# Computer Network (BCS603)

EduShine CLASSES By- ARMAN ALI

## Unit – 4 Transport Layer

| | |
|---|---|
| **IV** | **Transport Layer:** Process-to-process delivery, Transport layer protocols (UDP and TCP), Multiplexing, Connection management, Flow control and retransmission, Window management, TCP Congestion control, Quality of service. |

Edushine Classes

# Computer Network (BCS603)

EduShine CLASSES By- ARMAN ALI

## ❖ Introduction to Transport layer :

The **Transport Layer** is the **fourth layer** in the OSI (Open Systems Interconnection) model.

🔑 **Think of it like a delivery service** that ensures your data goes from **one software (process)** on a device to the **right software on another device** — correctly, in order, and without loss.

Transport layer is responsible for delivery of entire message from one process running on source to another process running on destination. This is called process-to-process delivery.

**For example:**

When you watch a video on YouTube, your browser (application) is receiving data from YouTube's server. The transport layer makes sure that the video data reaches your browser smoothly.

## ✓ Main Responsibilities of the Transport Layer(Design Issue):- (IMP)

### 1. Process-to-Process Delivery

Sending data from one specific software on one device to another specific software on another device.

**🎁 Example:**

When you send a message through WhatsApp on your phone, it should go to **WhatsApp app** on your friend's phone — **not to YouTube or Facebook**!

This is done using **port numbers**. (Like room numbers in a hotel.)

### 2. Multiplexing and Demultiplexing

Many applications can use the internet at the same time (multiplexing).

The transport layer helps send and receive data from the correct app (demultiplexing).

**🎁 Example:**

You're using **Zoom**, **WhatsApp**, and **YouTube** at the same time. The transport layer keeps the data of each app separate so that video calls, chats, and videos don't mix.

### 3. Connection Control

Some protocols like TCP create a **connection before sending data**.

**Example:**

Before talking to someone, we say “Hello” first. Similarly, TCP does a **3-way handshake** to start a connection. UDP, however, sends without saying "Hello" (connectionless).

### 4. Reliable Delivery

Making sure that the data **reaches safely, completely, and in the right order**.

**Example:**

If you're downloading a file, the transport layer checks whether **all parts** of the file have arrived. If any part is missing, it requests it again.

This is done using **acknowledgments (ACKs)** and **retransmissions**.

### 5. Error Control

Checks for errors in the data and fixes them if possible.

**Example:**

If a video frame is corrupted during download, the transport layer can **detect it and request again**. It uses a method like a checksum to do this.

### 6. Flow Control

Prevents fast senders from **overloading slow receivers**.

**Example:**

You are sending messages to a friend, but your friend is replying slowly. You should wait. The transport layer ensures this wait happens using a mechanism called a **sliding window**.

### 7. Congestion Control

Prevents the **entire network from getting overloaded**.

**Example:**

If many people are watching YouTube at the same time, the transport layer may slow down some connections so the network doesn't crash.

# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI

## What is a Process?

A **process** is simply a **running program** or **application** on a computer.

**Example:**

- WhatsApp is one process
- YouTube is another process
- Gmail is also a process

## What is Process-to-Process Delivery?(Port to port)

When you send data (like a message, video, or file) from one device to another, the data **should not just reach the computer** — it should reach the **correct application (process)** on that computer.

✓ This is called **Process-to-Process Delivery**.

# Computer Network (BCS603)


🔷 **Real-Life Example:**

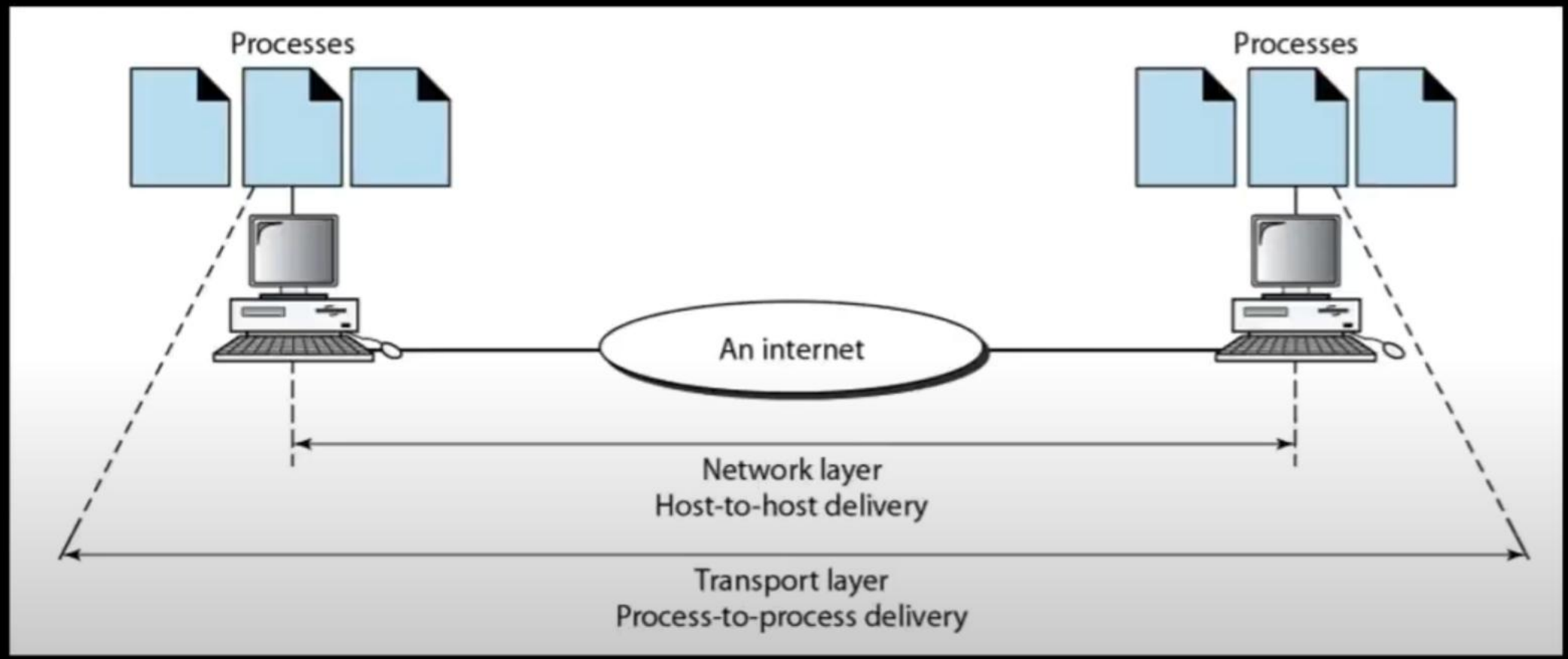
Imagine you are sending letters to a big building 🏢 (a computer). Inside the building, there are many people (processes).

You don't just send the letter to the building — you write the name of the person on the envelope (port number) so the letter reaches the **correct person**.

💡 Similarly, the **transport layer** uses **port numbers** to deliver data to the **correct application/process**.

"Transport layer is responsible for making sure that the message goes from the **right app on sender's computer** to the **right app on receiver's computer**. This is called process-to-process delivery"

# Computer Network (BCS603)



# Computer Network (BCS603)

EduShine CLASSES By- ARMAN ALI

## ❖ Services Provided by Transport Layer :

Two type of services provided by transport layer –

i. Connection Oriented Services
ii. Connection Less Services

### 🔗 1. What is Connection-Oriented Service?

Think of it like making a **phone call** 📞.

- First, you **establish a connection** (like calling and saying "hello").
- Then you **talk** (send and receive data).
- At the end, you **hang up** (close the connection).

### 🤔 Protocols that use this:

i. **TCP** (Transmission Control Protocol)
ii. **SCTP** (Stream Control Transmission Protocol)

# Computer Network (BCS603)


## 📨 2. What is Connectionless Service?

Think of it like **sending a letter** or **a message** without checking if the other person is there.

- You just **send the data**, no need to connect first.
- You don't know if it arrived, or if it came in order.

❌ No guarantee it reaches

✔ But it's **very fast** and has **less overhead**

⁉ **Protocol that uses this:**

- **UDP** (User Datagram Protocol)

# Computer Network (BCS603)

| Feature | Connection-Oriented Service | Connection-Less Service |
|---|---|---|
| **Setup** | Needs to **establish a connection** first | No need to establish a connection |
| **Reliability** | **Reliable** – guarantees delivery of data | **Unreliable** – does not guarantee delivery |
| **Order of data** | Data is received **in correct order** | Data may arrive **out of order** |
| **Example Protocol** | **TCP** (Transmission Control Protocol) | **UDP** (User Datagram Protocol) |
| **Data tracking** | Tracks sent and received data (acknowledgment) | No tracking of data |
| **Overhead (extra work)** | **More overhead** – uses more resources | **Less overhead** – simple and fast |
| **Speed** | Slower due to extra steps (handshaking, ACK, etc.) | Faster because it skips extra steps |
| **Use case examples** | File transfer, emails, web browsing | Live streaming, online gaming, video calls |

# Computer Network (BCS603)

EduShine CLASSES By:- ARMAN ALI

## Transport layer Protocol :




Now let discuss each protocol in detail →

# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI

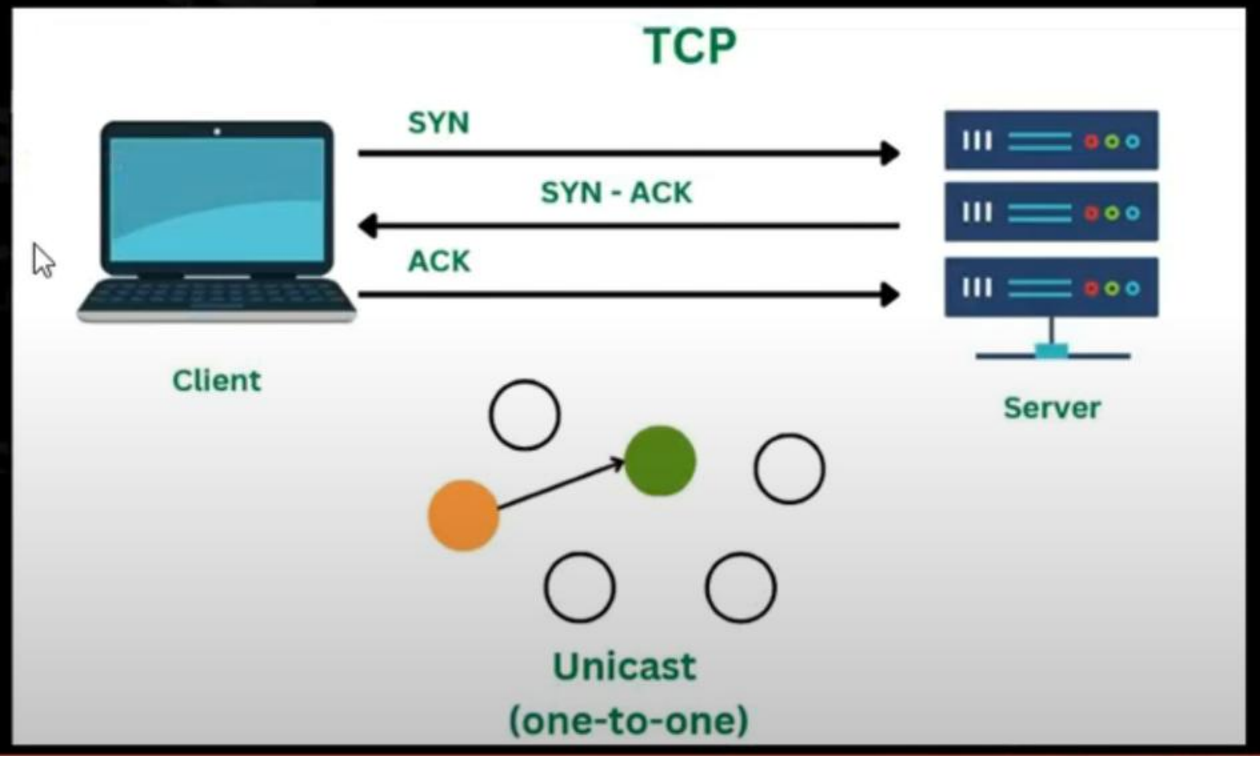
## 1. 💡 What is TCP?

**TCP (Transmission Control Protocol)** is a **connection-oriented** protocol.
This means before sending data, it **sets up a connection** between two devices.

☞ It makes sure the data is:

- Delivered **correctly**
- Delivered **in the correct order**
- Delivered **without loss**

TCP uses something called a **3-way handshake** to start the connection.



# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI

## 📄❓ Top Part – TCP 3-Way Handshake :

This is how two devices **create a secure connection** before sending data.

**Step 1: SYN**

- Client says: **"Hey, I want to connect!"**
- It sends a message called **SYN** (Synchronize).

**Step 2: SYN-ACK**

- Server replies: **"Okay, I got you. Let's connect!"**
- It sends back a message called **SYN-ACK** (Synchronize + Acknowledgment).

**Step 3: ACK**

- Client confirms: **"Thanks, I'm ready!"**
- Sends final message **ACK** (Acknowledgment).

✓ Now connection is **successfully established!**

This whole process ensures that **both devices are ready** and can **communicate reliably**.

# Computer Network (BCS603)


### Bottom Part – Unicast (One-to-One)

This part shows how **TCP sends data to one specific device only**, not to many.

- The **orange circle** is the sender.
- The **green circle** is the receiver.
- Arrows show that data is going from **one to one**.

**Example:** When you visit a website, your browser (client) talks only to **that specific server** — not to all servers on the internet.

# Computer Network (BCS603)

EduShine CLASSES By- ARMAN ALI

## Features of TCP (in easy words):

### (i) Numbering of Bytes

- It gives each byte of data a **number**.

This helps the receiver **know the order** of the data, even if it arrives **out of order**.

Like numbering boxes: Box 1, Box 2, Box 3.

### (ii) Stream Data Transfer

- It sends data as a **continuous stream**, not as fixed chunks.

This means TCP lets programs keep **sending and receiving smoothly** without worrying about how data is broken down internally.

### (iii) Reliability

- TCP makes sure that data **reaches the receiver**.
- If any data is **lost or damaged**, TCP **resends** it.

# Computer Network (BCS603)


**(iv) Flow Control**

- It **controls the speed** of data.

If the receiver is **slow**, TCP tells the sender to **slow down**, so no data is lost.

**(v) Full-Duplex**

- Data can go **both ways at the same time**.
- Just like a phone call — you can talk and listen at once.

**(vi) Multiplexing**

- It allows **multiple connections** from one device to another.

Like opening several tabs in your browser — all using the same network but for **different apps or websites**.

(vii) **Connection-Oriented Service :** discussed previous

# Computer Network (BCS603)


## ❖ TCP Header Format : (V.V.VIMP)

Imagine you're sending a file through a chat app:

The file is **broken into segments**.

Each segment has:

i. **Header** (info about where it should go, how to put the pieces back together),

ii. **Data** (a piece of your file).

These segments are reassembled at the receiver's side into the **original file**.

➢ Each TCP segment has **two parts**:

**Header Part**

– This part contains **control information** like source port, destination port, sequence number, etc.

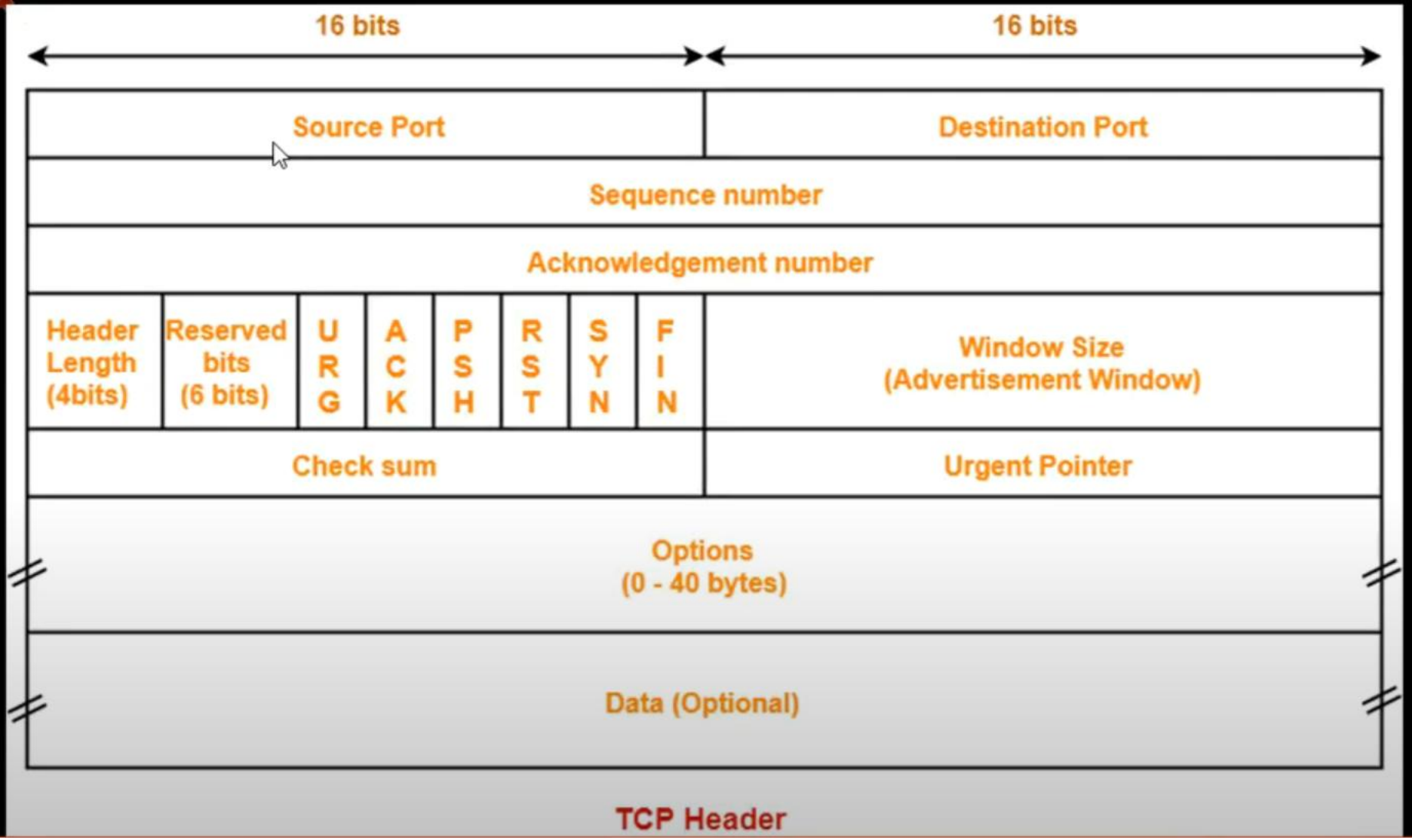
– Think of it like a **label on a parcel** with instructions for delivery.

**Data Part**

– This is the **actual content/message** you want to send.

– Like the **item inside the parcel**.

# Computer Network (BCS603)



## Fields of a TCP Header :

### 1. Source Port Number

This is the port number of the device that is **sending** the data.

***Example:*** Your browser might use port **3000** to send a request.

Think of ports like doors on a computer. The **source port** is the door number from where the data is coming,

### 2. Destination Port Number

This is the port number of the device that is **receiving** the data.

***Example:*** A web server receives data on port **80** or **443**.

Imagine **destination port** is the door number where the data is going.

### 3. Sequence Number:

It shows the **position of the first byte** of data in this segment. It helps in putting pieces of data in the correct order.

Imagine you're sending a big letter in 10 envelopes. Each one gets a number, so the person receiving can arrange them properly. The sequence number tells the receiver what position this piece of data is in.

### 4. Acknowledgment Number

This is used to **confirm** that data was received.

***Example:*** If a device gets data with sequence number 1001, it will send back an acknowledgment number of 1002.

This is used to say "Hey, I received your data successfully up to this number." It confirms that the other side got the data.

### 5. Data Offset (Header Length)

It tells how **long the header** is so the receiver knows where the actual data starts.

This tells the receiver where the TCP header ends and the real data (message) begins

### 6. Reserved

These are a few bits **kept for future use**. Normally set to 0.

These are just empty boxes saved for future features. They are not used currently and are se

### 7. Control Flags

These are **important signals** that help manage the connection:

- **SYN**: Start a connection
- **ACK**: Acknowledge received data
- **FIN**: Finish or close connection
- **RST**: Reset the connection
- **PSH**: Push data to the application
- **URG**: Data is urgent

### 8. Window Size

This is used for **flow control**. It tells how much data the sender can send **before waiting for an acknowledgment**.

This tells how much data the sender can send before needing an acknowledgment. It helps ir **flow control**, so the receiver doesn't get overloaded.

### 9. Checksum

A value used to **check for errors** in the data during transmission. If the data is changed or broken, the checksum will help detect it.

# Computer Network (BCS603)


### 10. Urgent Pointer

If the **URG flag** is set, this field tells where the **urgent data ends**.

If some data is marked as "urgent", this field tells where that urgent data ends.

### 11. Options

This is an **optional field** for extra settings like setting the maximum segment size.

Sometimes, extra information is added. Options field allows that.

### 12. Padding

This is just extra 0s added to make sure the header length is a **multiple of 4 bytes**.

# Computer Network (BCS603)

EduShine CLASSES By:- ARMAN ALI

**Q. The following is the dump of TCP header in hexadecimal format :(V.VV.IMP)**
**05320017 00000001 00000000 500207FF 00000000**

a. What is the sequence number
b. what is the destination port number
c. what is the acknowledgement number
d. what is the window size solve in easy way

Solution : Ye niche table yaad honi chahiye

| Field | Size |
|---|---|
| Source Port | 2 Bytes (4 hex digits) -> 16 bits |
| Destination Port | 2 Bytes (4 hex digits) -> 16 Bits |
| Sequence Number | 4 Bytes (8 hex digits) -> 32 Bits |
| Acknowledgment Number | 4 Bytes (8 hex digits) -> 32 Bits |
| Data Offset + Flags | 2 Bytes (4 hex digits) -> 16 Bits |
| Window Size | 2 Bytes (4 hex digits)-> 16 bits |
| (And more, but not needed here) | |

# Computer Network (BCS603)


Ye yaad rakhna → Har 2 character = 1 byte

**05320017 00000001 00000000 500207FF 00000000**

0532 → Source Port

0017 → Destination Port

00000001 → Sequence Number

00000000 → Acknowledgment Number

5002 → Data Offset + Flags

07FF → Window Size

✓ **a. What is the Sequence Number?**

Sequence number is: 00000001

☞ In decimal: **1**

✓ **b. What is the Destination Port Number?**

Destination port is: 0017

☞ Hex 0017 = **23** in decimal

### ✓ c. What is the Acknowledgment Number?

Acknowledgment number is: 00000000

☞ In decimal: **0**

### ✓ d. What is the Window Size?

Window size is: 07FF

☞ Hex 07FF = **2047** in decimal

# Computer Network (BCS603)


## ❖ Three way handshaking Protocol (Connection Establishment) : (V.V.VIMP)

The **3-way handshake** is a process used by the **TCP (Transmission Control Protocol)** to establish a **reliable connection** between a client and a server before starting data transmission.

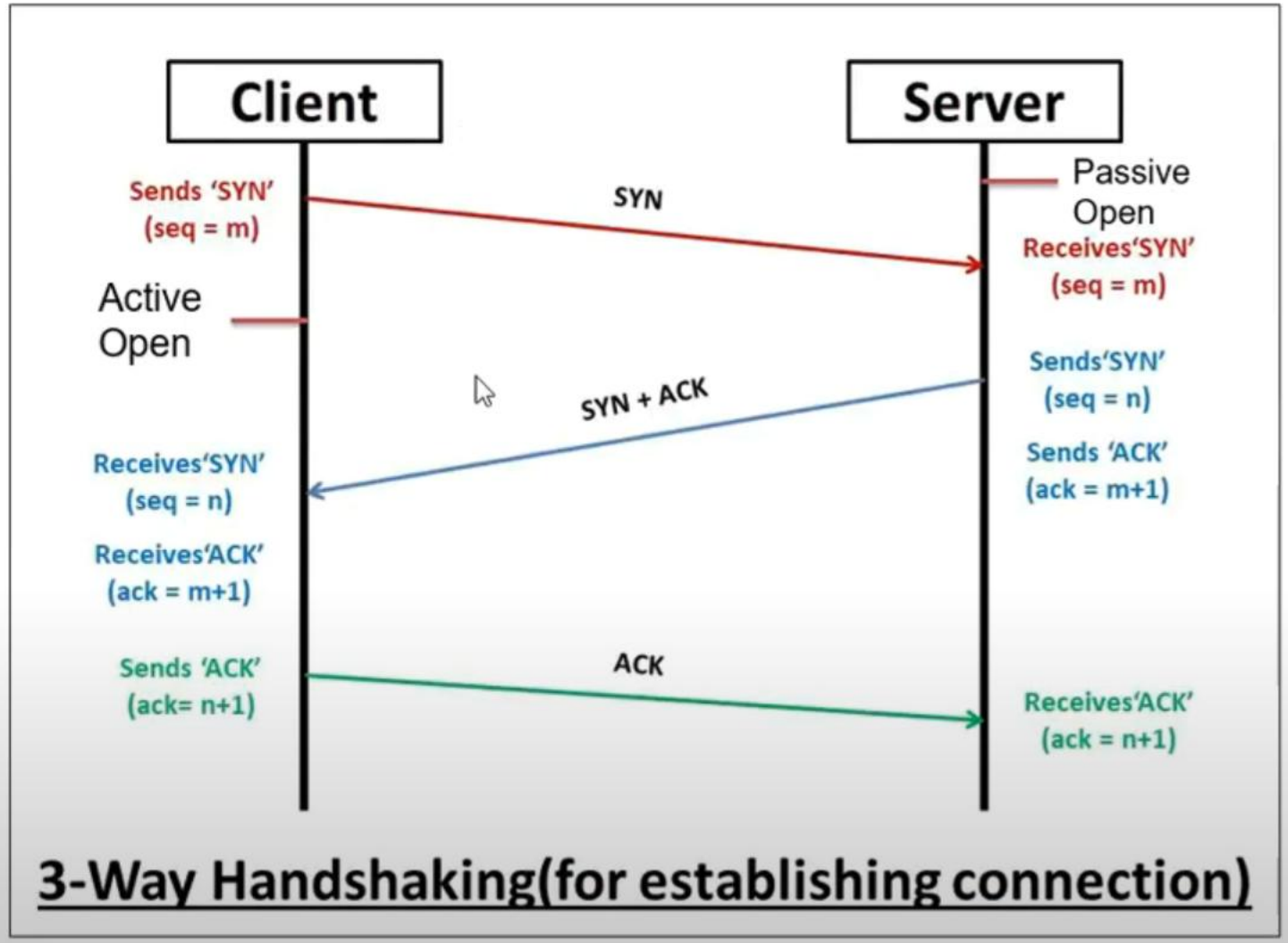
This handshake ensures that:

- Both client and server are ready.
- Sequence numbers are synchronized.
- The connection is reliable and secure.

### ✓ Step 1: Client sends SYN

- The client wants to start a connection.
- It sends a TCP segment with the **SYN (synchronize)** flag set.
- It also sends an **initial sequence number** (say, m).
- This means: “I want to connect. My sequence number is m.”

# Computer Network (BCS603)




3-Way Handshaking(for establishing connection)

### ✓ Step 2: Server sends SYN + ACK

- The server receives the SYN request.
- It replies with a TCP segment that has both **SYN and ACK flags** set.
- The server sends:
  - Its own sequence number (say, n)
  - Acknowledgement number = m + 1 (to confirm it received the client's SYN)
- This means: "Okay, I'm ready. I received your message. My sequence number is n."

### ✓ Step 3: Client sends ACK

- The client receives the server's SYN + ACK.
- It sends back an **ACK (acknowledgement)**.
- Acknowledgement number = n + 1 (to confirm it received the server's SYN)
- This means: "I received your message. Now we are connected."

# Computer Network (BCS603)


## 🌐 User Datagram Protocol (UDP) :

- UDP stands for **User Datagram Protocol**. It is a communication protocol used for **fast and simple data transfer** over the internet.
- UDP works at the **Transport Layer** (Layer 4 of the OSI model), just like TCP, but it is much **faster** and **lighter**.

## ✅ Key Features of UDP:

### 1. Connectionless:

- UDP does **not establish any connection** before sending data.
- It just sends data directly — like sending a message without checking if the receiver is ready.

### 2. No Acknowledgement:

- UDP does **not wait** for confirmation (ACK) from the receiver.
- No guarantee that the data will reach correctly.

**3. Faster Speed**:

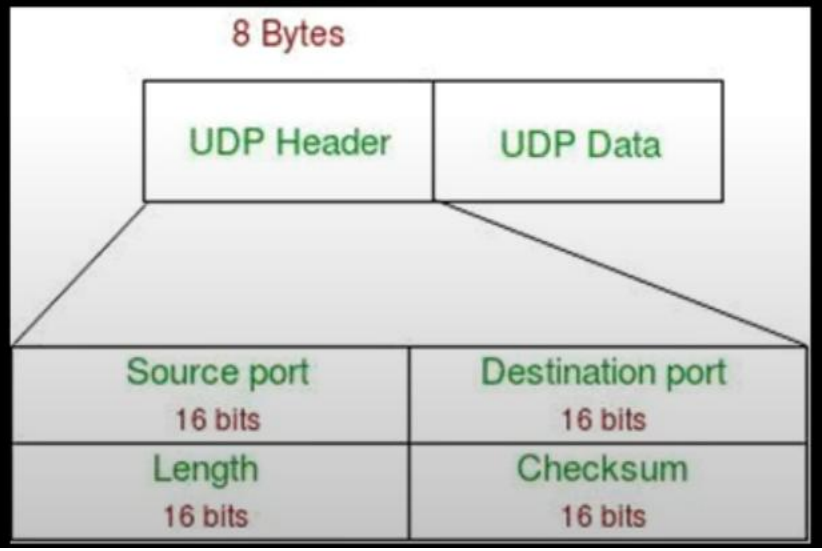
Because there is **no connection setup** and **no error checking**, UDP is very fast.

**4. Lightweight Protocol**:

UDP has **less overhead** (no extra steps or headers), making it efficient for time-sensitive applications.

❖ **UDP Header Format (Total 8 Bytes = 64 bits) : (V.V.VIMP)**

The UDP header is **very small and simple** — it is always **8 bytes** long, and contains **4 fields**, each 16 bits (2 bytes).



# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI

**1. Source Port (16 bits)**

- It tells **from which port** the data is coming.
- Used by the receiver to reply back to the correct sender.
- Example: If a message comes from port **12345**, this value will be **12345**.

**2. Destination Port (16 bits)**

- It tells **to which port** the data should go on the receiving side.
- Example: Port **53** is used for DNS.

**3. Length (16 bits)**

- It gives the **total length** of the UDP segment (header + data).
- Minimum value = **8 bytes** (if there is no data).
- For example, if the header is 8 bytes and data is 20 bytes → total = **28 bytes**.

## 4. Checksum (16 bits)

- It is used for **error checking**.
- The sender calculates a value using the data, and the receiver checks it to see if the data is correct or got corrupted.
- Optional in IPv4 but **mandatory in IPv6**.

### ✅ In Summary:

| Field Name | Size (bits) | Purpose |
|---|---|---|
| Source Port | 16 bits | Who sent the data |
| Destination Port | 16 bits | Where the data should go |
| Length | 16 bits | Total size of header + data |
| Checksum | 16 bits | For error detection |

## ❖ Application of UDP (2 Mark ) :

- **Live Streaming** – Used for video/audio streaming (e.g., YouTube Live)
- **Online Gaming** – Fast data transfer for real-time games (e.g., PUBG)
- **VoIP** – Voice calls over internet (e.g., WhatsApp, Skype)
- **DNS** – Resolving domain names to IP addresses
- **DHCP** – Assigning IP addresses in networks

# Computer Network (BCS603)

EduShine CLASSES By- ARMAN ALI

## ❖ Difference between TCP & UDP : (IMP)

| Feature | TCP (Transmission Control Protocol) | UDP (User Datagram Protocol) |
|---|---|---|
| **Type of Connection** | Connection-oriented (makes a connection first) | Connectionless (no setup needed) |
| **Speed** | Slower (more checks and control) | Faster (less checking) |
| **Reliability** | Reliable (guarantees delivery) | Not reliable (no guarantee) |
| **Error Checking** | Yes, with acknowledgment and retransmission | Yes, but no retransmission |
| **Order of Data** | Keeps data in correct order | May arrive out of order |
| **Use Case Examples** | Web browsing, email, file transfer | Online games, video streaming, VoIP |
| **Data Flow Control** | Yes (uses window size) | No flow control |
| **Overhead** | More (extra bits for control) | Less (lightweight protocol) |

# Computer Network (BCS603)

EduShine CLASSES By- ARMAN ALI

## ❖ Difference between TCP & UDP in context of Header format : (IMP)

| Feature / Field | TCP Header | UDP Header |
|---|---|---|
| Header Size | 20 to 60 bytes (because of options) | 8 bytes (fixed size) |
| Source Port | Yes | Yes |
| Destination Port | Yes | Yes |
| Sequence Number | ✓ Yes (for ordering data) | ✗ No |
| Acknowledgment Number | ✓ Yes (for confirming data receipt) | ✗ No |
| Header Length Field | ✓ Yes (Data Offset) | ✗ No (fixed size) |
| Flags | ✓ Yes (like SYN, ACK, FIN, etc.) | ✗ No |
| Window Size | ✓ Yes (for flow control) | ✗ No |
| Checksum | ✓ Yes | ✓ Yes |
| Urgent Pointer | ✓ Yes (if urgent flag is used) | ✗ No |
| Options Field | ✓ Yes (optional extra features) | ✗ No |
| Total Control | More control, reliable | Less control, faster |

## ❖ Multiplexing :

**Multiplexing** in the transport layer means:

➢ **Sending data from many applications through a single network connection.**

Imagine your phone is running:

- WhatsApp
- YouTube
- Instagram

All these apps want to send and receive data — **but your phone only has one IP address and one network connection (like Wi-Fi or mobile data).**

So how does your device know which data belongs to which app?

That's where **Multiplexing** comes in.

# Computer Network (BCS603)

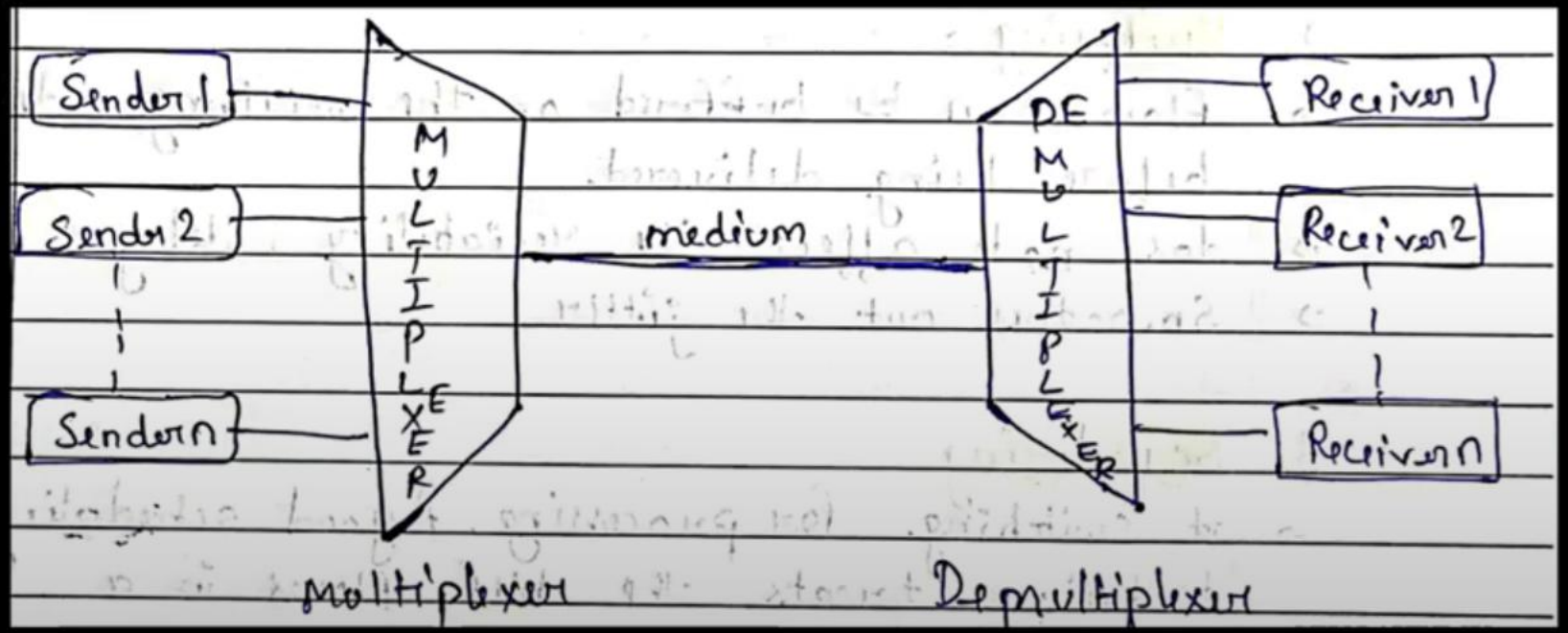
EduShine CLASSES
By:- ARMAN ALI

## 1. Multiplexing

- Happens at **Sender side**
- Takes data from multiple apps
- Adds headers (with **port numbers**)
- Sends it over the network

## 2. Demultiplexing

- Happens at **Receiver side**
- Looks at port number in the header
- Sends data to the correct app



# Computer Network (BCS603)


**Real-Life Example (To Understand Easily):**

Think of a **delivery truck** (the network). It's carrying **parcels** for different people (apps like WhatsApp, Instagram).

Each parcel has:

- The name of the sender (source port)
- The name of the receiver (destination port)

This way, when the truck reaches, each parcel goes to the right person.

In networking:

- The **delivery truck** is the transport layer
- The **parcels** are segments
- The **name tags** (ports) help deliver the data to the correct app

**Example:**

You're watching a YouTube video and chatting on WhatsApp at the same time.

- YouTube uses port **8080**
- WhatsApp uses port **5222**

When packets come in, your system checks:

- "Is this for port 8080?" → send to YouTube
- "Is this for port 5222?" → send to WhatsApp

That's **demultiplexing** in action.

## 💡 Why is Multiplexing Important?

- Lets you run many apps over one internet connection
- Keeps data organized and directed to the right place
- Avoids confusion between different services

# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI

## What is Flow Control?

**Flow Control** is like traffic control between **sender** and **receiver** in data communication.
It makes sure the **sender doesn't send data too fast**, and the **receiver doesn't get overloaded**.

## Real-Life Example:

Imagine you're pouring water (data) from a **jug (sender)** into a **glass (receiver)**.

- If you pour too fast, the glass will overflow (data loss).
- So you pour slowly or stop until the glass is ready again.

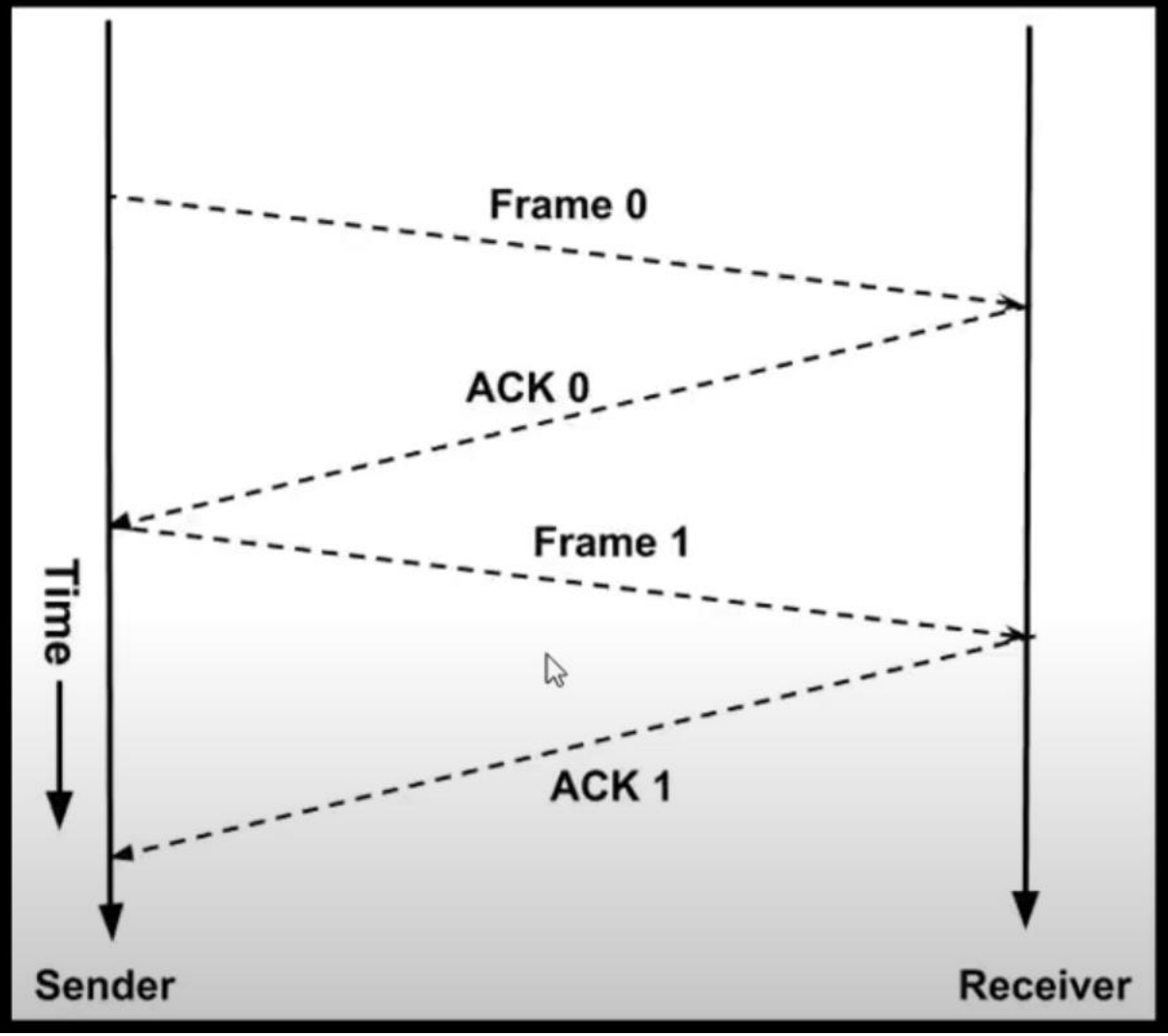
That's exactly what flow control does!

## Why is Flow Control Needed?

i. Sender is usually **faster** than receiver
ii. Receiver has **limited buffer (memory)**
iii. To **avoid data loss**
iv. To ensure **smooth communication**

# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI



# Computer Network (BCS603)


## 🔁 What is Retransmission?

**Retransmission** means:

- **Sending the data again** if it was **lost**, **damaged**, or **not acknowledged** by the receiver.
- It's a method used by **TCP (Transmission Control Protocol)** to ensure **reliable delivery** of data.

## 🤔 Real-Life Example:

Imagine you sent a WhatsApp message to a friend, but they didn't reply.
You might think: "Maybe they didn't get it."
So, you send the message again.
That's **retransmission** — repeating the message to make sure it's received.

# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI

## When Does Retransmission Happen?

In networking, retransmission happens if:

- A data packet is **lost**
- A packet is **damaged** (fails checksum)
- The sender **doesn't receive an acknowledgment (ACK)** in time

## How Does TCP Retransmit?

- **TCP assigns a sequence number** to each segment.
- **Receiver sends an acknowledgment (ACK)** for received data.
- If **no ACK comes within a timeout**, TCP assumes the segment is lost.
- The sender **retransmits** that segment.

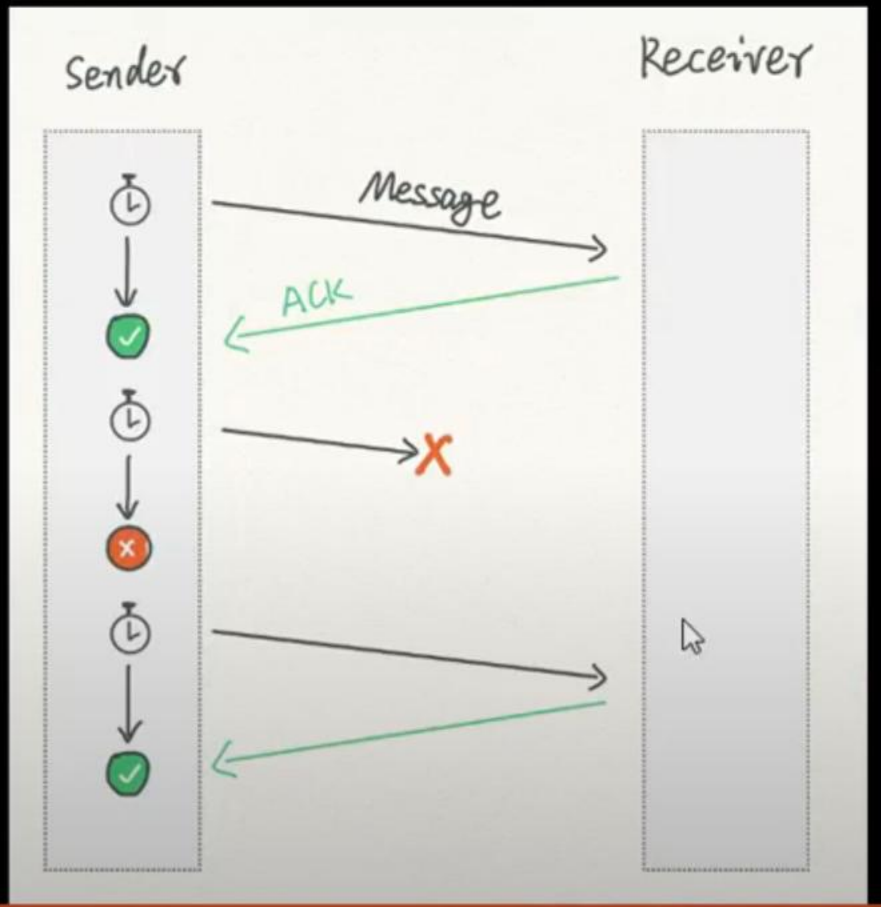
# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI

A **timer** is started when a segment is sent.
If the timer ends **before** receiving ACK → **retransmit** the segment.

## 🕮 Types of Retransmission Techniques

- ✓ **1. Go-Back-N(Already Covered)**
- ✓ **2. Selective Repeat(Already Covered)**



## ❖ TCP Window Management :

TCP Window Management is a method used to control how much data the sender can transmit before it must stop and wait for an acknowledgment from the receiver.

This helps in:

- Efficient use of the network
- Preventing the receiver from being overloaded
- Managing flow control

**Think of it like this:**

Imagine you're sending parcels to a friend.

They say:

- "I can only open **3 parcels at a time**. Don't send more until I tell you I'm ready again."

So you send 3 parcels → wait for a signal → send the next batch.

That's **window management**.

Now again if you want to explain more write sliding window protocol(Already Covered)

# Computer Network (BCS603)


## 🔁 How It Works (Step-by-Step):

- Sender sends multiple segments (as per window size)
- Receiver receives them and sends ACKs
- As ACKs arrive, the window "slides," allowing more data to be sent

## 💡 Example:

Let's say the receiver window size is 3 segments.

- Sender sends segment 1, 2, 3
- Receiver receives all → sends ACKs
- Window slides → sender can now send 4, 5, 6
- If segment 5 is lost → sender may retransmit 5, 6, etc.

# Computer Network (BCS603)

EduShine CLASSES By:- ARMAN ALI

## ⁉️ Why It Matters:

- ✓ Helps avoid congestion
- ✓ Adjusts to receiver's capacity
- ✓ Speeds up or slows down based on need
- ✓ Supports reliable and efficient communication

❖ **Congestion Control Techniques : Already Covered Same to same Unit 3 me hai.**

## 🔄 How Does TCP Handle Congestion?(IMP) :

TCP uses **4 phases** to control congestion:

# Computer Network (BCS603)

EduShine CLASSES
By- ARMAN ALI

## 1. Slow Start

- TCP starts sending data **slowly**.
- Begins with **1 segment**, then **doubles** the data each time (1 → 2 → 4 → 8...).
- This continues until it reaches a limit called the **threshold** or **congestion window (cwnd)**.

✓ Goal: Check how much the network can handle.

## 2. Congestion Avoidance

- When TCP reaches the threshold, it **stops doubling**.
- Now it **increases slowly** (1 segment at a time).
- This avoids sudden overload.

✓ Goal: Grow carefully without causing congestion.

### 3. Congestion Detection (Loss Occurs)

If packets are **lost** or **no acknowledgment (ACK)** comes, it means: → The network is **congested**.

TCP reacts by:

- **Reducing the congestion window** to half
- Setting a new threshold
- Going back to slow start or avoidance phase

✓ Goal: Back off when there is trouble.

### 4. Fast Retransmit & Fast Recovery

- If TCP gets **3 duplicate ACKs**, it guesses a packet was lost.
- It **quickly resends** the lost packet (**fast retransmit**)
- Instead of slow start, it does **fast recovery** to avoid restarting completely.

✓ Goal: React quickly without starting from scratch.

# Computer Network (BCS603)


**Simple Example:**

- Start: Send 1 packet
- Got ACK → Send 2 packets
- Got ACK → Send 4 packets
- Got ACK → Send 8 packets
- Then network drops a packet ✕ → congestion!

Now:

- TCP reduces sending speed
- Slowly increases again

# Computer Network (BCS603)



## ❖ Quality of Service (QoS) : (V.VIMP)

**Quality of Service (QoS)** means **how good and reliable the network service is** while sending and receiving data.

### ✓ Main Parameters of QoS:

i. **Bandwidth** – Maximum data that can be sent per second
ii. **Delay (Latency)** – Time taken for data to reach the destination
iii. **Jitter** – Variation in delay
iv. **Packet loss** – Number of lost or dropped packets

### 🔧 Techniques to Improve QoS :

Here are **common methods** used to improve Quality of Service:

### ✓ 1. Scheduling

- Traffic is arranged so that **important packets go first**.

**Example:** Give priority to video call data over file downloads.

### 🛠 Method:

- FIFO (First In First Out)
- Priority Queuing
- Weighted Fair Queuing (WFQ)

### ✓ 2. Traffic Shaping

- Controls the flow of outgoing data.
- It **delays packets** if too many are being sent at once.

🛠 **Example:** Leaky Bucket or Token Bucket algorithm

### ✓ 3. Resource Reservation

- Before sending, the sender **reserves bandwidth and resources** in the network.
- Ensures that when data is sent, the network is **ready and not congested**.

🛠 **Protocol:** RSVP (Resource Reservation Protocol)

### ✓ 4. Admission Control

- Before accepting a new traffic flow, the network checks if it can handle it.
- If not, it **rejects or delays** that flow to maintain QoS for existing traffic.

### ✓ 5. Packet Classification and Marking

- Packets are **marked** based on type (e.g., video, voice, file).
- Routers then treat each type according to its **priority**.

**Quality of Service (QoS)** is the ability of a network to provide better service to certain types of traffic. It ensures that important applications like voice, video, and gaming work smoothly without delay or data loss. To improve QoS, techniques like **scheduling, traffic shaping, resource reservation, admission control**, and **packet marking** are used. These help manage bandwidth, reduce delay, and give priority to critical data.

# Thank You...