetooth, Satellite communication**.

Wireless Links: Using the Radio Spectrum

📡 Links = Physical Media

In wireless communication, **radio waves** are used instead of physical wires to transmit signals.

🌐 Wireless Radio Communication

- **Signal travels** in different frequency **bands** of the **electromagnetic spectrum**
- No physical wire — transmission is **over the air**
- Typically **broadcast** and **half-duplex** (only one device can transmit at a time)

⚠️ Propagation Environment Effects

Wireless signals can be affected by the environment:

- **Reflection** – signal bounces off surfaces
- **Obstruction** – blocked by walls or objects
- **Interference/Noise** – from other electronic devices or overlapping signals

To overcome the environment effect we need to have options.

Type	Range	Speed	Purpose
Wi-Fi (Wireless LAN)	10–50 meters	10s to 100s of Mbps	Local area networking (home, office)
4G Cellular (Wide-area)	~10 kilometers	10s of Mbps	Mobile communication
Bluetooth	Very short (1–10 m)	Limited (1–3 Mbps)	Cable replacement between devices
Terrestrial Microwave	Point-to-point (line-of-sight)	~45 Mbps per channel	Long-distance data transmission
Satellite	Global (via orbit)	Up to 45 Mbps/channel	Long-distance, but with high delay (~270 ms end-to-end)

⚠️ Propagation Environment Effects

Wireless signals can be significantly affected by their surrounding environment. **Reflection** occurs when signals bounce off surfaces like walls or buildings, possibly causing signal distortion or delay. **Obstruction** happens when physical objects block the direct path between the sender and receiver, weakening or even completely disrupting the signal. **Interference and noise** come from other electronic devices or overlapping wireless signals, leading to reduced signal quality or data loss. To overcome these environmental effects, we need to implement various solutions such as **using stronger antennas, frequency selection, signal repeaters, error correction algorithms, and adaptive communication technologies** to ensure reliable wireless transmission.

What is Multihoming?

Multihoming is a network setup where a **host or network is connected to more than one Internet Service Provider (ISP)** or network path at the same time.

This provides:

- **Redundancy** – if one connection fails, the other keeps working
- **Load balancing** – traffic can be distributed for better performance
- **Improved reliability and availability**

🌐 The Network Edge – Simply Explained

The **network edge** is where all the devices we use (like computers, phones, laptops) are connected to the Internet. These devices are called **hosts** or **end systems**, and they run apps like **web browsers, email, or YouTube**.

👷 Two Main Ways Devices Communicate at the Edge:

1. Client-Server Model

- One device (called **client**) asks for something
- Another device (called **server**) is always on and gives the service
- Example:
    - When you open a **web browser**, it sends a request to a **web server** to load a website
    - **Email** also works like this

2. Peer-to-Peer (P2P) Model

- No fixed server
- All devices (peers) can share data directly with each other
- Example:
    - **BitTorrent**: people download and upload files directly from/to each other
    - **Skype**: makes direct voice/video calls without needing a central server

---

📌 In short:

- **Client-server** = one asks, one replies
- **P2P** = both can ask and reply

🔗 Connection-Oriented Service

In a **connection-oriented service**, the sender and receiver first establish a connection **before** sending any data. This process ensures **reliable data transfer**.

☑ Handshake Mechanism

- A **handshake** is a process where both devices agree to communicate.
- It helps **set up the connection**, verify availability, and ensure both sides are ready.
- It also allows **initial settings** like buffer size and sequence numbers to be exchanged.

📦 TCP for Reliable Transfer

- **TCP (Transmission Control Protocol)** is used in connection-oriented services.
- It ensures:
  - **Reliable delivery** (no missing or duplicate data)
  - **In-order packet delivery**
  - **Error detection and correction**

🧠 **Example**:
When you open a website, your browser uses TCP to **handshake** with the web server before starting the data exchange.

📦 TCP (Transmission Control Protocol)

- TCP is the **Internet's connection-oriented protocol**
- It uses a **three-way handshake** to:
  - Establish the connection
  - Ensure both ends are ready
  - Start **reliable data transfer**

🤝 TCP Three-Way Handshake

The **three-way handshake** is a process used by **TCP (Transmission Control Protocol)** to establish a **reliable connection** between two hosts before data transfer begins.

📜 Steps of the Three-Way Handshake:

1. **SYN (Synchronize)**
   - The **client** sends a segment with the **SYN** flag set to the **server**.
   - This is like saying:
     "Hello! I want to connect. Here's my sequence number."
2. **SYN-ACK (Synchronize + Acknowledge)**
   - The **server** replies with a **SYN-ACK**.
   - This means:
     "Hello back! I acknowledge your request and here's my sequence number."
3. **ACK (Acknowledge)**
   - The **client** sends an **ACK** to confirm the connection is established.
   - Like saying:
     "Okay! Let's start communicating."

📦 After this handshake:

- The **TCP connection is established**.
- Both sides are **ready to send and receive data reliably**.

📦 Features of TCP (Transmission Control Protocol)

Defined in RFC 793, TCP provides a **reliable, connection-oriented service** for data transmission over the Internet.

☑ Key Features of TCP:

1. **Reliable, In-Order Byte-Stream Delivery**
   - TCP ensures that all data is **delivered correctly** and in the **same order** it was sent.
   - Data is treated as a **continuous stream of bytes**, not separate packets.
2. **Loss Recovery (Acknowledgements & Retransmissions)**
   - If a packet is lost, the receiver doesn't acknowledge it.
   - The sender **detects the loss** and **resends** the missing data.
3. **Flow Control**
   - Prevents the sender from **overwhelming the receiver**.
   - Uses a **sliding window** mechanism to match the receiver's capacity.
4. **Congestion Control**
   - Prevents the **network from being overloaded**.
   - When congestion is detected, TCP **reduces its sending rate**.
   - This helps **protect the Internet from gridlock** or complete slowdown.

🧠 In Short:

TCP is designed to make sure:

- Data arrives **reliably**
- In the **correct order**
- Without **overloading the receiver or the network**

🔄 What is Flow Control?

Flow control is a technique used in networking to ensure that the **sender does not overwhelm the receiver** by sending data too fast.

☑ Why is Flow Control Needed?

- The sender might be faster than the receiver.
- If too much data is sent too quickly, the receiver's buffer may overflow, causing **data loss**.

📦 How TCP Handles Flow Control:

- **TCP uses a "sliding window" mechanism**.
- The **receiver tells the sender** how much data it can handle at a time (called the **window size**).
- The sender adjusts the amount of data it sends based on that window.

🧠 Simple Example:

Imagine a teacher speaking to a student:

- If the student says, "Wait, I need time to write," the teacher pauses.
- Once the student is ready, the teacher continues.

This is exactly how **flow control** works between sender and receiver.

🚦 What is Congestion Control?

Congestion control is a method used to prevent the **network from becoming overloaded** with too much data at once.

When **many senders transmit large amounts of data** through the network at the same time, routers and links can become congested — leading to:

- **Packet loss**
- **Delays**
- **Low network performance**

☑ TCP Congestion Control Features:

1. **Detection of Congestion:**
   - If packets are lost or delayed (no acknowledgment received), TCP assumes the network is congested.
2. **Adjusting Sending Rate:**
   - TCP **slows down the data transmission rate** when congestion is detected.

   - It **gradually increases the rate** again when the network seems to be free.
3. **Protecting the Network:**
   - This helps prevent the Internet from **gridlock** or total breakdown of communication.

🌱 Common TCP Congestion Control Techniques:

- **Slow Start:**
  - TCP starts sending a small amount of data and increases it gradually.
- **Congestion Avoidance:**
  - Once the rate reaches a certain point, it increases more slowly to avoid congestion.
- **Fast Retransmit & Fast Recovery:**
  - Quickly detect lost packets and recover without restarting the connection from scratch.

🧠 Simple Analogy:

Imagine many cars entering a small road. If all try to go fast at once, traffic jams happen. If cars slow down when it's crowded, the traffic flows better.
→ That's **congestion control** in a network!

🌐 Network Edge: Connectionless Service

Just like connection-oriented services, the goal here is still the same:

👉 **To transfer data between end systems** (like computers, phones, etc.)

📦 UDP – User Datagram Protocol

Defined in **RFC 768**, **UDP** is a **connectionless protocol**, meaning:

- No connection is set up before sending data
- Just sends the data and hopes it reaches the destination
- Often used when **speed is more important than reliability**

⚠️ Key Characteristics of UDP:

1. **Connectionless**
   - No handshake or setup between sender and receiver before sending data
2. **Unreliable Data Transfer**
   - Data might be **lost**, **duplicated**, or **received out of order**
   - No guarantee of delivery
3. **No Flow Control**
   - Sender can send data as fast as it wants, even if receiver is not ready
4. **No Congestion Control**
   - Doesn't care if the network is busy or overloaded
   - Keeps sending at a fixed rate

🧠 When to Use UDP?

- Real-time apps like:
  - **Voice calls (VoIP)**
  - **Online gaming**
  - **Streaming**

These apps prefer **speed** over **perfect accuracy**

🔬 UDP Relies on the IP Layer

UDP (User Datagram Protocol) is a **simple, lightweight protocol** that relies heavily on the **IP layer** to deliver data across the network.

☑ What Does This Mean?

- **UDP does not guarantee delivery** of data — it just **hands the data to the IP layer**.
- The **IP layer** is responsible for **routing the datagram** (packet) from the sender to the receiver.
- UDP **adds only minimal functionality** (like port numbers and a checksum), and leaves the rest (delivery, ordering, reliability) to the **application** or **IP layer**.

🧠 In Simple Terms:

UDP just passes the message to IP and says, "Here, deliver this if you can."
It **does not check** if it arrives or arrives correctly

🚚 What is Best-Effort Service (BES)?

Best-Effort Service means the network **does its best to deliver data**, but there is **no guarantee** that:

- The data will arrive
- It will arrive in the correct order
- Or it will arrive on time

📦 Where is Best-Effort Used?

- **UDP** (User Datagram Protocol) uses best-effort delivery
- Also applies to **standard Internet service** (like browsing or watching YouTube)

🧠 Simple Example:

Think of sending a regular letter by post (not a registered or tracked one).
The post office **does its best to deliver it**, but:

- It might get delayed
- It might get lost
- There's no way to know what happened

That's **Best-Effort Service** – fast and simple, but no promises!

🧠 The Network Core – Simply Explained

The **network core** is the **central part of the Internet**. It's made up of a **mesh (web) of interconnected routers** that help move data across the world.

🔁 Key Functions of the Network Core:

1. **Mesh of Routers**
    - Routers are connected in a complex web (mesh)
    - They work together to **forward data packets** toward their destination
2. **Packet Switching**
    - End systems (like computers or phones) **break large messages into small packets**
    - These **packets are sent one by one** across the network

3. **Forwarding Packets**
    - Each packet travels **router by router**
    - Routers **decide the next best hop** using routing algorithms
    - Packets follow a **path from the source to the destination**

📦 In Simple Terms:

The network core is like a **postal system** made up of routers. Each router is like a post office that helps **deliver pieces of your message (packets)** step-by-step until they reach the destination.

🚚 What is Forwarding?

Forwarding is the process of **moving a data packet from one router to the next** on its way to the destination.

🔁 How It Works:

- When a packet arrives at a router:
    0. The router **checks the destination address**
    1. It looks up the best **next hop** in its **forwarding table**
    2. It **sends the packet** to the next router or to the final device

---

🧠 In Simple Words:

Forwarding is like passing a letter to the next post office until it reaches the correct address.

🧭 What is Routing?

Routing is the process of **finding the best path** for data packets to travel **from the source to the destination** across the network.

🔁 What Does Routing Do?

- Routers use **routing algorithms** to:
    - Discover all possible paths
    - Select the **most efficient path** (based on distance, speed, cost, etc.)

- Update paths when the network changes (e.g., a link fails)

🧠 In Simple Words:

Routing is like Google Maps for the Internet.
It **plans the full journey** for your data, while **forwarding** is each step of the trip.

⚙️ Related to:

- **Forwarding** = moving packet to next router
- **Routing** = deciding **which path** the packet should follow

⚙️ What is the Work of a Forwarding Algorithm?

A **forwarding algorithm** decides **where to send a packet next** when it arrives at a router.

📜 Main Tasks:

1. **Look at the destination address** in the packet
2. **Search the forwarding table** (also called routing table)
3. **Choose the correct outgoing link** (next hop) based on the best match
4. **Send the packet** to the selected next router or the final device

🧭 What is the Work of a Routing Algorithm?

A **routing algorithm** is responsible for **finding the best paths** through the network so that data packets can travel from the source to the destination efficiently.

📜 Main Tasks:

1. **Discover the network topology** — find out which routers are connected and how
2. **Calculate the best path(s)** based on criteria like shortest distance, lowest cost, or fastest route
3. **Create and update routing tables** in each router with the best paths
4. **Adapt to network changes** — if a link goes down or a router fails, the algorithm recalculates new paths

5. • **Forwarding is local:**
   Each router **makes a decision for each incoming packet** about **where to send it next** based only on its **own forwarding table**.
   It doesn't know the entire path—just the **next hop**.
6. • **Routing is global:**
   Routing algorithms work across the **whole network** to **find the best paths** from any source to any destination.
   They use information about the **entire network topology** and update the forwarding tables accordingly.

Packet switching is a method used in networks where **data is broken into small pieces called packets** before being sent.

🔑 Key Points:

- Each packet contains part of the data plus information like **destination address**
- Packets are sent **independently** and can take **different paths** to the destination
- At the destination, packets are **reassembled** in the correct order to recreate the original data

🌐 Network Core: Packet Switching

- Each **end-to-end data stream** is divided into small units called **packets**.
- Packets from multiple users (e.g., User A and User B) **share the same network resources**.
- Each packet can use the **full bandwidth** of a link while it's being transmitted.
- Network resources are used **only when needed**, making the system efficient.

⚠️ Resource Contention and Congestion

- When the **total demand for resources exceeds the available capacity**, **resource contention** occurs.
- This causes **congestion**, where packets are **queued and must wait** before they can be sent.

🔁 Store-and-Forward Switching

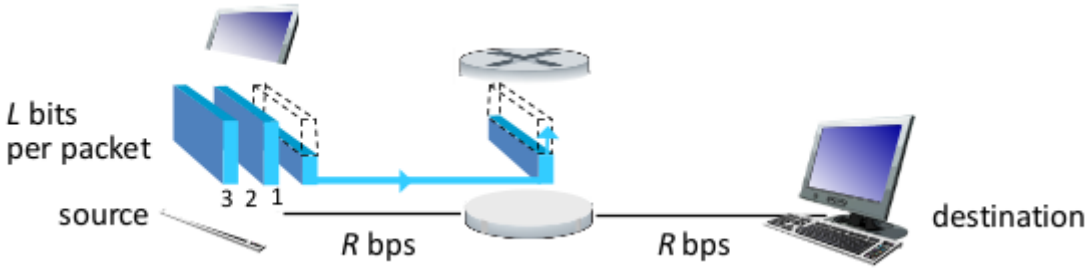
- Packets are sent **one hop at a time** through the network.
- Each **router (node) receives the entire packet before forwarding** it to the next hop.

🧠 Simple Summary:

Packet switching allows multiple users to efficiently share network resources by sending data in small packets that are forwarded hop-by-hop, but congestion can cause delays when the network is busy

In **store-and-forward**, a router **receives the entire packet first**, stores it temporarily, and only then **forwards the packet to the next router or destination**.

This means the packet must be completely received before sending it onward, which can cause a small delay at each hop



📦 Packet Switching: Store-and-Forward

- **Packet transmission delay:**
  It takes **L/R seconds** to transmit (push out) an **L-bit packet** into a link with bandwidth **R bps** (bits per second).
- **Store-and-forward:**
  The **entire packet must be received** at the router **before** it can be transmitted to the next link.

🔢 Numerical Example

- Packet size, L = 10 Kbits (10,000 bits)
- Link transmission rate, R = 100 Mbps (100,000,000 bits per second)

Transmission delay per hop:

$$\text{Delay} = \frac{L}{R} = \frac{10,000 \text{ bits}}{100,000,000 \text{ bits/s}} = 0.0001 \text{ seconds} = 0.1 \text{ milliseconds}$$

📋 Queueing

When packets arrive at a router **faster than the router can send them out**, they form a **queue (line)** waiting for transmission.

For example, if the router's output link speed is **R = 1.5 Mb/s**, but packets arrive at a faster rate (like **100 Mbps**), packets will **wait in a queue** before being sent.

This happens when the **arrival rate of packets exceeds the service rate** (transmission capacity) of the output link.

🧠 Simple Example:

Imagine a **narrow bridge** (output link) where cars (packets) arrive faster than they can cross. The cars will line up and wait — this is **queueing**.

📦 Buffer vs. Queue

- **Buffer:**
  A temporary storage space in a router or device used to **hold incoming packets** before they are processed or forwarded.
- **Queue:**
  The **ordered list** or **line of packets waiting** inside the buffer to be transmitted out on the next link.

🧠 In simple terms:

The **buffer is the storage area**, and the **queue is the waiting line** of packets inside that storage.

Alternative to Packet Switching: Circuit Switching

- In **circuit switching**, **end-to-end resources** (like channels or circuits) are **reserved and dedicated** for the entire duration of a call or connection between the source and destination.
- For example, if a link has **four circuits**, a call might use the **2nd circuit on one link** and the **1st circuit on another link**.
- **Dedicated resources** mean no sharing with other users during the call, providing **guaranteed performance**.
- However, if the call isn't using the circuit at a moment, that **circuit segment remains idle**—wasting resources.
- Circuit switching is **commonly used in traditional telephone networks**.

Circuit Switching: Frequency Division Multiplexing (FDM) and Time Division Multiplexing (TDM)

- **Frequency Division Multiplexing (FDM):**
  The physical link's frequency spectrum is divided into many narrow frequency bands.
  Each user (call) is assigned a unique frequency band and can transmit simultaneously but only within that band.
  This means each call transmits at the **maximum rate of its assigned narrow band**.
- **Time Division Multiplexing (TDM):**
  The physical link's time is divided into fixed-duration time slots.
  Each user (call) is allocated periodic time slots during which it can transmit using the **entire frequency band**.
  Users take turns to transmit in their assigned slots, each at the **maximum rate of the whole band but only during their time slot**.

🔁 Circuit Switching: FDM and TDM (Short)

- **FDM (Frequency Division Multiplexing):**
  The link's frequency is divided into separate bands. Each user gets a band and transmits **at the same time** but on different frequencies.
- **TDM (Time Division Multiplexing):**
  The link's time is divided into slots. Each user gets a time slot and transmits **one after another** using the full frequency during their slot.

🌐 Internet Structure: A "Network of Networks"

- The Internet is made up of **many smaller networks** interconnected together—hence called a **network of networks**.
- **Hosts connect** to the Internet through **Access ISPs** (Internet Service Providers).
- These **Access ISPs must be interconnected** so any two hosts anywhere can communicate.
- Directly connecting each Access ISP to every other ISP is **not scalable** because it requires **$O(N^2)$ connections** (too many).
- Instead, Access ISPs connect to **transit ISPs**, which provide a backbone for routing packets across networks.
- Economic agreements between **customer ISPs** and **provider ISPs** govern these connections.
- Because there isn't a single global ISP (competition exists), multiple **regional ISPs** and **content provider networks** (like Google, Microsoft, Akamai) operate their own networks.
- At the core of the Internet are a small number of **Tier-1 ISPs** (e.g., Level 3, Sprint, AT&T, NTT) with national and international coverage.
- **Content provider networks** often have private networks to connect their data centers, sometimes bypassing traditional Tier-1 or regional ISPs to improve performance.

🧠 Summary:

Layer	Description
Access ISPs	Connect end users (hosts)
Transit ISPs	Connect Access ISPs, provide backbone
Tier-1 ISPs	Large networks with wide coverage
Content Provider Nets	Private networks delivering content

🕓 How Do Packet Delay and Loss Occur?

1. **Packet Delay:**
   - When packets arrive at a router, they may need to **wait in a queue** before being transmitted.
   - If many packets arrive at once, the **queue gets longer**, increasing the **queueing delay**.
   - A packet also experiences **transmission delay** while it is being pushed onto the link.
2. **Packet Loss:**
   - Routers have **limited memory (buffers)** to hold queued packets.
   - If too many packets arrive and the buffer is full, **new packets are dropped** — this is called **packet loss**.

🧠 In Simple Terms:

If packets arrive faster than the router can send them out, they wait.
If there's no space left to wait (buffer is full), new packets are thrown away.

⏳ Queueing Delay

Queueing delay is the time a packet **spends waiting in the router's buffer** (queue) **before it is transmitted**.

🚀 Transmission Delay

Transmission delay is the time it takes to **push all the bits** of a packet **onto the link**

1. 🚀 Transmission Delay

Definition:
Transmission delay is the amount of time required to **push all the bits** of a packet onto the communication link. It depends on the **packet's size** and the **transmission rate** (bandwidth) of

the link. The larger the packet or the slower the link, the greater the delay.
Formula:

Transmission Delay=LR\text{Transmission Delay} = \frac{L}{R}Transmission Delay=RL

Where LLL is the packet length (in bits), and RRR is the transmission rate (in bits per second).

---

2. 🌐 Propagation Delay

Definition:
Propagation delay is the time it takes for a single bit to **physically travel** from the sender to the receiver over the transmission medium. It depends on the **distance** between the two points and the **propagation speed** of the signal in the medium (which is usually a fraction of the speed of light).
Formula:

Propagation Delay=ds\text{Propagation Delay} = \frac{d}{s}Propagation Delay=sd

Where ddd is the length of the physical link, and sss is the propagation speed of the medium.

---

3. ⏳ Queueing Delay

Definition:
Queueing delay is the time a packet spends **waiting in the queue** (buffer) of a router or switch before it can be transmitted. This delay occurs when packets arrive at a router **faster than they can be forwarded**, leading to congestion. Queueing delay can vary greatly depending on **traffic load**, and under high congestion, it may increase significantly.

---

4. 🧠 Nodal Processing Delay

Definition:
Nodal processing delay is the time a router takes to **process the packet header**, check for **bit-level errors**, and determine the appropriate **outgoing link**. This delay also includes time taken for security checks or forwarding decisions. It is usually very small (microseconds), but it depends on the **router's processing capability**.

⏱️ Four Sources of Packet Delay

When a packet travels from source to destination, it experiences **four types of delay** at each network node (router):

---

1. 🚀 Transmission Delay

- **Time to push the packet onto the link**
- Depends on packet size LLL and link bandwidth RRR
- **Formula:** LR\frac{L}{R}RL

---

2. 🌐 Propagation Delay

- **Time for a bit to travel across the physical link**
- Depends on **link length** and **propagation speed**
- **Formula:** ds\frac{d}{s}sd
  where d=d =d= distance, s=s =s= propagation speed

---

3. ⏳ Queueing Delay

- **Time the packet spends waiting** in the router's buffer
- Varies with **network congestion**
- Can be **very small or very large**

---

4. 🧠 Nodal Processing Delay

- **Time to check for bit errors** and determine where to forward the packet
- Usually **very small** (microseconds)

In network communication, the **end-to-end delay** increases mainly due to **transmission delay** and **queueing delay**, especially when **packet sizes are large** or **network congestion is high**.

⏱️ End-to-End Delay Across Multiple Routers

If a packet travels through **n routers**, the **total end-to-end delay** can be approximated as:

$$d_{\text{end-to-end}} = n \times (d_{\text{nodal}})$$

Where:

- nnn = number of routers (or links)
- dnodald_{\text{nodal}}dnodal = total delay at each node (router), which includes:
  - **Transmission delay**
  - **Propagation delay**
  - **Queueing delay**
  - **Processing delay**

So, total delay = **sum of all delays across each node**

🔁 Retransmission Delay

Retransmission delay is the extra time needed to **re-send a lost or corrupted packet**. It includes the waiting time before resending and the time to transmit the packet again.

Link-Dependent vs Router-Dependent

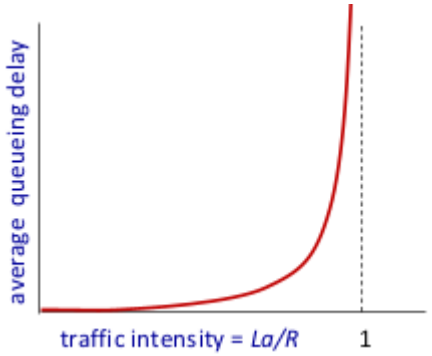
- **Link-dependent:** Delays caused by the physical link (e.g., propagation and transmission delay).
- **Router-dependent:** Delays caused inside routers (e.g., queueing and processing delay).

For efficient network performance, **link-dependent delays** (like transmission and propagation) and **router-dependent delays** (like processing and queueing) need to be **balanced**. Too much delay in either can slow down data transmission.

Example: Balancing Delays

- If the **internet cable (link)** is super fast but the **router** is slow or busy, your internet will still feel slow because the router takes time to handle each packet.
- If the **router** is fast but the **cable** is slow or very long (like satellite internet), data takes a long time to travel.

So, for fast internet, **both the cables and routers need to be fast and work well together**.



Calculate $\frac{L \times a}{R}$:

- If it's **close to 0**, queueing delay is **small**.
- If it's **close to 1**, queueing delay is **big**.
- If it's **more than 1**, packets come faster than the link can send, so delay becomes **infinite** (queue keeps growing).

🔁 Router Load and Utilization

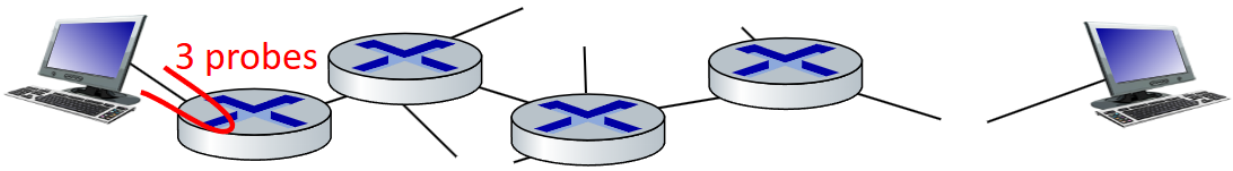
- To keep the network efficient, the **router's load** (traffic intensity) should be kept around **70-80%** of its capacity.
- At this level, the router is **well utilized** without causing excessive queueing delay or packet loss.
- Pushing the load above this can lead to **long delays and congestion**.

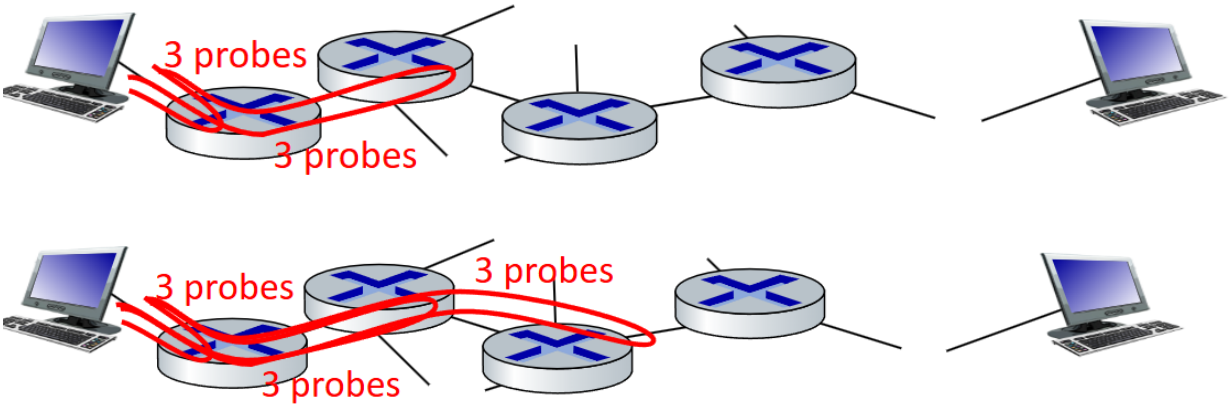
🧠 Simply put:

Keep the router busy but **not overloaded** to maintain good performance.

🌐 Real Internet Delays and Routes

- To understand real Internet delays and losses, we use a tool called **traceroute**.
- **Traceroute** measures the delay from your computer (source) to each router along the path to the destination.
- How it works:
  - For each router iii on the path, traceroute sends **3 probe packets** with a special setting (TTL = iii) that causes the packet to stop at router iii.
  - Router iii sends a reply back to the sender.
  - The sender measures the **round-trip time** between sending the probe and receiving the reply.
- This way, traceroute shows the **delay to each router** and the route packets take across the Internet.





📉 Packet Loss

- Routers have a **buffer (queue)** before each outgoing link to hold packets waiting to be sent.
- The buffer has **limited space**.
- When **packets arrive faster** than they can be sent, the buffer fills up.
- **New packets arriving when the buffer is full are dropped (lost)**.
- Lost packets may be **retransmitted** by the previous router, the source system, or sometimes may not be retransmitted at all.

📊 Throughput (Kurose and Ross)

Throughput is the **rate at which bits are transferred between sender and receiver** in a network, measured in bits per second (bps). It reflects the **actual achieved data transfer rate**, which can be affected by network congestion, errors, and protocol overhead. Throughput is often less than the raw link capacity (bandwidth) due to these real-world factors.

☑ Goodput

Goodput is the rate at which **useful application-level data** is successfully delivered to the receiver, excluding protocol overhead, retransmissions, and headers.

📌 In Simple Words:

Goodput = **Only the actual data** received correctly
(Not counting retransmissions or extra control bits)

🧠 Example:

If you send 12 packets but only 10 are useful data (2 were retransmissions),

- **Throughput** = all 12 packets
- **Goodput** = only the 10 correct packets

📊 Throughput vs Goodput

- **Throughput** is the total rate at which bits are transferred from sender to receiver, including all transmitted packets (original plus retransmissions).
- For example, if you send 10 data packets but 2 packets need to be retransmitted due to errors, the throughput counts all 12 packets sent.
- **Goodput** is the rate of **useful data** successfully received, excluding retransmissions. In this example, goodput is based on the original 10 packets only.

Our main goal in networking is to **maximize goodput** — the amount of **useful data** successfully delivered to the receiver.
This means we aim to reduce **retransmissions, delays, and losses** so that more of the data sent is actually useful.

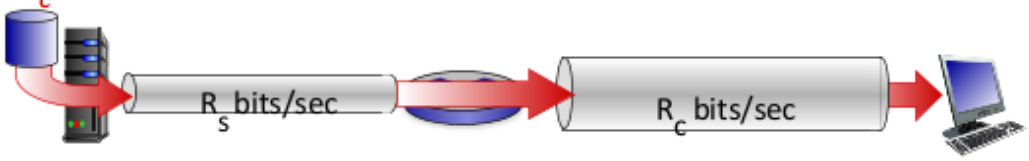
The **bottleneck link** is the **slowest link** along the path from sender to receiver.
It **limits the maximum throughput** of the entire connection — no matter how fast the other links are.

🧠 Simple Example:

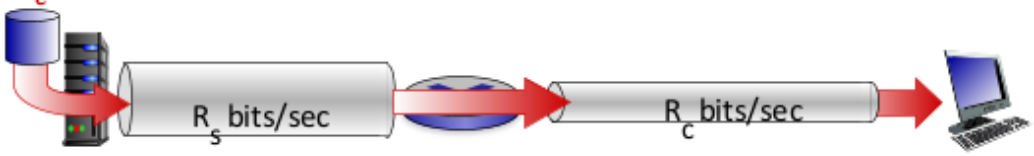
If data passes through 3 links and one of them supports only 1 Mbps while the others support 100 Mbps,
➡️ the **throughput will be limited to 1 Mbps**, because of the **bottleneck link**.

Throughput

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



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



bottleneck link

link on end-end path that constrains end-end throughput

📚 Layering Concept in Computer Networking

When we use computer networks, they perform many different tasks—like sending data, routing, error checking, etc.
Instead of doing everything in one place, these tasks are **divided into separate modules**, each handling a specific function.

This idea of **breaking network functions into layers** is called the **layering concept**.

Each layer is responsible for a specific job and **communicates with the layer above and below it**, making the network system easier to design, understand, and manage.

🧱 Protocol Layers & Reference Models

Computer networks are **complex systems** made up of many parts:

- **Hosts** (like computers, phones)
- **Routers and switches**
- **Wired/wireless links**
- **Applications** (like web, email)
- **Protocols** (like TCP, IP)
- **Hardware and software**

To **manage this complexity**, network functions are **divided into layers** — each layer handles a specific task.

This structure is called a **reference model**, like:

- **OSI Model** (7 layers)
- **TCP/IP Model** (5 layers)

Each layer talks to the layer above and below it, making it **easier to design, understand, and troubleshoot** networks.

📚 Why Layering?

(As described in Kurose & Ross)

Layering provides a **structured approach** for designing and discussing complex network systems.

✳️ Key Reasons for Using Layers:

1. **Explicit Structure:**
   Layering helps identify and organize the various components and their relationships in a network system.
2. **Modularization:**
   - Each layer performs a **specific function**.
   - Makes the system **easier to design, update, and maintain**.
   - Enhancements or changes to one layer's implementation can be made **without affecting other layers**.
3. **Layered Reference Model:**
   - Offers a **framework for discussion**, education, and protocol design.
   - Enables **standardization** across systems and vendors.
4. **Transparency Across Layers:**
   - A change in a layer's internal implementation remains **invisible** to the other layers.
   - 🔁 *Example:* A new method of performing routing (in the network layer) can be introduced without modifying the transport or application layer.

📚 Layered Reference Model (Simple)

A **Layered Reference Model** is like a **blueprint** for how networks work.

- It **breaks down** all the tasks of networking into **separate layers**.
- Each layer has a **specific job** and talks only to the layers right above and below it.

- This makes networks easier to **build, fix, and improve**.

🌐 Layered Internet Protocol Stack with Functions

1. **Application Layer**
   - Provides network services directly to user applications.
   - Examples: Web browsing, email, file transfer.
   - Works by defining protocols like HTTP, SMTP to communicate data between apps.
2. **Transport Layer**
   - Ensures **process-to-process communication** (from one program on a device to another).
   - Provides **reliable data transfer** (TCP) or **fast, connectionless transfer** (UDP).
   - Handles **flow control**, **error detection**, and **retransmission**.
3. **Network Layer**
   - Routes data packets from the sender to receiver across multiple networks.
   - Determines the **best path** using routing protocols.
   - Responsible for **logical addressing** (IP addresses).
4. **Link Layer**
   - Transfers data between two devices directly connected (neighbors) on the same physical link.
   - Deals with **framing**, **error detection/correction** on the link, and **MAC addressing**.
   - Examples: Ethernet, WiFi.
5. **Physical Layer**
   - Transmits raw bits over a physical medium (cables, radio waves).
   - Deals with **signal encoding**, **modulation**, and **hardware specifications**.

Or

🌐 Internet Protocol Layers and What They Do

1. **Application Layer**
   - Lets you use the internet apps you know (like web browsers, email).
   - It sends and receives messages using rules like HTTP or email protocols.
2. **Transport Layer**
   - Sends data from one program to another on different computers.
   - Makes sure data gets there correctly (TCP) or just quickly without checking (UDP).
3. **Network Layer**
   - Finds the best route for your data to travel across many networks.
   - Uses IP addresses to know where to send data.
4. **Link Layer**
   - Sends data between devices that are directly connected (like your computer and WiFi router).
   - Makes sure the data doesn't have errors on this short trip.

5. **Physical Layer**
    - Actually sends the data as electrical signals, radio waves, or light through cables or air

📦 What is Encapsulation?

- **Encapsulation** is the process where each layer of the network adds its own header (and sometimes footer) to the data before passing it down to the next layer.
- This helps each layer include the information needed for its specific function.
- When data moves down the layers, it gets wrapped like layers of an onion — each layer adding its own "wrapper."
- At the receiver, the process is reversed (called **decapsulation**) — each layer removes its header to understand the data.

🧠 Simple example:

When you send an email, the application layer creates the message, then the transport layer adds info like ports, the network layer adds IP addresses, and so on, wrapping the message with all necessary info before sending.

- A **switch** mainly works at **Layer 2 (Data Link Layer)** of the network.

    - It uses **MAC addresses** to forward data between devices on the same network.
    - It helps devices communicate within a local area network (LAN).

- **Routers primarily operate at Layer 3 (Network Layer)** of the Internet protocol stack.

    - They **route packets** based on IP addresses, finding the best path to send data from source to destination.
- Routers also handle functions related to:
    - **Layer 2 (Data Link Layer):** For sending and receiving frames on each link (e.g., Ethernet frames).
    - **Layer 1 (Physical Layer):** For transmitting bits over physical media.

So, routers work with:

- **Physical Layer (Layer 1):** For actual signal transmission
- **Data Link Layer (Layer 2):** For local link communication
- **Network Layer (Layer 3):** For routing and forwarding packets

Read the works of different layer from slide.

🔐 Network Security

- The **Internet was not originally designed with strong security** features.
- Early design assumed **trusted users and networks** — everyone was cooperative and safe.
- Today, the Internet connects millions of users, including **untrusted and malicious actors**, so security is critical.
- Because of this, **security features are now added at every layer** of the network protocol stack to protect data, users, and systems.
- Protocol designers are constantly working to **catch up and improve security** in response to new threats.

⚠️ What is a DoS Attack?

DoS (Denial of Service) attack is a malicious attempt to make a network service or website **unavailable** to its users.

How it works:

- The attacker **floods the target** (server, website, or network) with **excessive fake requests** or traffic.
- This overloads the system's resources (like bandwidth, CPU, memory).
- Legitimate users cannot access the service because it's overwhelmed.

Impact:

- Website or service becomes **slow or completely unreachable**.
- Can cause serious disruption to businesses or users.

⚠️ What is a DDoS Attack?

DDoS (Distributed Denial of Service) attack is like a DoS attack, but much bigger:

- Instead of one attacker, **many compromised computers (called a botnet)** send huge amounts of traffic to overwhelm the target.
- This makes it harder to stop because the attack comes from **many different sources**.

How DDoS works:

- Botnet machines send massive fake traffic or requests simultaneously.
- The target's network, servers, or services get overwhelmed and become unavailable to real users.

Common Protection Methods:

1. **Traffic Filtering:**
   - Block suspicious or abnormal traffic patterns.
2. **Rate Limiting:**
   - Limit the number of requests from a single source.
3. **Use of CDNs and Load Balancers:**
   - Distribute traffic across many servers to handle large volumes.
4. **Firewalls and Intrusion Detection Systems:**
   - Detect and block attack traffic.
5. **Blackholing or Sinkholing:**
   - Redirect attack traffic to a non-functional route to protect the network.

🔒 Types of Network Attacks

1. Active Attack

- Involves **changing or disrupting** system resources or operations.
- The attacker **modifies data**, injects false information, or interrupts normal communication.
- Examples: data tampering, impersonation, or denial-of-service attacks.

2. Passive Attack

- Involves **eavesdropping or monitoring** data without altering it.
- The attacker tries to **gain unauthorized access to information** but does not affect system resources or operations.
- Examples: wiretapping, traffic analysis.

🦹 Packet Sniffing

Packet sniffing is a type of **passive attack** where an attacker **captures and monitors data packets** traveling over a network.

- The attacker uses a tool called a **packet sniffer** or **network analyzer** to intercept network traffic.

- It allows the attacker to **read sensitive information** such as passwords, emails, or credit card numbers if the data is unencrypted.
- Packet sniffing does **not alter** the data; it only listens silently, making it hard to detect.

or

🦹 Packet Sniffing

- Packet sniffing is an example of a **passive attack**.
- It happens mostly on **broadcast networks** like shared Ethernet or wireless, where data is sent to multiple devices.
- A network interface in **promiscuous mode** can **read and record all packets** passing through, even if they are not addressed to it.
- This can include sensitive data like **passwords, messages, or other private info**.
- Tools like **Wireshark** are popular free packet sniffers used for analyzing network traffic (also used in labs).

🎭 IP Spoofing

- **IP Spoofing** is an **active attack** where the attacker **forges (fakes) the source IP address** in a packet.
- The attacker sends packets that appear to come from a trusted or different IP address to trick the receiver.
- This helps the attacker to **hide their identity**, **bypass security measures**, or **impersonate another device**.
- Commonly used in attacks like **Denial of Service (DoS)** and **man-in-the-middle attacks**.

or

🎭 IP Spoofing

- IP spoofing means **sending packets with a fake (false) source IP address**.
- For example, an attacker sends a packet that looks like it's coming from Host B, but actually it's from someone else.
- This tricks the receiver into thinking the packet is from a trusted source.