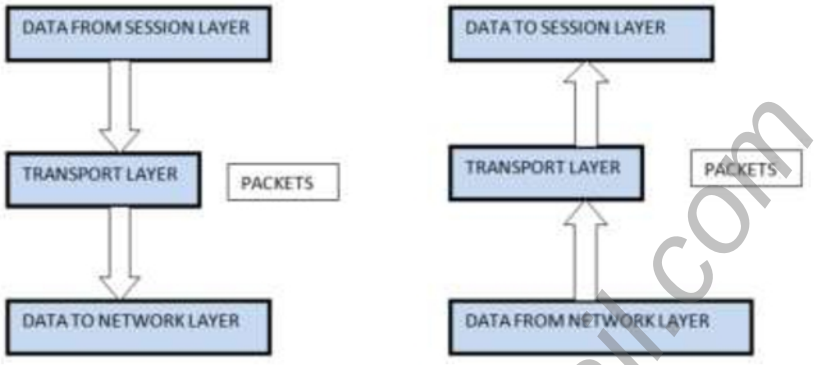
# Chapter-5
# Transport Layer Protocols

## Introduction

- Transport layer resides between the session and network layer of OSI model.
- A transport layer protocols provides for logical communication between application processes running on different host.
- The TCP/IP and UDP protocols resides at this layer.




- The primary functions of this layer are
  i. Provision of connection oriented and connectionless service
  ii. End to end data transport
  iii. Breaking large message into small segments called packets and reassembling them at destination.
  iv. Establishing and closing connections across the network
  v. Packet sequencing
  vi. Flow control
  vii. Multiplexing and DE multiplexing
  viii. Congestion control.

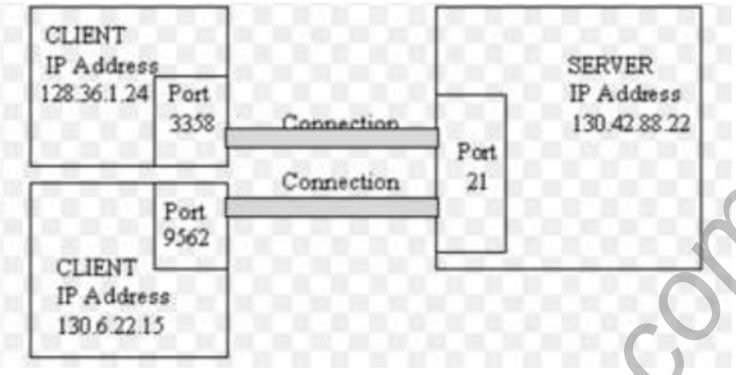
## Relationship between transport layer and network layer

- Transport layer resides just above the network layer in protocol stack.
- A transport layer provides logical communication between **processes** running on different host whereas a network layer protocol proves logical communication between **hosts**.
- A transport layer moves messages from application processes to the network edge and vice versa but it doesn't say how the message are moved within the network core.
- A computer network may have multiple transport layer protocol (TCP and UDP), with each protocol offering a different service model to application.
- A transport layer protocol can offer reliable data transfer service while the underlying network layer protocol are unreliable.

## Port number and port addressing overview

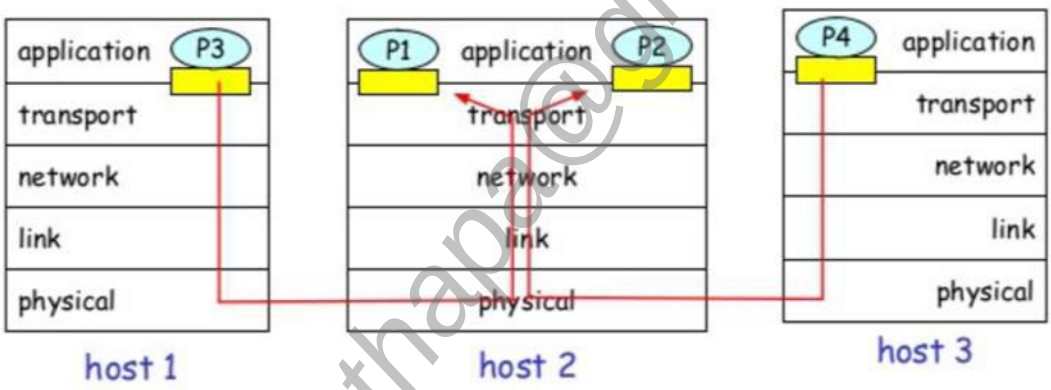
- A port is a type of programming related docking point through which information flows from a program on one computer to another computer in the network.
- So two programs running on different host connects via a port.
- Port are numbered 0 to 65535.
- Ports which are numbered from 0 to 1023 are also called as well-known port and are reserved. Example: HTTP protocol uses port number 80 and similarly FTP uses 21.
- Ports ranging from 1024 to 49151 are called registered port. They can be used under IANA (Internet Assigned Numbers Authority) registration for specific protocols by software corporations.
- Ports ranging from 49152 to 65535 can be used without any consideration. This range ports are also called as dynamic or private port.
- A port is always associated with an IP address of a host and the protocol type of the communication, and thus completes the destination or origination network address of a communication session.

- A port is identified for each address and protocol by a 16-bit number, commonly known as the port number.
- For example, an address may be "protocol: TCP, IP address: 192.168.31.4, port number: 80", which may be written 192.168.31.4:80 when the protocol is known from context.
- Ports become necessary only when computers are executing more than one program at a time and are connected to modern packet-switched networks.




## Process to Process Delivery: Multiplexing and De-Multiplexing

- The transport layer provides multiplexing and de-multiplexing service i.e. it extends the host to host delivery service provided by network layer to a process to process delivery service for applications running on different host.
- Each process can have one or more sockets i.e. a door through which data passes from the process to the network.




- At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing. The protocol accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, the transport layer passes the packet to the network layer.
- The job of gathering data chunks at the source host from different socket, encapsulating/enveloping each data chunk with header information (needed for de-multiplexing) to create segments and passing the segments to the network layer is called multiplexing.
- At the receiver site, the relationship is one-to-many and requires de-multiplexing. After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.
- The job of delivering the data in a transport layer segment to the correct socket is called de-multiplexing.
- When host receives IP datagram. Each datagram has source IP address, destination IP address. Each segment has source, destination port number. Host uses IP addresses & port numbers to direct segment to appropriate socket

## Connectionless and Connection Oriented Service

A transport layer protocol can either be connectionless or connection-oriented.

i. Connectionless Service

- In connection less service, when one side of an application wants to send packets to another side of an application, the sending application simply sends the packet without handshaking.
- Since there in no handshaking procedure prior to the transmission of packets, data can be delivered faster.
- But there is no acknowledgement either, so a source never knows for sure which packets arrive in destination.

- This service also has no provision for flow control, congestion control.
- The connectionless service is provided by transport layer protocol called UDP i.e. user datagram protocol.

ii. Connection Oriented Service

- When an application uses the connection oriented service, the client and server residing in different end system sends control packet to each other before sending packets with real data.
- The procedure of sending control packet is also called as handshaking that alert the client and server to be read for transmission of packets.
- Once handshaking is finished, a connection is established between two end system hence called as connection oriented.
- The connection oriented service provides other service like reliable data transfer, flow control, congestion control.
- The connection oriented service is provided b transport layer protocol called TCP i.e. transmission control protocol.

## Transmission control protocol(TCP)

### i. Introduction

- TCP corresponds to the transport layer of OSI.
- The main goal of TCP protocol is to see that data is reliably received and sent, that the packet data reaches the proper program in the application layer, and that the data reaches the program in the right order.
- TCP is connection oriented service because before an application data to another, the two processes must first handshake with each other.
- TCP connection is always point to point connection between sender and receiver.
- The connection establishment procedure in TCP is also called as 3 way handshake because here first client sends a special TCP segment, then server responds with second special TCP segment and finally client responds again with third segment which consists of application layer data.

### ii. Features of TCP

a. **Reliable Data transfer:** A reliable protocols ensures that data sent from one machine to another will eventually be communicated correctly. Reliability is achieved through the use of acknowledgement and retransmission policy. When a source send packet to some destination then destination acknowledge the source. If source receive acknowledgement then it knows that the packet has definitely received but if no acknowledge is received then sender assume that the packet it sent was not received by destination, so it retransmits the packet.

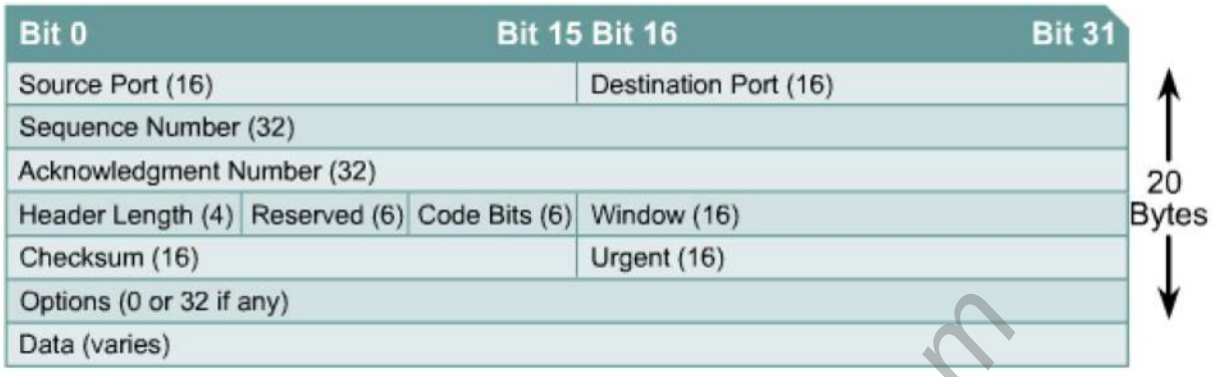
b. **Flow control:** Flow control makes sure that neither side of a connection overwhelms the other side by sending too many packets too fast. The flow control service forces the sending end system to reduce its pumping rate whenever there is such risk.

c. **Congestion control:** Congestion control service helps to prevent the network from entering a state of grid lock. If every pair of communication end system continue to pump as fast as they can, grid lock set in and few packets are delivered to destination because packet loss occur as router buffer overflows. So congestion control forces the end system to reduce the rate at which they send packet into the network during the period of congestion.

d. **Full duplex:** It provides bidirectional mode of communication i.e. both side can send and receive cconcurrently.

### iii. TCP header segment

The TCP header segment is shown in the figure below. It consists of header field and data field




- Source port – Number of the port that sends data. Basically used for multiplexing and de-multiplexing.
- Destination port – Number of the port that receives data. Basically used for multiplexing and de-multiplexing.
- Sequence number – the 32 bit sequence number is used to ensure the data arrives in the correct order
- Acknowledgment number – Next expected TCP segment.
- Header length – The four bit header length specify the length of TCP header as TCP header is of variable length due to TCP option field.
- Reserved – Set to zero
- Code bits or flag bit – Control functions, such as setup and termination of a session. It consists of 6 bits each for URG(urgent), ACK(acknowledgement), PSH (push), RST(reset), SYN(synchronization) and FIN
- Window – The 16-bit Window field is used for flow control.
- Checksum – Calculated checksum of the header and data fields for error detection.
- Options: The variable length options field is used when sender and receiver negotiate to maximum segment size for use in high speed network.
- Urgent pointer – indicates a location in the data field where urgent data resides.

## User Datagram Protocol(UDP)

### i. Introduction

- UDP is an unreliable, connectionless datagram protocol.
- It is called connectionless because there is no handshaking between the sending and receiving entity.
- It is useful for application that do not require TCP's sequencing or flow control.
- It is used for one shot, request reply applications where prompt delivery is important.
- Example of these application includes DNS and transmission of speech, video.

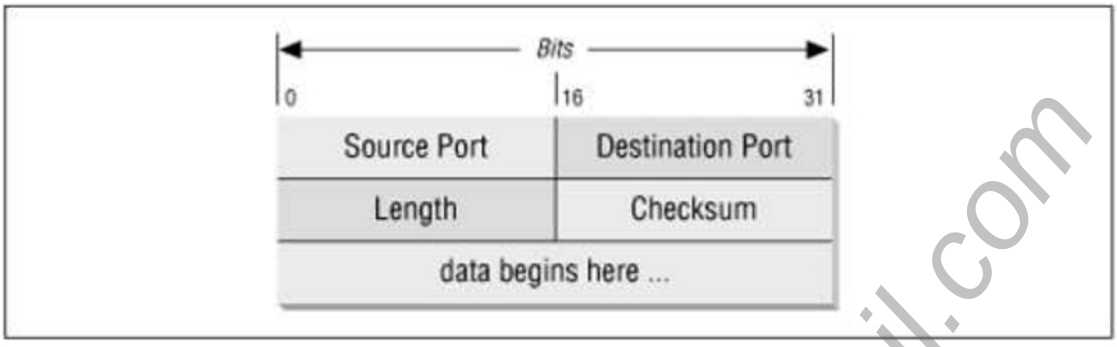
### ii. Features

- **Connectionless paradigm:** A process using UDP doesn't need to establish a connection between the sending and receiving entity. So it is faster than TCP.
- **Robustness:** UDP gives best effort delivery which means that segments can be lost, duplicated, or corrupted during transmission. This is why UDP is suitable for real time data transfer like audio/video transmission and unsuitable for transmission of text/character.
- **Unreliability:** UDP is fast, but unreliable in nature. This means when data bits are transferred via UDP, their reception acknowledgment cannot be attained in an automated fashion, unlike TCP. This feature of UDP prevents it from being used for text or character transmission/reception on computer networks
- **Disarrangement:** Data packets, when sent over a protocol like TCP, arrive in an arranged and assembled manner at the receiver's end. This property is also missing in UDP, since it takes no guarantee of transferring data bits or packets in an arranged manner

- **Reduced Overhead:** If the amount of data being transmitted is small, the overhead of creating connections and ensuring reliable delivery may be greater than the work of re-transmitting the entire data set. So UDP is better option as transport layer protocol

iii. **UDP header format**

- The format of a UDP header is shown in figure below




i. **Source port** – Number of the port that sends data
ii. **Destination port** – Number of the port that receives data
iii. **Length** – Number of bytes in header and data
iv. **Checksum** – Calculated checksum of the header and data fields
v. **Data** – Upper-layer protocol data

## TCP vs UDP

| S.no | TCP - Transmission Control Protocol | UDP - User Datagram Protocol |
|---|---|---|
| 1 | connection-oriented, reliable (virtual circuit) | connectionless, unreliable, does not check message delivery |
| 2 | Divides outgoing messages into segments | sends "datagrams" |
| 3 | reassembles messages at the destination | does not reassemble incoming messages |
| 4 | re-sends anything not received | Does-not acknowledge. |
| 5 | provides flow control | provides no flow control |
| 6 | more overhead than UDP (less efficient) | low overhead - faster than TCP |
| 7 | Examples:HTTP, NFS, SMTP | Eg. VOIP,DNS,TFTP |

## Introduction to Congestion Control

- Congestion in a network may occur if the load on the network (the number of packets sent to the network) is greater than the capacity of the network (the number of packets a network can handle).
- Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.
- When too many packets are pumped into the system, congestion occur leading into degradation of performance (i.e. resources are idle), packet loss (Router buffer may overflow), and queuing delay.
- We may ask why there is congestion on a network. Congestion happens in any system that involves waiting.
- Congestion in a network or internetwork occurs because routers and switches have queues-buffers that hold the packets before and after processing. A router, for example, has an input queue and an output queue for each interface. When a packet arrives at the incoming interface, it undergoes three steps before departing.
    i. The packet is put at the end of the input queue while waiting to be checked.

ii. The processing module of the router removes the packet from the input queue once it reaches the front of the queue and uses its routing table and the destination address to find the route.

iii. The packet is put in the appropriate output queue and waits its tum to be sent.

- we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal).

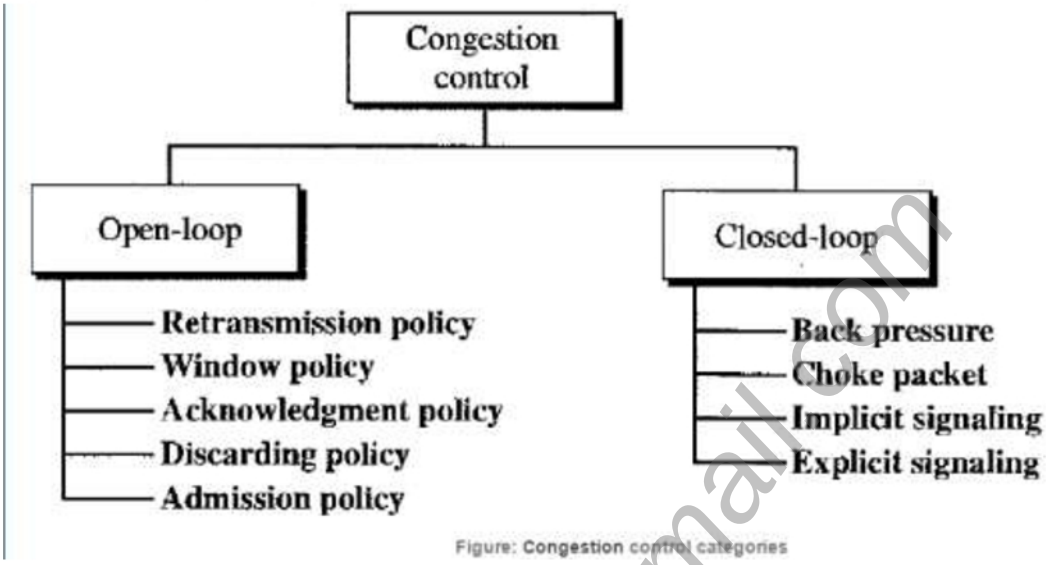



Figure: Congestion control categories

**Congestion prevention Policies / Open loop congestion control mechanism**

- In open-loop congestion control, policies are applied to prevent congestion before it happens.
- In these mechanisms, congestion control is handled by either the source or the destination

1. Retransmission Policy:
   - Retransmission is sometimes unavoidable. 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. For example, the retransmission policy used by TCP is designed to prevent or alleviate congestion.

2. Window Policy:
   The type of window at the sender may also affect congestion. The Selective Repeat window is better than the Go-Back-N window for congestion control. In the Go-Back-N window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse. The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted.

3. Acknowledgment Policy:
   The acknowledgment policy imposed by the receiver may also affect congestion. If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion. Several approaches are used in this case. A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires. We need to know that the acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing less load on the network.

4. Discarding Policy:
   A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission. 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.

5. Admission Policy:
   An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. 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 Control Mechanism

Closed-loop congestion control mechanisms try to reduce congestion after it happens. Several mechanisms have been used by different protocols.

1. 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. The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is coming.

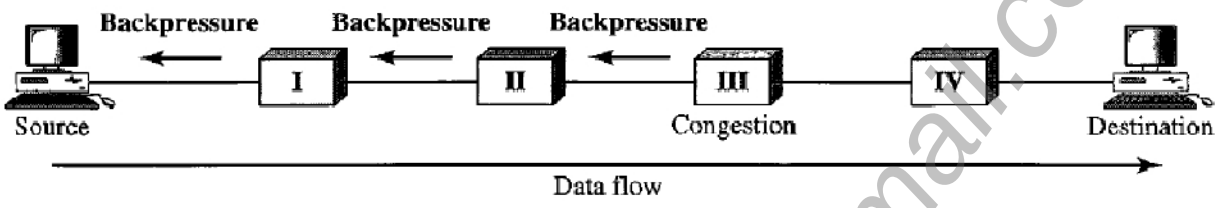



Figure: Backpressure method for alleviating congestion

Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down. Node II, in turn, may be congested because it is slowing down the output flow of data. If node II is congested, it informs node I to slow down, which in turn may create congestion. If so, node I inform the source of data to slow down. This, in time, alleviates the congestion. Note that the pressure on node III is moved backward to the source to remove the congestion.

2. Choke Packet

A choke packet is a packet sent by a node to the source to inform it of congestion. Note the difference between the backpressure and choke packet methods. 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 travelled are not warned. We have seen an example of this type of control in ICMP. When a router in the Internet is overwhelmed datagrams, it may discard some of them; but it informs the source. host, using a source quench ICMP message. The warning message goes directly to the source station; the intermediate routers, and does not take any action. Figure shows the idea of a choke packet.

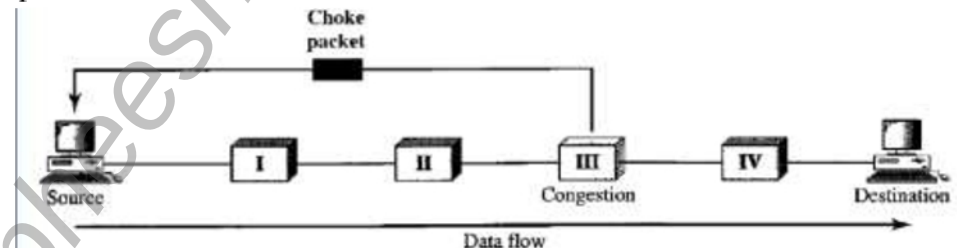
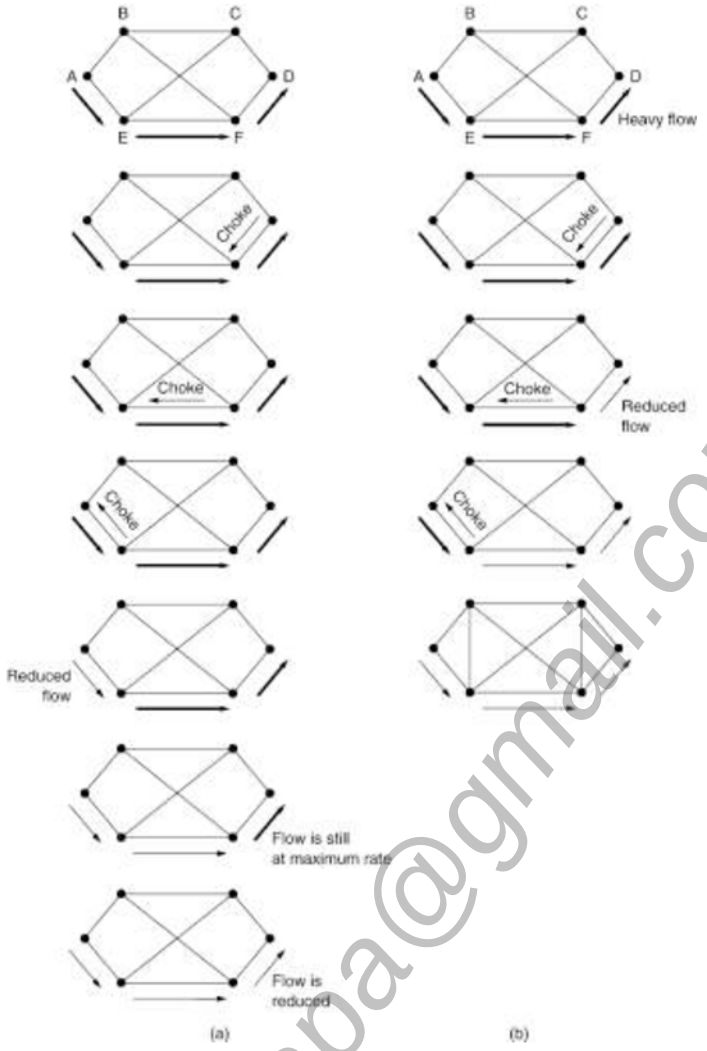



Figure: Choke packet

**Hop by Hop choke packet**

Over long distances or at high speeds choke packets are not very effective. A more efficient method is to send to choke packets hop-by-hop. This is similar to **back pressure** in which each hop to reduces its transmission even before the choke packet arrive at the source




a. Choke packet that affect only source b. choke packet that affect each hop it passes through

3. Implicit Signalling

In implicit signalling, there is no communication between the congested node or nodes 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 delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down. This approach is implemented in TCP congestion control mechanism.

4. Explicit Signalling

The node that experiences congestion can explicitly send a signal to the source or destination. The explicit signalling 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 signalling method, the signal is included in the packets that carry data(Piggybacking). Explicit signalling, can occur in either the forward or the backward direction.

1. **Backward Signalling:** A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.
2. **Forward Signalling:** A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

5. Load Shedding
   In this technique, when buffers of the router become full and can't be handled, then routers simply discard packets. The key question for a router drowning in packets is which packets to drop. Which packet is chosen to be the victim i.e. packet to be thrown out, depends on the application and on the error strategy used in the data link layer. For a file transfer, for, e.g. cannot discard older packets since this will cause a gap in the received data. For real-time voice or video, it is probably better to throw away old data and keep new packets. Also we can get the application to mark packets with discard priority.

## Traffic Shaping

- Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network.
- It is about regulating average rate of data flow.
- It is a method of congestion control by providing shape to data flow before entering the packet into the network. Two techniques can shape traffic: leaky bucket and token bucket

### 1. Leaky bucket algorithm

- The Leaky Bucket Algorithm used to control rate in a network.
- It is implemented as a single-server queue with constant service time.
- If the bucket (buffer) overflows, then packets are discarded.
- In this algorithm the input rate can vary but the output rate remains constant.
- This algorithm saves burst traffic into fixed rate traffic by averaging the data rate


Fig: A leaky bucket with water

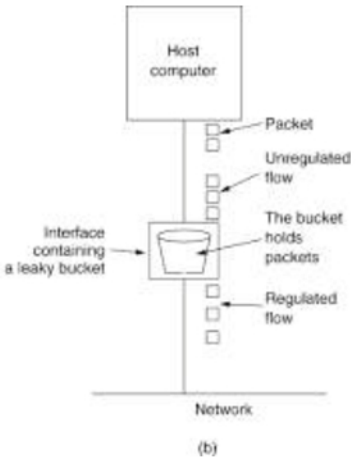
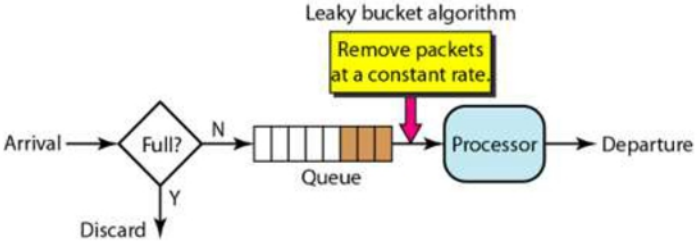


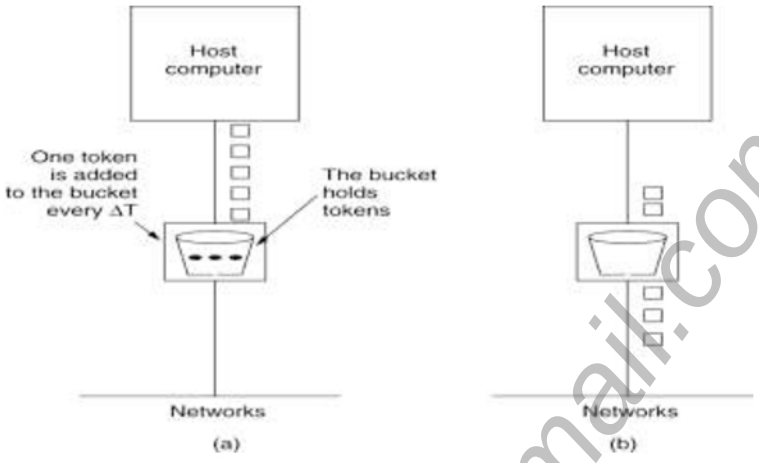

Fig: A leaky bucket with packet

The below figure shows the implementation of leaky bucket algorithm.

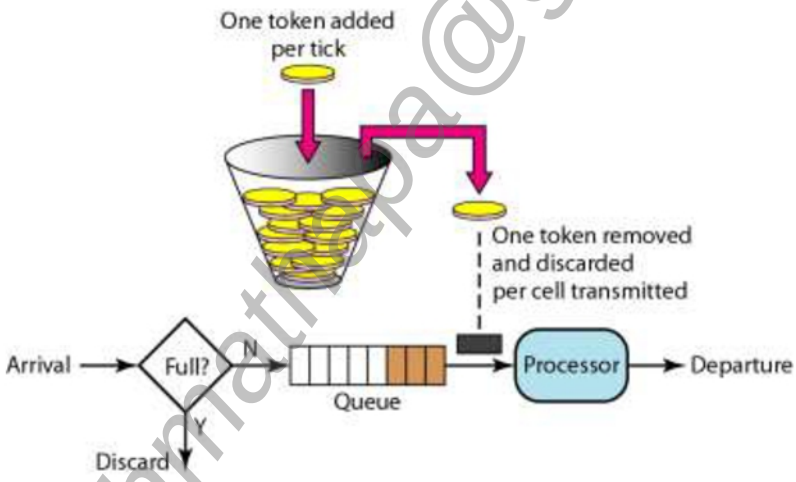


## 2. Token bucket algorithm

- The token bucket allows burst traffic at a regulated maximum rate.
- The Token Bucket Algorithm compare to Leaky Bucket Algorithm allow the output rate vary depending on the size of burst.
- In this algorithm the buckets hold token and to transmit a packet, the host must capture and destroy one token.
- Tokens are generated by a clock at the rate of one token every Δt sec.
- Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.




The below figure shows the implementation of leaky bucket algorithm.



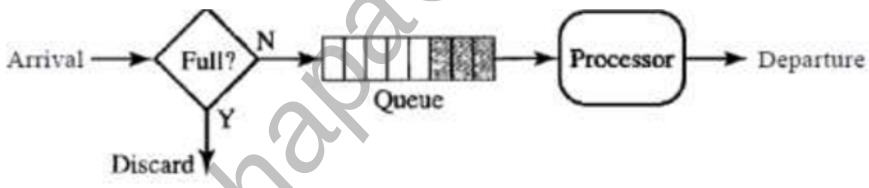
## Token bucket vs leaky bucket

| TOKEN BUCKET | LEAKY BUCKET |
|---|---|
| Token dependent. | Token independent. |
| If bucket is full token are discarded, but not the packet. | If bucket is full packet or data is discarded. |
| Packets can only transmitted when there are enough token | Packets are transmitted continuously. |
| It allows large bursts to be sent faster rate after that constant rate | It sends the packet at constant rate |
| It saves token to send large bursts. | It does not save token. |

## Queuing techniques for scheduling

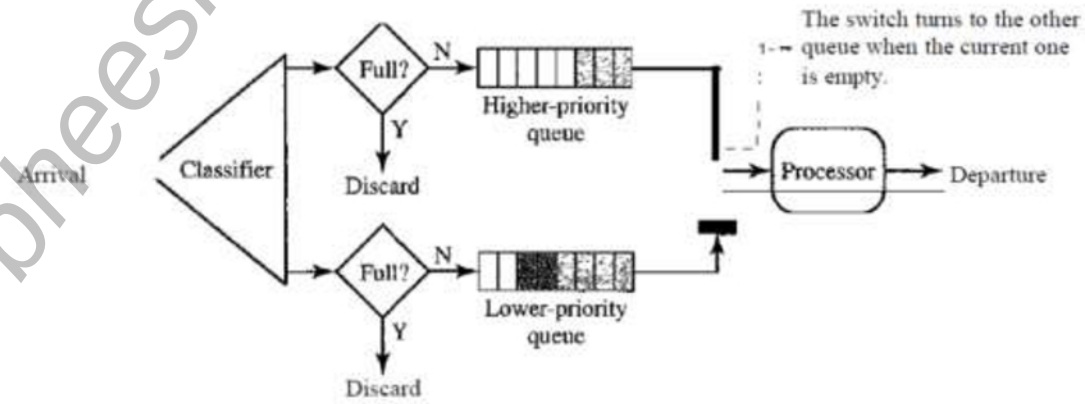
- Packets from different flows arrive at a switch or router for processing.
- A good scheduling technique treats the different flows in a fair and appropriate manner.
- Several scheduling techniques are designed to improve the quality of service. Some of them are

1. FIFO queue




   - In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them.
   - If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded

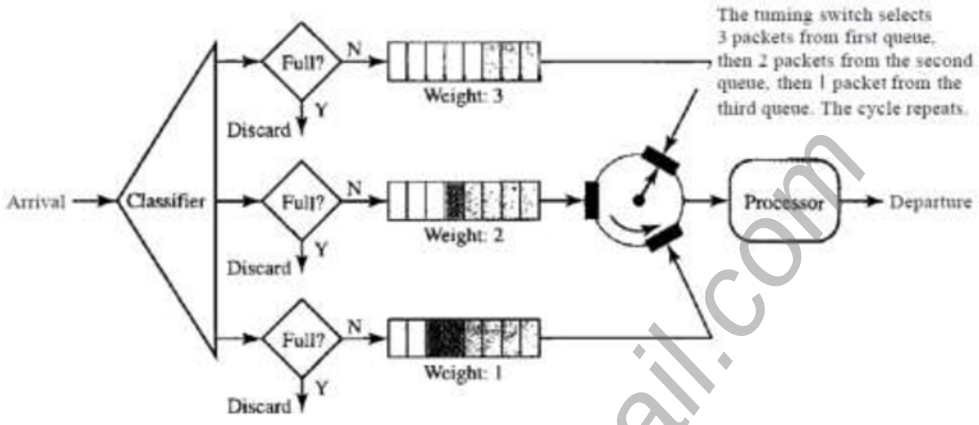
2. Priority queue




   - In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue.
   - The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last.
   - A priority queue can provide better QoS than the FIFO queue because higher priority traffic, such as multimedia, can reach the destination with less delay.

- However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called starvation.

3. Weighted queue




- A better scheduling method is weighted fair queuing. In this technique, the packets are still assigned to different classes and admitted to different queues.
- The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight.
- The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight.
- For example, if the weights are 3, 2, and 1, three packets are processed from the first queue, two from the second queue, and one from the third queue.
- If the system does not impose priority on the classes, all weights can be equal. In this way, we have fair queuing with priority.

## Overview of TCP congestion Control

- In this approach, the TCP limits the rate at which the sender can send traffic into its connection at a function of perceived network congestion.
- Network congestion is perceived based on observed network packet loss, delay etc.
- The TCP connection controls its transmission rate by limiting the number of transmitted but not yet acknowledged segments.
- Let us denote this number of permissible unacknowledged segments as w, often referred to as the TCP window size.
- W is dynamic, function of perceived network congestion.
- So the sender limits the transmission by keeping the amount of unacknowledged segments less than the value of w.

  Last byte sent – last byte acknowledged <=W
- The size of window is determined as below.
- TCP's general policy for handling congestion is based on three phases: slow start, congestion avoidance, and congestion detection.
  1. Slow-start, exponential increase phase
     - In this phase, the sender starts with a very slow rate of transmission, but increases the rate rapidly to reach a threshold.
     - This algorithm is based on the idea that the size of the congestion window (w) starts with one maximum segment size (MSS).
     - This means that the sender can send only one segment. After receipt of the acknowledgment for segment 1, the size of the congestion window is increased by 1, which means that w is now 2. Now two more

segments can be sent. When each acknowledgment is received, the size of the window is increased by 1 MSS i.e. w is now 4. So the size of window is increasing exponentially.

- So whenever the size of window reaches some threshold, slow start stops and the next phase starts. In most implementations the value of threshold is 65,535 bytes

2. Congestion Avoidance, Additive Increase Phase
   - To avoid congestion before it happens, one must slow down the exponential growth. TCP defines another algorithm called congestion avoidance, which undergoes an additive increase instead of an exponential one.
   - When the size of the congestion window reaches the slow-start threshold, the slow-start phase stops and the additive phase begins.
   - In this algorithm, each time the **whole window of segments** is acknowledged (one round), the size of the congestion window is increased by 1.
   - In the congestion avoidance algorithm, the size of the congestion window increases additively until congestion is detected.

3. Congestion Detection, Multiplicative Decrease phase
   - If congestion occurs, the congestion window size must be decreased. The only way the sender can guess that congestion has occurred is based on observed network packet loss, delay etc.
   - So in this cases, the size of the threshold is dropped to one-half, a multiplicative decrease , the window size is set to one MSS and the slow phase is again started.

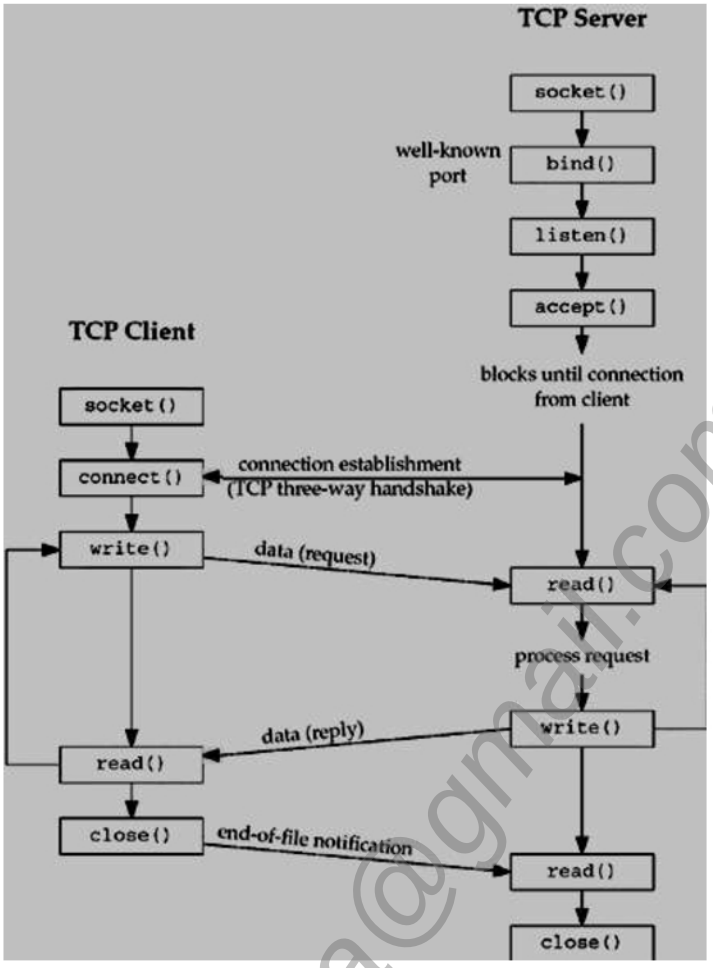
## Concept of Socket Programming using TCP socket

- Networks in which certain computers have special dedicated tasks, providing services to other computers (in the network) are called client-server networks.
- Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.
- In client-server architecture two types of process are involved in communication as below.

  i. Client Process:
  This is the process which typically makes a request for information. After getting the response this process may terminate or may do some other processing. For example: Internet Browser works as a client application which sends a request to Web Server to get one HTML web page.

  i. Server Process:
  This is the process which takes a request from the clients. After getting a request from the client, this process will do required processing and will gather requested information and will send it to the requestor client. Once done, it becomes ready to serve another client. Server process are always alert and ready to serve incoming requests. For example: Web Server keeps waiting for requests from Internet Browsers and as soon as it gets any request from a browser, it picks up a requested HTML page and sends it back to that Browser.

- In client-server communications the client needs to know of the existence and the address of the server, but the server does not need to know the address or even the existence of the client prior to the connection being established. Once a connection is established, both sides can send and receive information as shown in the figure below.




Example:

Write a socket program in which the client program receives radius from the user and sends to the server. The server then computes area of the circle and sends the result back to client. The client then displays the result.

**Client side program:**

```
import java.net.*;
import java.util.*;
import java.io.*;
public class Client
{
    public static void main(String [] args) throws Exception
    {
            Socket cs=new Socket("localhost",2000);

            Scanner sc=new Scanner(System.in);
            System.out.println("Enter the radius of circle");
            String rad=sc.nextLine();

            PrintStream out=new PrintStream(cs.getOutputStream());
            out.println(rad);

            DataInputStream in=new DataInputStream(cs.getInputStream());
            System.out.println("The area of circle is "+in.readLine());

            out.close();
            in.close();
    }
}
```

**Server side program:**

```
import java.net.*;
import java.io.*;
public class Server
{
        public static void main(String [] args) throws Exception
        {
                ServerSocket ss=new ServerSocket(2000);
                while(true)
                {
                        Socket cs=ss.accept();

                        DataInputStream dis=new DataInputStream(cs.getInputStream());
                        String rad=dis.readLine();

                        PrintStream ps=new PrintStream(cs.getOutputStream());
                        double area=3.14*Double.parseDouble(rad)*Double.parseDouble(rad);
                        ps.println(area);

                        ps.close();
                        dis.close();
                }
        }
}
```

**Assignment**

1. **What is piggybacking? Explain its usefulness.**