

# Chapter 5 - Transport Layer (book - 231 - 255)

<b><u>5.The Transport Layer</u></b>			
<b>5.1</b>	Functions of Transport Layer	1	7 Hrs.
<b>5.2</b>	Elements of Transport Protocols: Addressing, Establishing and Releasing Connection, Flow Control & Buffering, Error Control, Multiplexing & Demultiplexing, Crash Recovery	1	
<b>5.3</b>	User Datagram Protocol(UDP):User Datagram, UDP Operations, Uses of UDP, RPC	1	
<b>5.4</b>	Principles of Reliable Data Transfer:Building a Reliable Data Transfer Protocol, Pipelined Reliable Data Transfer Protocol, Go Back-N(GBN), Selective Repeat(SR)	2	
<b>5.5</b>	Transmission Control Protocol(TCP): TCP Services, TCP Features, TCP Segment Header	1	
<b>5.6</b>	Principle of Congestion Control	1	

## 5.1 Transport Layer

#Functions of transport layer

5.2 Elements of Transport Protocols: **Addressing**, Establishing and Releasing Connection, Flow Control & Buffering, **Error Control**, **Multiplexing** & Demultiplexing, Crash Recovery

#Transport Layer services

End-to-end delivery:

**Reliable delivery:**

Flow Control

Addressing

#Establishing and Releasing Connection

#Flow Control & Buffering

**1.Flow Control:**

**2.Buffering:**

#Multiplexing and De-Multiplexing

#Crash Recover

5.3 User Datagram Protocol(UDP):User Datagram, UDP Operations, Uses of UDP, RPC

#Concept

#Feature / Use of UDP

#UDP Header

#UDP Application

#Difference between TCP and UDP

#RPC (Remote Procedure Call)

##RPC Frame Format

###Network File Sharing

###SUN RPC

###Serialization

###Refinements

5.4 Principles of Reliable DataTransfer: Building a Reliable DataTransfer Protocol, Pipelined Reliable Data Transfer Protocol, Go Back-N(GBN), Selective Repeat(SR)

#Difference between RDT and UDT

#Building Reliable Data Transfer Protocol

#Pipelined Reliable Data Transfer Protocols

#Two Generic Forms of Pipelined protocols

## 5.5 Transmission Control Protocol(TCP): TCP Services, TCP Features, TCP Segment Header

#Feature

#Difference between TCP and UDP

#TCP Header

### 5.1 Principle of Congestion Control

#Congestion

##Slow Start Algorithm (exponential)

##Congestion Avoidance (add)

##Congestion Detection (time-out, 3ack)

#Quality of Service (QoS).

#Leaky and Token Bucket

**Read Me First (3times) Assumes Basic Key Terms while writing your unit 5 answer.**

bandwidth-delay product	client-server paradigm	congestion
congestion control	demultiplexing	ephemeral port number
finite state machine (FSM)	Go-Back-N protocol (GBN)	multiplexing
piggybacking	pipelining	port number
process-to-process communication	Selective-Repeat (SR) protocol	sequence number
sliding window	socket address	Stop-and-Wait protocol
well-known port number	congestion-avoidance	cookie
deadlock	denial of service attack	fast-recovery
fast retransmission	fragmentation	half-close
initial sequence number (ISN)	keepalive timer	persistence timer
primary address	retransmission time-out (RTO)	round-trip time (RTT)
silly window syndrome	slow-start algorithm	socket address
Stream Control Transmission Protocol	(SCTP)	stream identifier (SI)
stream sequence number (SSN)	SYN flooding attack	three-way handshaking
Transmission Control Protocol (TCP)	transmission sequence number (TSN)	user datagram
User Datagram Protocol (UDP)		

## 5.1 Transport Layer

- To provide end to end communication between application (port)
- Ensure the data is delivered reliably and correct data in order
- It decides if data transmission should be taken place in sing path or parallel path.
- Standard protocol
  - TCP ( Transmission control protocol )
  - UDP ( User datagram protocol )
  - DCCP ( Data congestion control protocol )

### #Functions of transport layer

#### ▼ Segmentation:

Large data packets are broken down into smaller, more manageable segments for transmission across the network. This improves efficiency by allowing for smaller packets to be transmitted and processed faster.

#### ▼ Error control:

The transport layer adds error-checking mechanisms to the data packets to detect and correct errors that may occur during transmission. This ensures that the data arrives at the receiver intact.

**Q. Why data link's error control is not enough?**

The data link layer checks for the error between each network. If an error is introduced inside one of the routers, then this error will not be caught by the data link layer. It only detects those errors that have been introduced between the beginning and end of the link. Therefore, the transport layer performs the checking for the errors end-to-end to ensure that the packet has arrived correctly.

▼ **Flow control:**

The transport layer manages the flow of data between the sender and receiver to prevent the sender from overwhelming the receiver with data packets faster than it can process them.

▼ **Multiplexing:**

Multiplexing allows multiple applications on a single device to share a single network connection. The transport layer keeps track of which data segments belong to which application by using port numbers. ( Facebook, Instagram, Youtube ) → same network

▼ **Connection Control**

▼ **Connectionless Transport Layer**

In this type of transmission, the receiver does not send an acknowledgment to the sender about the receipt of a packet. This is a faster communication technique

| Data pathauxha, receiver le ack pathaudaina → faster

▼ **Connection Oriented Transport Layer**

This Transport Layer creates a connection with the Transport Layer at the destination machine before transmitting the packets to the destination.

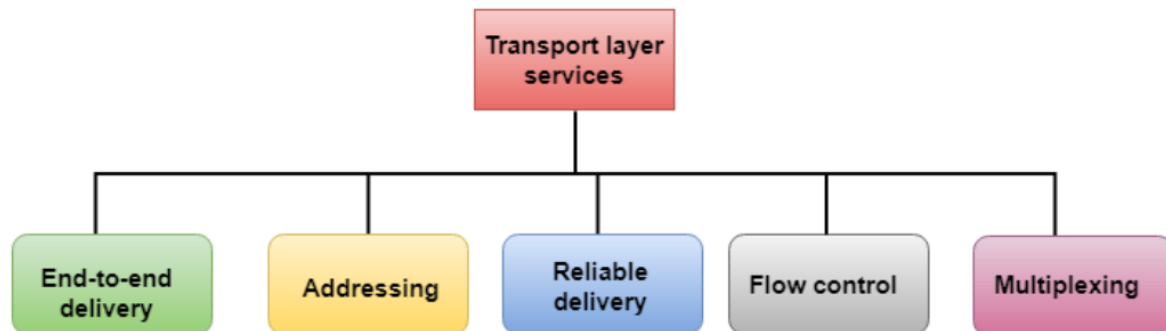
To Create a connection following three steps are possible:

- o Connection establishment
- o Data transfer
- o Connection termination

| Connection build garxha destination sanga ani data send garxha and terminate garxha

## 5.2 Elements of Transport Protocols: Addressing, Establishing and Releasing Connection, Flow Control & Buffering, Error Control, Multiplexing & Demultiplexing, Crash Recovery

### #Transport Layer services



#### ▼ End-to-end delivery:

The transport layer transmits the entire message to the destination. Therefore, it ensures the end-to-end delivery of an entire message from a source to the destination.

#### ▼ Reliable delivery:

The transport layer provides reliability services by retransmitting the lost and damaged packets.

**The reliable delivery has four aspects:**

##### ▼ Error Control

Kunai ne transmission error free hudaina, tei vayera transport layer le ne error free rakhxha. Data link layer ma ne thiyo error control tara tesle error control dine vaneko node to node lai matrai ho doesnt mean ki it will be reliable.

The primary role of reliability is **Error Control**.

In reality, no transmission will be 100 percent errorfree delivery. Therefore, transport layer protocols are designed to provide error-free transmission.

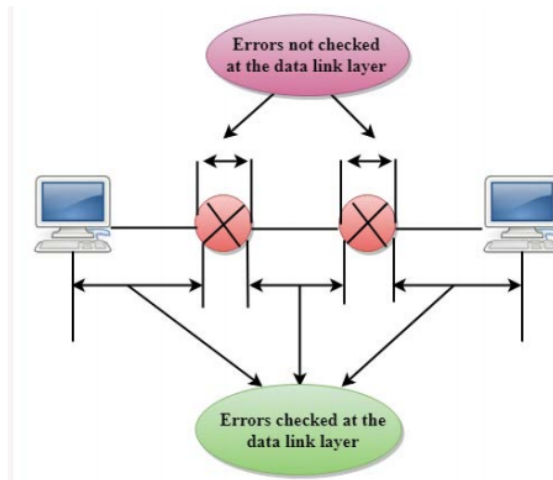
The data link layer also provides the error handling mechanism, but it ensures only node-to-node errorfree delivery.

However, node-to-node reliability does not ensure the end-to-end reliability.

#### **Q. Why data link's error control is not enough?**

The data link layer checks for the error between each network. If an error is introduced inside one of the routers, then this error will not be caught by the data link layer. It only detects those errors that have been introduced

between the beginning and end of the link. Therefore, the transport layer performs the checking for the errors end-to-end to ensure that the packet has arrived correctly.



#### ▼ Sequence control

Sender side ma chai upper to lower level dekhi packet use garrna pauna paryo  
 Reciver side ma chai assembly garna lai correctly

- The second aspect of the reliability is sequence control which is implemented at the transport layer.
- On the sending end, the transport layer is responsible for ensuring that the packets received from the upper layers can be used by the lower layers.
- On the receiving end, it ensures that the various segments of a transmission can be correctly reassembled.

#### ▼ Loss control

It ensures no part of the data gets lost during transmission.

#### ▼ Duplication control

Gaurantees that no duplicate data arrive at destination

#### ▼ Flow Control

Receiver ko side ma data overwhelming hunxha, tesle garera data loss hunxha ani retransmission garna parxha. That will cause network congesion badxha ani bad performance.

Flow control is used to prevent the sender from overwhelming the receiver.

If the receiver is overloaded with too much data, then the receiver discards the packets and asking for the retransmission of packets. This increases network congestion and thus, reducing the system performance. The transport layer is responsible for flow control.

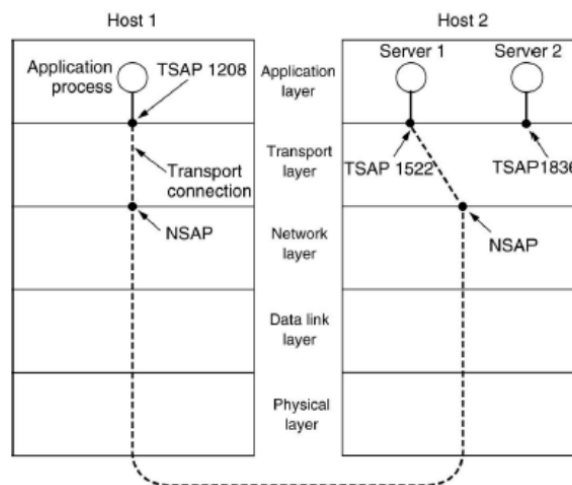
It uses the

sliding window protocol that makes the data transmission more efficient as well as it controls the flow of data so that the receiver does not become overwhelmed.

Sliding window protocol is byte oriented rather than frame oriented.

### ▼ Addressing

Port provide garxha so that data haru exchange hos ra track garna lai upper layer le k communicate garerako xha vanera



According to the layered model,

- The transport layer interacts with the functions of the session layer.
- Many protocols combine session, presentation, and application layer protocols into a single layer known as the application layer.
- In these cases, delivery to the session layer means the delivery to the application layer.
- Data generated by an application on one machine must be transmitted to the correct application on another machine.
- In this case, addressing is provided by the transport layer.
- The transport layer provides the user address which is specified as a port (They act like doorways for incoming and outgoing data on a device).
- The port variable represents a particular TS user of a specified station known as a Transport Service access point (TSAP).
- Each station has only one transport entity.
- The transport layer protocols need to know which upper-layer protocols are communicating.

## #Establishing and Releasing Connection

Establishing and releasing connections in the transport layer, such as in networking protocols like TCP/IP, involves several steps. Here's a general overview of the process:

### 1. Establishing a Connection:

- **Client Initiates Connection:** The client initiates the connection by sending a request to the server. This request typically contains information such as the destination IP address and port number.
- **Three-Way Handshake:** In TCP/IP, establishing a connection involves a three-way handshake:
  - **SYN:** The client sends a SYN (synchronize) packet to the server, indicating its intention to establish a connection.
  - **SYN-ACK:** The server responds with a SYN-ACK packet, acknowledging the client's request and indicating its readiness to establish a connection.
  - **ACK:** Finally, the client sends an ACK (acknowledge) packet back to the server, confirming the connection establishment. At this point, the connection is considered open.

## 2. Data Transfer:

- Once the connection is established, data transfer can occur bidirectionally between the client and the server. Data packets are sent and received according to the agreed-upon protocol (e.g., TCP).

## 3. Releasing the Connection:

- **Normal Connection Closure:**  
Either the client or the server can initiate the connection closure process by sending a FIN (finish) packet.
  - **FIN:** The party initiating the closure sends a FIN packet to the other party, indicating its intention to close the connection.
  - **ACK:** The receiving party acknowledges the FIN packet with an ACK.
  - **FIN-ACK:** If the receiving party also wants to close the connection, it can send its own FIN packet back, indicating mutual agreement to close the connection.
  - **Final ACK:** The initiator of the closure acknowledges the FIN packet from the other party with an ACK, and the connection is considered closed.
- **Abnormal Closure:**  
In some cases, a connection may be **terminated due to network issues or failures**.  
In such cases, the party detecting the issue may close the connection unilaterally without waiting for a response from the other party.

## 4. Connection Termination:

- After the connection is closed, any resources associated with the connection are released, and the connection is terminated. This may involve releasing buffers, freeing up ports, and deallocating memory.

Overall, the process of establishing and releasing connections in the transport layer is essential for reliable communication between networked devices, and following standardized protocols such as TCP/IP ensures orderly and efficient data exchange.

## #Flow Control & Buffering

Flow control and buffering are essential mechanisms in the transport layer, particularly in protocols like TCP/IP, to ensure efficient and reliable data

transmission between communicating entities. Here's an overview of flow control and buffering:

## 1.Flow Control:

- **Definition:** Flow control is the process of regulating the rate of data transmission between a sender and a receiver to prevent the receiver from being overwhelmed by incoming data.
- **Purpose:** It prevents the sender from transmitting data at a rate faster than the receiver can process, thus avoiding data loss, buffer overflow, and network congestion.
- **Mechanisms:**
  - **Sliding Window Protocol:** In protocols like TCP, flow control is often implemented using a sliding window mechanism. The receiver advertises its available buffer space (receive window) to the sender, indicating how much data it can accept without overflowing its buffers.
  - **Acknowledgment (ACK) Mechanism:** The receiver sends acknowledgment messages (ACKs) back to the sender to indicate successful receipt of data. These ACKs may also include information about the receiver's current buffer space.
  - **Window Size Adjustment:** Based on the receiver's advertised window size and ACKs received, the sender adjusts its transmission rate dynamically to match the receiver's processing capability.

## 2.Buffering:

- **Definition:** Buffering involves temporarily storing data in memory buffers before it is processed or transmitted.
- **Types of Buffers:**
  - **Send Buffer:**

Used by the sender to store outgoing data temporarily before transmission.
  - **Receive Buffer:**

Used by the receiver to store incoming data temporarily before it can be processed.
- **Buffer Management:**

Buffers need to be managed efficiently to avoid overflow or underflow conditions:

  - **Buffer Size:** The size of buffers should be appropriately chosen based on factors such as network bandwidth, latency, and the processing speed of the sender and receiver.
  - **Flow Control Interaction:** Buffers interact closely with flow control mechanisms to ensure that the sender does not overwhelm the receiver with data.

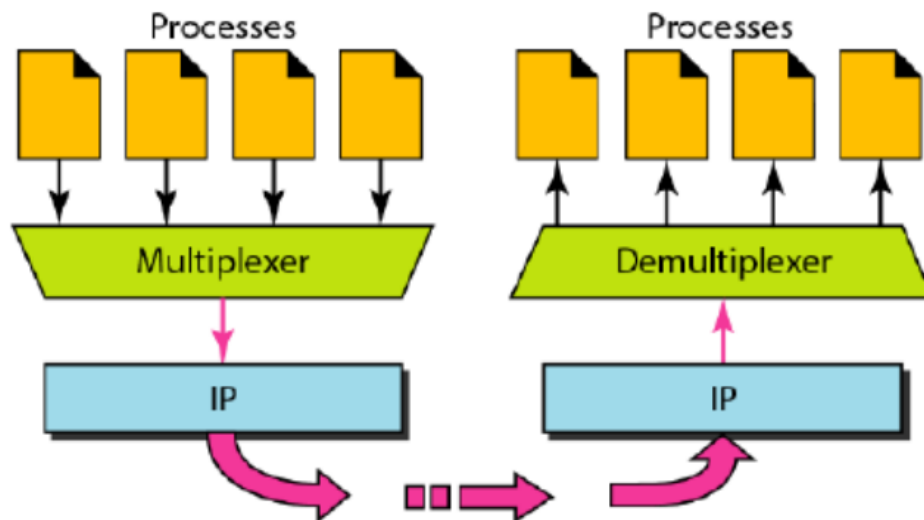
In summary, flow control and buffering mechanisms in the transport layer play crucial roles in regulating data transmission rates and managing data transfer efficiently, especially in scenarios where the sender and receiver operate at different speeds or where network conditions vary. These mechanisms contribute to the reliability and performance of communication protocols like TCP/IP.

---

## #Multiplexing and De-Multiplexing



Multiplexing and demultiplexing are fundamental processes in the transport layer, especially in protocols like TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Here's what each term means:



### 1. Multiplexing:

- **Definition:**

Multiplexing is the process of combining multiple data streams from different sources into a single stream for transmission over a shared medium or connection.

- **Purpose:** It allows multiple communication sessions to share the same network link or connection efficiently.

- **Mechanisms:**

- **Port Numbers:**

- Each communication session is associated with a unique port number on the sending and receiving sides.
    - When data is transmitted over a network, the combination of the source IP address, source port, destination IP address, and destination port uniquely identifies the communication session.

- **Socket:**

- A combination of IP address and port number
    - Multiplexing allows multiple sockets to be active simultaneously, with each socket representing a different communication session.

### 2. Demultiplexing:

- **Definition:**

Demultiplexing is the reverse process of multiplexing.

It involves extracting multiple data streams from a single transmission stream and delivering each data stream to the appropriate receiving entity based on predetermined criteria.

Multiple data stream nikalne from single data steam ra tyo data steam lai requested gareko thau ma pataune.

- **Purpose:** Demultiplexing ensures that data sent over a shared medium or connection **reaches** the intended recipient **correctly**.
- **Mechanisms:**
  - **Header Information:**  
In TCP/IP, demultiplexing **relies on** header information such as **IP addresses and port numbers**. When a packet is received, the networking stack examines the destination port number to **determine which application or process should receive the data**.
  - **Routing Tables:**  
Demultiplexing may also involve consulting routing tables or other lookup mechanisms to determine the appropriate destination for incoming data packets.

In summary, multiplexing and demultiplexing are essential processes in the transport layer of network protocols like TCP/IP. Multiplexing allows multiple communication sessions to share the same network connection efficiently, while demultiplexing ensures that data sent over the network reaches the intended recipients based on predetermined criteria such as port numbers. Together, these processes enable efficient and reliable communication between networked devices.

---

## #Crash Recover

Crash recovery is a vital function of the transport layer in the OSI model.

**It ensures reliable data delivery even when crashes (unexpected shutdowns) occur on the sender or receiver side.**

Here's how the transport layer implements crash recovery:

### Mechanisms for Crash Recovery:

- **Sequence Numbers:** (Data reliable vane si sequence no )  
Data segments are assigned unique sequence numbers by the sender. This allows the receiver to identify missing segments and request retransmission.
- **Acknowledgements (ACKs):**  
The receiver sends ACKs back to the sender to confirm successful reception of segments. Missing ACKs indicate a potential issue on the receiver's end (crash or lost packet).
- **Timers:**  
The sender sets timers for each transmitted segment. If an ACK isn't received within the timeout period, it assumes a network issue or receiver crash and retransmits the segment.
- **Selective Repeat:**  
When a gap in the received sequence is detected, the transport layer requests retransmission only for the missing segments, improving efficiency.

### Recovery Process:

1. **Crash Event:** A crash occurs on either the sender or receiver side during data transmission.
2. **Lost Segments:** Due to the crash, some data segments might not be received or processed by the receiver.
3. **Missing ACKs:** The sender doesn't receive ACKs for the missing segments within the timeout period.
4. **Retransmission:** The sender detects the missing ACKs and retransmits the corresponding segments.
5. **Reordering (if necessary):** The receiver reassembles the data stream in the correct order based on sequence numbers, even if segments arrive out of order.

#### **Benefits of Crash Recovery:**

- **Reliable Data Delivery:** Ensures that all data segments are eventually delivered to the receiver, even in case of crashes.
- **Error Handling:** Provides mechanisms to detect and rectify errors caused by crashes during transmission.
- **Data Integrity:** Maintains the integrity of the data by ensuring complete and correct delivery.

#### **Limitations:**

- **Increased Latency:** Retransmissions can introduce delays in data delivery.
- **Complexity:** Implementing and managing timers and retransmissions adds complexity to the transport layer protocols.

Crash recovery in the transport layer plays a crucial role in guaranteeing reliable communication and data transfer across networks, even in the presence of unexpected crashes.

---

## **5.3 User Datagram Protocol( UDP):User Datagram, UDP Operations, Uses of UDP, RPC**

### **#Concept**

- It is a unreliable and connection less protocol
  - Real time service like
    - Computer gaming
    - Voice
    - video communication
- we use UDP
- Its efficient for bandwidth and latency
  - No Error checking

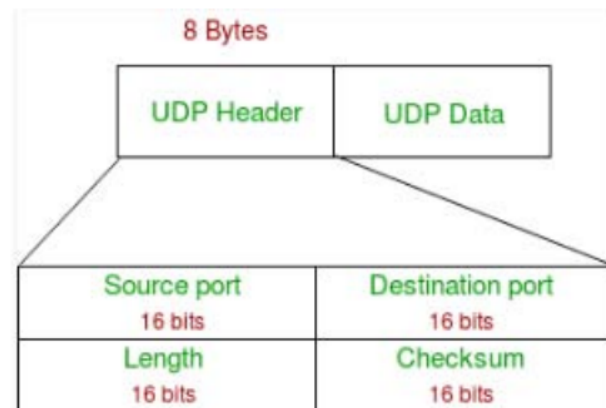
---

### **#Feature / Use of UDP**

- UDP is stateless
- Not Connection Oriented

- Good for Data flow in one direction only
- Do not guarantee ordered delivery of data
- Do not provide Congestion control mechanism
- Its better for streaming application like VOIP

## #UDP Header



### Notes -

Unlike TCP, Checksum calculation is not mandatory in UDP.

No Error control or flow control is provided by UDP.

Hence UDP depends on IP and ICMP for error reporting.

## #UDP Application

- DNS
- RIP (Routing informational protocol)
- Simple network managemnet protocol

UDP is often used in scenarios where speed and efficiency are prioritized over reliability.

Common applications include real-time multimedia streaming, online gaming, DNS (Domain Name System) queries, and VoIP (Voice over Internet Protocol) services.

## #Difference between TCP and UDP

TCP	UDP
It is connection oriendted protocol	It is connection less protocol
It is reliable	It is not reliable
Retransmission of Lost packets is possible	Not possible
Sends packets in order	Does not necessarily arrive in order
It is slower than TCP	It is faster than TCP
Header size is 20 bytes	Header size is 8 bytes

TCP	UDP
It uses HTTPS, HTTP, SMTP, FTP etc	It used VOIP, DNS etc

## #RPC (Remote Procedure Call)

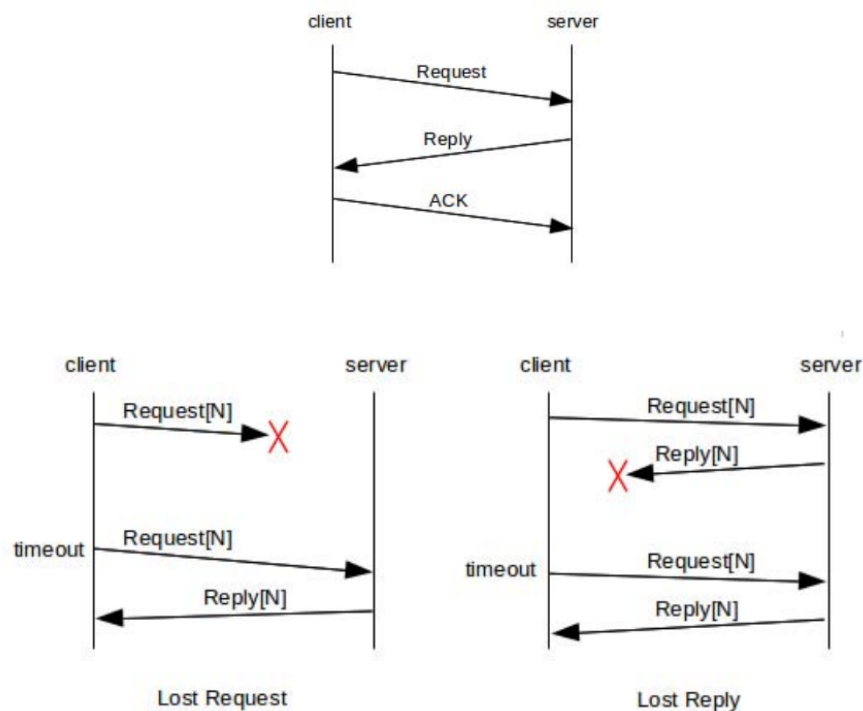
RPC stands for Remote Procedure Call.

It's a protocol that allows a program to execute code on a remote computer or server as if it were local.

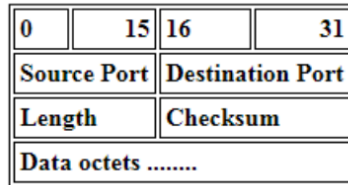
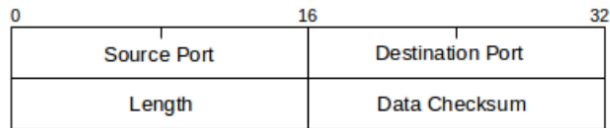
It's a protocol that allows a computer program to call a procedure on another computer program in a network, without having to know the details of its implementation.

This allows programs to communicate with each other across different machines, like different computers on a network or even across the internet.

Here's a basic analogy: imagine you have two co-workers in different offices. With RPC, it's like you can call your co-worker and ask them to do something (like get you some information), even though they're in a different location. You don't need to know exactly how they get the information, you just know they can do it for you.



## ##RPC Frame Format



- **Length**

The number of bytes in the packet. This includes the UDP header and the data (RPC packet in this case).

- **Checksum**

The checksum is the 16-bit one's complement of the one's complement sum of all 16-bit words in the pseudo-header, UDP header and raw data.

The UDP pseudo-header consists of the source and destination IP addresses, the Internet Protocol

Number for UDP (17 decimal) and the UDP length (see RFC 768). An implementation may choose not to compute a UDP checksum when transmitting a packet, in which case it must set the checksum field to zero.

- **Data Octets**

Provided by the protocol layer above UDP. In this case, this is the RPC request itself

### ###Network File Sharing

Yesle allow garthiyo server le clients haru lai file ma operation garna read or write

In terms of total packet volume, the application making the greatest use of early RPC was Sun's **Network File Sharing**, or NFS; this allowed for a filesystem on the server to be made available to clients.

**For read() operations**

**For write() operations,**

### ###SUN RPC

Sun RPC, also known as ONC RPC (Open Network Computing Remote Procedure Call), is a specific type of RPC protocol developed by Sun Microsystems in the 1980s.

It's a widely used protocol for communication between programs on a network.

### ###Serialization

In some RPC systems, even those with explicit ACKs, requests are executed serially by the server.

### ###Refinements

The main issue is that standard network protocols might fragment large data packets into smaller pieces. While this works fine on local networks (LAN), it can cause problems over longer distances.

The proposed solution is an RPC-level large-message protocol. This protocol would handle large messages differently:

- Fragmentation would happen at the RPC layer, instead of the network layer.
- Only the lost fragments would be retransmitted, improving efficiency.

This approach avoids unnecessary fragmentation and retransmissions, potentially improving performance for RPC over long distances.

---

## 5.4 Principles of Reliable DataTransfer: Building a Reliable DataTransfer Protocol, Pipelined Reliable Data Transfer Protocol, Go Back-N(GBN), Selective Repeat(SR)

### #Difference between RDT and UDT

Feature	Reliable Data Transfer	Unreliable Data Transfer
Guarantee	Ensures data arrives correctly and in order	No gaurantees on arrival, order
Focus	Accuracy and completness	Speed and efficiency
Protocol Eg	TCP	UDP
Advantage	Ensures data integrity and order	Faster due to minimal overhead
Disadvantage	Slower due to extra overhead	Dass loss, out of order delivery
Use Case	File transfer, web browsing Online transactions, viedo streaming	Online gaming, real time application

### #Building Reliable Data Transfer Protocol

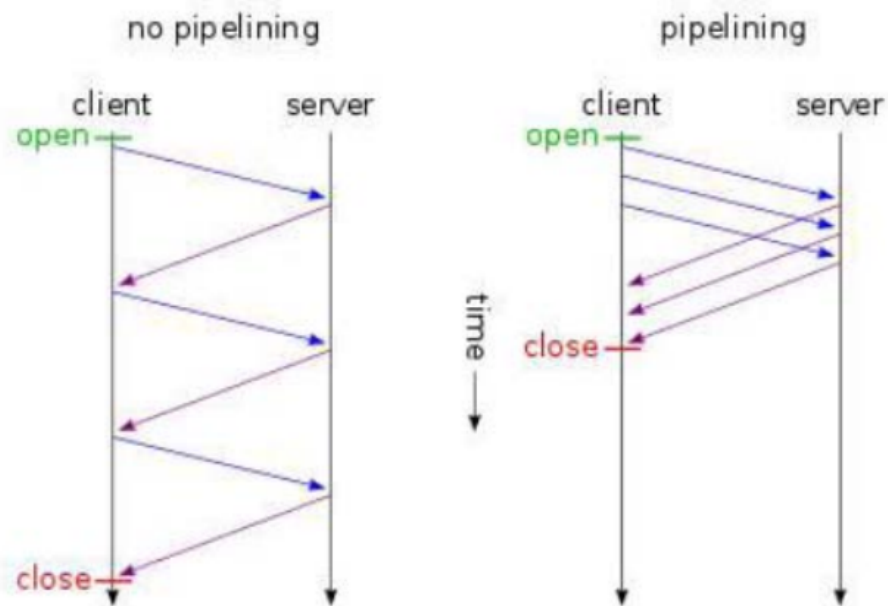
Imagine sending a letter. You wouldn't just toss it in a mailbox and hope for the best. Reliable data transfer works similarly. Here are some basic elements:

- **Sequence Numbers:**  
Each data piece gets a unique sequence number to ensure order upon arrival.
  - **Acknowledgments (ACKs):**  
The receiver sends an acknowledgment (ACK) signal for each correctly received piece.
  - **Timeouts and Retransmissions:**  
If an ACK doesn't arrive within a set time, the sender assumes the data was lost and retransmits it
  - **Checksums:**  
A checksum, like a digital fingerprint, is added to each data piece to detect corruption during transmission.
-

## #Pipelined Reliable Data Transfer Protocols

Basic protocols wait for an ACK before sending the next piece. Pipelining allows the sender to transmit multiple pieces before waiting, improving efficiency.

Pipelining can be used in various application layer network protocols, like HTTP/1.1, SMTP and FTP.



## #Two Generic Forms of Pipelined protocols

Feature	Go - Back - N	Selective Repeat
Retransmission Strategy	Retransmits all packets from the lost packet onwards (including correctly sent ones)	Retransmits only the specific lost packet
Efficiency	Less efficient for high packet loss or large window sizes	More efficient, reduces unnecessary retransmissions
Complexity	Simpler to implement	More complex due to receiver-side buffering
Suitable for	Networks with low to moderate packet loss	Networks with higher packet loss rates
Sender Window Size	Size N	Size N
Receive Window size	Size is 1	Size is N
Figure	—	—



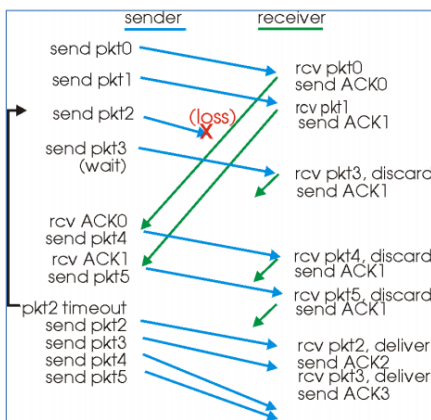
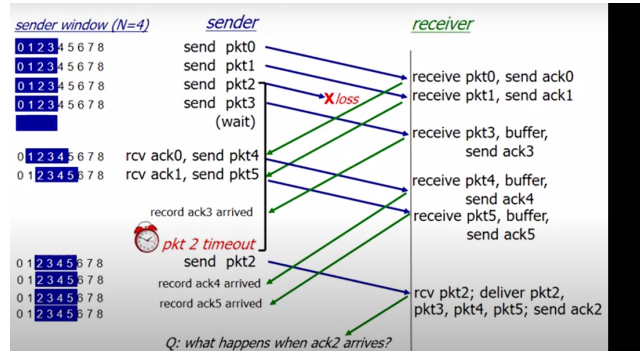
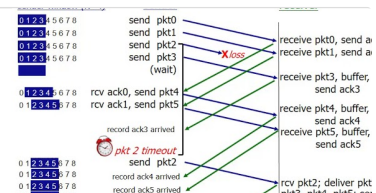


Figure 3.4-14: Go-Back-N in operation



Networking: Unit 3 - Transport Layer - Lesson 10 - Selective Repeat

[https://youtu.be/zBB1V\\_iQ9YM?si=1kLkMBf10G80mjxc](https://youtu.be/zBB1V_iQ9YM?si=1kLkMBf10G80mjxc)



## 5.5 Transmission Control Protocol(TCP): TCP Services, TCP Features, TCP Segment Header

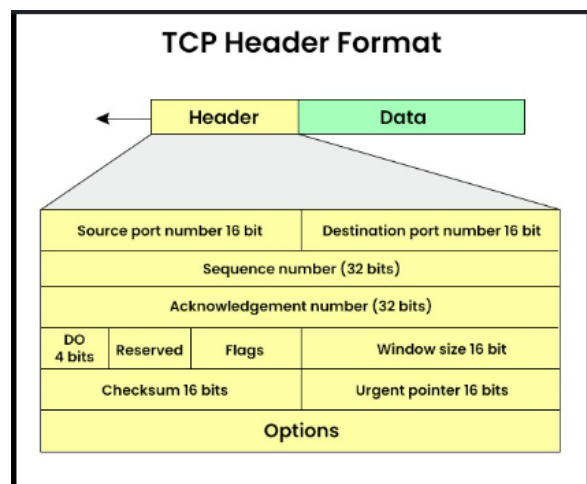
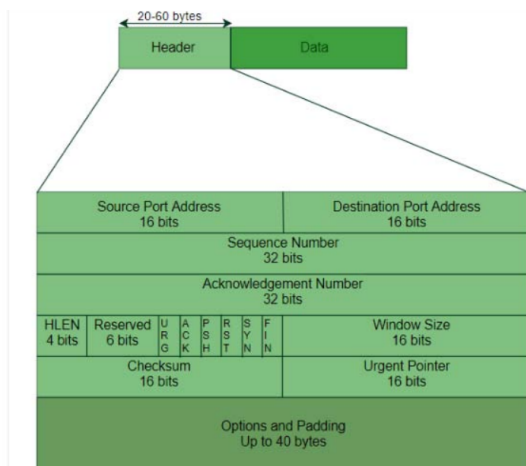
### #Feature

- Connection Oriented
- Reliable Protocol ( send ACK or NACK ) if needed retransmission
- Sends the data in order
- Error control and flow control
- Full duplex server

### #Difference between TCP and UDP

TCP	UDP
It is connection oriented protocol	It is connection less protocol
It is reliable	It is not reliable
Retransmission of Lost packets is possible	Not possible
Sends packets in order	Does not necessarily arrive in order
It is slower than TCP	It is faster than TCP
Header size is 20 bytes	Header size is 8 bytes
It uses HTTPS, HTTP, SMTP, FTP etc	It used VOIP, DNS etc

### #TCP Header



Sequence number and ACK is from Crash Recover → reliable  
→ TCP

DO → header length

- **Source Port (16 bits):**  
Identifies the sender application on the sending device.
- **Destination Port (16 bits):**  
Identifies the receiver application on the receiving device.
- **Sequence Number (32 bits):**  
Track for order
- **Acknowledgment Number (32 bits):**  
Used by the receiver to acknowledge received data and indicate the next expected byte sequence number.
- **Data Offset (4 bits):**  
Also known as the header length, specifies the number of 32-bit words in the TCP header (usually 5 for a standard header). This helps identify where the data payload starts within the segment.
- **Reserved (3 bits):**  
Currently unused and set to 0.
- **Flags (9 bits):**  
Control bits that signal different actions or statuses during data transfer, such as indicating the start or end of a connection, urgent data, or acknowledging data.
- **Window Size (16 bits):**  
Specifies the number of bytes the sender is willing to receive before needing an acknowledgment (ACK) from the receiver. This helps control the flow of data.
- **Checksum (16 bits):**  
A calculated value to check for errors during transmission. Both sender and receiver calculate the checksum and compare them. Any discrepancy indicates data corruption.
- **Urgent Pointer (16 bits):**  
(Optional) Points to urgent data within the segment if the URG flag is set.

- **Options (variable length):**  
(Optional) Provides additional capabilities beyond the basic header, but can increase header size.
- 

## 5.1 Principle of Congestion Control

Practical understanding

### #Congestion

- It occurs, if the load offered to any network is more than its capability
  - TCP controls congestion by means of window mechanism
  - TCP sets window size telling the other end how much data segment to send.
  - It uses four algorithms / components
    - **Slow Start:**  
Upon establishing a connection, TCP starts by sending a small amount of data to gradually probe the network's capacity.
    - **Congestion Window (CWND):**  
Determines the maximum amount of data that the sender can have outstanding at any given time.
    - **Congestion Avoidance Phase:**  
Once the slow start phase is complete, TCP enters the congestion avoidance phase, where it gradually increases the CWND by one segment (a unit of data) per acknowledgment (ACK) received from the receiver.
    - **Congestion Detection:**  
TCP detects congestion when it experiences packet loss or timeouts (when an ACK is not received within a specified time).
- 

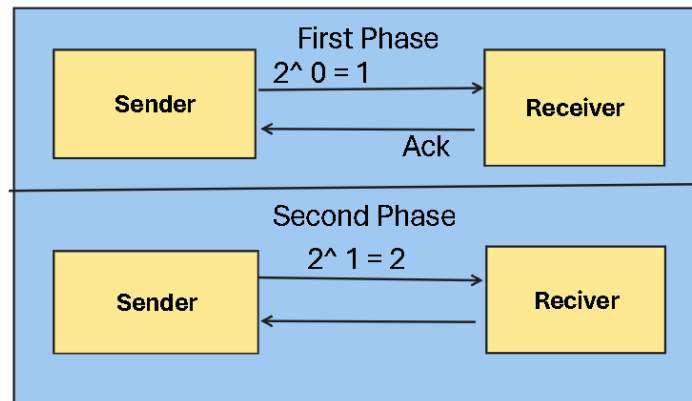
### ##Benefits of Congestion Control:

- **Reduced Packet Loss:** By preventing congestion, fewer data packets are dropped, improving data integrity and reducing the need for retransmissions.
  - **Lower Delays:** Efficient data flow minimizes delays, leading to faster and more responsive network communication.
  - **Improved Network Efficiency:** Congestion control ensures optimal utilization of network resources, preventing underutilization and congestion collapse.
  - **Fairness:** Different users share network resources fairly, preventing any single user from dominating the bandwidth.
- 

### ##Slow Start Algorithm (exponential)

- The size of the window is growing exponentially till
  - timeout occurs on the receiver window
  - receiver window it reached maximum threshold (64 kb)

EG:



Suppose sender starts with congestion window =  $2^0 = 1$  MSS  
(Max seg size, each segment consist of 1 byte)

After receiving ack from seg 1, the size of the window can be increased by one.

Thus, total size of congestion window =  $2^1 = 2$ .

When all the seg are, total size of the congestion window is  $2^3 = 8$

### ##Congestion Avoidance (add)

- In slow start algo, size of congestion window increases exponentially
- To avoid congestion before it happens, it is necessary to slow down the exponential growth
- We will use additive increase instead of exponential growth

After receiving ack, the segment of congestion window can be increased by 1.

EG:

Start with congestion window = 1

By adding, congestion window =  $1 + 1 = 2$ .

### ##Congestion Detection (time-out, 3ack)

- If congestion occurs, it is required to decrease the congestion window size.
- Two conditions in which congestion may occur
  - congestion time out
  - reception of three acks.
- In both cases, threshold size can be dropped to one half of the previous size

## #Quality of Service (QoS).

- Measure performance of computer network
- Character of QoS :
  - ▼ Reliability
 

It's about receiving ack. eg: email & file transfer
  - ▼ Delay
 

It needs minimal time delay

eg: video streaming

#### ▼ Jitter

Variaion in packet delay.

#### ▼ Bandwidth

Different app needs different bandwidth

eg: video streaming more bandwith then email

---

## #Leaky and Token Bucket

Leaky Bucket and Token Bucket are both rate limiting algorithms used for traffic control in networks. They fall under the category of **Quality of Service (QoS)** mechanisms.

Here's a breakdown of each:

### Leaky Bucket:

- Imagine a leaky bucket with a hole at the bottom and a tap at the top.
- Data packets (water) arrive at the bucket (reservoir) through the tap.
- The bucket has a limited capacity (maximum water level).
- The hole at the bottom allows a constant rate of water to leak out (data packets to be sent out).
- If incoming packets (water) arrive faster than the leak rate, they fill the bucket (buffer) until it's full.
- Any packets exceeding the bucket's capacity are discarded (dropped).

### Token Bucket:

- Imagine a bucket filled with tokens at a constant rate.
- Each data packet requires a token for permission to be sent.
- Packets arriving without a token are queued until a token becomes available.
- The bucket has a limited capacity for storing tokens (maximum number of packets that can be queued).
- If the bucket is full and no tokens are available, incoming packets are discarded (dropped).

### Choosing Between Leaky Bucket and Token Bucket:

- **Leaky Bucket:** Good for **guaranteed average rate**, but can't handle burst traffic well (packets arriving in a short time). Packets might be dropped even if the average rate isn't exceeded.
- **Token Bucket:** Good for **bursty traffic**, as it allows accumulating tokens for short bursts of data. Offers more flexibility in handling peak loads.


### In Summary:

Both Leaky Bucket and Token Bucket are QoS mechanisms that regulate data flow to prevent network congestion. They differ in how they handle incoming data:

- Leaky Bucket enforces a constant outflow rate, potentially dropping packets even if the average rate isn't exceeded.

- Token Bucket allows for bursts by accumulating tokens, offering more flexibility but potentially causing queuing delays.

leaky bucket algorithm | congestion control | networking | Bhanu Priya  
leaky bucket algorithm in networks

 <https://youtu.be/Db-tV8JJ3ZQ?si=KPxpKk08WMgJIT>

