RGPVNOTES.IN

Subject Name: **Computer Network**

Subject Code: **IT-5003**

Semester: **5th**

Transport Layer

Transport layer services are conveyed to an application via a programming interface to the transport layer protocols. The services may include the following features:

- **Connection-oriented communication:** It is normally easier for an application to interpret a connection as a data stream rather than having to deal with the underlying connection-less models, such as the datagram model of the User Datagram Protocol (UDP) and of the Internet Protocol (IP).
- **Same order delivery:** The network layer doesn't generally guarantee that packets of data will arrive in the same order that they were sent, but often this is a desirable feature. This is usually done using segment numbering, with the receiver passing them to the application in order. This can cause head-of-line blocking.
- **Reliability:** Packets may be lost during transport due to network congestion and errors. By means of an error detection code, such as a checksum, the transport protocol may check that the data is not corrupted, and verify correct receipt by sending an ACK or NACK message to the sender. Automatic repeat request schemes may be used to retransmit lost or corrupted data.
- **Flow control:** The rate of data transmission between two nodes must sometimes be managed to prevent a fast sender from transmitting more data than can be supported by the receiving data buffer, causing a buffer overrun. This can also be used to improve efficiency by reducing buffer underrun.
- **Congestion avoidance:** Congestion control can control traffic entry into a telecommunications network, to avoid congestive collapse by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets. For example, automatic repeat requests may keep the network in a congested state; this situation can be avoided by adding congestion avoidance to the flow control, including slow-start. This keeps the bandwidth consumption at a low level in the beginning of the transmission, or after packet retransmission.
- **Multiplexing:** Ports can provide multiple endpoints on a single node. For example, the name on a postal address is a kind of multiplexing and distinguishes between different recipients of the same location. Computer applications will each listen for information on their own ports, which enables the use of more than one network service at the same time. It is part of the transport layer in the TCP/IP model, but of the session layer in the OSI model.

TCP

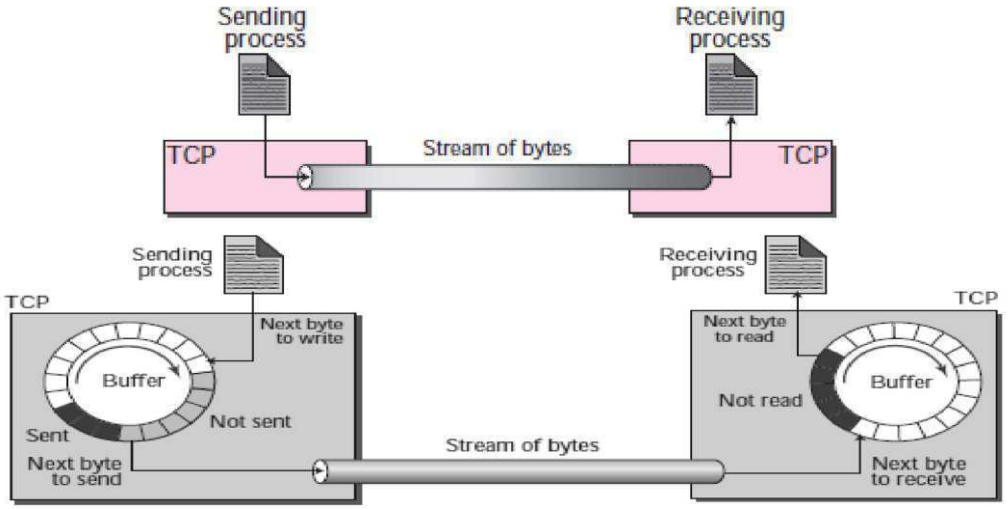
TCP lies between the application layer and the network layer and serves as the intermediary between the application programs and the network operations. Services offered by TCP to the processes at the application layer.

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

Stream Delivery Service: - 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 be connected by an imaginary "tube" that carries their bytes across the Internet.

Sending and Receiving Buffers: - The sending and the receiving processes may not necessarily write or read data at the same rate, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction.



Segments

Although buffering handles the disparity between the speed of the producing and consuming processes, we need one more step before we can send data. The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a segment. TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission.

Full-Duplex Communication

TCP offers full-duplex service, where data can flow in both directions at the same time. Each TCP endpoint then has its own sending and receiving buffer, and segments move in both directions.

Multiplexing and Demultiplexing

Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver. However, since TCP is a connection-oriented protocol, a connection needs to be established for each pair of processes.

Connection-Oriented Service

TCP, unlike UDP, is a connection-oriented protocol. when a process at site A wants to send to and receive data from another process at site B, the following three phases occur:

1. The two TCPs establish a virtual connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

Note that this is a virtual connection, not a physical connection. The TCP segment is encapsulated in an IP datagram and can be sent out of order, or lost, or corrupted, and then resent. Each may be routed over a different path to reach the destination. There is no physical connection. TCP creates a stream-oriented environment in which it accepts the responsibility of delivering the bytes in order to the other site.

Reliable Service

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

TCP FEATURES

Numbering System

Although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields called the sequence number and the acknowledgment number. These two fields refer to a byte number and not a segment number.

Byte Number

TCP numbers all data bytes (octets) that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, TCP stores them in the sending buffer and numbers them. The numbering does not necessarily start from 0. Instead, TCP chooses an arbitrary number between 0 and 232 − 1 for the number of the first byte.

Sequence Number

After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent.

Acknowledgment Number

As we discussed previously, communication in TCP is full duplex; when a connection is established, both parties can send and receive data at the same time. Each party numbers the bytes, usually with a different starting byte number.

Flow Control

TCP, unlike UDP, provides flow control. The sending TCP controls how much data can be accepted from the sending process, the receiving TCP controls how much data can to be sent by the sending TCP.

Error Control

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

Congestion Control

TCP, unlike UDP, considers 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, if any, in the network.

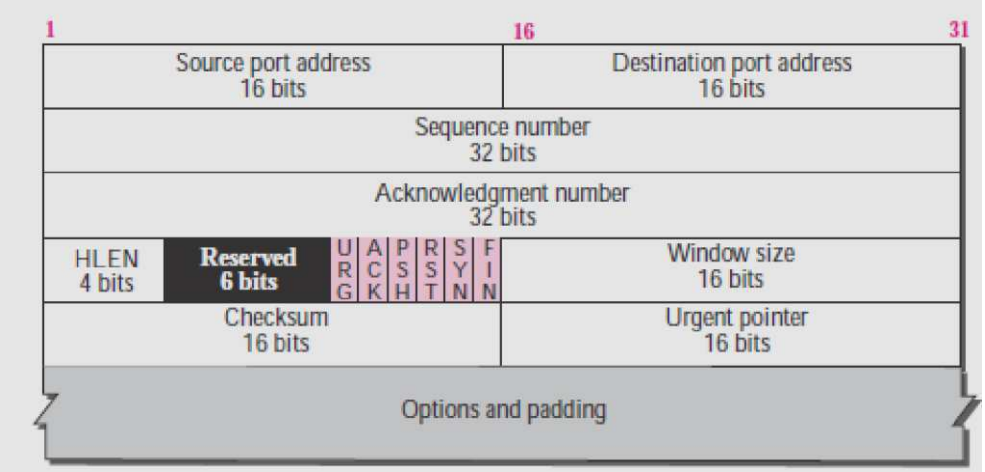
Addressing

TCP communication between two remote hosts is done by means of port numbers (TSAPs). Ports numbers can range from 0 – 65535 which are divided as:

System Ports (0 – 1023)
User Ports (1024 – 49151)
Private/Dynamic Ports (49152 – 65535)

SEGMENT

A packet in TCP is called a segment. The segment consists of a header of 20 to 60 bytes, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options. We will discuss some of the header fields in this section.



Source port address: This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.

Destination port address: This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

Sequence number: This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered.

Acknowledgment number: This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party.

Header length: This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes.

Reserved: This is a 6-bit field reserved for future use.

Control: This field defines 6 different control bits or flags.

URG: Urgent pointer is valid RST: Reset the connection

ACK: Acknowledgment is valid SYN: Synchronize sequence numbers

PSH: Request for push FIN: Terminate the connection

URG	ACK	PSH	RST	SYN	FIN
6 Bits					

Window size: This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes.

Checksum: This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP.

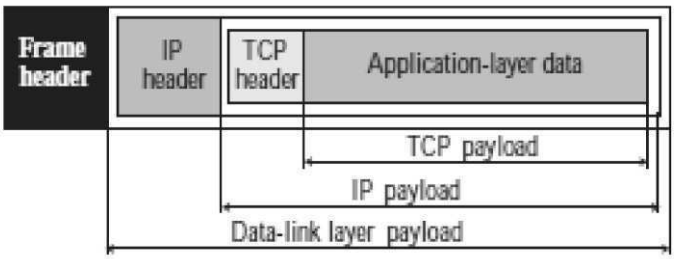
Windows Size: This field is used for flow control between two stations and indicates the amount of buffer (in bytes) the receiver has allocated for a segment, i.e. how much data is the receiver expecting.

Checksum: This field contains the checksum of Header, Data and Pseudo Headers.

Urgent Pointer: It points to the urgent data byte if URG flag is set to 1.

Options: It facilitates additional options which are not covered by the regular header. Options field is always described in 32-bit words. If this field contains data less than 32-bit, padding is used to cover the remaining bits to reach the 32-bit boundary.

Encapsulation: A TCP segment encapsulates the data received from the application layer. The TCP segment is encapsulated in an IP datagram, which in turn is encapsulated in a frame at the data-link layer.



Connection Management in TCP

TCP communication works in Server/Client model. The client initiates the connection and the server either accepts or rejects it. Three-way handshaking is used for connection management. A connection-oriented transport protocol establishes a virtual path between the source and destination. All of the segments belonging to a message are then sent over this virtual path. Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.

Connection Establishment

TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously. This implies that each party must initialize communication and get approval from the other party before any data are transferred.

Before the sending device and the receiving device start the exchange of data, both devices need to be synchronized. During the TCP initialization process, the sending device and the receiving device exchange a few control packets for synchronization purposes. This exchange is known as Three-way handshake.

The Three-way handshake begins with the initiator sending a TCP segment with the SYN control bit flag set.

TCP allows one side to establish a connection. The other side may either accept the connection or refuse it. If we consider this from application layer point of view, the side that is establishing the connection is the client and the side waiting for a connection is the server.

TCP identifies two types of OPEN calls:

Active Open: In an Active Open call, a device (client process) using TCP takes the active role and initiates the connection by sending a TCP SYN message to start the connection.

Passive Open: A passive OPEN can specify that the device (server process) is waiting for an active OPEN from a specific client. It does not generate any TCP message segment. The server processes listening for the clients are in Passive Open mode.

Three-way Handshaking:

Step 1. Device A (Client) sends a TCP segment with SYN = 1, ACK = 0, ISN (Initial Sequence Number) = 2000.

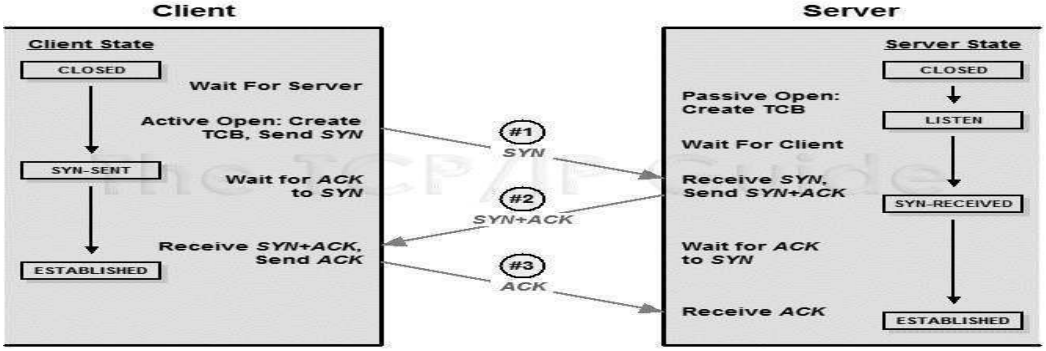
An Initial Sequence Number (ISN) is a random Sequence Number, allocated for the first packet in a new TCP connection.

The Active Open device (Device A) sends a segment with the SYN flag set to 1, ACK flag set to 0 and an Initial Sequence Number 2000 (For Example), which marks the beginning of the sequence numbers for data that device A will transmit. SYN is short for SYNchronize. SYN flag announces an attempt to open a connection.

Step 2. Device B (Server) receives Device A's TCP segment and returns a TCP segment with SYN = 1, ACK = 1, ISN = 5000 (Device B's Initial Sequence Number), Acknowledgment Number = 2001 (2000 + 1, the next sequence number Device B expecting from Device A).

Step 3. Device A sends a TCP segment to Device B that acknowledges receipt of Device B's ISN, with flags set as SYN = 0, ACK = 1, Sequence number = 2001, Acknowledgment number = 5001 (5000 + 1, the next sequence number Device A expecting from Device B)

This handshaking technique is referred to as TCP Three-way handshake or SYN, SYN-ACK, ACK.



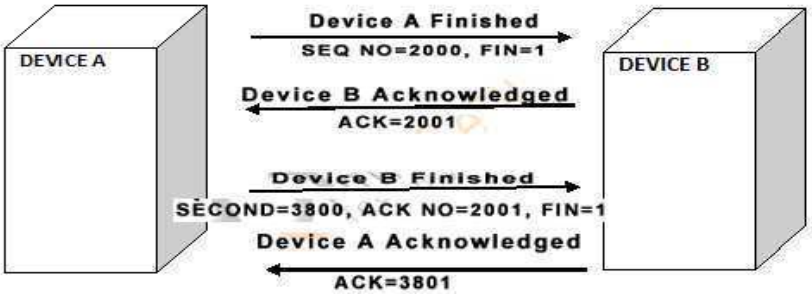
SYN Flooding Attack

The connection establishment procedure in TCP is susceptible to a serious security problem called SYN flooding attack. This happens when one or more malicious attackers send a large number of SYN segments

to a server pretending that each of them is coming from a different client by faking the source IP addresses in the datagrams.

Transmission Control Protocol (TCP) Connection Termination

When the data transmission is complete and the device wants to terminate the connection, the device initiating the termination places a TCP segment (Segment is the name of the data packet at the transport layer, if the protocol is TCP) with the FIN flag set to one. The purpose of FIN bit is to enable TCP to gracefully terminate an established session. The application then enters in a state called the FIN-WAIT state. When at FIN-WAIT state, Device A continues to receive TCP segments from Device B and processes the segments already in the queue, but no additional data is accepted from the application.



Comparison between TCP & UDP

Characteristic / Description	UDP	TCP
General Description	Simple, high-speed, low-functionality "wrapper" that interfaces applications to the network layer and does little else.	A full-featured protocol that allows applications to send data reliably without worrying about network layer issues.
Protocol Connection Setup	Connectionless; data is sent without setup.	Connection-oriented; connection must be established prior to transmission.
Data Interface To Application	Message-based; data is sent in discrete packages by the application.	Stream-based; data is sent by the application with no particular structure.
Reliability and Acknowledgments	Unreliable, best-effort delivery without acknowledgments.	Reliable delivery of messages; all data is acknowledged.
Retransmissions	Not performed. The application must detect lost data and retransmit if needed.	Delivery of all data is managed, and lost data is retransmitted automatically.
Features Provided to Manage Flow of Data	None	Flow control using sliding windows; window size adjustment heuristics; congestion avoidance algorithms.
Overhead	Very low	Low, but higher than UDP

Transmission Speed	Very high	High, but not as high as UDP
Data Quantity Suitability	Small to moderate amounts of data (up to a few hundred bytes)	Small to very large amounts of data (up to gigabytes)
Types of Applications That Use the Protocol	Applications where data delivery speed matters more than completeness, where small amounts of data are sent; or where multicast/broadcast are used.	Most protocols and applications sending data that must be received reliably, including most file and message transfer protocols.
Well-Known Applications and Protocols	Multimedia applications, DNS, BOOTP, DHCP, TFTP, SNMP, RIP, NFS (early versions)	FTP, Telnet, SMTP, DNS, HTTP, POP, NNTP, IMAP, BGP, IRC, NFS (later versions)

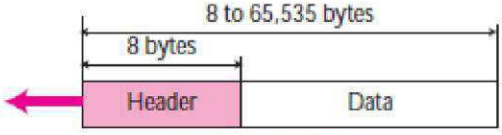
UDP

Requirement of UDP

A question may arise; why do we need an unreliable protocol to transport the data? We deploy UDP where the acknowledgment packets share a significant amount of bandwidth along with the actual data. For example, in the case of video streaming, thousands of packets are forwarded towards its users. Acknowledging all the packets is troublesome and may contain a huge amount of bandwidth wastage. The best delivery mechanism of underlying IP protocol ensures best efforts to deliver its packets, but even if some packets in video streaming get lost, the impact is not calamitous and can be ignored easily. Loss of few packets in video and voice traffic sometimes goes unnoticed.

Features

- UDP is used when acknowledgment of data does not hold any significance.
- UDP is a good protocol for data flowing in one direction.
- UDP is simple and suitable for query based communications.
- UDP is not connection oriented.
- UDP does not provide congestion control mechanism.
- UDP does not guarantee ordered delivery of data.
- UDP is stateless.
- UDP is a suitable protocol for streaming applications such as VoIP, multimedia streaming.



a. UDP user datagram

b. Header format

Source port number. This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535. If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the

process and chosen by the UDP software running on the source host. If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number.

Destination port number. This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

Length. This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. How- ever, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with the total length of 65,535 bytes. The length field in a UDP user datagram is not necessary. A user datagram is encapsulated in an IP datagram. There is a field in the IP datagram that defines the total length.

UDP length = IP length – IP header's length

Checksum. This field is used to detect errors over the entire user datagram (header plus data). The checksum is discussed in the next section.

UDP SERVICES

Process-to-Process Communication

UDP provides process-to-process communication using sock- etc, a combination of IP addresses and port numbers. Several port numbers used by UDP.

Connectionless Services

As mentioned previously, UDP provides a connectionless service. This 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 program.

Flow Control

UDP is a very simple protocol. There is no flow control, and hence no window mechanism. The receiver may overflow with incoming messages. The lack of flow control means that the process using UDP should provide for this service if needed.

Error Control

There is no error control mechanism in UDP except for the checksum. This means that 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.

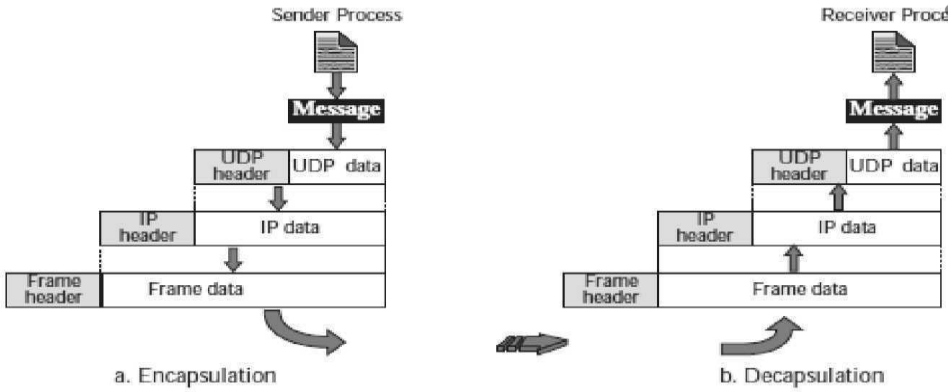
Checksum

UDP checksum calculation is different from the one for IP. Here the checksum includes three sections: a pseudo header, the UDP header, and the data coming from the application layer.

Congestion Control

Since UDP is a connectionless protocol, it does not provide congestion control. UDP assumes that the packets sent are small and sporadic, and cannot create congestion in the network. This assumption may or may not be true today when UDP is used for real-time transfer of audio and video.

Encapsulation and decapsulation



Encapsulation

When a process has a message to send through UDP, it passes the message to UDP along with a pair of socket addresses and the length of data. UDP receives the data and adds the UDP header. UDP then passes the user datagram to IP with the socket addresses. IP adds its own header, using the value 17 in the protocol field, indicating that the data has come from the UDP protocol.

Decapsulation

When the message arrives at the destination host, the physical layer decodes the signals into bits and passes it to the data link layer. The data link layer uses the header (and the trailer) to check the data. If there is no error, the header and trailer are dropped and the datagram is passed to IP. The IP software does its own checking. If there is no error, the header is dropped and the user datagram is passed to UDP with the sender and receiver IP addresses.

Integrated Services & Differentiated Services

INTEGRATED SERVICES

Two models have been designed to provide quality of service on the Internet:

- Integrated Services
- Differentiated Services

Both models emphasize the use of quality of service at the network layer (IP), although the model can also be used in other layers such as the data link layer. As we learned in Chapter 20, IP was originally designed for best-effort delivery. This means that every user receives the same level of services. This type of delivery does not guarantee the minimum of a service, such as bandwidth, to applications such as real-time audio and video. If such an application accidentally gets extra bandwidth, it may be detrimental to other applications, resulting in congestion. Integrated Services, sometimes called Int-Serv, is a flow-based QoS model, which means that a user needs to create a flow, a kind of virtual circuit, from the source to the destination and inform all routers of the resource requirement. Integrated Services is a flow based QoS model designed for IP.

Signaling

The reader may remember that IP is a connectionless, datagram, packet-switching protocol. How can we implement a flow-based model over a connectionless protocol? The solution is a signaling protocol to run over IP that provides the signaling mechanism for making a reservation. This protocol is called Resource Reservation Protocol (RSVP) and will be discussed shortly.

Flow Specification

When a source makes a reservation, it needs to define a flow specification. A flow specification has two

parts:

- Rspec (resource specification)
- Tspec (traffic specification)

Rspec defines the resource that the flow needs to reserve (buffer, bandwidth, etc.). Tspec defines the traffic characterization of the flow.

Admission

After a router receives the flow specification from an application, it decides to admit or deny the service. The decision is based on the previous commitments of the router and the current availability of the resource.

Service Classes

Two classes of services have been defined for Integrated Services:

- Guaranteed service
- Controlled-load service

Guaranteed Service Class

This type of service is designed for real-time traffic that needs a guaranteed minimum end-to-end delay. The end-to-end delay is the sum of the delays in the routers, the propagation delay in the media, and the setup mechanism. Only the first, the sum of the delays in the routers, can be guaranteed by the router. This type of service guarantees that the packets will arrive within a certain delivery time and are not discarded if flow traffic stays within the boundary of Tspec. We can say that guaranteed services are quantitative services, in which the amount of end-to-end delay and the data rate must be defined by the application.

Controlled-Load Service Class

This type of service is designed for applications that can accept some delays but are sensitive to an overloaded network and to the danger of losing packets. Good examples of these types of applications are file transfer, e-mail, and Internet access. The controlled load service is a qualitative type of service in that the application requests the possibility of low-loss or no-loss packets.

RSVP (Resource reservation protocol)

In the Integrated Services model, an application program needs resource reservation. As we learned in the discussion of the Int-Serv model, the resource reservation is for a flow. This means that if we want to use Int-Serv at the IP level, we need to create a flow, a kind of virtual-circuit network, out of the IP, which was originally designed as a datagram packet-switched network. A virtual-circuit network needs a signaling system to set up the virtual circuit before data traffic can start. The Resource Reservation Protocol (RSVP) is a signaling protocol to help IP create a flow and consequently make a resource reservation. Before discussing RSVP, we need to mention that it is an independent protocol separate from the Integrated Services model. It may be used in other models in the future.

Multicast Trees

RSVP is different from some other signaling systems we have seen before in that it is a signaling system designed for multicasting. However, RSVP can be also used for unicasting because unicasting is just a special case of multicasting with only one member of the multicast group. The reason for this design is to enable RSVP to provide resource reservations for all kinds of traffic including multimedia which often uses multicasting.

Receiver-Based Reservation

In RSVP, the receivers, not the sender, make the reservation. This strategy matches the other multicasting protocols. For example, in multicast routing protocols, the receivers, not the sender, make a decision to join

or leave a multicast group.

RSVP Messages

RSVP has several types of messages. However, for our purposes, we discuss only two of them:

- Path
- Resv

Path Messages Recall that the receivers in a flow make the reservation in RSVP. However, the receivers do not know the path traveled by packets before the reservation is made. The path is needed for the reservation. To solve the problem, RSVP uses Path messages. A Path message travels from the sender and reaches all receivers in the multicast path. On the way, a Path message stores the necessary information for the receivers. A Path message is sent in a multicast environment; a new message is created when the path diverges.

Resv Messages After a receiver has received a Path message, it sends a Resv message. The Resv message travels toward the sender (upstream) and makes a resource reservation on the routers that support RSVP. If a router does not support RSVP on the path, it routes the packet based on the best-effort delivery methods we discussed before.

Reservation Merging

In RSVP, the resources are not reserved for each receiver in a flow; the reservation is merged. Rc3 requests a 2-Mbps bandwidth while Rc2 requests an I-Mbps bandwidth. Router R3, which needs to make a bandwidth reservation, merges the two requests. The reservation is made for 2 Mbps, the larger of the two, because a 2-Mbps input reservation can handle both requests. The same situation is true for R2. The reader may ask why Rc2 and Rc3, both belonging to one single flow, request different amounts of bandwidth. The answer is that, in a multimedia environment, different receivers may handle different grades of quality. For example, Rc2 may be able' to receive video only at 1 Mbps (lower quality), while Rc3 may be able to receive video at 2 Mbps (higher quality).

Reservation Styles

When there is more than one flow, the router needs to make a reservation to accommodate all of them. RSVP defines three types of reservation styles.

Wild Card Filter Style In this style, the router creates a single reservation for all senders. The reservation is based on the largest request. This type of style is used when the flows from different senders do not occur at the same time. Fixed Filter Style In this style, the router creates a distinct reservation for each flow. This means that if there are n flows, n different reservations are made. This type of style is used when there is a high probability that flows from different senders will occur at the same time.

Shared Explicit Style In this style, the router creates a single reservation which can be shared by a set of flows.

Soft State

The reservation information (state) stored in every node for a flow needs to be refreshed periodically. This is referred to as a soft state as compared to the hard state used in other virtual circuit protocols such as ATM or Frame Relay, where the information about the flow is maintained until it is erased. The default interval for refreshing is currently 30 s.

Problems with Integrated Services

There are at least two problems with Integrated Services that may prevent its full implementation on the Internet:

- Scalability
- Service-type limitation

Scalability

The Integrated Services model requires that each router keeps information for each flow. As the Internet is growing every day, this is a serious problem.

Service-Type Limitation

The Integrated Services model provides only two types of services, guaranteed and control the load. Those opposing this model argue that applications may need more than these two types of services.

DIFFERENTIATED SERVICES

Differentiated Services (DS or Diffserv) was introduced by the IETF (Internet Engineering Task Force) to handle the shortcomings of Integrated Services. Two fundamental changes were made:

- The main processing was moved from the core of the network to the edge of the network. This solves the scalability problem. The routers do not have to store
- Information about flows. The applications, or hosts, define the type of service they need each time they send a packet.
- The per-flow service is changed to per-class service. The router routes the packet based on the class of service defined in the packet, not the flow. This solves the service-type limitation problem. We can define different types of classes based on the needs of applications.

INTERNETWORKING DEVICE

An internetworking device is a widely-used term for any hardware within networks that connect different network resources. Key devices that comprise a network are

- Routers
- Bridges
- Repeaters
- Gateways

Routers

Routers are highly intelligent network devices that are primarily used for large networks and provide the best data path for effective communication. Routers have memory chips which store large quantities of network addresses.

A router is a device that analyzes the contents of data packets transmitted within a network or to another network. Routers determine whether the source and destination are on the same network or whether data must be transferred from one network type to another, which requires encapsulating the data packet with routing protocol header information for the new network type.

Bridges

Bridges are used to connect two large networks by providing different network services.

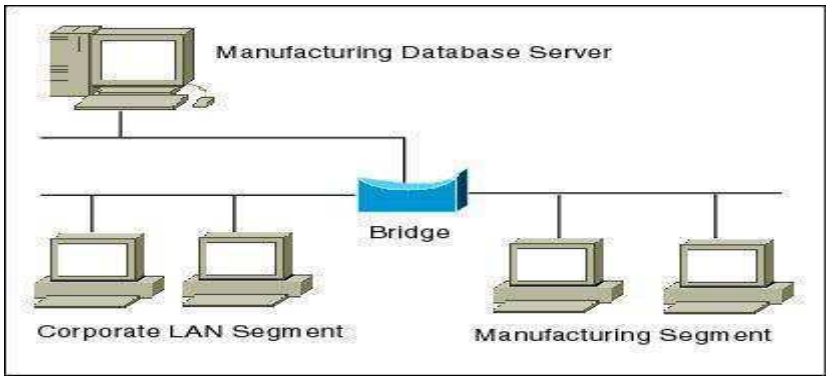
A bridge is a type of computer network device that provides interconnection with other bridge networks that use the same protocol.

Bridge devices work at the data link layer of the Open System Interconnect (OSI) model, connecting two different networks together and providing communication between them. Bridges are similar to repeaters and hubs in that they broadcast data to every node. However, bridges maintain the media access control (MAC) address table as soon as they discover new segments, so subsequent transmissions are sent to only to the desired recipient.

Bridges are also known as Layer 2 switches.

A bridge uses a database to ascertain where to pass, transmit or discard the data frame.

1. If the frame received by the bridge is meant for a segment that resides on the same host network, it will pass the frame to that node and the receiving bridge will then discard it.
2. If the bridge receives a frame whose node MAC address is of the connected network, it will forward the frame toward it.



Repeaters

Repeaters are used for signal and data regeneration and are primarily responsible for data amplification.

The term "repeater" originated with telegraphy in the 19th century and referred to an electromechanical device used to regenerate telegraph signals. Use of the term has continued in telephony and data communications.

In telecommunication, the term repeater has the following standardized meanings:

1. An analog device that amplifies an input signal regardless of its nature (analog or digital).
2. A digital device that amplifies, reshapes, retimes, or performs a combination of any of these functions on a digital input signal for retransmission. A repeater that includes the retiming function is also known as a regenerator.

In computer networking, because repeaters work with the actual physical signal, and do not attempt to interpret the data being transmitted, they operate on the physical layer, the first layer of the OSI model.

Gateways

Gateways are internetworking devices used to convert formats and are the backbone of any network architecture. The term gateway has the following meaning:

- Gateway is a router or a proxy server that routes between networks
- Gateway Rule - Gateway should belong to the same subnet to which your
- PC belongs
- In a communications network, a network node equipped for interfacing with another network that uses different protocols.
- A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.
- A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.
- Loosely, a computer or computer program configured to perform the tasks of a gateway. For a specific case, see default gateway.
- Gateways, also called protocol converters, can operate at any network layer. The activities of a

gateway are more complex than that of the router or switch as it communicates using more than one protocol.

Switch

A switch, in the context of networking, is a high-speed device that receives incoming data packets and redirects them to their destination on a local area network (LAN). A LAN switch operates at the data link layer (Layer 2) or the network layer of the OSI Model and, as such, it can support all types of packet protocols.

Essentially, switches are the traffic cops of a simple local area network.

A switch in an Ethernet-based LAN reads incoming TCP/IP data packets/frames containing destination information as they pass through one or more input ports. The destination information in the packets is used to determine which output ports will be used to send the data on to its intended destination.

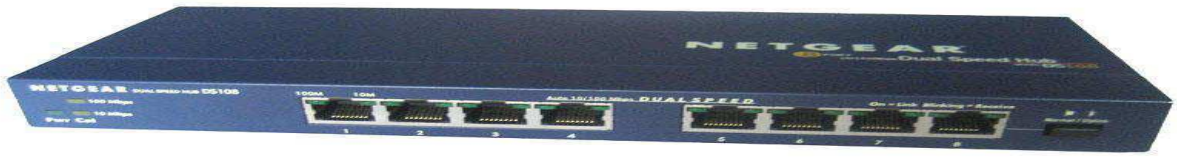
Switches are similar to hubs, only smarter. A hub simply connects all the nodes on the network communication is essentially in a haphazard manner with any device trying to communicate at any time, resulting in many collisions. A switch, on the other hand, creates an electronic tunnel between the source and destination ports for a split second that no other traffic can enter. This results in communication without collisions.

Switches are similar to routers as well, but a router has the additional ability to forward packets between different networks, whereas a switch is limited to node-to-node communication on the same network.

Hub

A hub is the connection point in a computer device where data from many directions converge and are then sent out in many directions to respective devices. A hub may also act as a switch by preventing specific data packets from proceeding to a destination.

In addition to receiving and transmitting communication data, a hub may also serve as a switch. For example, an airport acts much like a hub in the sense that passengers converge there and head out in many different directions. Suppose that an airline passenger arrives at the airport hub and is then called back home unexpectedly, or receives instructions to change his or her destination. The same may occur with a computing hub when it acts as a switch by preventing specific data packets from proceeding to a destination while sending other data packets on a specific route. Where packets are sent depends on attributes (MAC addresses) within the data packets. A switch may also act as a hub.



Layered Communication

Network communication models are generally organized into layers. The OSI model specifically consists of seven layers, with each layer representing a specific networking function. These functions are controlled by protocols, which govern end-to-end communication between devices. As data is passed from the user application down the virtual layers of the OSI model, each of the lower layers adds a header (and sometimes a trailer) containing protocol information specific to that layer. These headers are called Protocol Data Units (PDUs), and the process of adding these headers is referred to as encapsulation.

The PDU of each lower layer is identified with a unique term:

Commonly, network devices are identified by the OSI layer they operate at (or, more specifically, what header or PDU the device processes). For example, switches are generally identified as Layer-2 devices, as switches process information stored in the Data-Link header of a frame (such as MAC addresses in Ethernet). Similarly, routers are identified as Layer-3 devices, as routers process logical addressing information in the Network header of a packet (such as IP addresses).

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in