# Computer Network and Data Communication

# UNIT 6
# Transport Layer

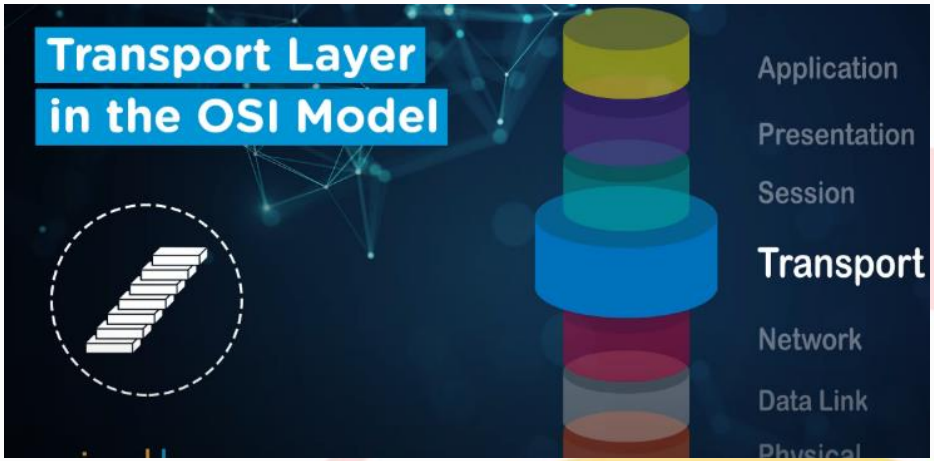
Rajan Sharma

# Course Outline

6. **Transport Layer** **[5 Hrs]**
   - 6.1. Functions of Transport Later
   - 6.2. Connection Management: TCP, UDP
   - 6.3. Port Addressing: Ports and Sockets
   - 6.4. Connection Establishment and Release
   - 6.5. Flow Control, Buffering
   - 6.6. Congestion Control: Token Bucket, Leaky Bucket
   - 6.7. IP Remapping: NAT

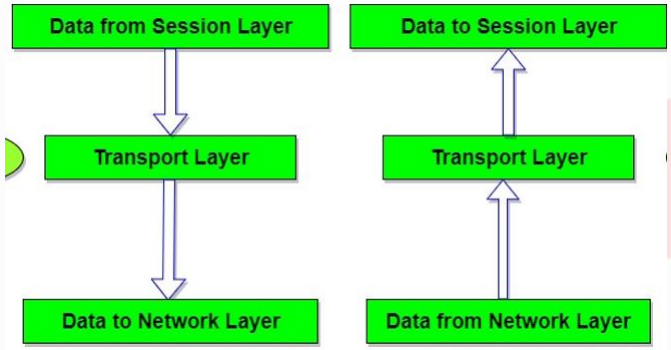
# TRANSPORT LAYER

- The transport Layer is the fourth layer in the OSI model.
- It is an end-to-end layer used to deliver messages to a host



# Working of Transport Layer

- **At the sender's side**: The transport layer receives data (message) from the Application/Session layer and then performs Segmentation, divides the actual message into segments, adds the source and destination's port numbers into the header of the segment, and transfers the message to the Network layer.
- **At the receiver's side**: The transport layer receives data from the Network layer, reassembles the segmented data, reads its header, identifies the port number, and forwards the message to the appropriate port in the Application layer.

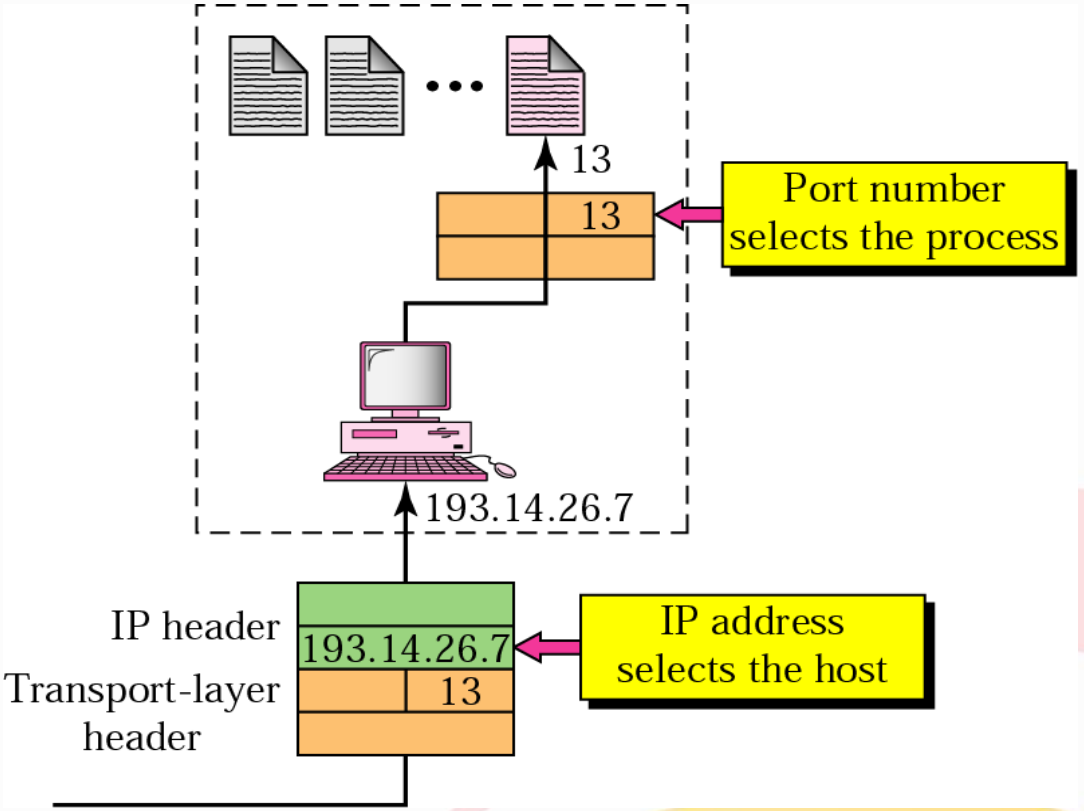


# Function of Transport Layer

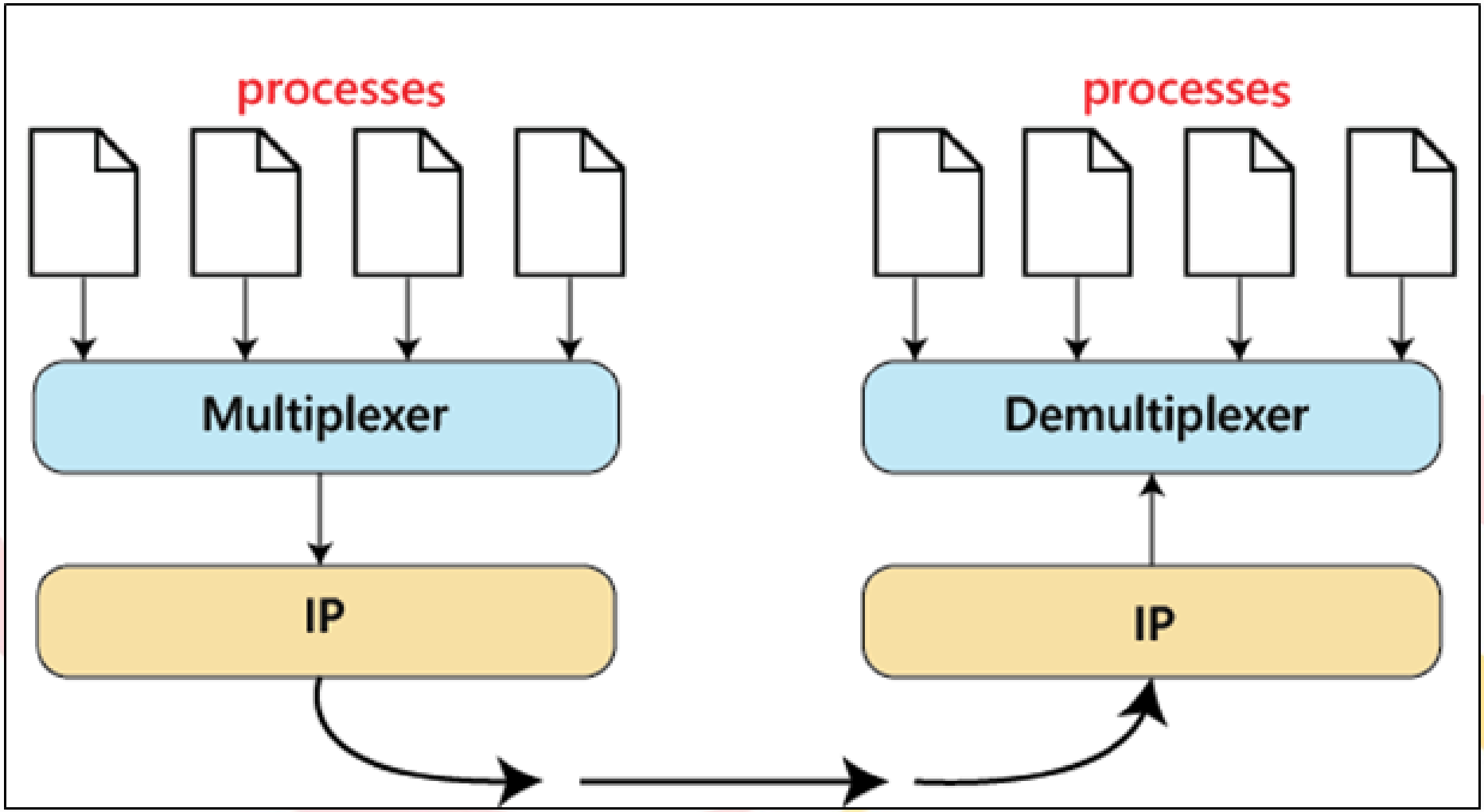
- 1. The Process to Process Delivery
- 2. End-to-End Delivery
- 3. Addressing
- 4. Multiplexing and Demultiplexing
- 5. Congestion Control
- 6. Error control
- 7. Flow control

# 1. Process to Process Delivery

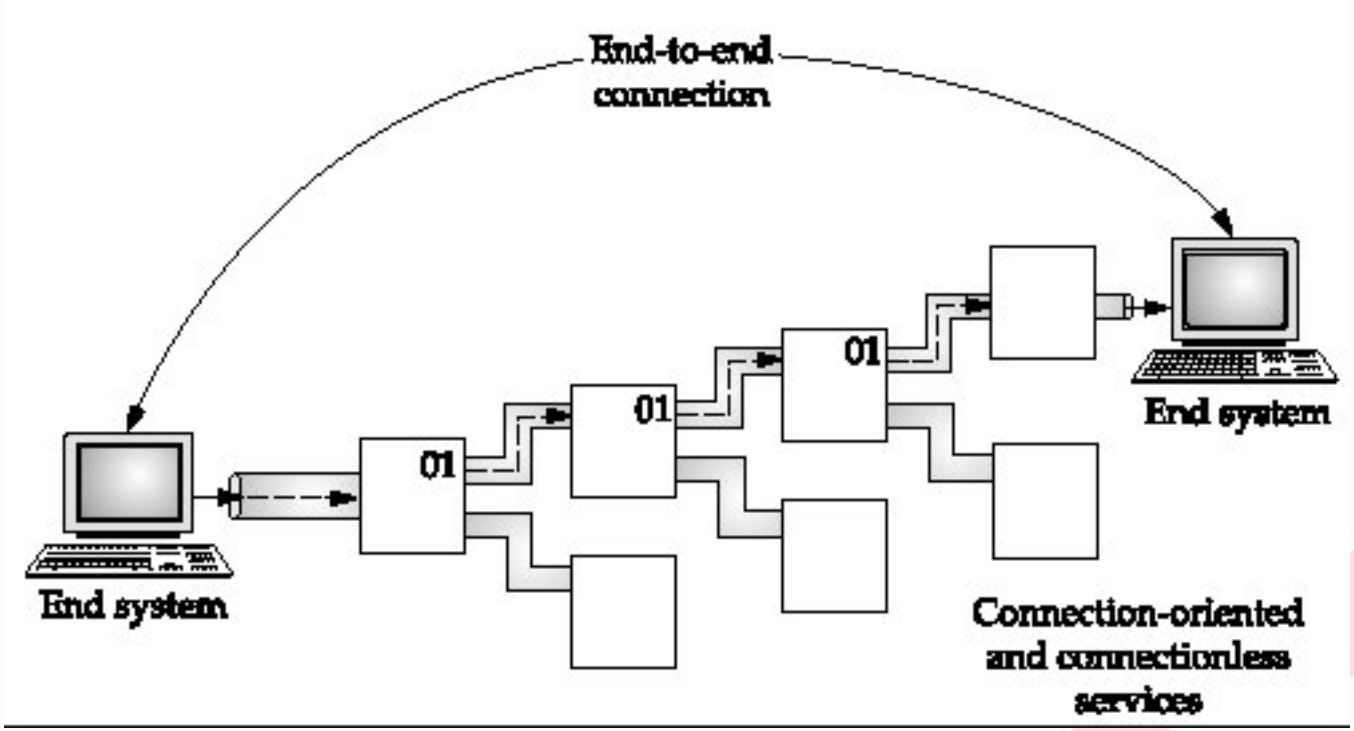
- Data Link Layer requires the MAC address
- Network layer requires the IP address for appropriate routing of packets
- Transport Layer requires a Port number to correctly deliver the segments of data to the correct process amongst the multiple processes running on a particular host.
- A port number is a 16-bit address used to identify any client-server program uniquely.



# 1. Process to Process Delivery



# 2. End to End Connection between hosts

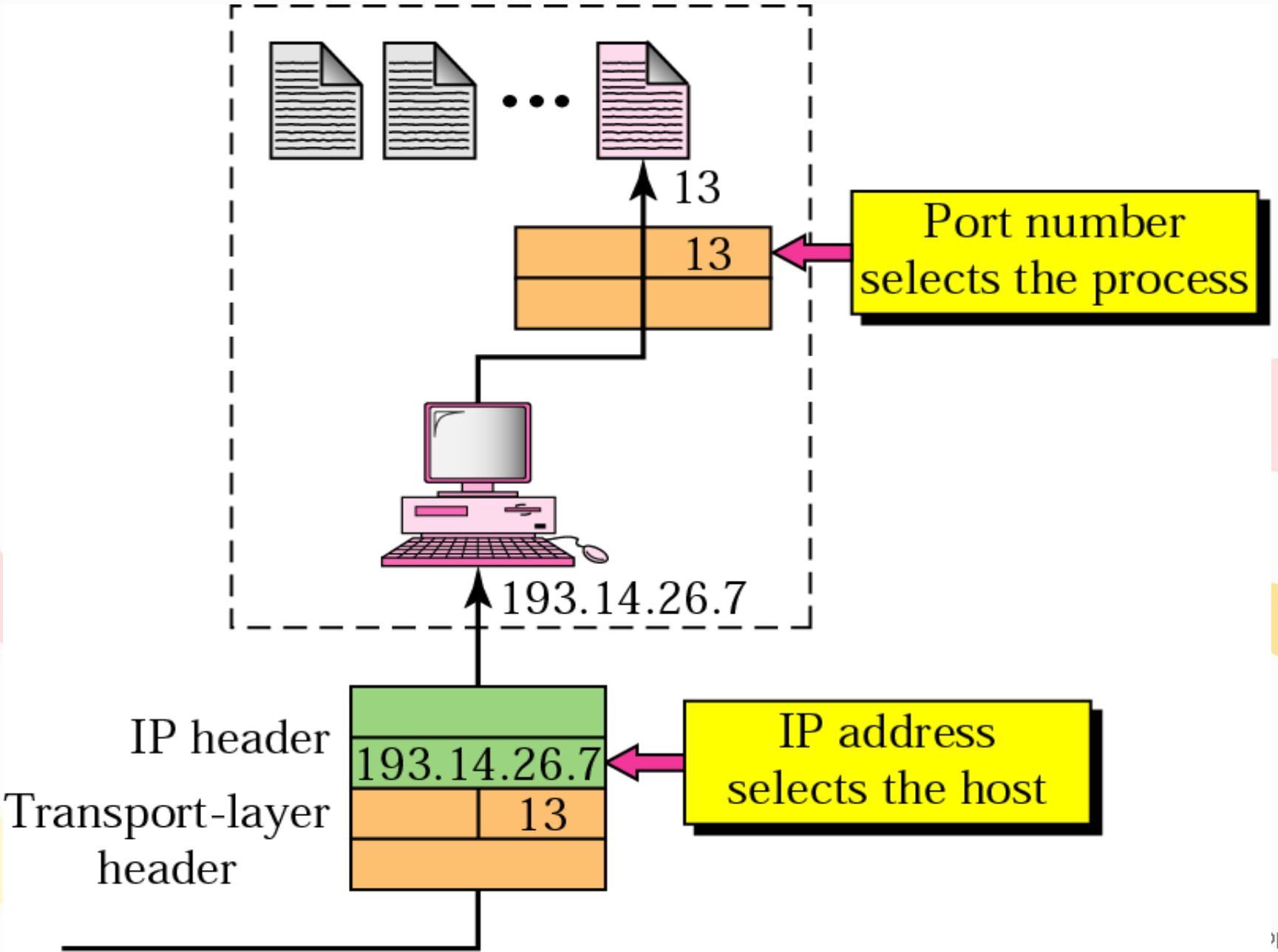



- The transport layer is also responsible for creating the end-to-end Connection between hosts for which it mainly uses TCP and UDP.
- TCP is a secure, connection-orientated protocol which ensures the reliable delivery of messages and is used in various applications.
- UDP, on the other hand, is a stateless and unreliable protocol that ensures best-effort delivery.

# 3. Addressing

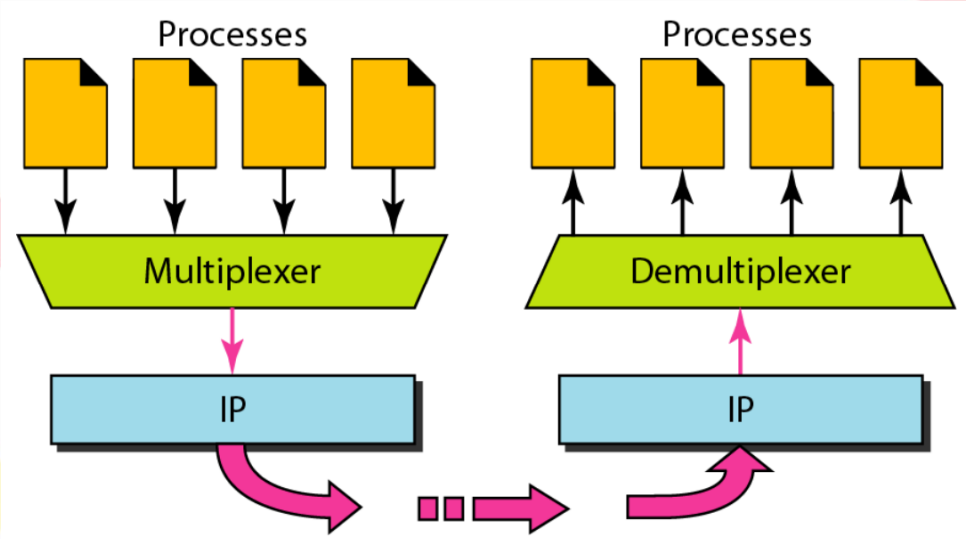
- Addressing in Transport Layer is the port Address
- Data generated by an application on one machine must be transmitted to the correct application on another machine. In this case, addressing is provided by the transport layer
- The transport layer provides the user address which is specified as a station or port
- The comnination of IP address and Port Address is called Socket Address

# 3. Addressing

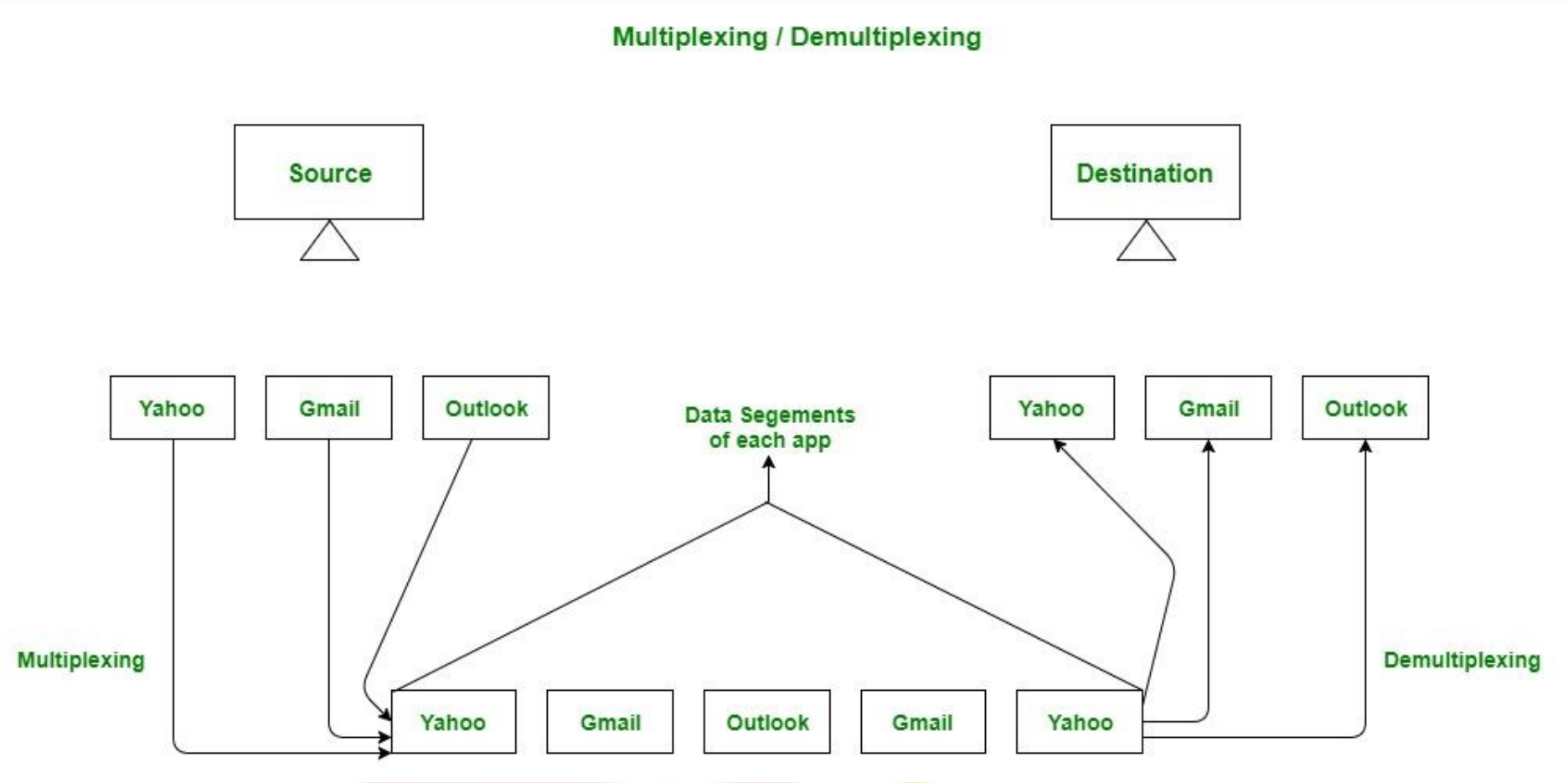


# 4. Multiplexing and Demultiplexing

- Multiplexing(many to one) is when data is acquired from several processes from the sender and merged into one packet along with headers and sent as a single packet.
- Multiplexing allows the simultaneous use of different processes over a network that is running on a host.
- The processes are differentiated by their port numbers.
- Similarly, Demultiplexing(one to many) is required at the receiver side when the message is distributed into different processes.
- Transport receives the segments of data from the network layer distributes and delivers it to the appropriate process running on the receiver's machine.



# 4. Multiplexing and Demultiplexing



# 6. Error Control

- The transport layer checks for errors in the messages coming from the application layer by using error detection codes, and computing checksums

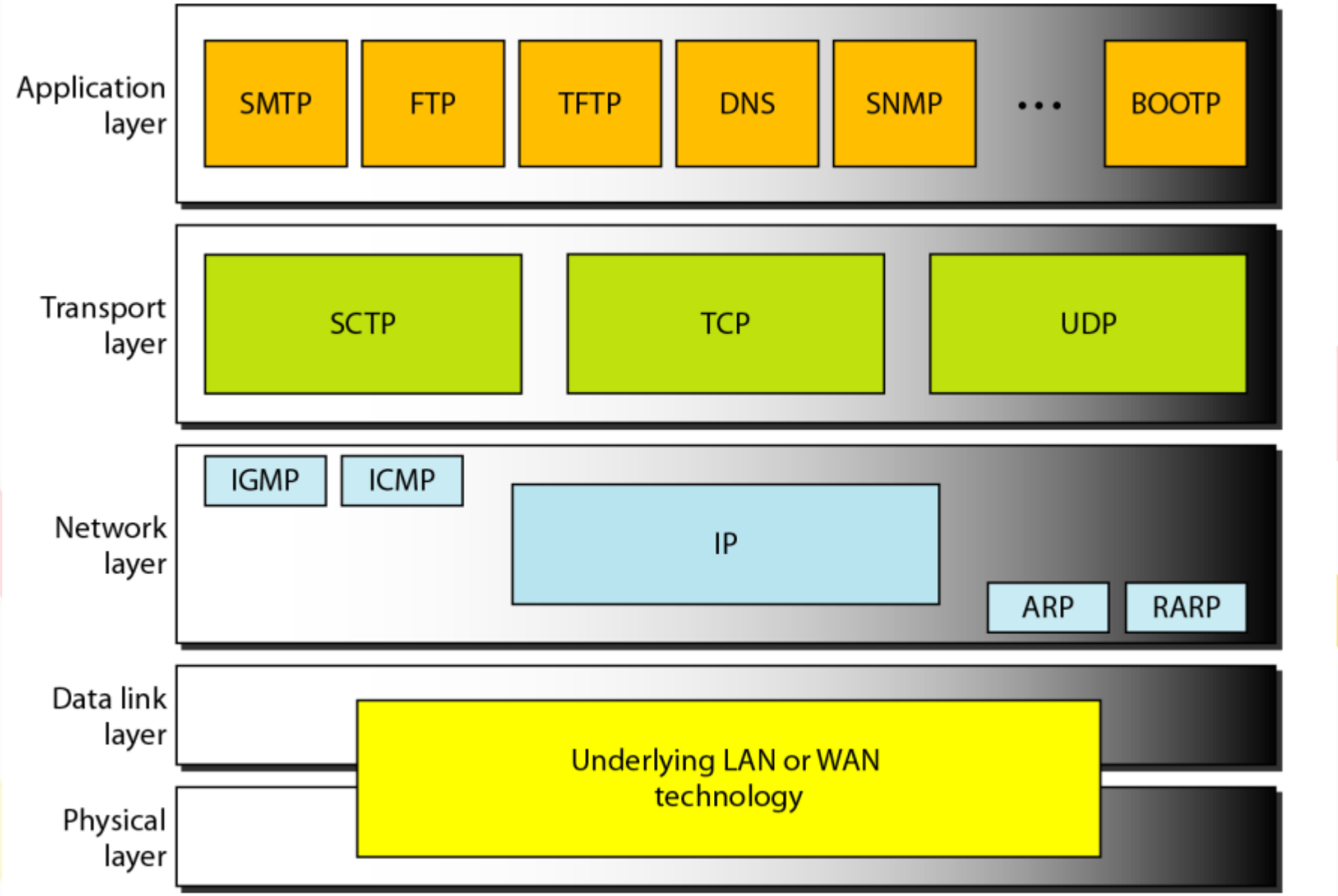
# 7. Flow Control

- The transport layer provides a flow control mechanism between the adjacent layers of the TCP/IP model.

# 8. QoS

- The Transport Layer maintains QoS via
  - Reliability
  - Flow control
  - Error Control
  - Congestion control

# Connection Management: TCP/UDP



# UDP

- UDP (User Datagram Protocol) is a connectionless transport layer protocol in the TCP/IP protocol suite.
- It provides a simple, unreliable, and low-overhead method for delivering datagrams (packets) between hosts on an IP network
- Key Features of UDP
  1. **Connectionless Protocol:** Unlike TCP, UDP does not establish a connection before transmitting data. Each UDP packet (datagram) is independent and can be sent without prior setup.
  2. **Unreliable Delivery:** UDP does not guarantee delivery or ensure that packets arrive in order.
  3. **Low Overhead:** UDP has minimal overhead compared to TCP, making it lightweight and suitable for applications where speed and efficiency are more important than reliability, such as real-time multimedia streaming and online gaming.
  4. **No Congestion Control:** UDP does not implement congestion control mechanisms like TCP.

# UDP

**5. Checksum for Error Detection:** UDP includes a checksum field in its header to detect errors in transmitted data.

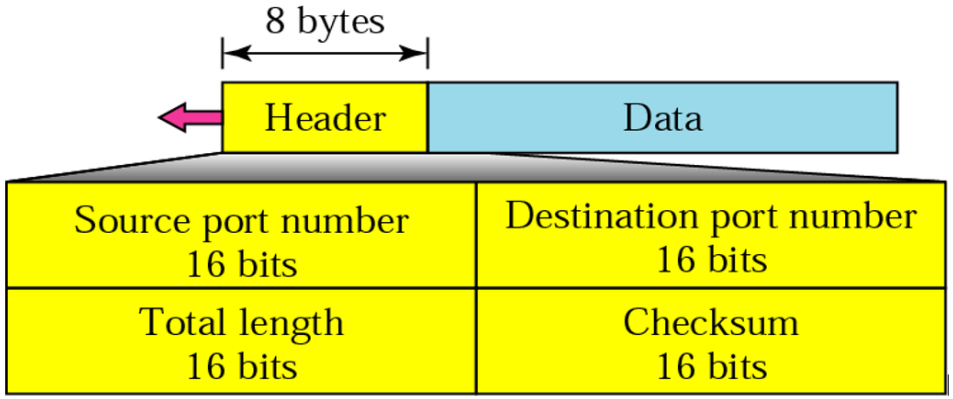
**6. Port Numbers:** UDP uses port numbers to identify different services or applications running on hosts.

**Examples of UDP Applications:**

- Domain Name System (DNS)
- Dynamic Host Configuration Protocol (DHCP)
- Trivial File Transfer Protocol (TFTP)
- Voice over IP (VoIP) and streaming media
- Online gaming and real-time communication protocols like UDP-based variants of Real-Time Protocol (RTP) and Real-Time Control Protocol (RTCP)

# UDP Datagram

**UDP Datagram**




## 1. Source Port-

- Source Port is a 16 bit field.
- It identifies the port of the sending applicatior

## 2. Destination Port-

- Destination Port is a 16 bit field.
- It identifies the port of the receiving application.

## 3. Length-

- Length is a 16 bit field.
- It identifies the combined length of UDP Header and Encapsulated data.

  Length = Length of UDP Header + Length of encapsulated data

## 4. Checksum-

- Checksum is a 16 bit field used for error control.
- It is calculated on UDP Header, encapsulated data and IP pseudo header.
- Checksum calculation is not mandatory in UDP.

# UDP Service Ports

- Applications use datagram sockets to establish host-to-host communications

| *Port* | *Protocol* | *Description* |
|---|---|---|
| 7 | Echo | Echoes a received datagram back to the sender |
| 9 | Discard | Discards any datagram that is received |
| 11 | Users | Active users |
| 13 | Daytime | Returns the date and the time |
| 17 | Quote | Returns a quote of the day |
| 19 | Chargen | Returns a string of characters |
| 53 | Nameserver | Domain Name Service |
| 67 | Bootps | Server port to download bootstrap information |
| 68 | Bootpc | Client port to download bootstrap information |
| 69 | TFTP | Trivial File Transfer Protocol |
| 111 | RPC | Remote Procedure Call |
| 123 | NTP | Network Time Protocol |
| 161 | SNMP | Simple Network Management Protocol |
| 162 | SNMP | Simple Network Management Protocol (trap) |

# TCP

- TCP (Transmission Control Protocol) is a connection-oriented transport layer protocol in the TCP/IP protocol suite.
- It provides reliable, ordered, and error-checked delivery of data between applications running on hosts in an IP network.
- TCP Provides process to process communication using Port numbers.
- TCP is one of the main protocols in TCP/IP networks and widely used for data communication in network such as internet.
- Where the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data.
- TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent

# Features of TCP

**1. Process to Process Communication:** TCP Provides process to process communication using Port numbers.

**2.Connection-Oriented:** TCP establishes a connection between sender and receiver before transmitting data.

**3.Reliable Delivery:** TCP ensures reliable delivery of data by using sequence numbers to track the order of transmitted segments and acknowledge receipt of each segment. It implements mechanisms for error detection, retransmission of lost segments, and flow control to prevent congestion and ensure efficient data transfer.

**4.Ordered Delivery:** TCP guarantees that data segments arrive in the same order they were sent. It reassembles segments at the receiving end before delivering them to the application layer.

**5.Flow Control:** TCP uses flow control mechanisms to manage the rate of data transmission between sender and receiver.

# Features of TCP

**6. Congestion Control:** TCP implements congestion control algorithms to avoid network congestion and optimize performance

**7. Windowing:** TCP uses a sliding window mechanism to optimize data transmission efficiency.

**8. Acknowledgments:** TCP requires the receiver to acknowledge received segments. The sender retransmits segments if acknowledgments are not received within a specified timeout period, ensuring reliable delivery.

**9. Checksum for Error Detection:** TCP includes a checksum field in its header to detect errors in transmitted data.

**10.Port Numbers:** TCP uses port numbers to identify different services or applications running on hosts. Port numbers are included in TCP headers to ensure that segments are delivered to the correct application.

# Examples TCP

- Web browsing (HTTP)
- File transfer (FTP, SFTP)
- Email (SMTP, IMAP, POP3)
- Remote access (SSH, Telnet)
- Secure communication (HTTPS, SSL/TLS)

# TCP Ports

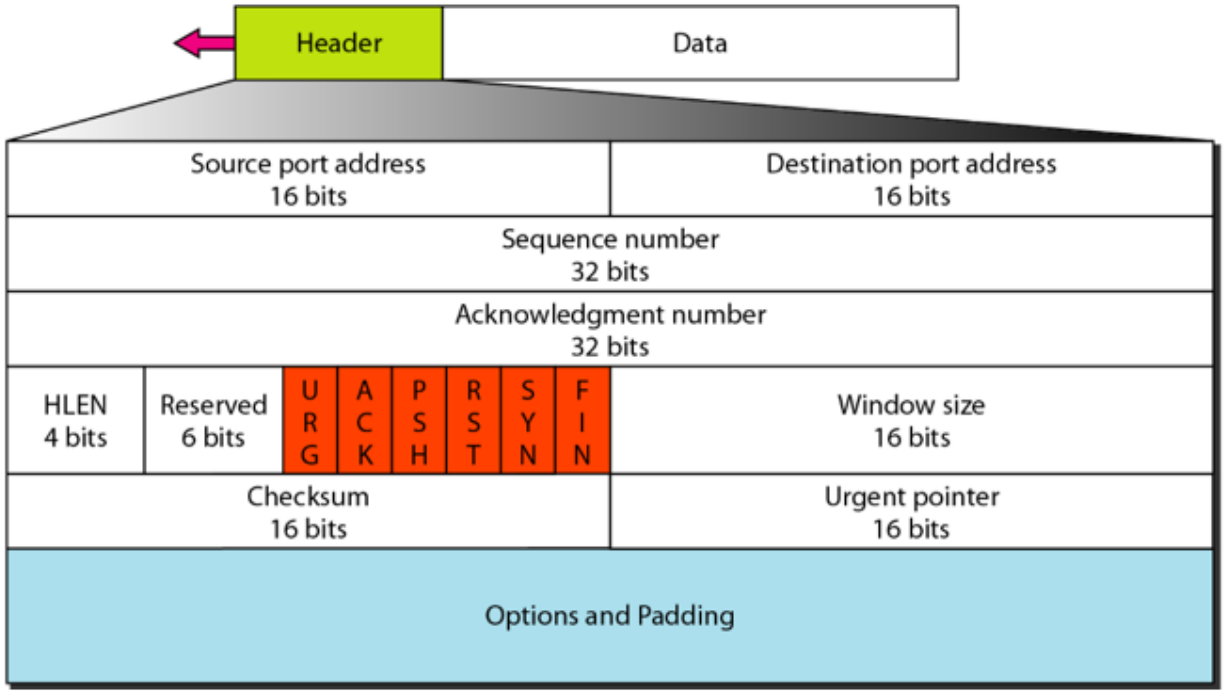
*Table 22.2 Well-known ports used by TCP*

| Port | Protocol | Description |
|---|---|---|
| 7 | Echo | Echoes a received datagram back to the sender |
| 9 | Discard | Discards any datagram that is received |
| 11 | Users | Active users |
| 13 | Daytime | Returns the date and the time |
| 17 | Quote | Returns a quote of the day |
| 19 | Chargen | Returns a string of characters |
| 20 | FTP, Data | File Transfer Protocol (data connection) |
| 21 | FTP, Control | File Transfer Protocol (control connection) |
| 23 | TELNET | Terminal Network |
| 25 | SMTP | Simple Mail Transfer Protocol |
| 53 | DNS | Domain Name Server |
| 67 | BOOTP | Bootstrap Protocol |
| 79 | Finger | Finger |
| 80 | HTTP | Hypertext Transfer Protocol |
| 111 | RPC | Remote Procedure Call |

# TCP Segment

- A packet in TCP is called a segment

**Figure 23.16** *TCP segment format*




No assigned to data

Seq no +1

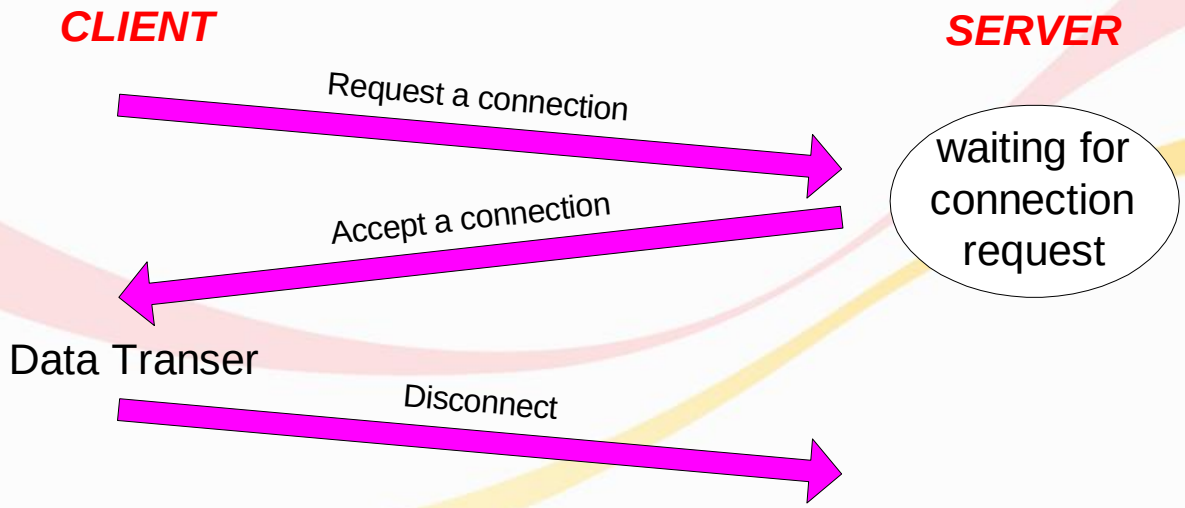
Valid only if Urgent Flag is set

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push

RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection

# Connection Management with TCP

- Before any data transfer, TCP establishes a connection:
  - One TCP entity is waiting for a connection (“server”)
  - The other TCP entity (“client”) contacts the server
- Each connection is full duplex
- Byte stream is broken up into chunks which are called segments
- Receiver sends acknowledgements (ACKs) for segments
- TCP maintains a timer. If an ACK is not received in time, the segment is retransmitted



# Connection Management with TCP

## TCP Connection (3-Way Handshake): Opening a Connection

TCP uses a three-way handshake to open a connection:

Host A sends a TCP SYNchronize packet to Host B

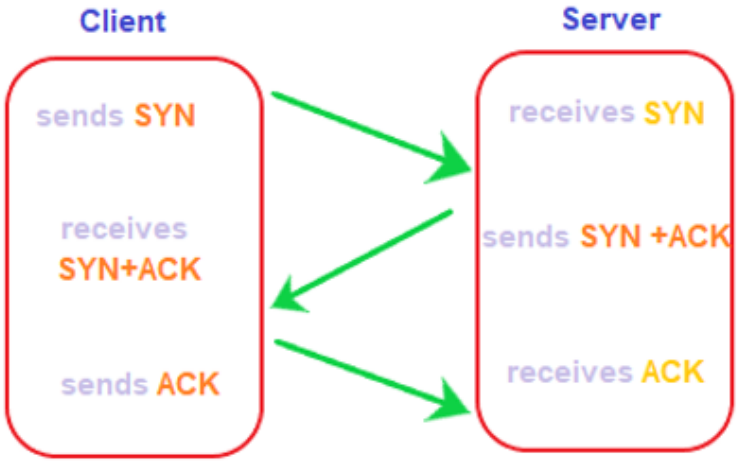
Host B receives A's SYN

Host B sends a SYNchronize-ACKnowledgement

Host A receives B's SYN-ACK

Host A sends ACKnowledge

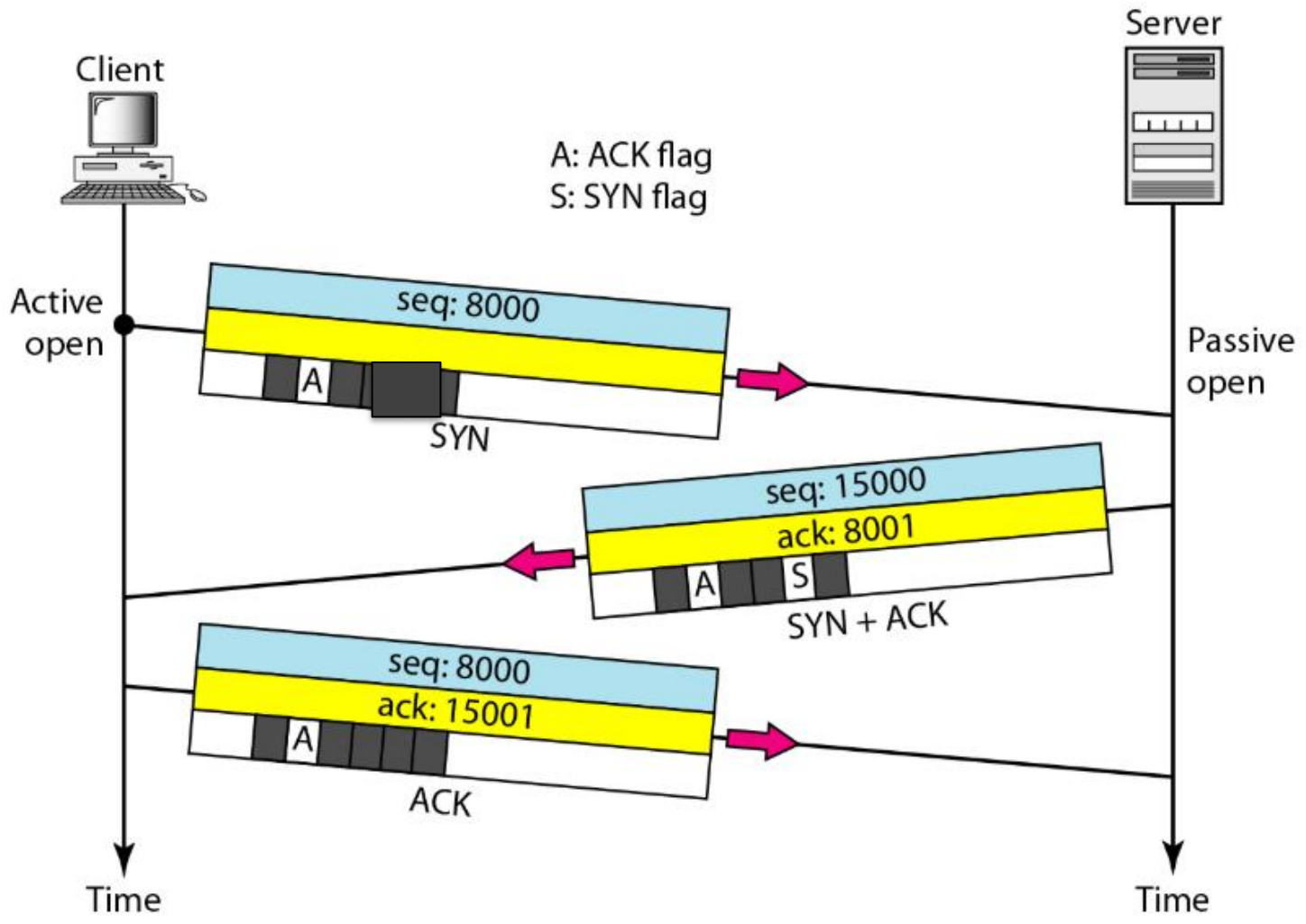
Host B receives ACK.




***TCP socket connection is ESTABLISHED.***

# Connection Management with TCP

**Figure 23.18** *Connection establishment using three-way handshaking*



# Connection Management with TCP

**Step 1: SYN**

- SYN is a segment sent by the client to the server.
- It acts as a **connection request** between the client and server.
- It informs the server that the client wants to establish a connection.
- Synchronizing sequence numbers also helps synchronize sequence numbers sent between any two devices, where the same SYN segment asks for the sequence number with the connection request.

**Step 2: SYN-ACK**

- It is an SYN-ACK segment or an SYN + ACK segment sent by the server.
- The ACK segment informs the client that the server has received the connection request and it is ready to build the connection.
- The SYN segment informs the sequence number with which the server is ready to start with the segments.

**Step 3: ACK**

- ACK (Acknowledgment) is the last step before establishing a successful TCP connection between the client and server.
- The ACK segment is sent by the client as the response of the received ACK and SN from the server.
- It results in the establishment of a reliable data connection.

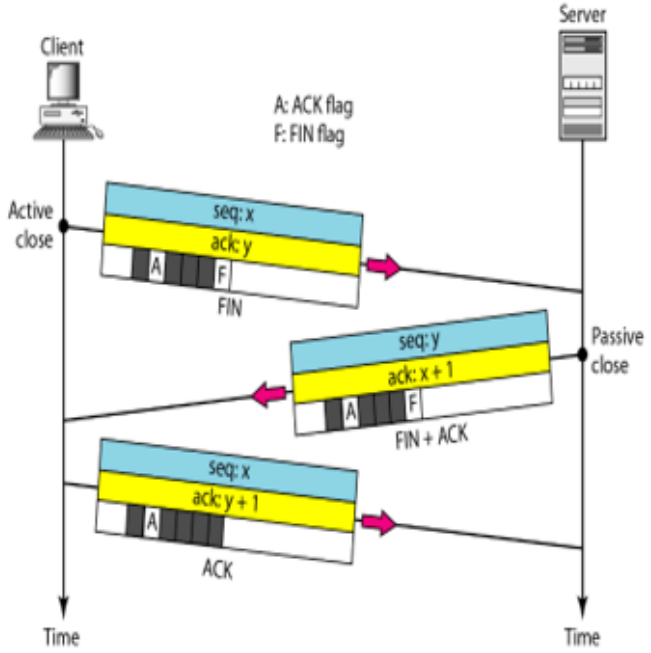
**After these three steps, the client and server are ready for the data communication process. TCP connection and termination are full-duplex.**

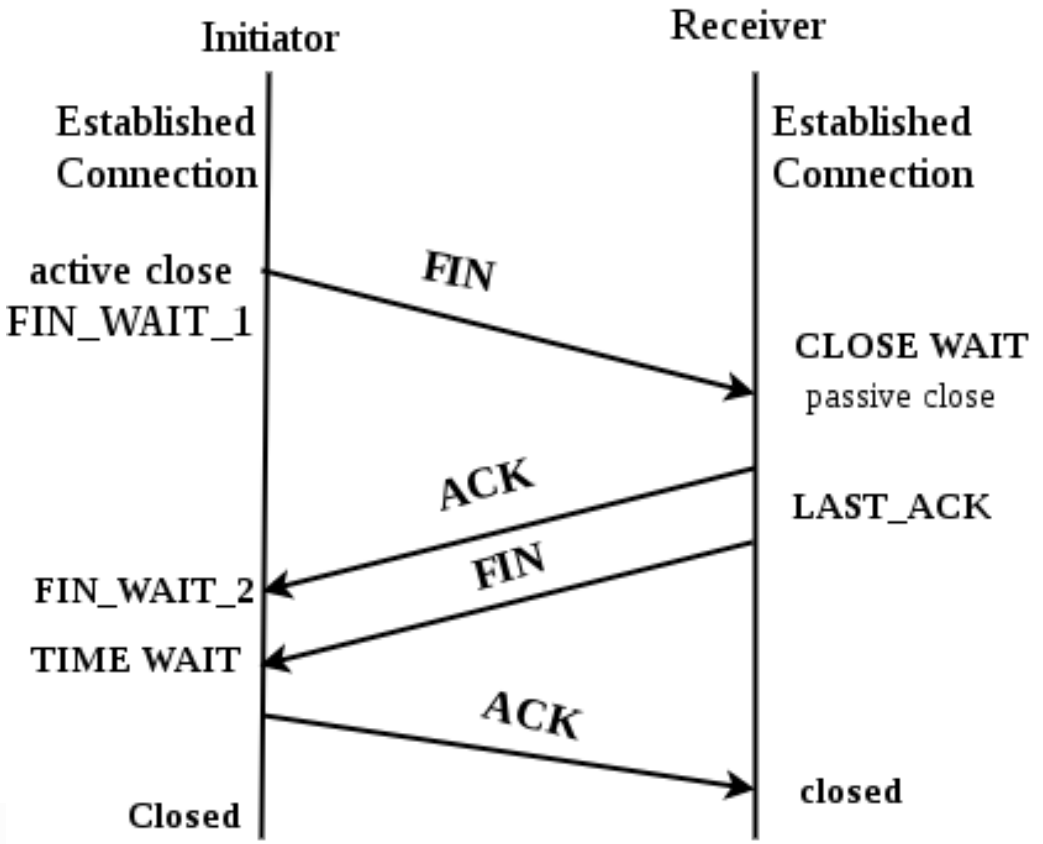
# Connection Management with TCP

## TCP: Terminating a Connection

- Each end of the data flow must be shut down independently
- If one end is done it sends a FIN segment
- This means that no more data will be sent

Figure 23.20 *Connection termination using three-way handshaking*



# Connection Management with TCP

**Step 1: FIN**

FIN refers to the **termination request** sent by the client to the server. The first FIN termination request is sent by the client to the server. It depicts the start of the termination process between the client and server.

**Step 2: ACK+ FIN**

The server sends the ACK (Acknowledgement) segment when it receives the FIN termination request. It depicts that the server is ready to close and terminate the connection.

The FIN segment is now sent by the server to the client. It is a confirmation signal that the server sends to the client. It depicts the successful approval for the termination.

**Step 3: ACK**

The client now sends the ACK (Acknowledgement) segment to the server that it has received the FIN signal, which is a signal from the server to terminate the connection. As soon as the server receives the ACK segment, it terminates the connection.

# Difference Between UDP and TCP

| Feature | TCP | UDP |
|---|---|---|
| **Connection-oriented** | Yes | No |
| **Reliability** | Reliable delivery with acknowledgment | Unreliable delivery (no acknowledgment) |
| **Ordered delivery** | Yes (Guaranteed in-order delivery) | No (Not guaranteed in-order delivery) |
| **Error checking** | Yes (Checksum for error detection) | Yes (Checksum for error detection) |
| **Flow control** | Yes (Through sliding window and congestion control) | No (No flow control mechanism) |
| **Congestion control** | Yes (Implemented with TCP window size and slow start algorithm) | No (No congestion control mechanism) |
| **Datagram size** | Variable | Fixed |
| **Header overhead** | Higher (Additional header fields for sequencing ,ack and flow control) | Lower (Minimal header overhead) |
| **Usage** | Suitable for applications requiring reliable and ordered delivery (e.g., web browsing, email, file transfer) | Suitable for real-time applications requiring low latency and high speed (video streaming, online gaming) |
| **Examples** | HTTP, FTP, SMTP, Telnet | DNS, DHCP, VoIP (Voice over IP), streaming media |

# Port Addressing: Ports and Sockets

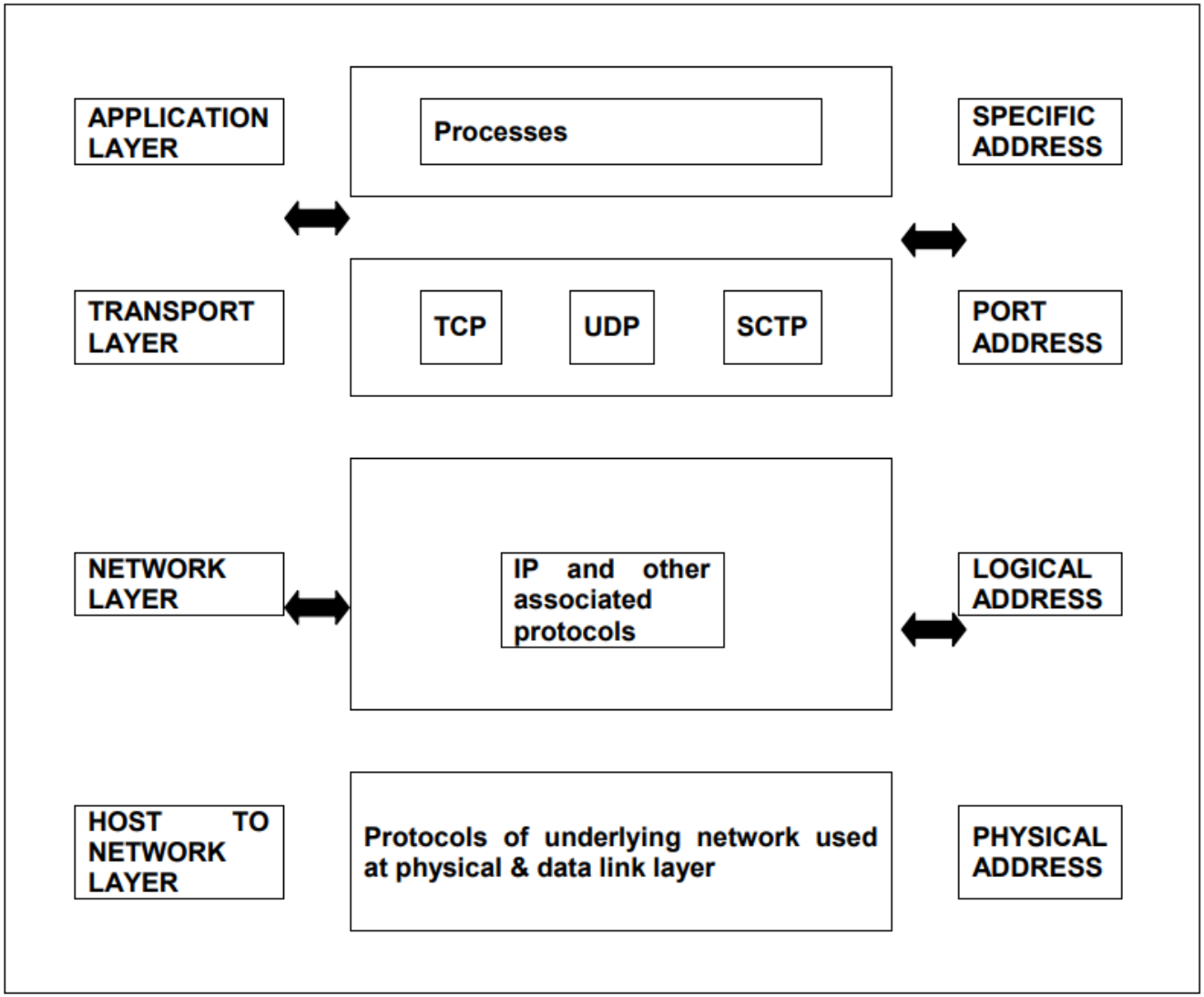
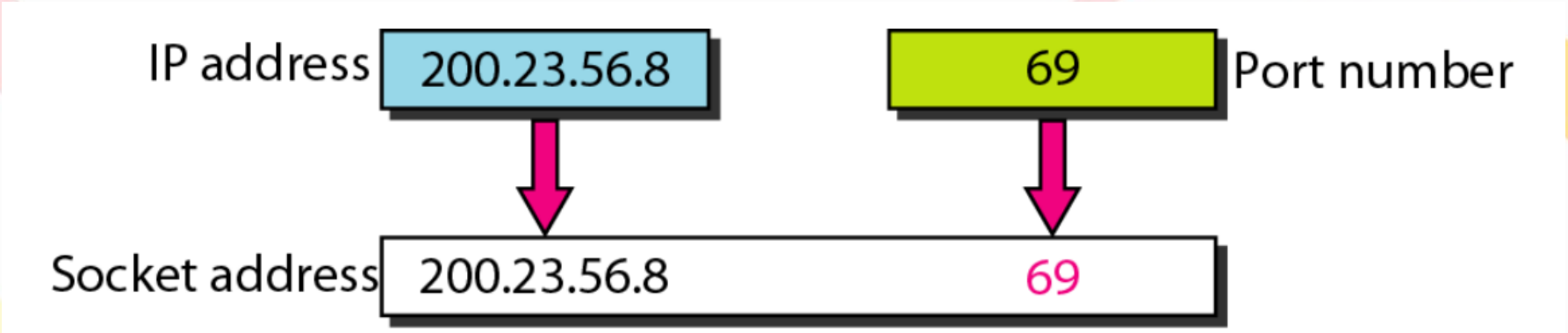


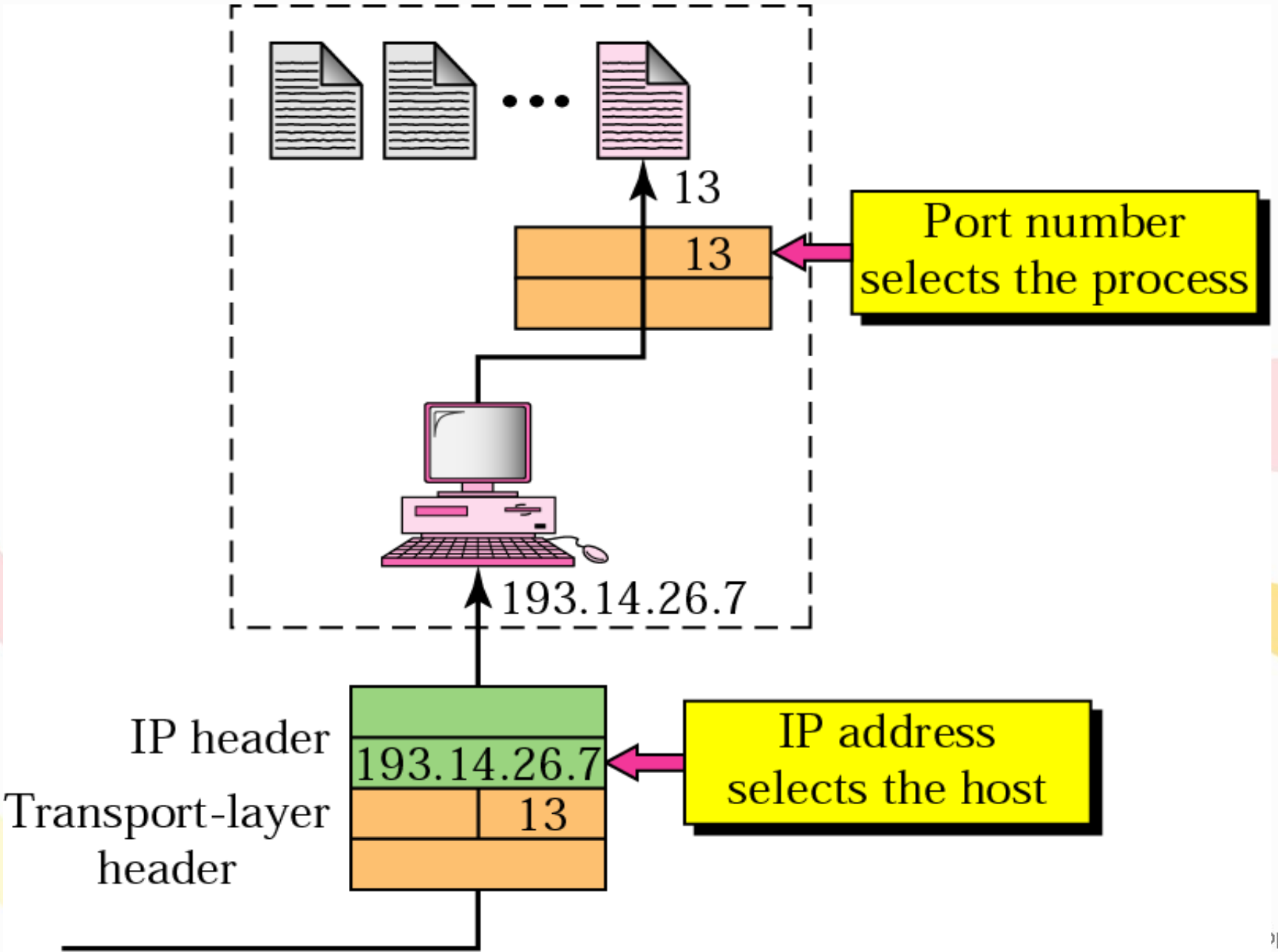

**Fig: Addressing in TCP/IP model**

# Port Addressing: Ports and Sockets

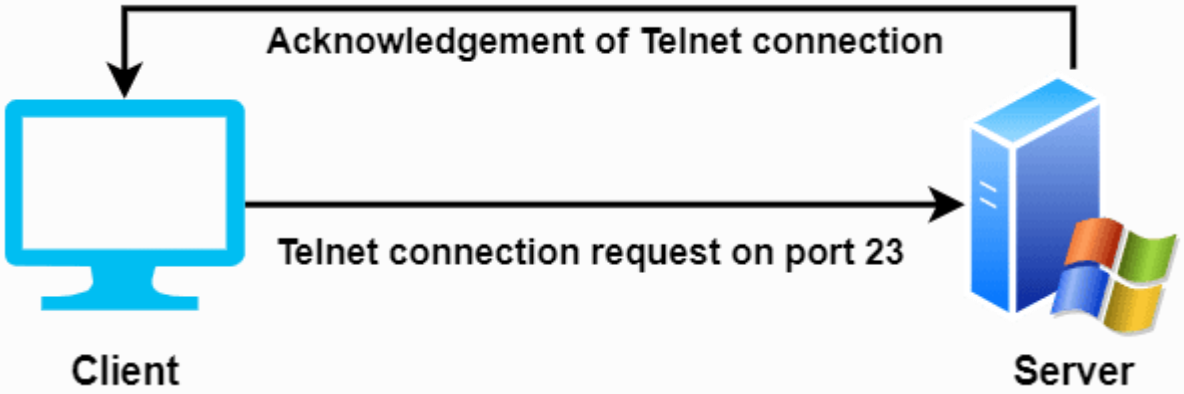
- Addressing in Transport Layer is the port Address
- Data generated by an application on one machine must be transmitted to the correct application on another machine. In this case, addressing is provided by the transport layer
- The transport layer provides the user address which is specified as a station or port
- The comnination of IP address and Port Address is called Socket Address



# Port Addressing



# Ports




- A port is a logical identifier assigned to a process in order to identify that process uniquely in a network system.
- When two network devices communicate, they do so by sending packets to each other.
- Each packet received by a receiver device contains a port number that uniquely identifies the process where the packet needs to be sent.
- Protocols like TCP, UDP, HTTP utilize a port for communication.
- Example: A client computer is requesting the server for a virtual connection with the port number . Telnet is a well-known protocol for establishing a remote connection over a TCP/IP and it uses port .

# Ports

## Classification of port numbers

**The port numbers are divided into three categories:**

- **Well-known ports**
- **Registered ports**
- **Dynamic ports**

| Port Number Range | Part Group |
|---|---|
| 0 to 1023 | Well Known (Contact) Ports |
| 1024 to 49151 | Registered Ports |
| 49152 to 65535 | Private and/or Dynamic Ports |

**Well Known Port**

- It is from the range 0 to 1023
- It is reserved for common and specifically used service
- It is used by some widely adopted protocols and services like HTTP(port 80), FTP(port 21), DNS(Port 53), SSH(port 22), etc.....

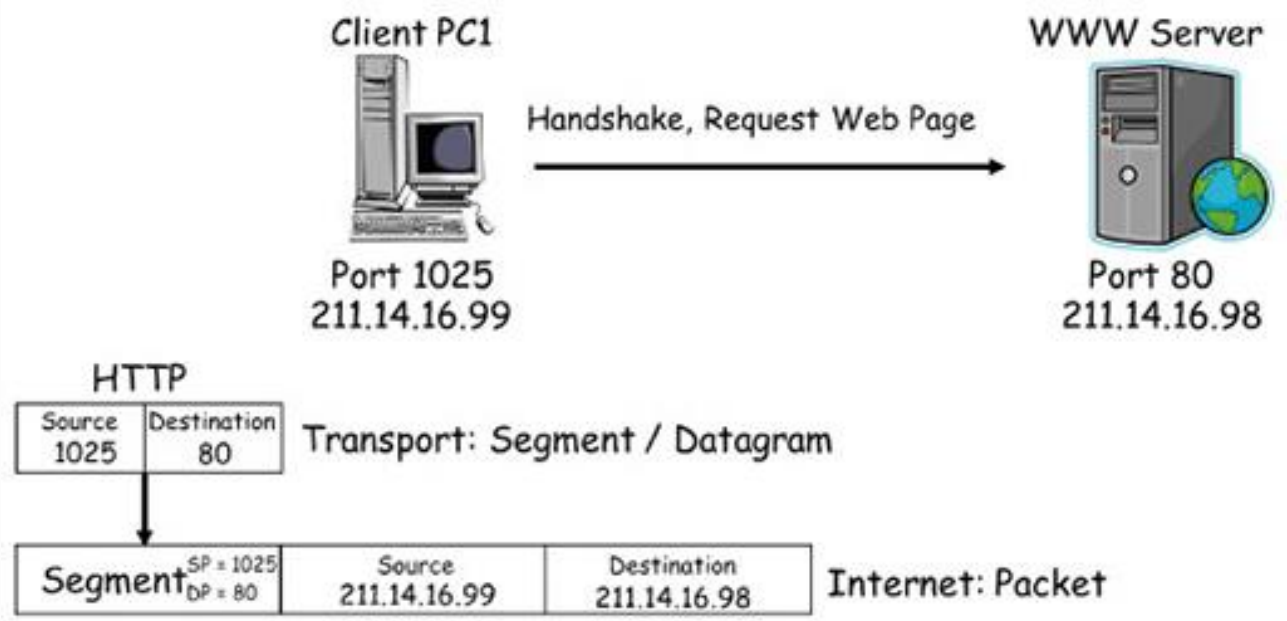
**Registered Port**

- It is from range 1024 to 49151
- These are used by applications or services that are not as common
- But it is used by those applications or services which require its specific port
- Organizations can ask IANA(Internet Assigned Number Authority) for any specific port number within this range

**Dynamic Port**

- It is from range 49152 to 65535
- It is also known as Ephemeral or Private Port
- It is used for those connections that are temporary or short-lived
- It is not registered or assigned and can be used by any process

# Sockets




- Two processes that are running on a computer or running on two different systems can communicate via a socket.
- A socket works as an inter-process communicator and seen as the endpoint of the process communication.
- A socket consists of the IP address of a system and the port number of a program within the system.
- The IP address corresponds to the system and the port number corresponds to the program where the data needs to be sent:

# Sockets

**Sockets can be classified into three categories: stream, datagram, and raw socket.** Stream sockets use connection-oriented network point to send and receive data. This type of sockets generally utilizes TCP to permit processes to communicate with each other.

Datagram sockets use connectionless network protocols like UDP to allow process communication. Raw sockets are datagram oriented and allow the processes to use ICMP for communication purpose.

| Port | Socket |
|---|---|
| Port specifies a number that is used by a program in a computer. | A socket is a combination of IP address and port number. |
| A program running on different computers can use the same port number. Hence port numbers can't be used to identify a computer uniquely. | It identifies a computer as well as a program within the computer uniquely. |
| Port number is used in the transport layer. | Sockets are involved in the application layer. A socket is an interface between the transport and application layer. |
| Port uses a socket to drop the data to a correct application. | A server and a client uses a socket to keep an eye on the data request and responses. |

# Flow Control in Transport Layer

- Flow control is a technique for optimizing the exchange of data between systems.
  - If too much data is sent at once, the receiving node can become overwhelmed and start dropping packets
  - If too little data is sent the receiver sits idle waiting for more data to arrive.
- Flow control in transport layer ensures the delivery of the message globally, as the two points of connection over this protocol are logically connected.
- Flow Control basically means that TCP will ensure that a sender is not overwhelming a receiver by sending packets faster than it can consume.

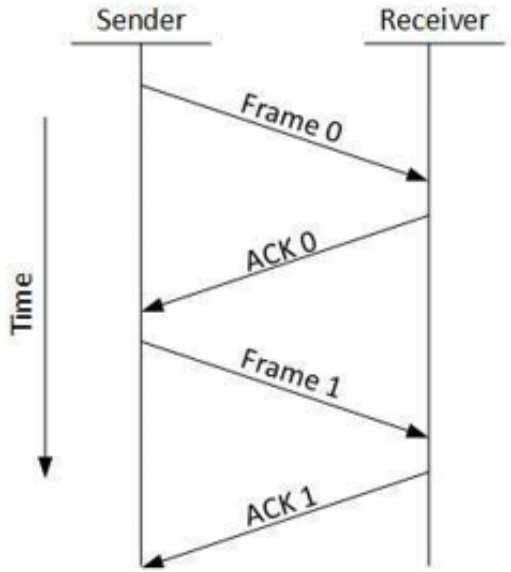
**The major Techniques used in flow control are**

1. **buffering and**
2. **windowing**

# Flow Control in Transport Layer

The major approach used for Flow control are

1. Stop and Wait - This flow control mechanism forces the sender after transmitting a data frame to stop and wait until the acknowledgement of the data-frame sent is received.

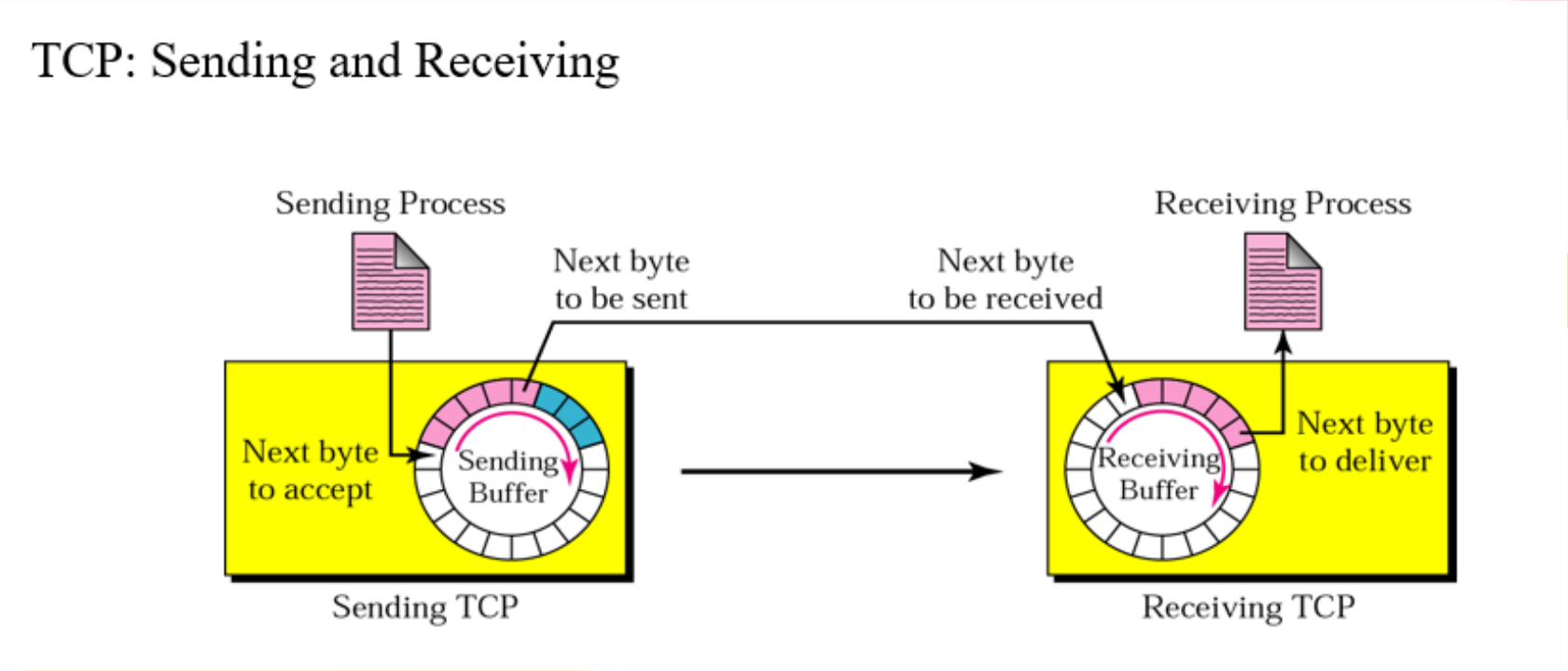



2. Sliding Window - In this flow control mechanism, both sender and receiver agree on the number of data-frames after which the acknowledgement should be sent. As we learnt, stop and wait flow control mechanism wastes resources, this protocol tries to make use of underlying resources as much as possible.

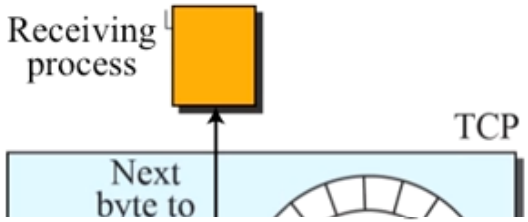
**Detail: Same as of Data Link Layer**

# Buffering in Transport Layer

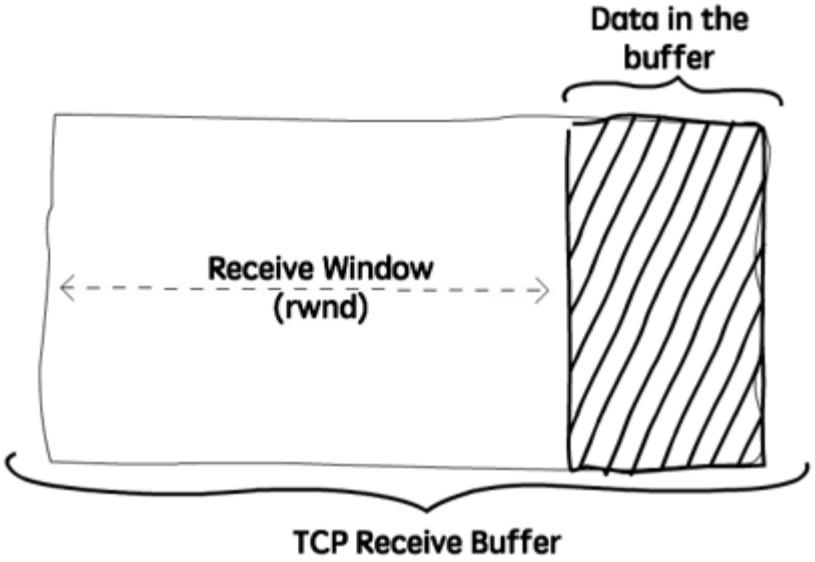
- Buffering in the transport layer is a mechanism used to temporarily store data packets while waiting for transmission, processing, or delivery.
- It plays a crucial role in flow control by managing the flow of data between sender and receiver, ensuring efficient and reliable communication.



# Buffering in Transport Layer


- Flow Control is all about making sure we don't send more packets when the receive buffer is already full, as the receiver wouldn't be able to handle them and would need to drop these packets.
- To control the amount of data that TCP can send, the receiver will advertise its *Receive Window (rwnd)*, that is, the spare room in the receive buffer.
- Every time TCP receives a packet, it needs to send an ack message to the sender, acknowledging it received that packet correctly, and with this ack message it sends the value of the current receive window, so the sender knows if it can keep sending data.

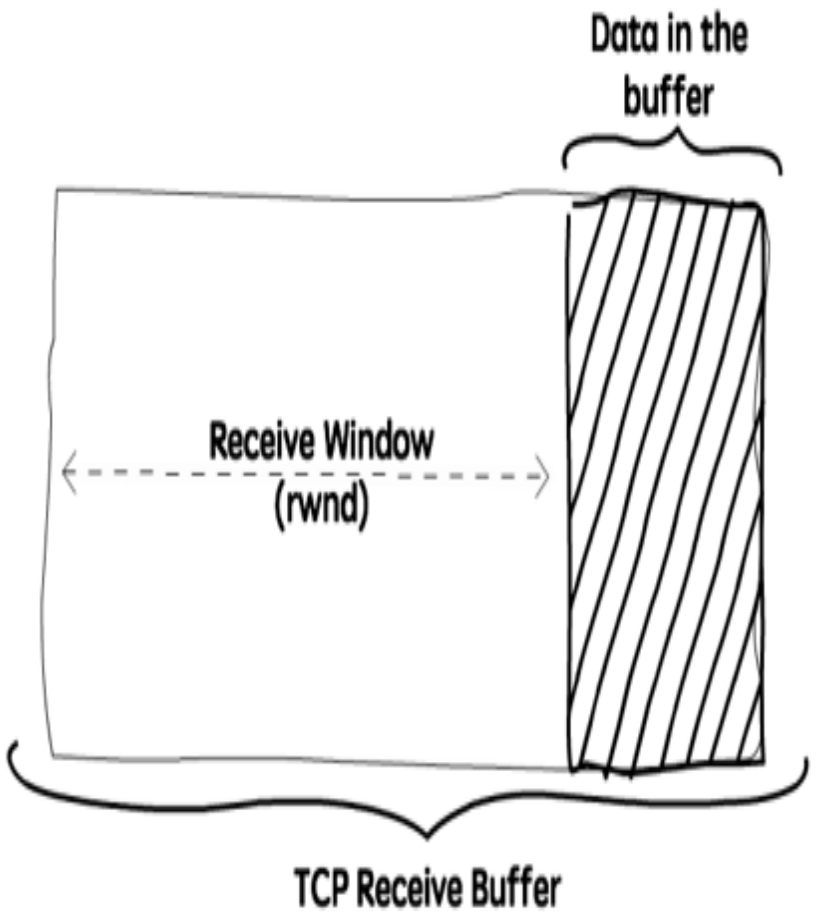




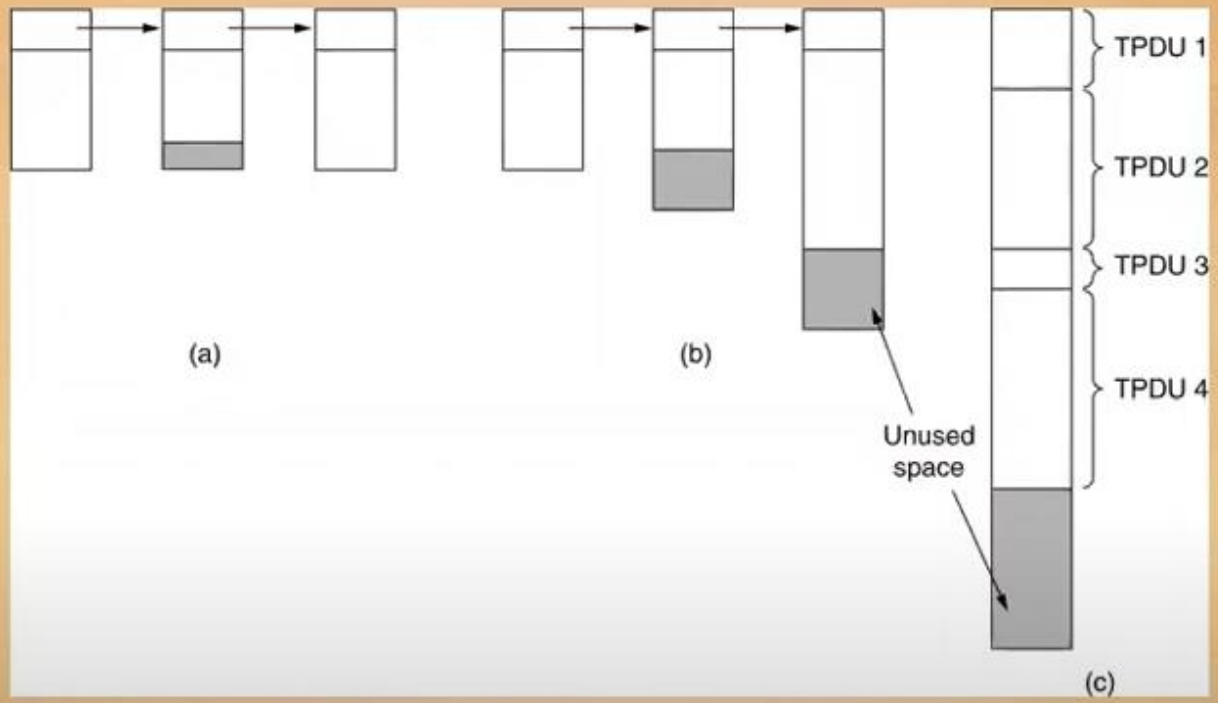
- TCP stores the data it needs to send in the ***send buffer***, and the data it receives in the ***receive buffer***. When the application is ready, it will then read data from the receive buffer.

# Buffering in Transport Layer

- Flow Control is all about making sure we don't send more packets when the receive buffer is already full, as the receiver wouldn't be able to handle them and would need to drop these packets.
- To control the amount of data that TCP can send, the receiver will advertise its *Receive Window (rwnd)*, that is, the spare room in the receive buffer.
- Every time TCP receives a packet, it needs to send an ack message to the sender, acknowledging it received that packet correctly, and with this ack message it sends the value of the current receive window, so the sender knows if it can keep sending data.



# Types of Buffer




(a) Chained fixed-size buffers. (b) Chained variable-sized buffers.
(c) One large circular buffer per connection.

# Types of Buffer

## Fixed Size Buffers (a)

- If the buffer size is chosen to be equal to the largest possible segment, space will be wasted whenever a short segment arrives. If the buffer size is chosen to be less than the maximum segment size, multiple buffers will be needed for long segments, with the attendant complexity.

## Variable size buffers (b)

- Another approach to the buffer size problem is to use variable-sized buffers, as in (b). The advantage here is better memory utilization, at the price of more complicated buffer management.

## Circular Buffer (c)

- A third possibility is to dedicate a single large circular buffer per connection, as in Fig(c). This system is simple and elegant and does not depend on segment sizes, but makes good use of memory only when the connections are heavily loaded.

# Flow control in TCP

**In TCP, Flow control is maintained via Windowing by following mechanisms**

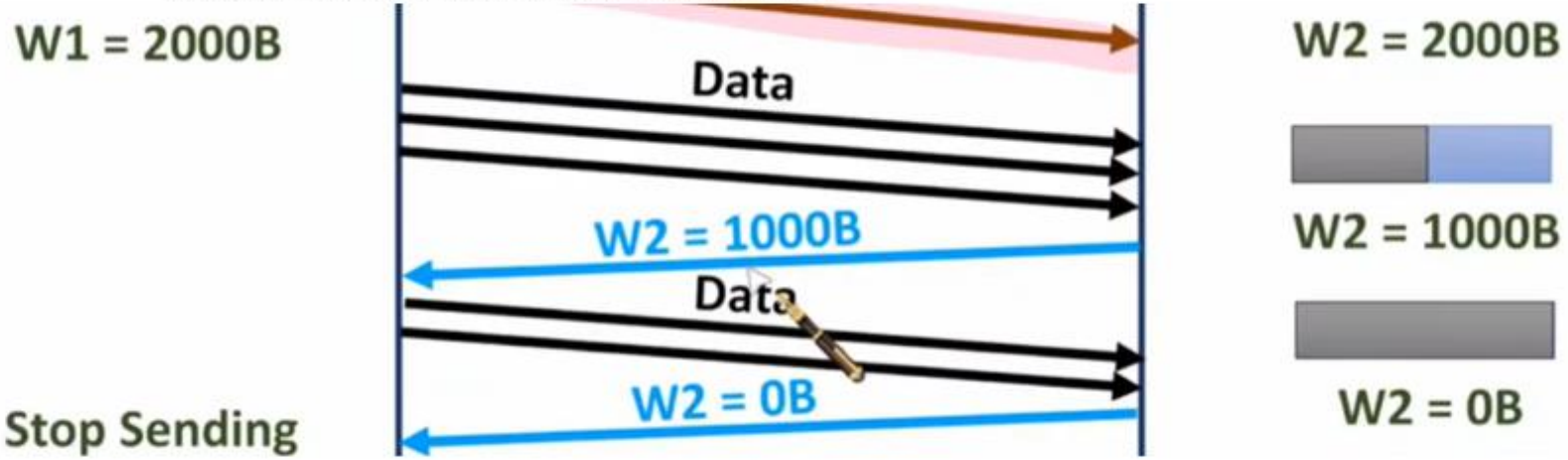
1. **Window Scaling**
   - Sender and Receiver Dynamically adjust/scale the window size (minimum of sender or receiver)


In the example window size of H1=3000 and H2=2000
H1 scale the window size to 2000 to match with the H2

# Flow control in TCP

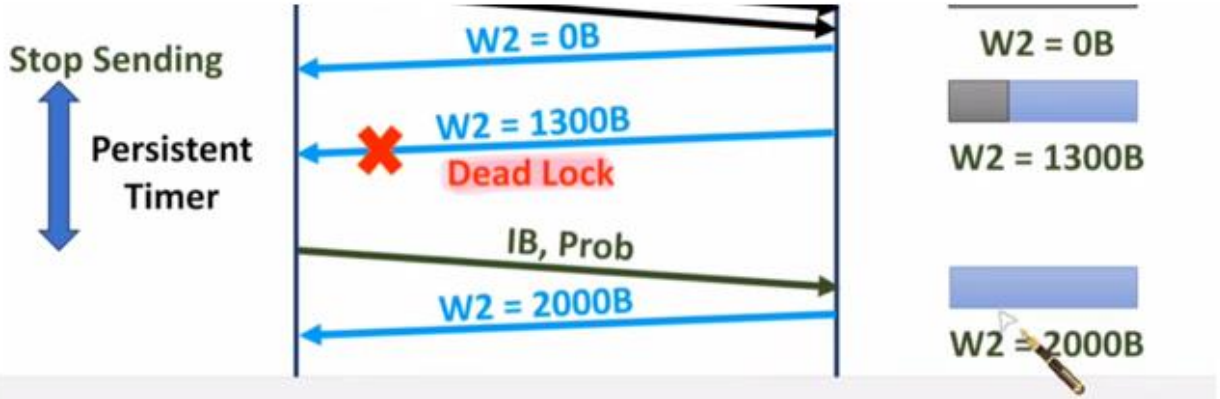
## 2. Advertisment Window size

- Window size are advertised by hosts during transmission of data to inform other side about the availability of the buffer

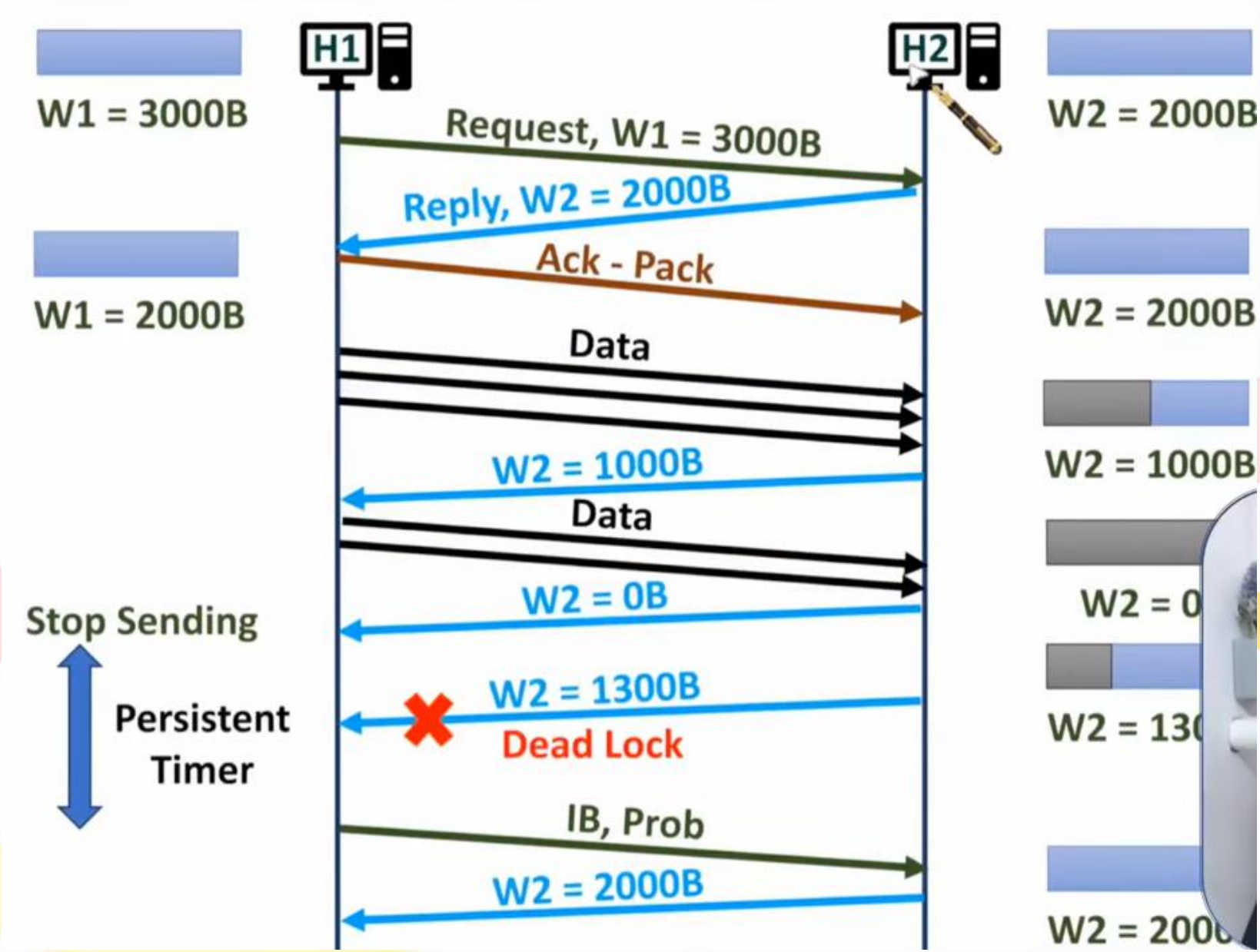



## 3. Persistent Timer

- To avoid the deadlock hosts sends the probe message to ask the available buffer from other hosts



# Flow control in TCP

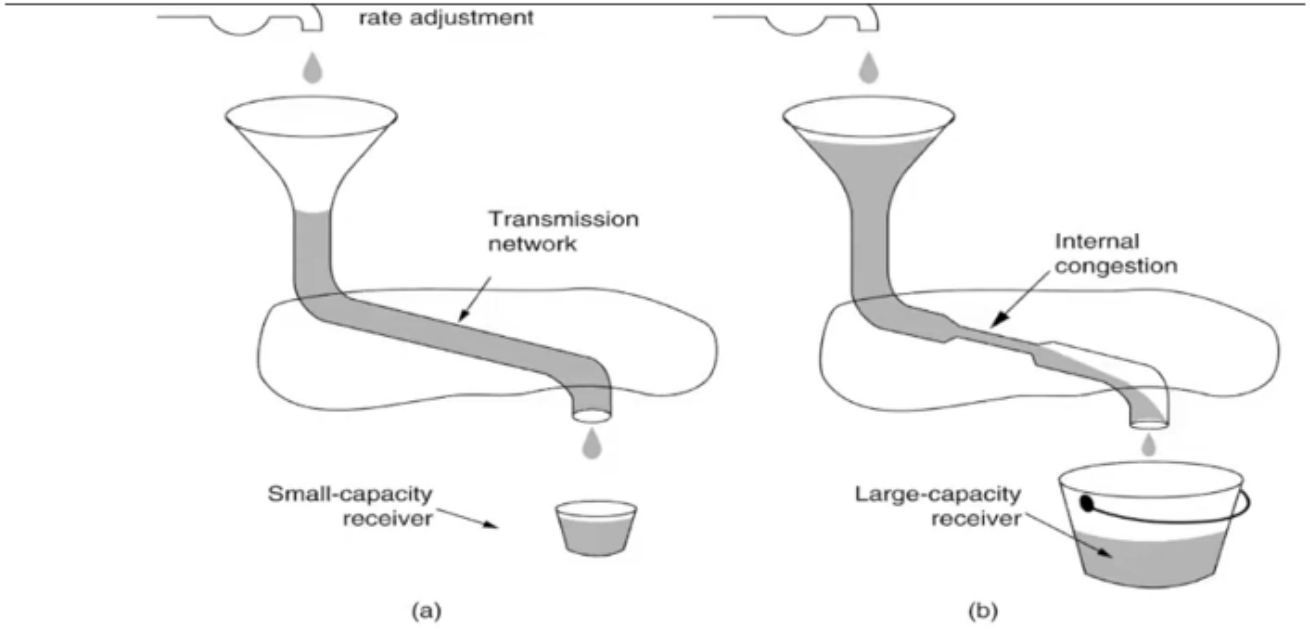


# CONGESTION CONTROL

- Congestion control in the transport layer is a set of techniques and mechanisms used to manage and alleviate congestion in computer networks.
- It ensures that the network operates efficiently without becoming overloaded, which can lead to packet loss, increased latency, and decreased throughput.

Congestion in the network happen when

- Network Capacity is High but Receiver capacity is low
- Network Capacity is Low but Receiver capacity is high




(a) A fast network feeding a low capacity receiver.
(b) A slow network feeding a high-capacity receiver.

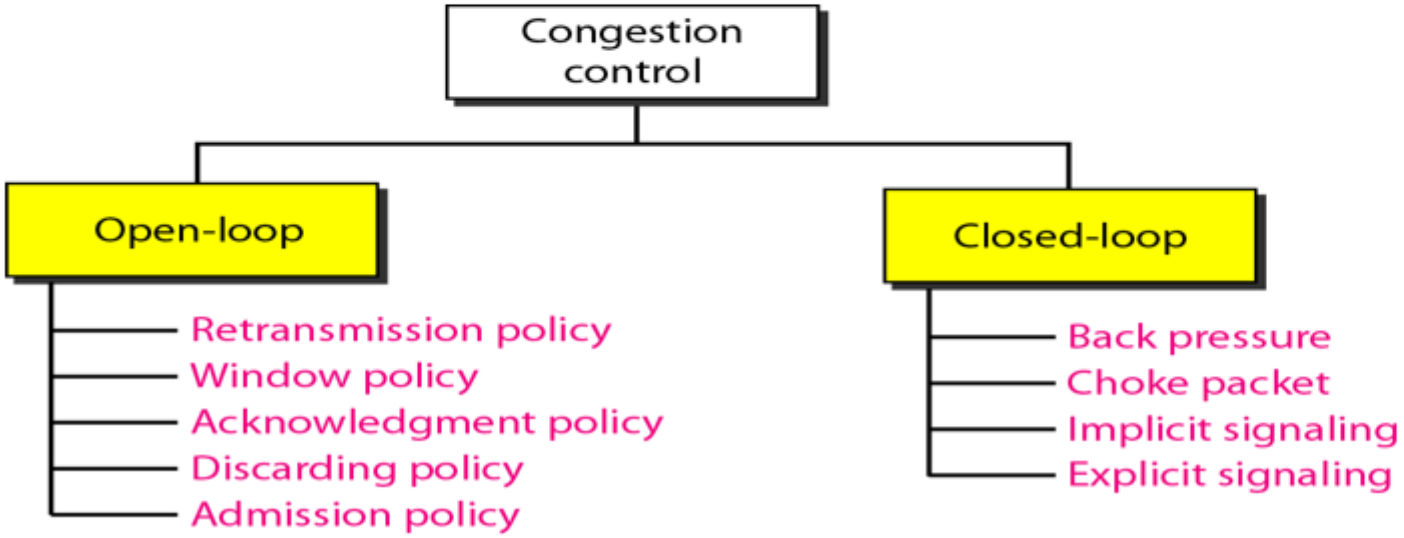
# Goals/Benefits of CONGESTION CONTROL

1. **Preventing Network Congestion:** by regulating the rate of data transmission
2. **Maintaining Quality of Service (QoS):** It ensures that the network can deliver data with acceptable levels of delay, jitter, and packet loss,
3. **Efficiency:** By optimizing network utilization and minimizing packet loss and delays, congestion control improves the overall efficiency
4. **Improved Performance:** improves network performance
5. **Fair Resource Allocation:** It ensures fair allocation of network resources among competing users and flows, preventing any single flow from monopolizing bandwidth.
6. **Enhanced Reliability:** By minimizing packet loss and delays, congestion control enhances the reliability of data transmission
7. **Optimized Utilization:** Congestion control optimizes network utilization by dynamically adjusting the transmission rate

# Congestion Control Mechanisms

- Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.

1. Open Loop – congestion Prevention
   - policies are applied to prevent congestion before it happens
   - In these mechanisms, congestion control is handled by either the source or the destination
2. Closed Loop- Congestion Removal
   - Closed-loop congestion control mechanisms try to alleviate congestion after it happens



# Open Loop/ Congestion Prevention Mechanisms

- **Retransmission Policy**
  - If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted
  - Retransmission in general may increase congestion in the network, However, a good retransmission policy can prevent congestion.
  - The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion
- **Window-based Flow Control:**
  - TCP uses window-based flow control mechanisms such as sliding window protocol to regulate the rate of data transmission based on the available network capacity and the receiver's ability to process data.
  - Generally Selective Repeat ARQ is used for Congestion Control.

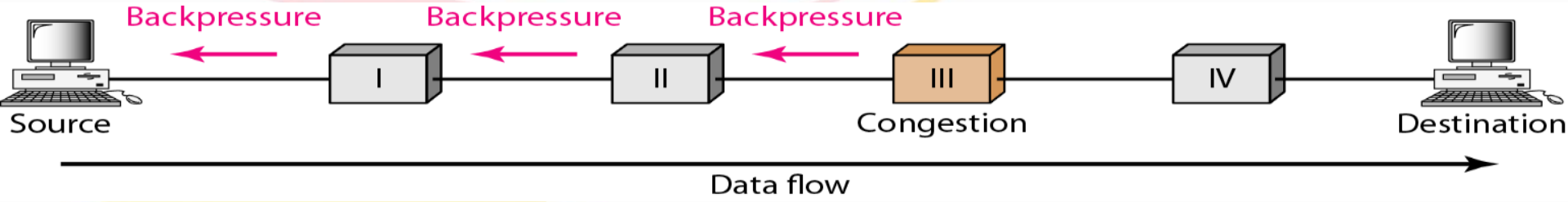
# Open Loop/ Congestion Prevention Mechanisms

- **Acknowledgment Policy**
  - If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion
  - A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires
  - Sending fewer acknowledgments means imposing less load on the network.
- **Discarding Policy**
  - Routers and switches implement packet discard policies to selectively drop packets before congestion occurs, preventing network congestion.
  - For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.
- **Admission Policy**
  - Resources are checked first before admission in the network
  - If resources are free then only the packets are admitted
  - A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility of future congestion

# Closed Loop/ Congestion Removal Mechanisms

## Back pressure

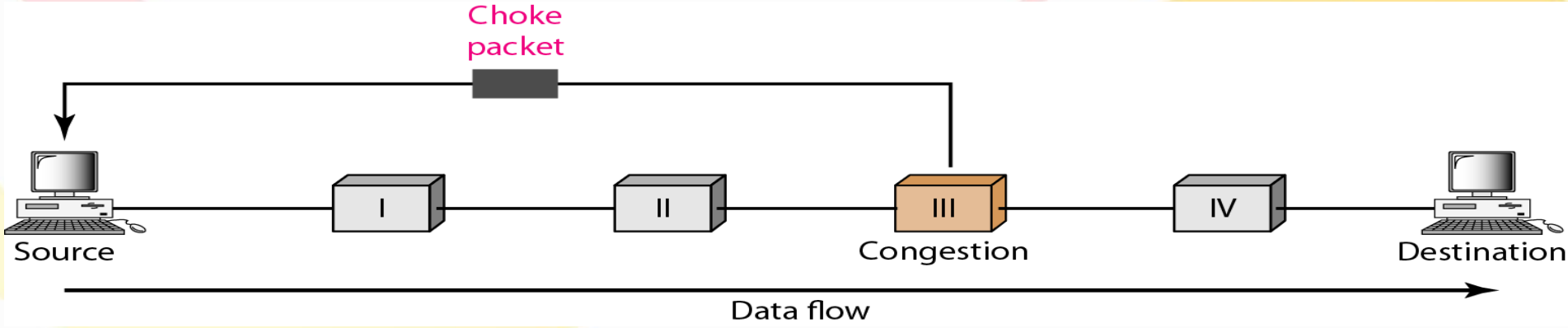
- The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes
- This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes, and so on
- Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source



# Closed Loop/ Congestion Removal Mechanisms

## Choke packet

- A choke packet is a packet sent by a node to the source to inform it of congestion
- In backpressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station
- In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly
- The intermediate nodes through which the packet has traveled are not warned and takes no action.



# Closed Loop/ Congestion Removal Mechanisms

**Implicit Signaling**

- In implicit signaling, there is no communication between the congested node and the source
- The source guesses that there is a congestion somewhere in the network from other symptoms
- For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested→ the source should slow down

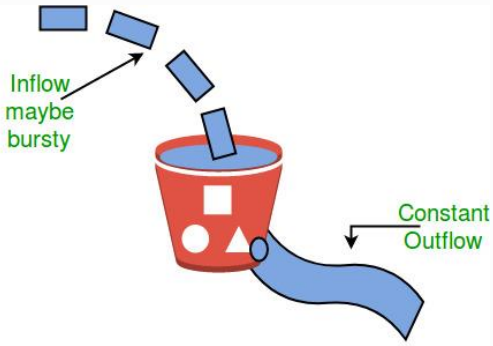
**Explicit Signaling**

- The node that experiences congestion can explicitly send a signal to the source or destination
- The explicit signaling method, however, is different from the choke packet method
- In the choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data

# Congestion control techniques: Traffic Policing and Shaping

- Traffic policing and shaping techniques are used to control the rate of incoming and outgoing traffic, ensuring that traffic flows conform to predefined traffic profiles and bandwidth limits.
- It is also measure of Congestion control
- Two techniques can shape traffic:
  - **Leaky bucket**
  - **Token bucket**

# Leaky Bucket



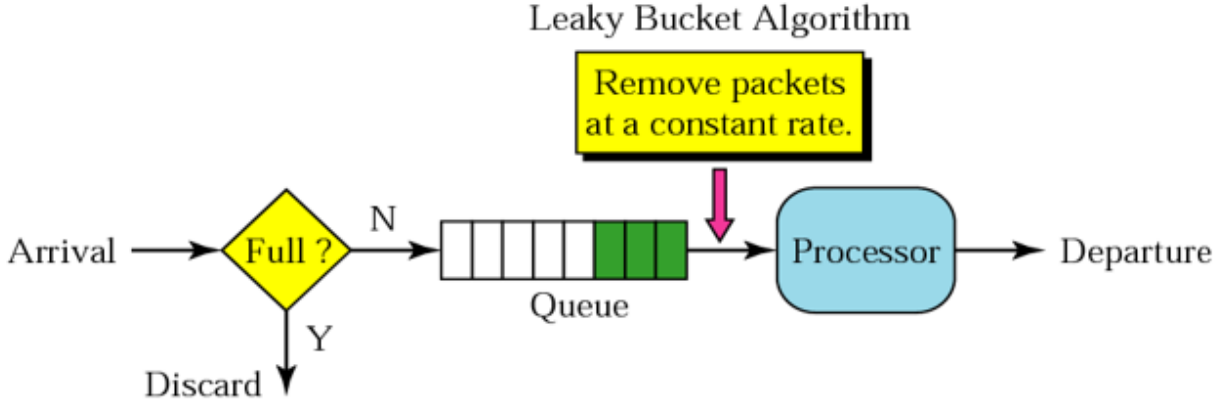

- Imagine a bucket with a small hole in the bottom.
- The water leaks from the bucket at a constant rate as long as there is water in the bucket
- The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty
- The input rate can vary, but the output rate remains constant
- Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic
- Bursty chunks are stored in the bucket and sent out at an average rate
- A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full

# Leaky Bucket Algorithm

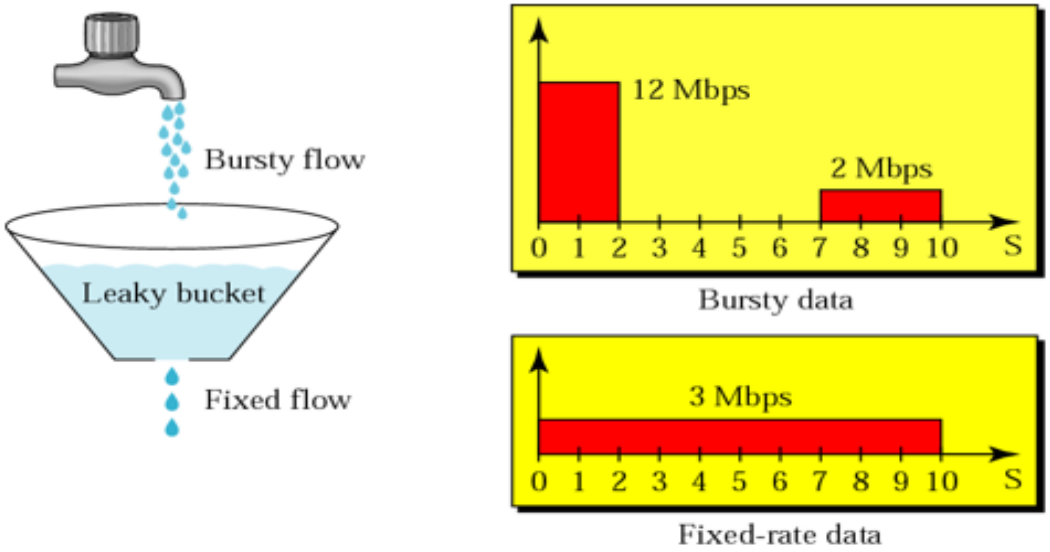
## Leaky Bucket Algorithm

Each network interface contains a leaky bucket and the following steps are involved in leaky bucket algorithm:

1. When host wants to send packet, packet is thrown into the bucket.
2. The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
3. Bursty traffic is converted to a uniform traffic by the leaky bucket.
4. In practice the bucket is a finite queue that outputs at a finite rate.



# Leaky Bucket Example




- In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host
- The use of the leaky bucket shapes the input traffic to make it conform to this commitment
- the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data
- The host is silent for 5s and then sends data at a rate of 2 Mbps for 3s, for a total of 6 Mbits of data
- In all, the host has sent 30 Mbits of data in 1O s
- The leaky bucket smoothers the traffic by sending out data at a rate of 3 Mbps during the same 10 s
- Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host
- We can also see that the leaky bucket may prevent congestion

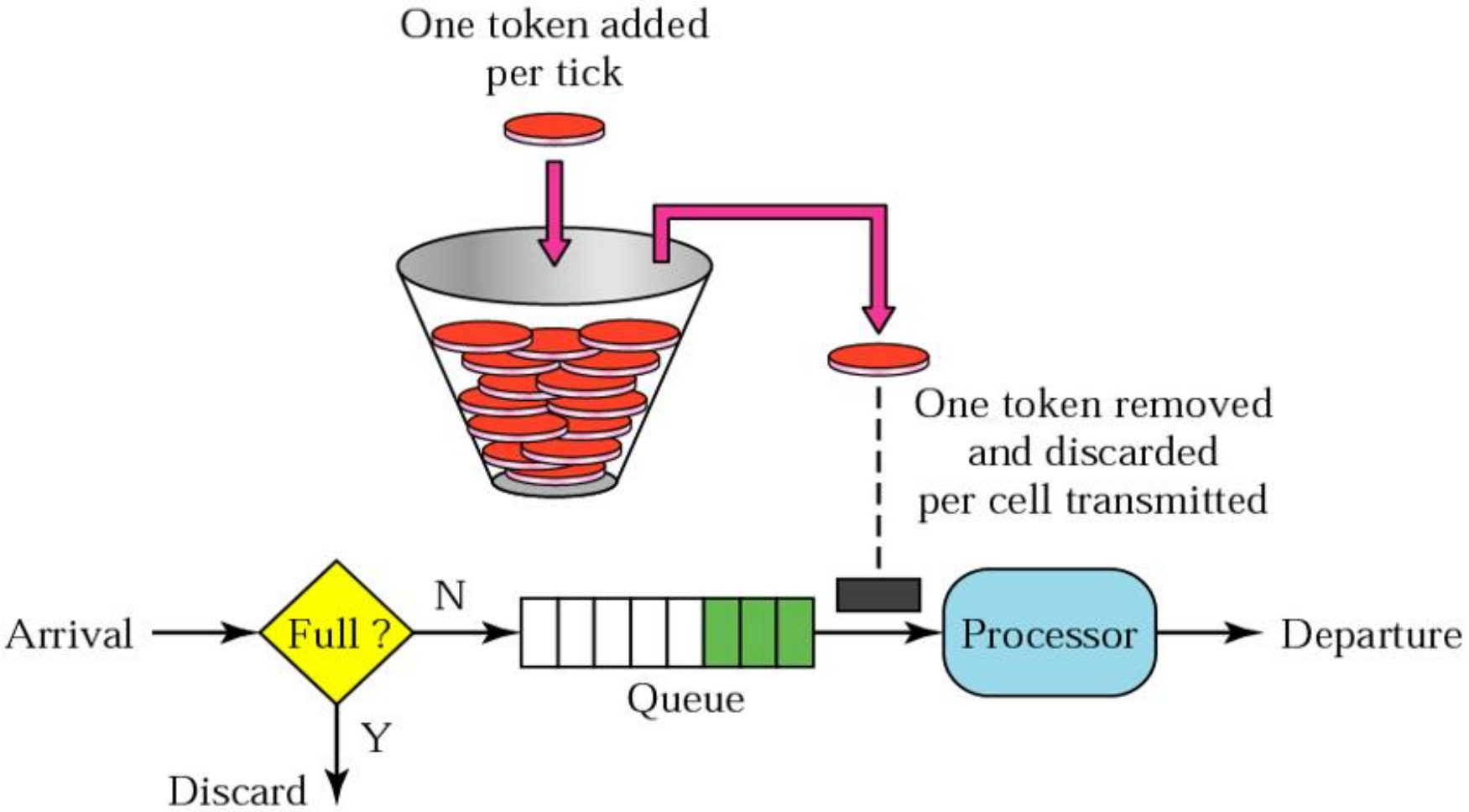
# Token Bucket

- The leaky bucket algorithm has a rigid output design at an average rate independent of the bursty traffic.
- In some applications, when large bursts arrive, the also need to speed up. This calls for a more flexible algorithm
- Therefore, a token bucket algorithm finds its uses in network traffic shaping or rate-limiting.
- It is a control algorithm that indicates when traffic should be sent.
- The bucket contains tokens. Each of the tokens defines a packet of predetermined size.
- When tokens are shown, a flow to transmit traffic appears in the display of tokens.
- No token means no flow sends its packets.
- Hence, a flow transfers traffic up to its peak burst rate in good tokens in the bucket.

# Token Bucket

**Tocken Bucket Algorithm**

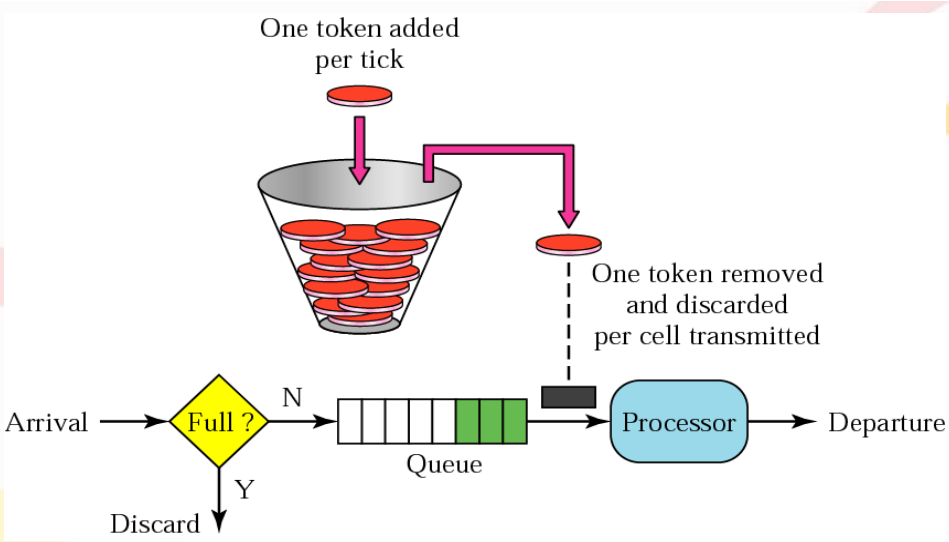
1. In regular intervals tokens are thrown into the bucket. ƒ
2. The bucket has a maximum capacity. ƒ
3. If there is a ready packet, a token is removed from the bucket, and the packet is sent.
4. If there is no token in the bucket, the packet cannot be sent.



# Token Bucket Example

**For example**

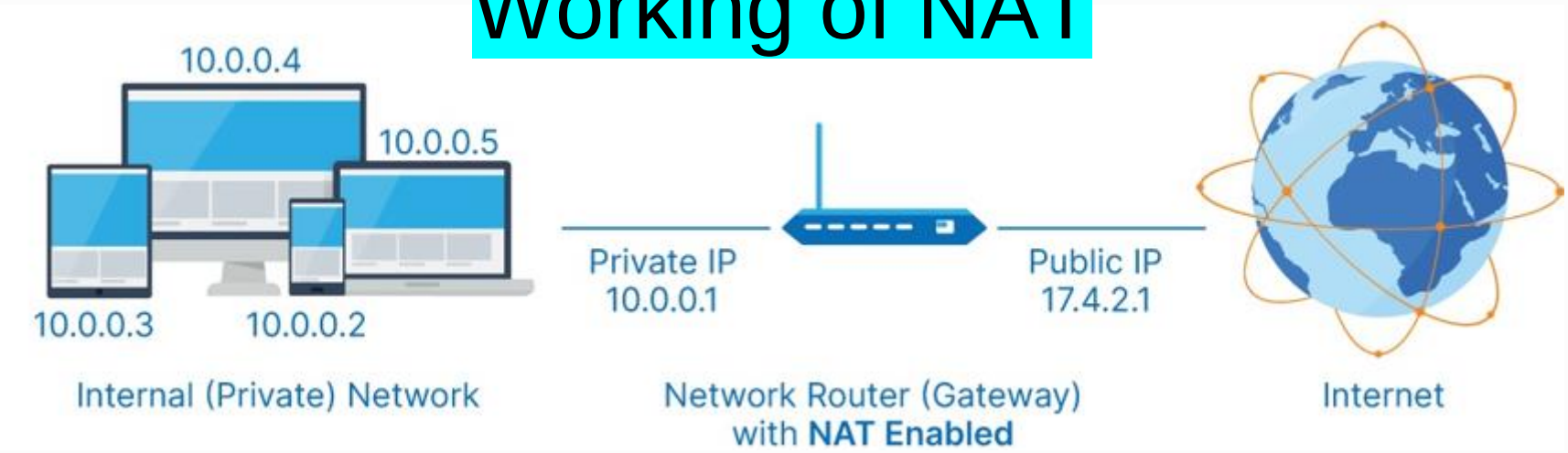
- For each tick of the clock, the system sends *n* tokens to the bucket
- The system removes one token for every cell (or byte) of data sent
- if n is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens
- Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick
- In other words, the host can send bursty data as long as the bucket is not empty
- The token bucket can easily be implemented with a counter
- The token is initialized to zero
- Each time a token is added, the counter is incremented by 1
- Each time a unit of data is sent, the counter is decremented by 1
- When the counter is zero, the host cannot send data



# IP Remapping:NAT

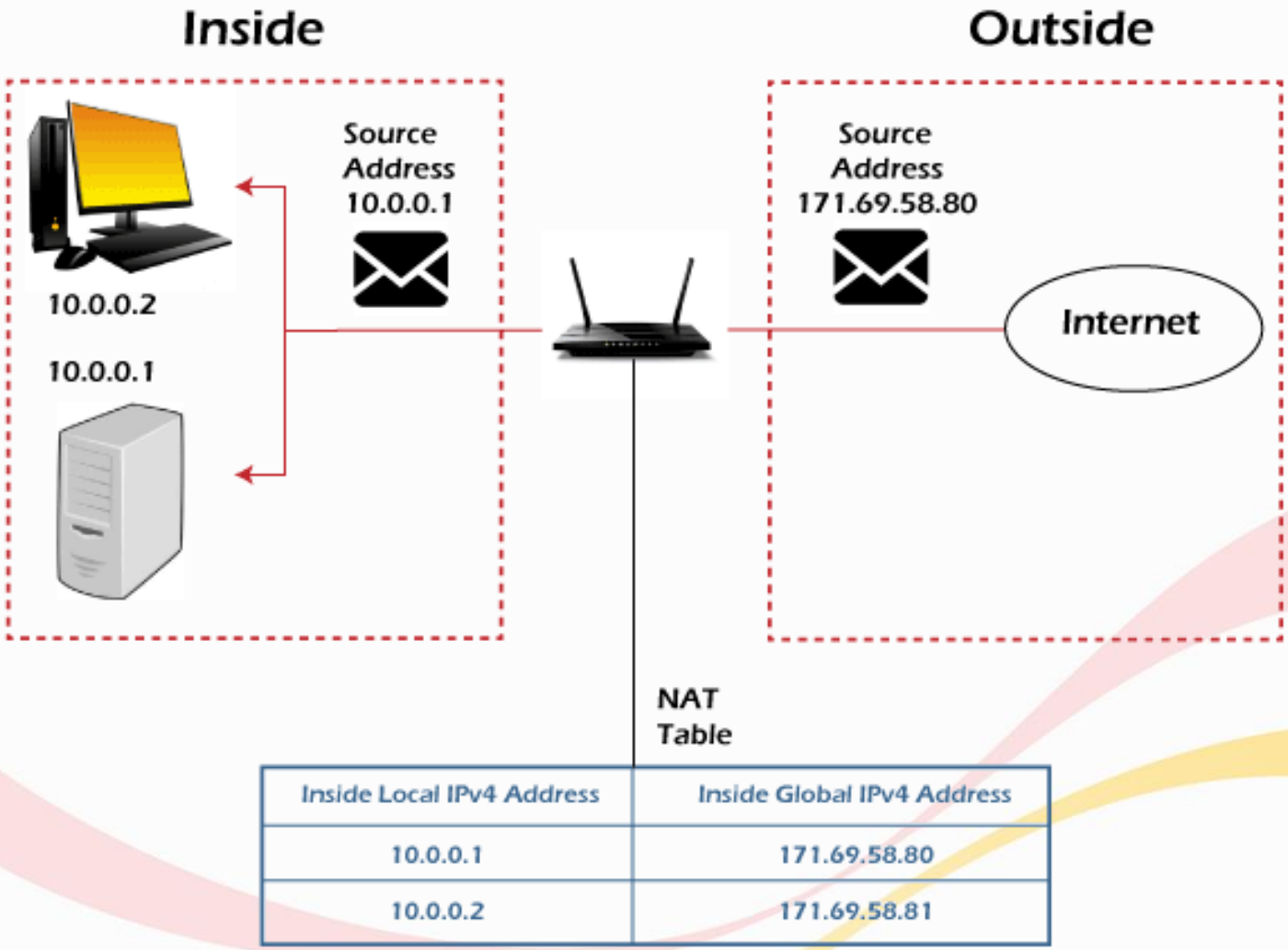
- To access the Internet, one public IP address is needed, but we can use a private IP address in our private network.
- The idea of NAT is to allow multiple devices to access the Internet through a single public address. To achieve this, the translation of a private IP address to a public IP address is required.
- **Network Address Translation (NAT)** is a process in which one or more local IP address is translated into one or more Global IP address and vice versa in order to provide Internet access to the local hosts.
- NAT (Network Address Translation) is a networking technique used to modify network address information in packet headers typically between private and public networks.
- Private and Public IP Addresses: NAT is commonly used in networks where private IP addresses (e.g., 192.168.x.x or 10.x.x.x) are assigned to devices within the local network, while public IP addresses are used for communication with external networks such as the internet.

# Working of NAT




- Generally, the border router is configured for NAT i.e the router which has one interface in the local (inside) network and one interface in the global (outside) network.
- When a device from the private network initiates communication with an external network, the NAT router modifies the source IP address and port number in the packet header to its own public IP address and a dynamically allocated port number.
- When the response packet is received from the external network, the NAT router translates the destination IP address and port number back to the original private IP address and port number of the requesting device.

# Working of NAT




| Inside Local IPv4 Address | Inside Global IPv4 Address |
|---|---|
| 10.0.0.1 | 171.69.58.80 |
| 10.0.0.2 | 171.69.58.81 |

**Mapping Table:** NAT routers maintain a mapping table that keeps track of the translation between private and public IP addresses and port numbers. This table is used to route incoming response packets to the correct internal device.

# Types of NAT

1. **Static NAT:**

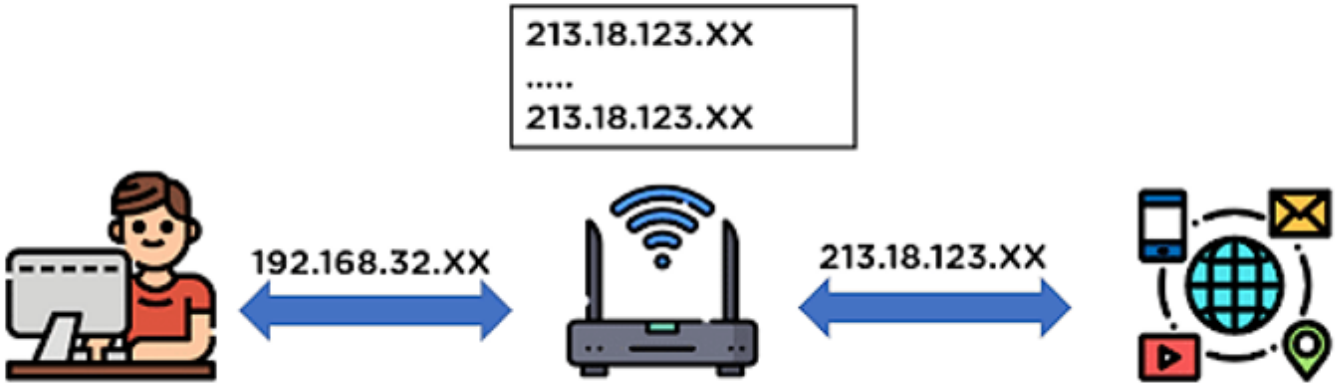
- In this NAT process, one single private network is mapped to an individual public IP address.
- Each private IP address is permanently mapped to a specific public IP address.
- This process of translation is also known as one-to-one NAT,
- Used generally for private network connections. This is generally used for Web hosting
- These are not used in organizations as there are many devices that will need Internet access and to provide Internet access, a public IP address is needed.


Static Network Address Translation

# Types of NAT

2. **Dynamic NAT**:

- For this NAT translation process, the private network address is converted to a public IP address by choosing it from a pool of public IP addresses available to the network model.
- Private IP addresses are dynamically assigned public IP addresses from a pool of available addresses.
- If the IP address of the pool is not free, then the packet will be dropped as only a fixed number of private IP addresses can be translated to public addresses.
- Suppose, if there is a pool of 2 public IP addresses then only 2 private IP addresses can be translated at a given time. If 3rd private IP address wants to access the Internet then the packet will be dropped
- This is also very costly as the organization has to buy many global IP addresses to make a pool.




Dynamic Network Address Translation

# Types of NAT

3. **PAT - Port Address Translation**:

- This translation process is configured to convert all the private IP addresses available to a single public IP address, but with a different port number assigned to each of the public addresses. Due to the process of translation, it is also known as NAT Overload.
- Allows multiple devices within the private network to share a single public IP address by using different port numbers to distinguish between connections.
- In this, many local (private) IP addresses can be translated to a single registered IP address. Port numbers are used to distinguish the traffic i.e., which traffic belongs to which IP address.
- This is most frequently used as it is cost-effective as thousands of users can be connected to the Internet by using only one real global (public) IP address.

# Benefits of NAT

- **IP Address Conservation:** NAT allows organizations to conserve public IP addresses by using private IP addresses internally and translating them to a smaller pool of public IP addresses
- **Network Segmentation:** NAT enables the creation of private networks with non-routable IP addresses, providing an additional layer of security by isolating internal devices from external networks.
- **Simplified Network Management:** NAT simplifies network management by allowing multiple devices to share a single public IP address
- **Security and Privacy:** NAT provides a level of security by hiding the internal IP addresses of devices from external networks, making them less susceptible to direct attacks from the internet.
- **Improves network performance**: By allowing multiple devices to share a single internet connection, NAT can improve network performance. This reduces the number of internet connections required and can improve bandwidth utilization.

# Disadvantages of NAT

1. **Complexity:** NAT device translates the IP addresses and port numbers of the network's devices; hence it increases the complexity factor of a network.
2. **Limited connectivity:** Network Address Translation limits the connectivity of some devices and applications because it may not support all protocols or allow certain types of connections.
3. **Reduced performance:** As Network Address Translation requires the NAT device to translate the IP addresses and port numbers of the network's devices, it can reduce the performance

END of UNIT 6

Thank You.