Unit 5: Transport Layer

The transport layer is responsible for process-to-process delivery of the entire message. A process is a network application running on a host. Whereas the network layer oversees source-to-destination delivery of packets, it does not recognize any relationship between those packets. The transport layer ensures that the whole message arrives at the host application.

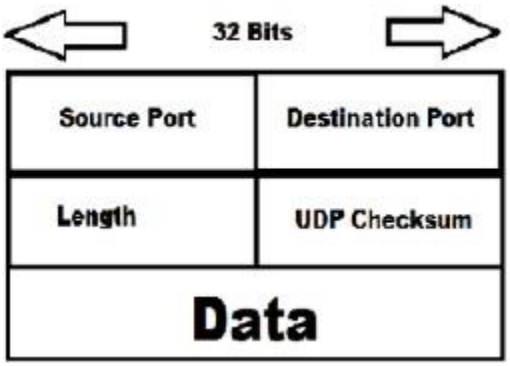
Transport Layer Services & Responsibilities:

- Process to Process Delivery
- Multiplexing and Demultiplexing – simultaneous use of different applications
- Segmentation and reassembly
- Congestion Control
- Connection Control – TCP or UDP
- Flow Control
- Error Control

User Datagram Protocol (UDP):

The user datagram protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.

The UDP packet structure is as follows:



UDP Operations:

UDP uses concepts common to the transport layer.

Connectionless Services: UDP provides connectionless service which means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination process. The user

datagrams are not numbered and there is no connection establishment and no connection termination, which means each user datagram can travel on a different path. Stream of data cannot be sent; it should get fragmented.

Flow and Error Control: UDP is a very simple, unreliable transport protocol which does not provide flow control. The receiver may overflow with incoming messages. There is error control mechanism in UDP except checksum, which means the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded.

Encapsulation and Decapsulation: To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.

Queuing: Queuing in UDP simply refers to requesting port number for client processes and using that port number for process-to-process delivery of messages.

UDP Characteristics and Applications:

- UDP allows very simple data transmission without any error detection. So, it can be used in networking applications like VOIP, streaming media, etc. where loss of few packets can be tolerated and still function appropriately.
- UDP is suitable for multicasting since multicasting capability is embedded in UDP software but not in TCP software.
- UDP is used for management processes such as SNMP.
- UDP is used for some route updating protocols such as RIP.
- UDP is used by Dynamic Host Configuration Protocol (DHCP) to assign IP addresses to systems dynamically.
- UDP is an ideal protocol for network applications where latency is critical such as gaming and voice and video communications.

Transmission Control Protocol (TCP):

TCP is a transport layer protocol for process-to-process communication like UDP. It is a connection oriented, reliable transport protocol. TCP creates a virtual connection between two TCP clients to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

TCP Operations:

Various services and operations of TCP are as follows:

Process to Process Communication: Like UDP, TCP provides process-to-process communication using port numbers.

Stream Delivery Service: TCP is a stream-oriented protocol. It allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. TCP creates an environment in which the two processes seem to have a dedicated connection that carries their data across the internet. For flow control, TCP uses sending and receiving buffer that provides some storage for data packets in case of overflow. This prevents the loss of packets by synchronizing the flow rate.

Full-Duplex Communication: TCP offers full-duplex services in which data can flow in both directions at the same time. Each TCP then has sending and receiving buffer, and segments move in both directions.

Connection-Oriented Service: When a process at site A wants to send and receive data from another process at site B, the following occurs:

- The two TCPs establish a connection between them.
- Data are exchanged in both directions.
- The connection is terminated.

Note that this a virtual connection, not a physical connection.

Reliable Service: TCP is a reliable transport protocol. It uses an acknowledgement mechanism to check the safe arrival of data. This is possible due to efficient error control mechanisms.

TCP Features and Characteristics:

To support the services of TCP, following are some features:

Numbering System: TCP keeps track of segments being transmitted or received. There are two fields called the sequence number and the acknowledgement number for numbering the bytes within the segments and acknowledgements respectively.

Flow control: TCP provides flow control mechanism. The receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being overloaded with data. The numbering system allows TCP to use a byte-oriented flow control.

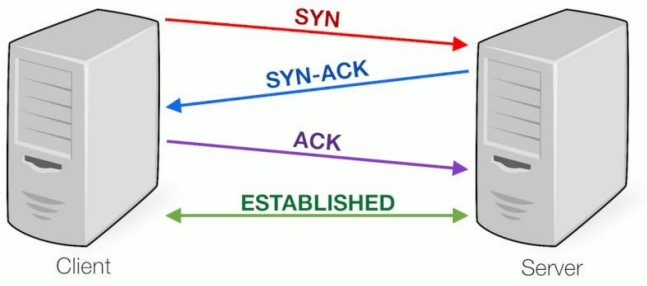
Error Control: To provide reliable service, TCP implements an error control mechanism. Although error control mechanism considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.

Congestion Control: TCP takes into account about the congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network.

Three-Way Handshaking:

Since TCP is a connection-oriented service, it requires three phases: connection establishment, data transfer, and connection termination.

The connection establishment in TCP is called three-way handshaking. The figure below shows the handshaking process.



Similarly, three-way handshaking can also be used for connection termination.

Key Differences Between TCP and UDP:

- TCP is a connection-oriented protocol, whereas UDP is a connectionless protocol.
- The speed for TCP is slower while the speed of UDP is faster.
- TCP is reliable as it guarantees the data delivery while UDP is not reliable.
- TCP uses handshake protocol like SYN, SYN-ACK, ACK while UDP uses no handshake protocols.
- TCP does error checking and also makes error recovery, on the other hand, UDP performs error checking, but it discards erroneous packets.
- TCP has acknowledgment segments, but UDP does not have any acknowledgment segment.
- TCP is heavy-weight, and UDP is lightweight.
- Data packets arrive in order at the receiver in TCP while no sequencing in UDP.
- TCP doesn't support broadcasting while UDP does.
- TCP is used by HTTP, HTTPS, FTP, SMTP and TELNET while UDP is used by DNS, DHCP, SNMP, RIP and VoIP.

Connection Oriented Services:

A connection-oriented service needs an established connection between peers before data can be sent between the connected terminals. This method is often called a "reliable" network service. This handles real-time traffic more efficiently than connectionless protocols because data arrives in the same order as it was sent. Connection-oriented protocols are also less error-prone. There is a sequence of operation to be followed by the users of connection-oriented service.
These are:
1. Connection is established.
2. Information is sent.
3. Connection is released.

In connection-oriented service, we have to establish a connection before starting the communication. When connection is established, we send the message or the information and then we release the connection. Example of connection oriented is TCP (Transmission Control Protocol) protocol.

Virtual Circuits:

- A virtual circuit (VC) is a means of transporting data over a packet switched computer network in such a way that it appears as though there is a dedicated physical layer link between the source and destination end systems of this data.
- In all major computer network architectures to date (Internet, ATM, frame relay, and so on), the network layer provides either a host-to-host connectionless service or a host-to-host connection service, but not both.
- Computer networks that provide only a connection-oriented service at the network layer are called **virtual-circuit** (VC) networks; computer networks that provide only a connectionless service at the network layer are called datagram networks.

- While the Internet is a datagram network, many alternative network architectures— including those of ATM and frame relay—are virtual-circuit networks and, therefore, use connections at the network layer. These network-layer connections are called **virtual circuits (VCs)**.

There are three identifiable phases in a virtual circuit: VC Setup, Data Transfer, VC Teardown

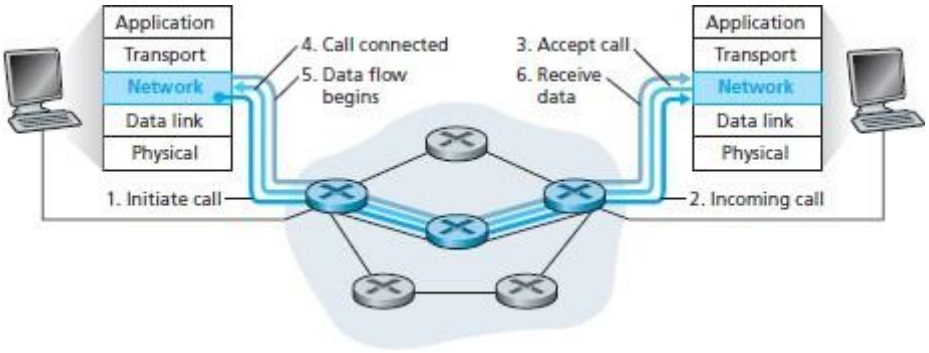


Fig: Virtual Circuit Setup

Connection less Services:

Connectionless service means that a terminal or node can send data packets to its destination without establishing a connection to the destination. A session connection between the sender and the receiver is not required, the sender just starts sending the data. The message or datagram is sent without prior arrangement, which is less reliable but faster transaction than a connection-oriented service. This works because of error handling protocols, which allow for error correction like requesting retransmission. It is similar to the postal services, as it carries the full address where the message (letter) is to be carried. Each message is routed independently from source to destination. The order of message sent can be different from the order received.
LANs are actually connectionless systems with each computer able to transmit data packets as soon as it can access the network. The Internet is a large connectionless packet network in which all packet delivery is handled by Internet providers. Example of Connectionless service is UDP (User Datagram Protocol) protocol.

The connectionless services at the network layer are called datagram networks. In a datagram network, each time an end system wants to send a packet, it stamps the packet with the address of the destination end system and then pops the packet into the network. As shown in Figure, there is no VC setup and routers do not maintain any VC state information (because there are no VCs).

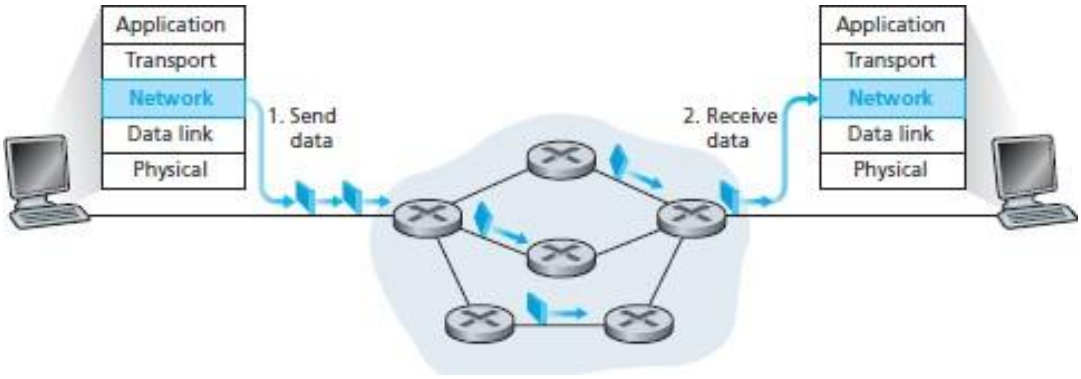


Fig: Datagram Network

Congestion Control:

Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called congestion. In other words, 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).

The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets. However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

Effects of Congestion:

- As delay increases, performance decreases.
- If delay increases, retransmission occurs, making situation even worse

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

The General Principles of Congestion Control are as follows:

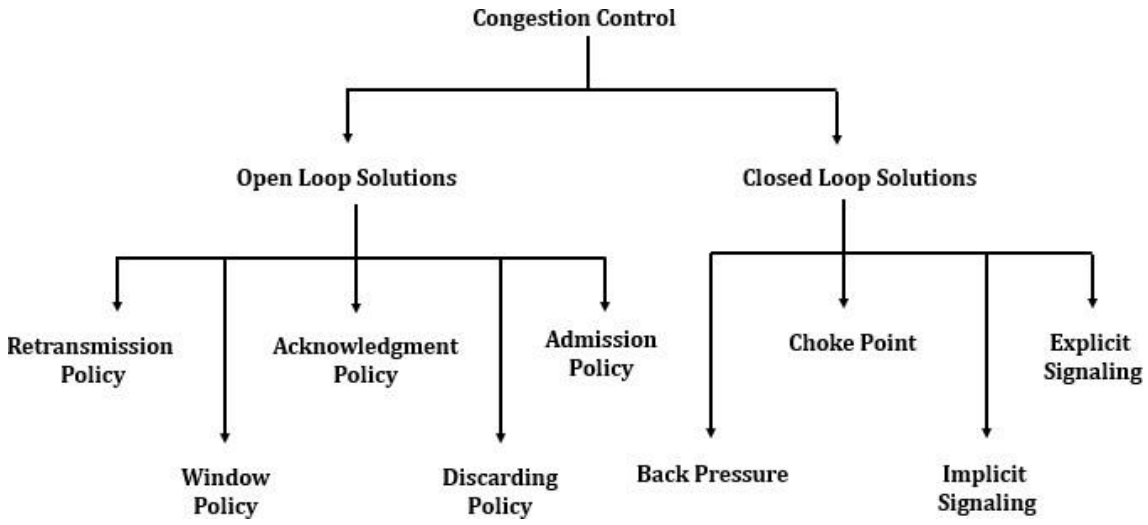
Open Loop Principle:

- attempt to prevent congestion from happening
- after system is running, no corrections made

Closed Loop Principle:

- monitor system to detect congestion

- pass information to where action is taken
- adjust system operation to correct problem



Open Loop Congestion Control:

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.

The list of policies that can prevent congestion are:

Retransmission Policy:

It is the policy in which retransmission of the packets are taken care. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. This transmission may increase the congestion in the network.

To prevent congestion, retransmission timers must be designed to prevent congestion and also able to optimize efficiency

Window Policy

The type of window at the sender side may also affect the congestion. Several packets in the Go-back-n window are resent, although some packets may be received successfully at the receiver side. This duplication may increase the congestion in the network and making it worse.

Therefore, Selective repeat window should be adopted as it sends the specific packet that may have been lost.

Acknowledgement Policy

Since acknowledgement are also the part of the load in network, the acknowledgment policy imposed by the receiver may also affect congestion. Several approaches can be used to prevent congestion related to acknowledgment.

The receiver should send acknowledgement for N packets rather than sending acknowledgement for a single packet. The receiver should send an acknowledgment only if it has to send a packet or a timer expires.

Discarding Policy

A good discarding policy adopted by the routers is that the routers may prevent congestion and at the same time partially discards the corrupted or less sensitive package and also able to maintain the quality of a message.

In case of audio file transmission, routers can discard fewer sensitive packets to prevent congestion and also maintain the quality of the audio file.

Admission Policy

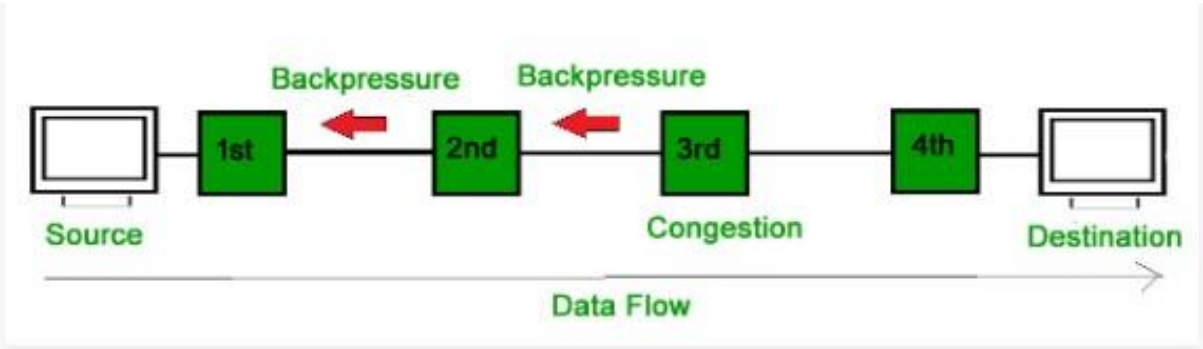
In admission policy a mechanism should be used to prevent congestion. Switches in a flow should first check the resource requirement of a network flow before transmitting it further. If there is a chance of a congestion or there is a congestion in the network, router should deny establishing a virtual network connection to prevent further congestion.

Closed Loop Congestion Control

Closed-Loop congestion control mechanisms try to reduce effects of congestion after it happens.

Back Pressure:

Backpressure is a technique in which a congested node stops receiving packet from upstream node. This may cause the upstream node or nodes to become congested and rejects receiving data from above nodes. Backpressure is a node-to-node congestion control technique that propagate in the opposite direction of data flow. The backpressure technique can be applied only to virtual circuit where each node has information of its above upstream node.

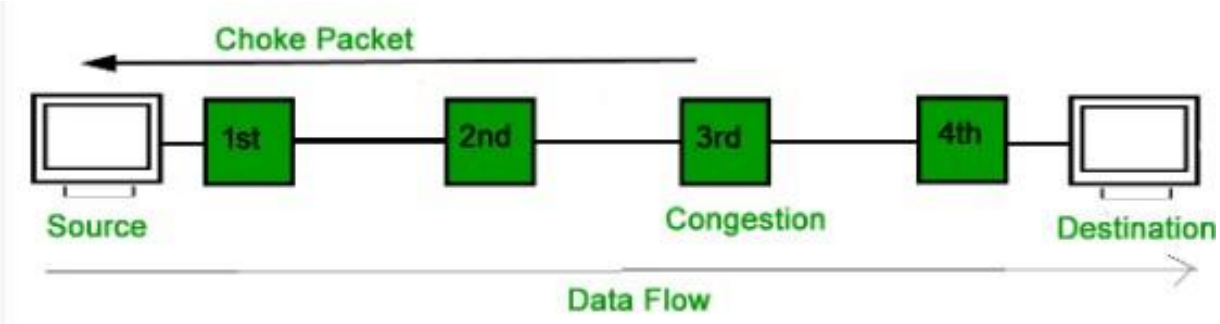


In above diagram the 3rd node is congested and stops receiving packets as a result 2nd node may be get congested due to slowing down of the output data flow. Similarly, 1st node may get congested and informs the source to slow down.

Choke Packet:

Choke packet technique is applicable to both virtual networks as well as datagram subnets. A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitors its resources and the utilization at each of its output lines. whenever the resource utilization exceeds the threshold value

which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets have traveled are not warned about congestion.



Implicit Signaling:

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

Explicit Signaling:

In explicit signaling, if a node experiences congestion it can explicitly sends a packet to the source or destination to inform about congestion. The difference between choke packet and explicit signaling is that the signal is included in the packets that carry data rather than creating different packet as in case of choke packet technique.

Explicit signaling can occur in either forward or backward direction.

Forward Signaling: In forward signaling signal is sent in the direction of the congestion. The destination is warned about congestion. The receiver in this case adopts policies to prevent further congestion.

Backward Signaling: In backward signaling signal is sent in the opposite direction of the congestion. The source is warned about congestion and it needs to slow down.

TCP Congestion Control:

Congestion Control in TCP (Virtual-Circuit Subnet) is closed loop-based design for connection-oriented services which can be done during connection set up. The basic principle is that when setting up a virtual circuit, we have to make sure that congestion be avoided.

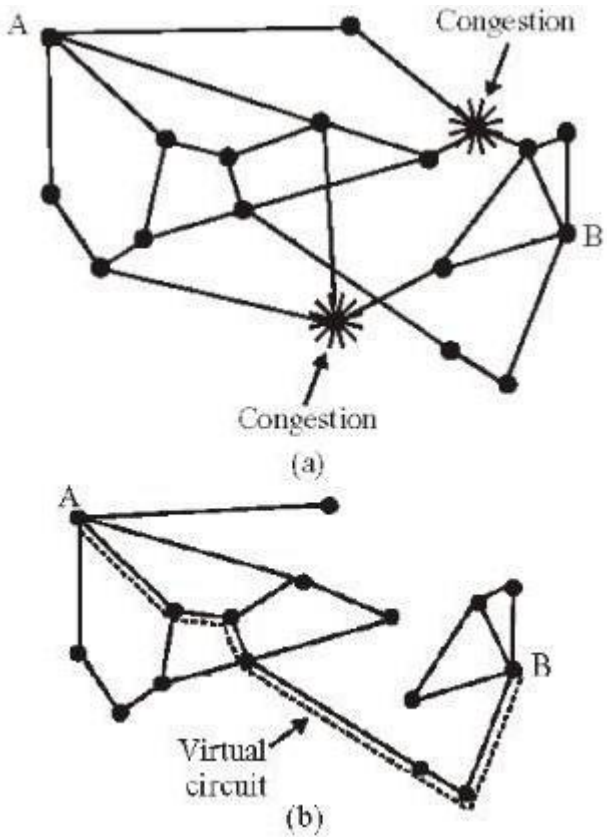
The following method is used for congestion control in TCP:

- Admission Control

- Once the congestion has been signaled, no newer virtual circuits can be set up until the problem has been solved.
- This type of approach is often used in normal telephone networks. When the exchange is overloaded, then no new calls are established.

- Another Approach: Alternative routes
  - To allow new virtual connections, route these carefully so that none of the congested router (or none of the problem area) is a part of this route i.e., to avoid the part of the network that is overloaded.
  - Yet another approach can be: To negotiate different parameters between the host and the network, when the connection is setup. During the setup time itself, Host specifies the volume and shape of traffic, quality of service, maximum delay and other parameters, related to the traffic it would be offering to the network. Once the host specifies its requirement, the resources needed are reserved along the path, before the actual packet follows.

    In the figure below, Normally when router A sets a connection to B, it would pass through one of the two congested routers, as this would result in a minimum-hop route. To avoid congestion, a temporary subnet is redrawn by eliminating congested routers. A virtual circuit can then be established to avoid congestion.



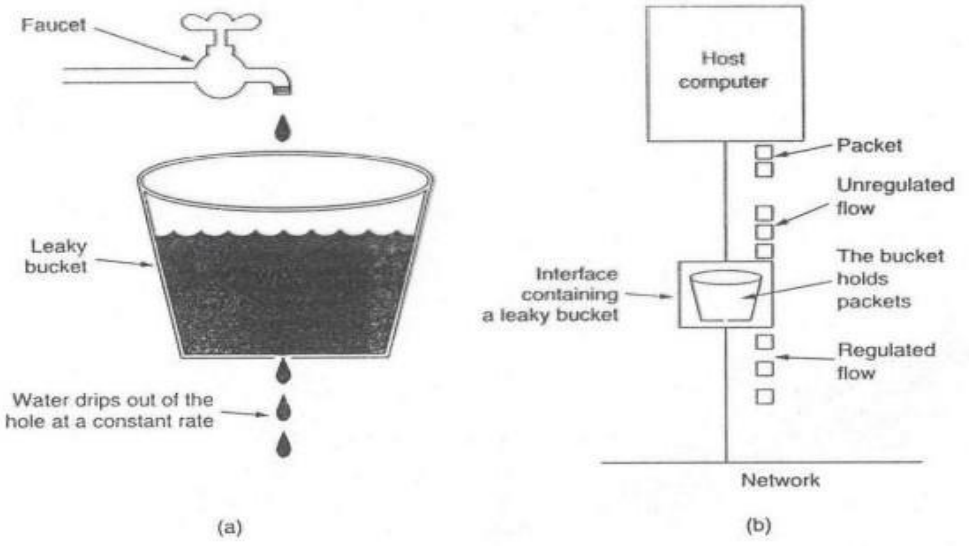
Traffic Shaping:

It is a mechanism to control the amount and the rate of the traffic sent to the network.

Two techniques can shape traffic: Leaky bucket and Token bucket.

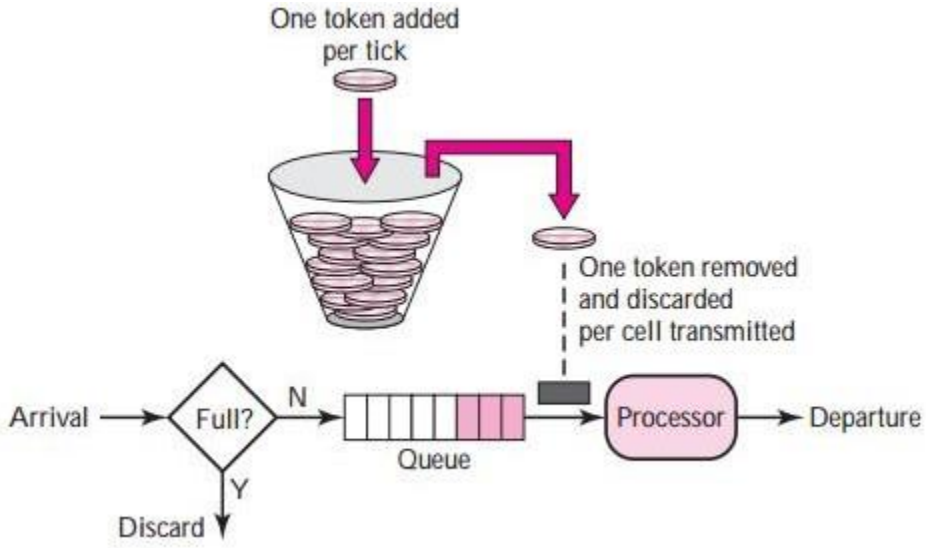
Leaky bucket algorithm:

- Each host is connected to the network by an interface containing a leaky bucket, a finite internal queue.
- If a packet arrives at the queue when it is full, the packet is discarded.
- In fact, it is nothing other than a single server queuing system with constant service time.



Token bucket algorithm:

- The leaky bucket holds tokens, generated by a clock at the rate of one token every T second.
- For a packet to be transmitted, it must capture and destroy one token.
- The token bucket algorithm allows idle hosts to save up permission to the maximum size of bucket n for burst traffic latter.



Parameter	Leaky Bucket	Token Bucket
Token Dependency	Token independent.	Dependent on Token.
Filled bucket for token	When bucket is full, data or packets are discarded.	If bucket is full, token are discard not packets.
Packet transmission	Leaky bucket sends packets at constant rate.	Token bucket can send large burst of packets at faster rate.
Condition for packet transmission	In Leaky bucket algorithm, Packets are transmitted continuously.	In Token bucket algorithm, Packets can only transmit when there is enough token.
Token saving	It does not save any token.	It saves token for the burst of packet transmission.
Restrictive Algorithm	Leaky bucket algorithm is more restrictive as compared to Token bucket algorithm.	Token bucket algorithm is less restrictive as compared to Leaky bucket algorithm.

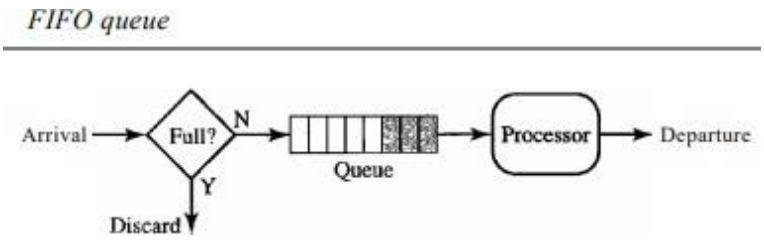
Queuing Techniques for Scheduling:

QoS traffic scheduling is a scheduling methodology of network traffic based upon QoS (Quality of Service).

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. Major scheduling techniques are: FIFO Queuing, Priority Queuing and Weight Fair Queuing.

FIFO Queuing:

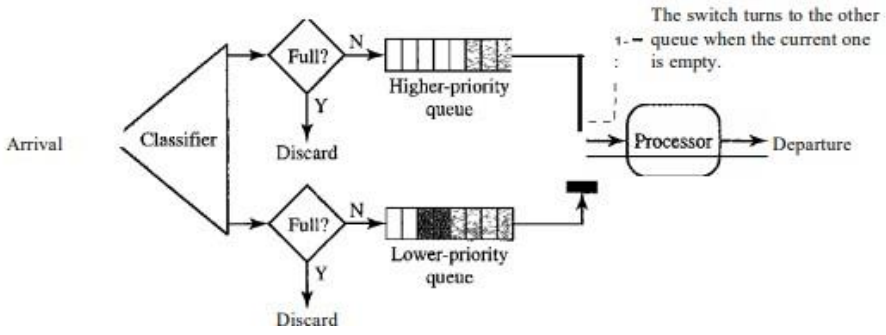
In first- in first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded. A FIFO queue is familiar to those who have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



Priority Queuing:

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. Note that the system does not stop serving a queue until it is empty.

Figure 24.17 *Priority queuing*

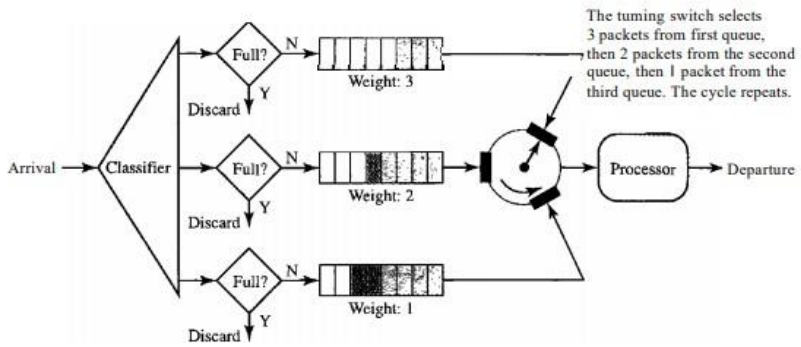


A priority queue can provide better QoS than the FIFO queue because higher priority traffic, such as multimedia, can reach the destination with less delay. However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called starvation.

Weighted Fair Queuing:

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.

Figure 24.18 *Weighted fair queuing*



Introduction to Ports and Sockets: (Port Addressing/Socket Addressing)

The IP address and the physical address are necessary for a quantity of data to travel from a source to the destination host. However, arrival at the destination host is not the final objective of data communications on the Internet. A system that sends nothing but data from one computer to another is not complete.

Today, computers are devices that can run multiple processes at the same time. The end objective of Internet communication is a process communicating with another process.

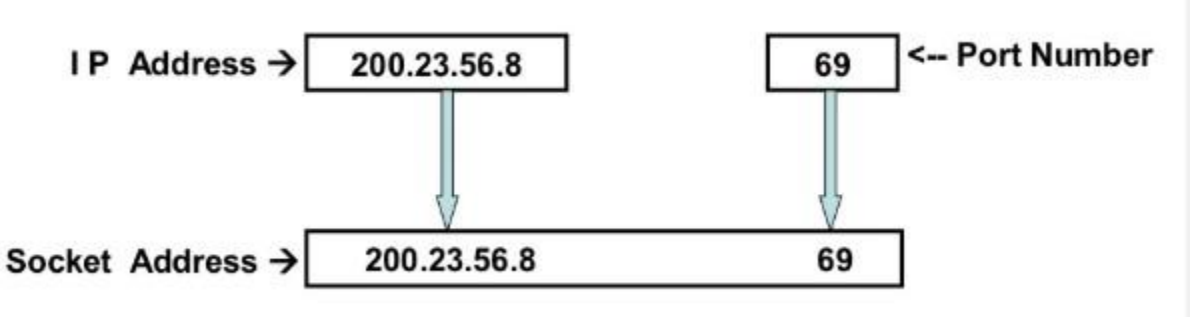
For example, computer A can communicate with computer C by using TELNET. At the same time, computer A communicates with computer B by using the File Transfer Protocol (FTP). For these processes to receive data simultaneously, we need a method to label the different processes.

In other words, they need addresses. In the TCP/IP architecture, the label assigned to a process is called a port address. A port address in TCP/IP is 16 bits in length.

Source and destination addresses are found in the IP packet, belonging to the network layer. A transport layer datagram or segment that uses port numbers is wrapped into an IP packet and transported by it.

The network layer uses the IP packet information to transport the packet across the network (routing). Arriving at the destination host, the host's IP stack uses the transport layer information (port number) to pass the information to the application.

Port Number	Assignment
21	File Transfer Protocol (FTP)
23	TELNET remote login
25	SMTP
80	HTTP
53	DNS



IP address and Port Number is combinedly called Socket Address that identifies the host along with the networking application running in the host.

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to. An endpoint is a combination of an IP address and a port number.

Socket Programming:

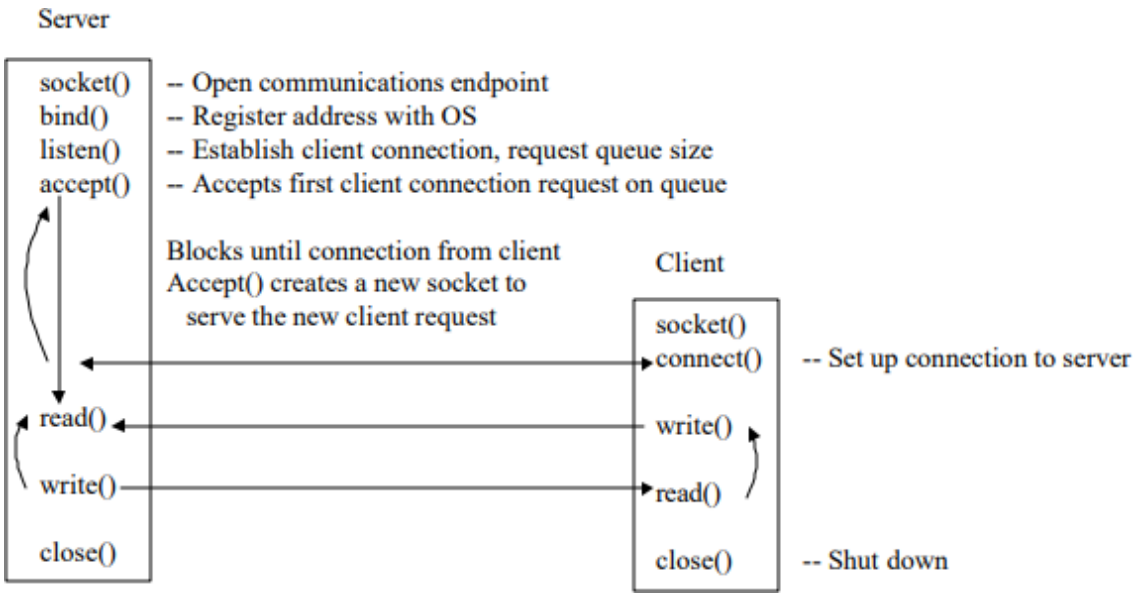
A typical network application consists of a pair of programs—a client program and a server program—residing in two different end systems. When these two programs are executed, a client process and a server process are created, and these processes communicate with each other by reading from, and writing to, sockets. When creating a network application, the developer's main task is therefore to write the code for both the client and server programs, called socket programming.

There are two types of network applications. One type is an implementation whose operation is specified in a protocol standard, such as an RFC or some other standards document; such an application is sometimes referred to as "open," since the rules specifying its operation are known to all. For such an implementation, the client and server programs must conform to the rules dictated by the RFC.

The other type of network application is a proprietary network application. In this case the client and server programs employ an application-layer protocol that has not been openly published in an RFC or elsewhere. A single developer (or development team) creates both the client and server programs, and the developer has complete control over what goes in the code. But because the code does not implement an open protocol, other independent developers will not be able to develop code that interoperates with the application.

During the development phase, one of the first decisions the developer must make is whether the application is to run over TCP or over UDP.

When a web page is opened, automatically a socket program is initialized to receive/send to the process. The socket program at the source communicates with the socket program at the destination machine with the associated source port/destination port numbers. When a web page is terminated, automatically the socket programs will be terminated.



The following sequence of events occur in the client-server application using socket programming for both TCP and UDP:

- The client reads a line of characters (data) from its keyboard and sends the data to the server.
- The server receives the data and converts the characters to uppercase.
- The server sends the modified data to the client.
- The client receives the modified data and displays the line on its screen.

BSD Socket API is the well-known socket programming API that defines a set of standard function calls made available at the application level.

BSD- Berkeley Software Distribution, API-Applications Programming Interface

These functions allow programmers to include Internet communications capabilities in their products. BSD Sockets generally relies upon client/server architecture. For TCP communications, one host listens for incoming connection requests. When a request arrives, the server host will accept it, at which point data can be transferred between the hosts. UDP is also allowed to establish a connection, though it is not required. Data can simply be sent to or received from a host. The Sockets API makes use of two mechanisms to deliver data to the application level: ports and sockets.