4.6 Congestion Avoidance

TCP impose some methods to control congestion once it happens, instead of trying to avoid congestion.It is a prevention mechanism while congestion control is a recovery mechanism.

DECbit

DECbit means destination experiencing congestion bit. This mechanism was developed for use on the Digital Network Architecture (DNA), a connectionless network with a connection-oriented transport protocol.This mechanism could, therefore, also be applied toTCP and IP.It split the responsibility for congestion control between the routers and the end nodes.

Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur. This notification is implemented by setting a binary congestion bit in the packets that flow through the router, hence the name DECbit.The destination host then copies this congestion bit into the ACK it sends back to the source. Finally, the source adjusts its sending rate so as to avoid congestion.

A router set this bit in a packet if its average queue length is greater than or equal to 1 at the time the packet arrives.This average queue length is measured over a time interval as , the last busy+idle cycle, plus the current busy cycle.The source records how many of its packets has set the congestion bit.If less than 50% of the packets had the bit set, then the source increases its congestion window byone packet. If 50% or more of the last window of packets had the congestion bit set, then the source decreases its congestion window to0.875 times the previous value.

Random Early Detection (RED)

RED provide congestion control at the router for TCP flows.RED was designed to work with TCP. Red notifies the sender by dropping packets. Packet dropping probability is increased as the average queue length increases. The moving average of the queue length is used to detect the long term congestion and allows short term bursts to arrive.

Properties of RED

1.RED drops packets before queue is full, in the hope of reducing the rates of some flows.

2.Drops packet for each flow roughly in proportion to its rate.

3. Red maintains average queue length.

4. Random drops desynchronize the TCP sources.

5. RED calculates the average que length using a weighted running average.

The Formula is as follows.

Average length = (1- Weight) x Average length + Weight x Sample length

Sample length is the queue length each time a packet arrives. The weight parameter is between 0 and 1.

RED has two queue length thresholds that trigger certain activity: MinThreshold and MaxThreshold.

When a packet arrives at the gateway, RED compares the current AvgLen with these two thresholds,according to the following rules:

ifAvgLen≤ MinThreshold

Then queue the packet

ifMinThreshold<AvgLen<MaxThreshold

calculate probability P

drop the arriving packet with probability P

ifMaxThreshold≤AvgLength

!drop the arriving packet

If the average queue length is smaller than the lower threshold, no action is taken, and if the average queue length is larger than the upper threshold, then the packet is always dropped. If the average queue length is between the two thresholds, then the newly arriving packet is dropped with some probability P.

4.5 FLOW CONTROL

Flow control balances the rate a producer creates data with the rate a consumer can use the data. Figure 4.5.1 shows unidirectional data transfer between a sender and a receiver; bidirectional data transfer can be deduced from the unidirectional process.

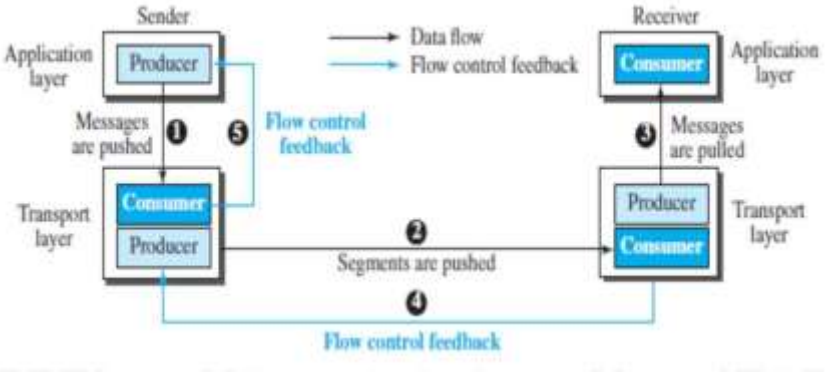

Fig4.5.1: Data flow and flow control feedbacks in TCP.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-763]

The figure shows that data travel from the sending process down to the sending TCP, from the sending TCP to the receiving TCP, and from the receiving TCP up to the receiving process (paths 1, 2, and 3). Flow control feedbacks, are traveling from the receiving TCP to the sending TCP and from the sending TCP up to the sending process (paths 4 and 5). In other words, the receiving TCP controls the sending TCP; the sending TCP controls the sending process.Flow control feedback from the sending TCP to the sending process (path 5) is achieved through simple rejection of data by the sending TCP when its window is full.This means that our discussion of flow control concentrates on the feedback sent from the receiving TCP to the sending TCP (path 4).

Opening and Closing Windows

To achieve flow control, TCP forces the sender and the receiver to adjust their window sizes, although the size of the buffer for both parties is fixed when the connection is established. The receive window closes (moves its left wall to the right) when more bytes arrive from the sender; it opens (moves its right wall to the right) when more bytes are pulled by the process. We assume that it does not shrink (the right wall does not move to the left).The opening, closing, and shrinking of the send window is controlled by the receiver. The send window closes (moves its left wall to the right) when a new acknowledgment allows it to do so.

ERROR CONTROL

TCP is a reliable transport-layer protocol. This means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, without error, and without any part lost or duplicated.

TCP provides reliability using error control. Error control includes mechanisms for detecting and resending corrupted segments, resending lost segments, storing out-of order segments until missing segments arrive, and detecting and discarding duplicated segments. Error control in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.

Checksum

Each segment includes a checksum field, which is used to check for a corrupted segment.If a segment is corrupted, as detected by an invalid checksum, the segment is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment.

TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data, but consume a sequence number, are also acknowledged. ACK segments are never acknowledged.

CONGESTION CONTROL IN TCP

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. Congestion control refers to the mechanisms and techniques to control the congestion.TCP uses a congestion window and a congestion policy that avoid congestion.If the network cannot able to deliver the data as fast as it is created by the sender, it must tell the sender to slow down.

Congestion policy in TCP

Slow Start Phase: starts slowly increment is exponential to threshold

1.Congestion Avoidance Phase: After reaching the threshold increment is by 1.

2.Congestion Detection Phase: Sender goes back to Slow start phase or Congestion avoidance phase.

Slow Start , exponential increase – In this phase after every RTT(round trip time) the congestion window size increments exponentially.

The slow-start algorithm is based on the idea that the size of the congestion window (cwnd) starts with one maximum segment size (MSS), but it increases one MSS each time an acknowledgment arrives. The sender starts with cwnd ?1. This means that the sender can send only one segment as in figure 4.5.2. After the first ACK arrives, the acknowledged segment is purged from the window, which means there is now one empty segment slot in the window. The size of the congestion window is also increased by 1 because the arrival of the acknowledgment is a good sign that there is no congestion in the network. The size of the window is now 2. After sending two segments and receiving two individual acknowledgments for them, the size of the congestion window now becomes 4, and so on.

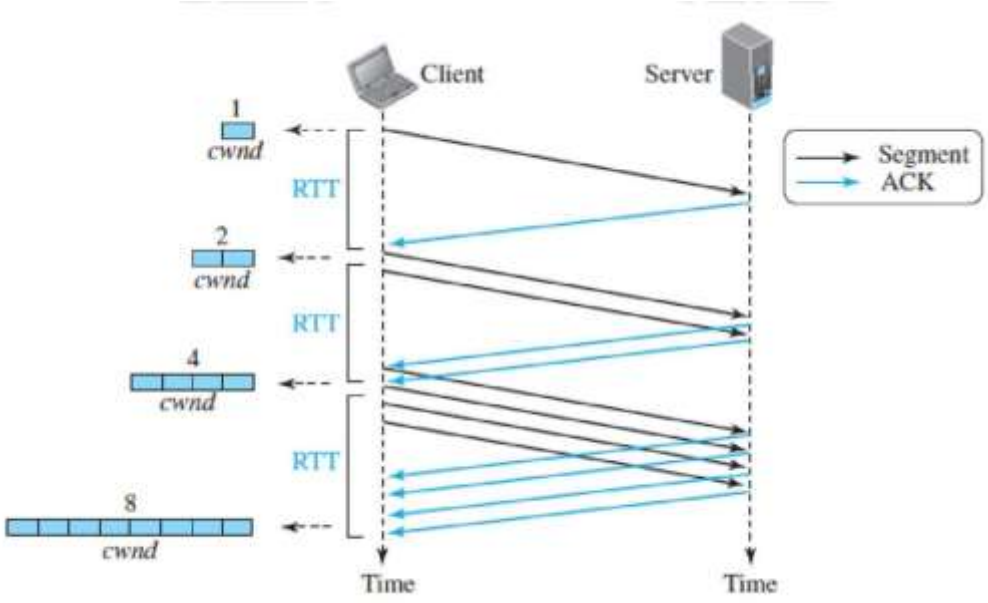

Fig4.5.2: Slow start exponential increase.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-779]

Start	→	$cwnd = 1 \rightarrow 2^0$
After 1 RTT	→	$cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1$
After 2 RTT	→	$cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2$
After 3 RTT	→	$cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3$

Congestion Avoidance Phase : additive increase

To avoid congestion before it happens.This phase starts after the threshold value is denoted as ssthresh. The size of cwnd(congestion window) increases additive as shown in figure 4.5.3.When the size of the congestion window reaches the slow-start threshold in the case where cwnd =i, the slow-start phase stops and the additive phase begins.

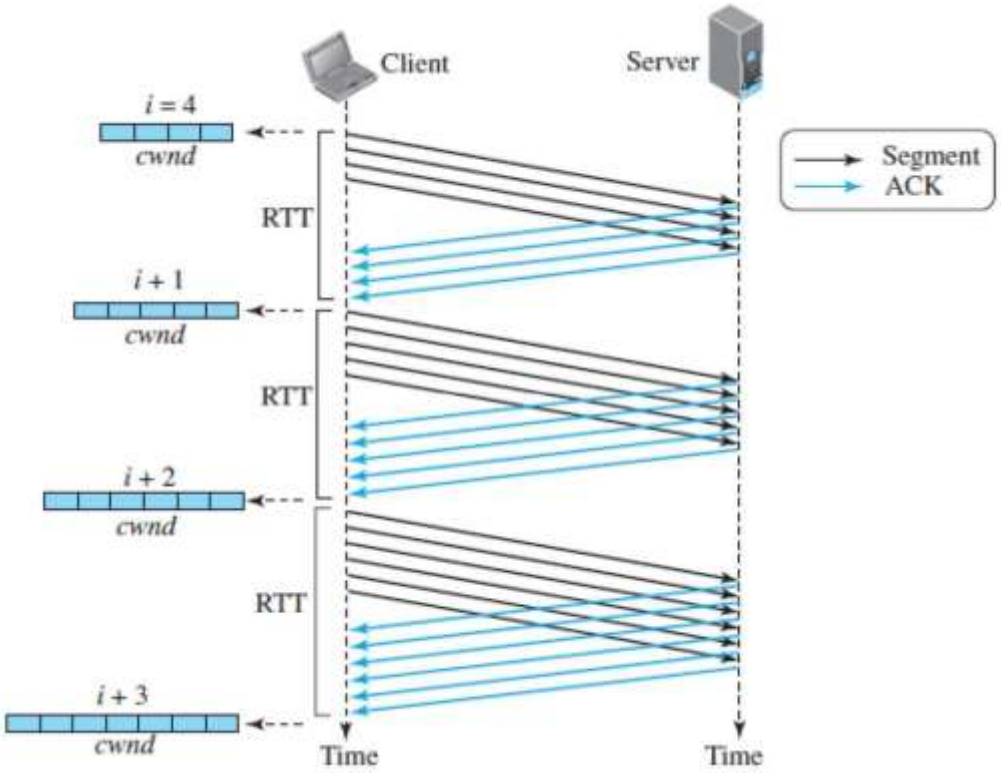

Fig4.5.3:TCP congestion avoidance.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-780]

To avoid congestion before it happens.This phase starts after the threshold value is denoted as ssthresh. The size of cwnd(congestion window) increases additive as shown in figure 4.5.3.When the size of the congestion window reaches the slow-start threshold in the case where cwnd =i, the slow-start phase stops and the additive phase begins. In this algorithm, each time the whole "window" of segments is acknowledged, the size of the congestion window is increased by one. After each RTT cwnd = cwnd + 1.

Congestion Detection Phase : multiplicative decrement – If congestion occurs, the congestion window size is decreased. The only way a sender knows that congestion has occurred is the need to retransmit a segment. Retransmission is needed to recover a missing packet which is dropped by a router due to congestion.

Retransmission can occur in one of two cases: when the RTO timer times out or when three duplicate ACKs are received.

Case 1 : Retransmission due to Timeout . In this case congestion possibility is high.

A),ssthresh (Slow start threshold) is reduced to half of the current window size.

B).set cwnd = 1

C).start with slow start phase again.

Case 2 : Retransmission due to 3 Acknowledgement Duplicates .

In this case congestion possibility is less.

(a) ssthresh value reduces to half of the current window size.
(b) set cwnd= ssthresh
(c) start with congestion avoidance phase

4.7 QOS (QUALITY OF SERVICE)

Quality of service (QoS) refers to any technology that manages data traffic to reduce packet loss, latency and jitter on the network. QoS controls and manages network resources by setting priorities for specific types of data on the network.

FLOW CONTROL TO IMPROVE QOS

At the routers place a packet may be delayed, suffer from jitters, be lost.

A good scheduling technique treats the different flows in a fair and appropriate manner.

Several scheduling techniques are used to improve the quality of service.

FIFO Queuing

In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router) 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. Packets from different applications (with different sizes) arrive at the queue, are processed, and depart.

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.The system does not stop serving a queue until it is empty.

A packet priority is determined from a specific field in the packet header. That is the ToS field of an IPv4 header, the priority field of IPv6 and apriority number assigned to a destination address.

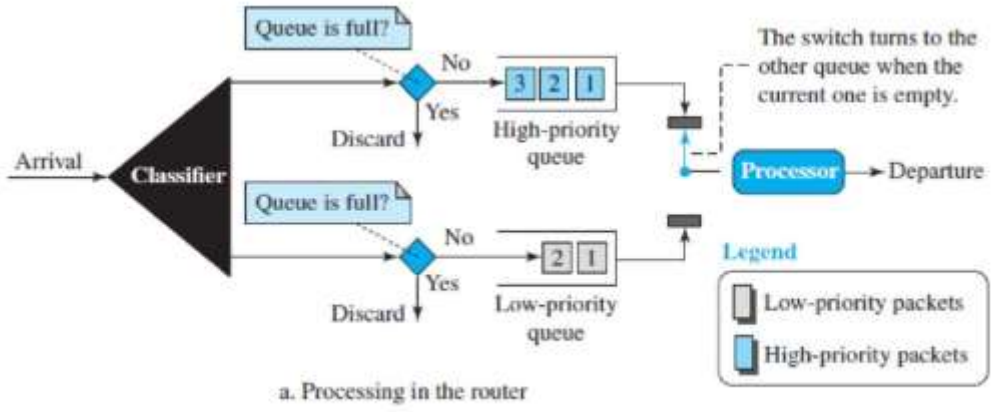

Fig4.7.1: Priority queuing.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-1058]

Figure 4.7.1 shows the priority queuing with two priority levels (for simplicity). 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.

Traffic Shaping or Policing

Traffic shaping term is used when the traffic leaves a network. The policing term is used when the data enters the network.

Leaky Bucket

If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket.

The rate at which the water leaks does not depend on the rate at which the water is input unless the bucket is empty. Here the input rate can vary, but the output rate remains constant.

Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.

Figure4.7.2 shows the leaky bucket concept.

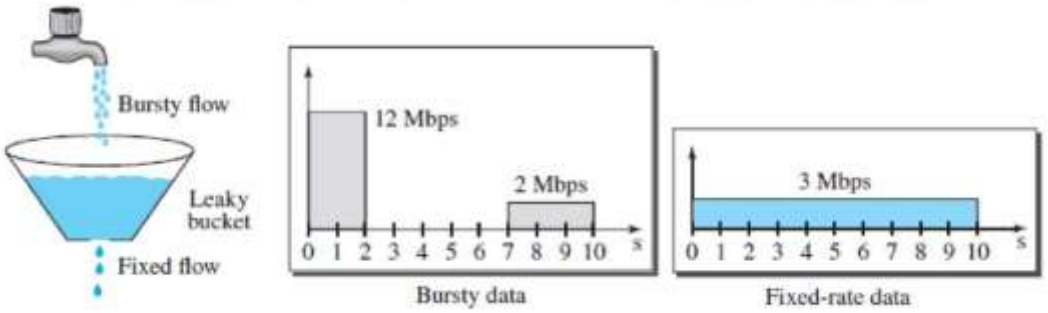

Fig4.7.2: Leaky bucket concept.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-1059]

Here the host sends a burst of data at a rate of 12 Mbps for 2 seconds, for a total of 24 Mb of data. The host is silent for 5 seconds and then sends data at a rate of 2 Mbps for 3 seconds, for a total of 6 Mb of data.In all, the host has sent 30 Mb of data in 10 seconds. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 seconds.

A simple leaky bucket implementation is shown below. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The algorithm for variable-length packets:

Initialize a counter to n at the tick of the clock.

1. If n is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until the counter value is smaller than the packet size.
2. Reset the counter to n and go to step 1.

Token Bucket

The leaky bucket algorithm does not consider the idle time of the host. The token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens as shown in figure 4.7.3. Assume the capacity of the bucket is c tokens and tokens enter the bucket at the rate of r tokens per second. The system removes one token for every cell of data sent. The maximum number of cells that can enter the network during any time interval of length is shown below.

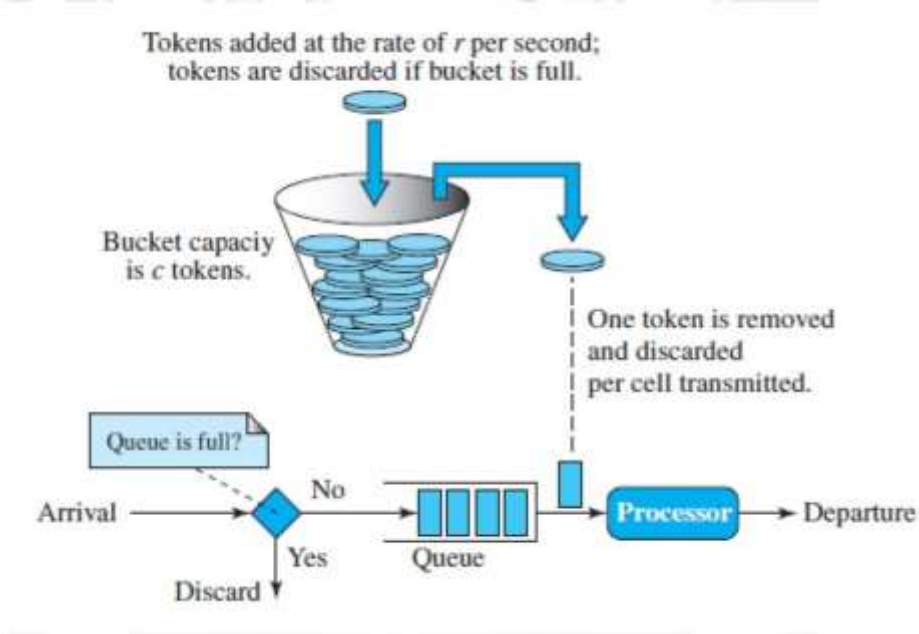

Fig4.7.3: Token bucket concept.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-1061]

Maximum number of packets r * t +c

The maximum average rate for the token bucket is shown below.

Maximum average rate. r * t +c/ t , packets per second.This means that the token bucket limits the average packet rate to the network.

Resource Reservation

A flow of data needs resources such as a buffer, bandwidth and CPU time. The quality of service is improved if these resources are reserved.

Integrated Services, depends heavily on resource reservation to improve the quality of service.

Admission Control

Admission control refers to the mechanism used by a router or a switch to accept or reject a flow based on predefined parameters called flow specifications. Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity can handle the flow.

INTEGRATED SERVICES (INTSERV)

Today internet provide only the best-effort delivery service to all users .

To provide different QoS for different applications, IETF developed the Integrated Services (IntServ)model.

This model, is a flow-based architecture. Resources such as bandwidth are explicitly reserved for a given data flow.

The model is based on three schemes:

The packets are first classified according to the service they require.

The model uses scheduling to forward the packets according to their flow characteristics.

Devices like routers use admission control to determine if the device has the capability (available resources to handle the flow) before making a commitment.

For example, if an application requires a very high data rate, but a router in the path cannot provide such a data rate, it denies the admission.

Flow Specification

To define a specific flow, a source needs to define a flow specification, which has two parts:

Rspec (resource specification). Rspec defines the resource that the flow needs to reserve (buffer, bandwidth, etc.).

Tspec (traffic specification). Tspec defines the traffic characterization of the flow.

Service Classes

Two classes of services for Integrated Services 1.guaranteed service 2.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.

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.

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.For example, types of applications are file transfer, e-mail, and Internet access. The controlled load service is a qualitative service in that the application requests the possibility of low-loss or no-loss packets.

Resource Reservation Protocol (RSVP)

RSVP is based on multicast communication.

Receiver-Based Reservation

In RSVP, the receivers make the reservation. This strategy matches the other multicasting protocols. For example, in multicast routing protocols, the receivers make a decision to join or leave a multicast group.

RSVP Messages

Path and Resv.

Path Messages

The path is needed for the reservation. RSVP uses Path messages as shown in figure 4.7.4.

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.

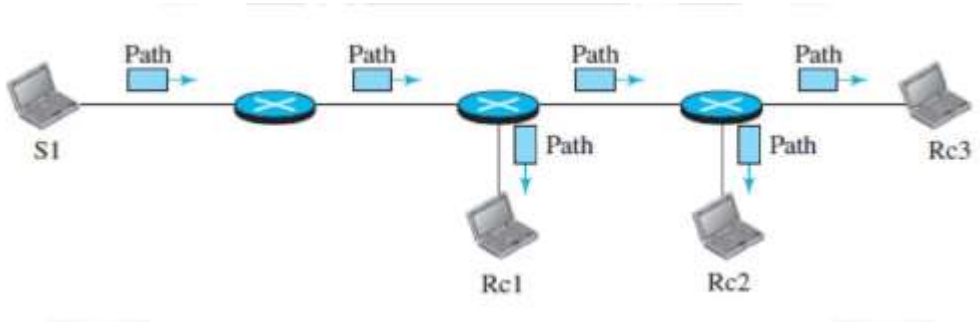

Fig4.7.4: RSVP path message.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-1064]

Resv Messages

After a receiver has received a Path message, it sends a Resv message as shown in figure 4.7.5. The Resv message travels toward the sender (upstream) and makes a resource reservation on the routers that support RSVP.If a router on the path does not support RSVP, it routes the packet based on the best effort delivery methods .

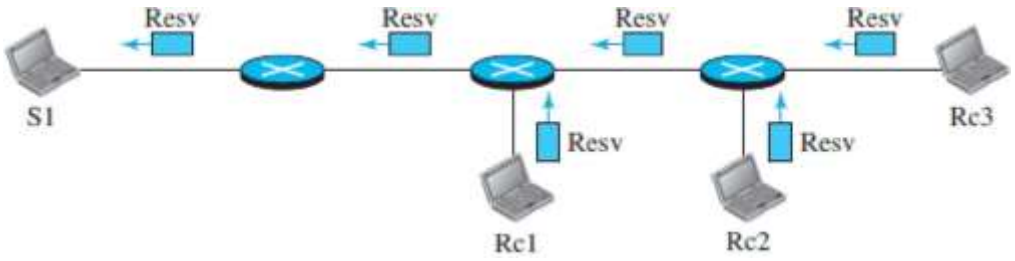

Fig4.7.5: Rsvp reserve message.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-1065]

Reservation Merging

In RSVP, the resources are not reserved for each receiver in a flow; the reservation is merged.

In Figure 4.7.6, Rc3 requests a 2-Mbps bandwidth while Rc2 requests a 1-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.

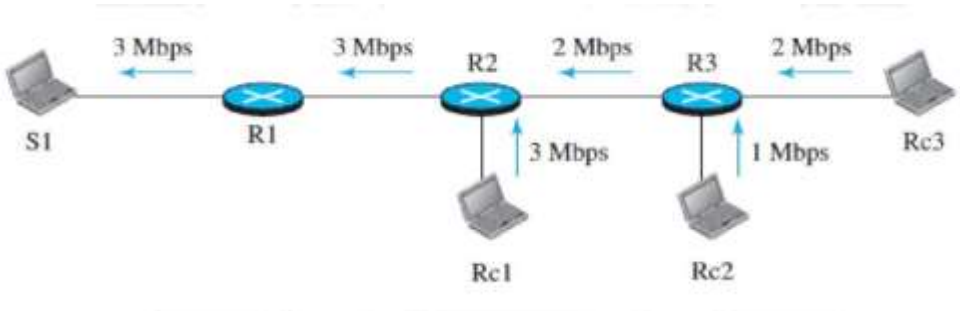

Fig4.7.6: Reservation merging.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-1065]

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: wildcard filter (WF), fixed filter (FF), and shared explicit (SE).

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 that can be shared by a set of flows.

DIFFERENTIATED SERVICES (DIFFSERV)

In this model, called DiffServ**,** packets are marked by applications into classes according to their priorities.Routers and switches, using various queuing strategies, route the packets. This model 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.

DS Field

In DiffServ, each packet contains a field called the DS field. The value of this field is set at the boundary of the network by the host or the first router designated as the boundary router.

IETF proposes to replace the existing ToS (type of service) field in IPv4 or the priority class field in IPv6 with the DS field.The DS field contains two subfields: DSCP and CU.

The DSCP (Differentiated Services Code Point) is a 6-bit subfield that defines the perhop behavior (PHB). The2-bit CU (Currently Unused) subfield is not currently used.

Per-Hop Behavior

The DiffServ model defines per-hop behaviors (PHBs) for each node that receives a packet. Three PHBs are defined: DE PHB, EF PHB, and AF PHB.

DE PHB

The DE PHB (default PHB) is the same as best-effort delivery, which is compatible with ToS.

EF PHB

The EF PHB (expedited forwarding PHB) provides the following services: A**.** Low loss.

B.Low latency. C.Ensured bandwidth.

This is the same as having a virtual connection between the source and destination.

AF PHB

The AF PHB (assured forwarding PHB) delivers the packet with a high assurance as long as the class traffic does not exceed the traffic profile of the node.

The users of thenetwork need to be aware that some packets may be discarded.

Traffic Conditioners

To implement DiffServ, the DS node uses traffic conditioners such as meters, markers,shapers, and droppers as shown in figure 4.7.7.

Meter

The meter checks to see if the incoming flow matches the negotiated traffic profile. The meter also sends this result to other components. The meter can use several tools such as a token bucket to check the profile.

Marker

A marker can re-mark a packet that is using best-effort delivery (DSCP: 000000) or down-mark a packet based on information received from the meter. Down marking (lowering the class of the flow) occurs if the flow does not match the profile. A marker does not up-mark a packet (promote the class).

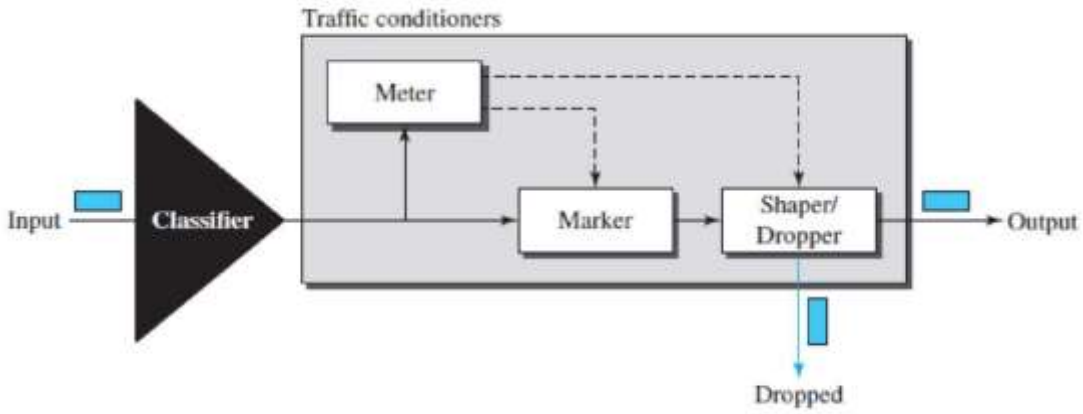

Fig4.7.7: Traffic conditioners.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-1067]

Shaper

A shaper uses the information received from the meter to reshape the traffic if it is not compliant with the negotiated profile.

Dropper

A dropper, which works as a shaper with no buffer, discards packets if the flow severely violates the negotiated profile.

4.4 STATE TRANSITION DIAGRAM

To observe the events happening during connection establishment, connection termination, and data transfer,TCP is specified as the finite state machine (FSM**)** as shown in Figure 4.4.1.

Here two FSMs used by the TCP client and server combined in one diagram.The rounded-corner rectangles represent the states. The transition from one state to another is shown using directed lines. Each line has two strings separated by a slash. The first string is the input, what TCP receives. The second is the output, what TCP sends.

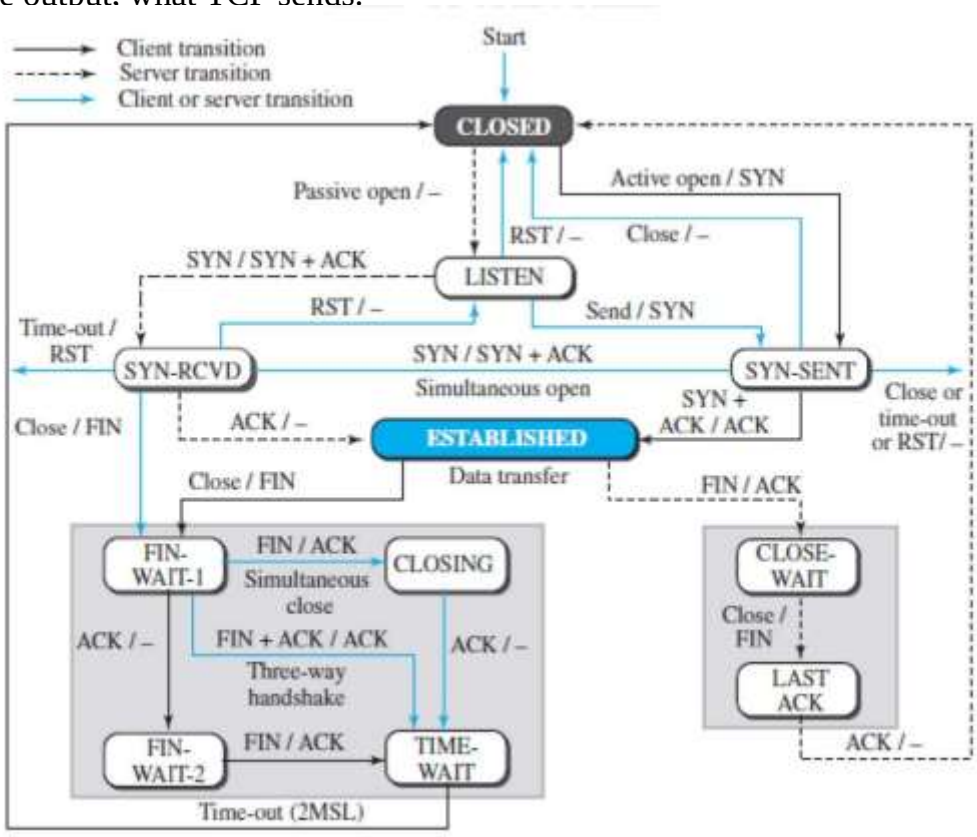

Fig4.4.1: State transition diagram.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-758]

.

The dotted black lines in the figure represent the transition that a server normally goes through; the solid black lines show the transitions that a client normally goes through.In some situations, a server transitions through a solid line or a client transitions through a dotted line. The colored lines show special situations. The rounded-corner rectangle marked ESTABLISHED has two sets of states, a set for the client and another for the server, that are used for flow and error control.

Consider the scenario. Figure 4.4.2 shows the state transition diagram for this scenario. The client process issues an active open command to its TCP to request a connection to a specific socket address.

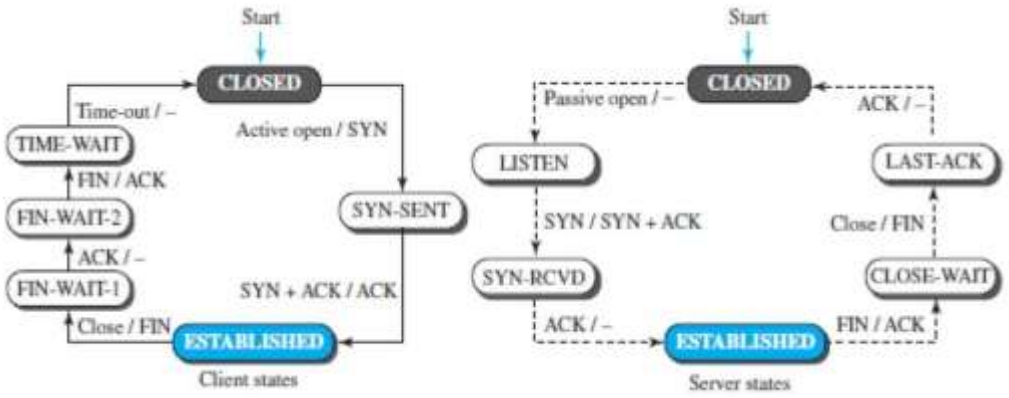

Fig4.4.2: State transition diagram.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-759]

TCP sends a SYN segment and moves to the SYN-SENT state. After receiving the SYN +ACK segment, TCP sends an ACK segment and goes to the ESTABLISHED state. Data are transferred, possibly in both directions, and acknowledged. When the client process has no more data to send, it issues a command called an active close. The TCP sends a FIN segment and goes to the FINWAIT-1 state. When it receives the ACK segment, it goes to the FIN-WAIT-2 state. When the client receives a FIN segment, it sends an ACK segment and goes to the TIME-WAIT state.The client remains in this state for 2 MSL .MSL is the maximum time a TCP segment is expected to live, or stay in the network.When the corresponding timer expires, the client goes to the CLOSED state.The server process issues a passive open command. The server TCP goes to the LISTEN state and remains there passively until it receives a SYN segment.

The TCP then sends a SYN +ACK segment and goes to the SYN-RCVD state, waiting for the client to send an ACK segment. After receiving the ACK segment, TCP goes to the ESTABLISHED state, where data transfer can take place. TCP remains in this state until it receives a FIN segment from the client signifying that there are no more data to be exchanged and the connection can be closed.The server, upon receiving the FIN segment, sends all queued data to the server with a virtual EOF marker, which means that the connection must be closed.

It sends an ACK segment and goes to the CLOSEWAIT state, but postpones acknowledging the FIN segment received from the client until it receives a passive close command from its process. After receiving the passive close command, the server sends a FIN segment to the client and goes to the LASTACK state, waiting for the final ACK. When the ACK segment is received from the client,the server goes to the CLOSE state.

4.3 TCP Connection

TCP is connection-oriented.

A connection-oriented transport protocol establishes a logical path between the source and destination. All of the segments belonging to a message are then sent over this logical path.

If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering.

In TCP, connection-oriented transmission requires three phases: connection establishment,data transfer, and connection termination.

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.

Three-Way Handshaking

The connection establishment in TCP is called three-way handshaking.

For example,an application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport-layer protocol.

The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This request is called a passive open.

The client program issues a request for an active open. A client that wishes to connect to an open server tells its TCP to connect to a particular server.

TCP function is enabled by three-way handshaking process, as shown in Figure 4.3.1 .

Three step process.

1.The client sends the first segment, a SYN segment, in which only the SYN flag is set.

This segment is for synchronization of sequence numbers.The client chooses a random number as the first sequence number and sends this number to the server. This sequence number is called the initial sequence number (ISN).

The SYN segment is a control segment and carries no data. It consumes one sequence number because it needs to be acknowledged. SYN segment carries one imaginary byte.

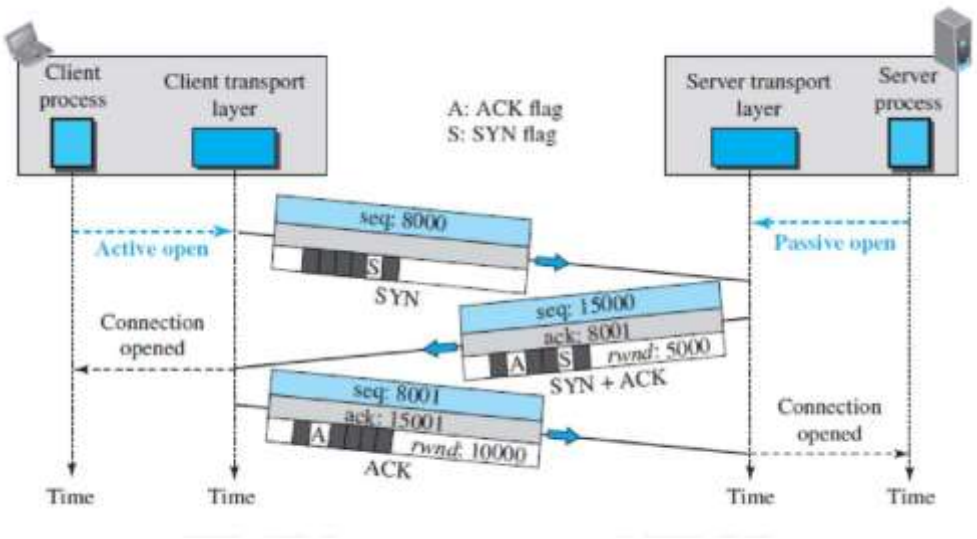

Fig4.3.1:TCP connection.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-751]

2. The server sends the second segment, a SYN + ACK segment with two flag bits set as: SYN and ACK. This segment has a dual purpose.

First, it is a SYN segment for communication in the other direction. The server uses this segment to initialize a sequence number for numbering the bytes sent from the server to the client.

The server also acknowledges the receipt of the SYN segment from the client by setting the ACK flag and displaying the next sequence number it expects to receive from the client.

3. The client sends the third ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field.

Note that the ACK segment does not consume any sequence numbers if it does not carry data.

Data Transfer

After connection is established, bidirectional data transfer take place. The client and server can send data and acknowledgments in both directions.The acknowledgment is piggybacked with the data. Figure 4.3.2 shows an example.

In this example, after a connection is established, the client sends 2,000 bytes ofdata in two segments. The server then sends 2,000 bytes in one segment.

The client sends one more segment. The first three segments carry both data and acknowledgment,but the last segment carries only an acknowledgment because there is no more data to be sent. Note the values of the sequence and acknowledgment numbers.

The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.

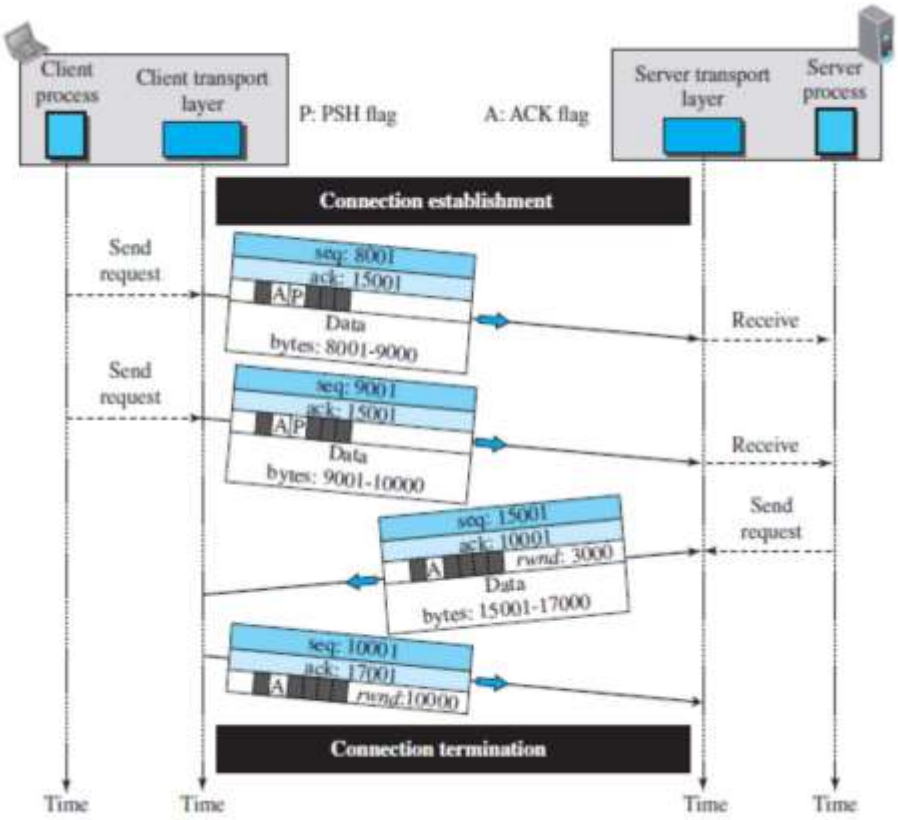

Fig4.3.2:Data Transfer.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-751]

TCP can handle push operation.

The application program at the sender can request a push operation. This means that the sending TCP must not wait for the window to be filled. It must create a segment and send it immediately.

The sending TCP set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come.

Urgent Data

TCP is a stream-oriented protocol. TCP supports an application program which needs to send urgent bytes, some bytes that need to be treated in a special way by the application at the other end.

The sending application program tells the sending TCP that the piece of data is urgent. The sending TCP creates a segment and inserts the urgent data at the beginning of the segment. The rest of the segment can contain normal data from the buffer.

The urgent pointer field defines the end of the urgent data (the last byte of urgent data).

Connection Termination: Three-Way Handshaking

It has three phases as shown in figure 4.3.3.

1. In this situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client or it can be just a control segment as shown in the figure. If it is only a control segment, it consumes only one sequence number because it needs to be acknowledged. The client TCP, after receiving a close command from the client process, sends the first segment called FIN segment.

The FIN segment has the last data sent by the client or it can be just a control segment.

2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN + ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number because it needs to be acknowledged.

3.The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is one plus the sequence number received in the FIN segment from the server. This segment cannot carry data and has no sequence numbers.

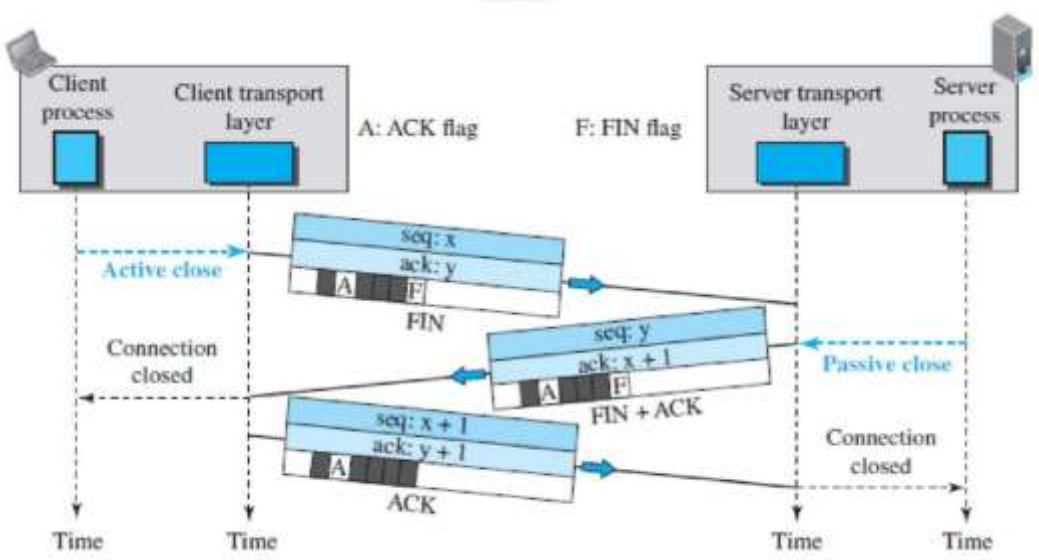

Fig4.3.3: Connection termination.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-713]

Half-Close

In TCP, one end can stop sending data while still receiving data. This is called a half- close. Either the server or the client can issue a half-close request. It can occur when the server needs all the data before processing can begin

Connection Reset

TCP at one end can oppose a connection request, or may terminate an idle connection. These are done with the RST (reset) flag.

4.2 TRANSMISSION CONTROL PROTOCOL

Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol. TCP defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.

TCP uses a combination of GBN and SR protocols to provide reliability. To achieve this ,TCP uses checksum(for error detection), retransmission of lost or corrupted packets, cumulative and selective acknowledgments, and timers.

TCP Services

Process-to-Process Communication

TCP provides process-to-process communication using port numbers.

Stream Delivery Service

TCP is a stream-oriented protocol.

TCP 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. The sending process produces (writes to) the stream and the receiving process consumes (reads from) it.

Segments

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 network layer for transmission. The segments are encapsulated in an IP datagram and transmitted.

Full-Duplex Communication

TCP offers full-duplex service, where data can flow in both directions at the same time.

Multiplexing and Demultiplexing

Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver.

Connection-Oriented Service

When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:

The two TCP's establish a logical connection between them.

1.Data are exchanged in both directions.

2.The connection is terminated.

Reliable Service

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

TCP Features

The sequence number and the acknowledgment number is used in TCP.

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.TCP chooses an arbitrary number between 0 and $2^{32} - 1$ for the number of the first byte. For example, if the number happens to be 1057 and the total data to be sent is 6000 bytes, the bytes are numbered from 1057 to 7056.

Sequence Number

After the bytes have been numbered, TCP allot a sequence number to each segment that is being sent.The sequence number of the first segment is the ISN (initial sequence number), which is a random number.The sequence number of any other segment is the sequence number of the previous segment plus the number of bytes (real or imaginary) carried by the previous segment.

Acknowledgment Number

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.The sequence number in each direction shows the number of the first byte carried by the segment.Each party uses an acknowledgment number to confirm the bytes it has received.

TCP Segment

A packet in TCP is called a segment. The format of a segment is shown in Figure 4.2.1.

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.

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.

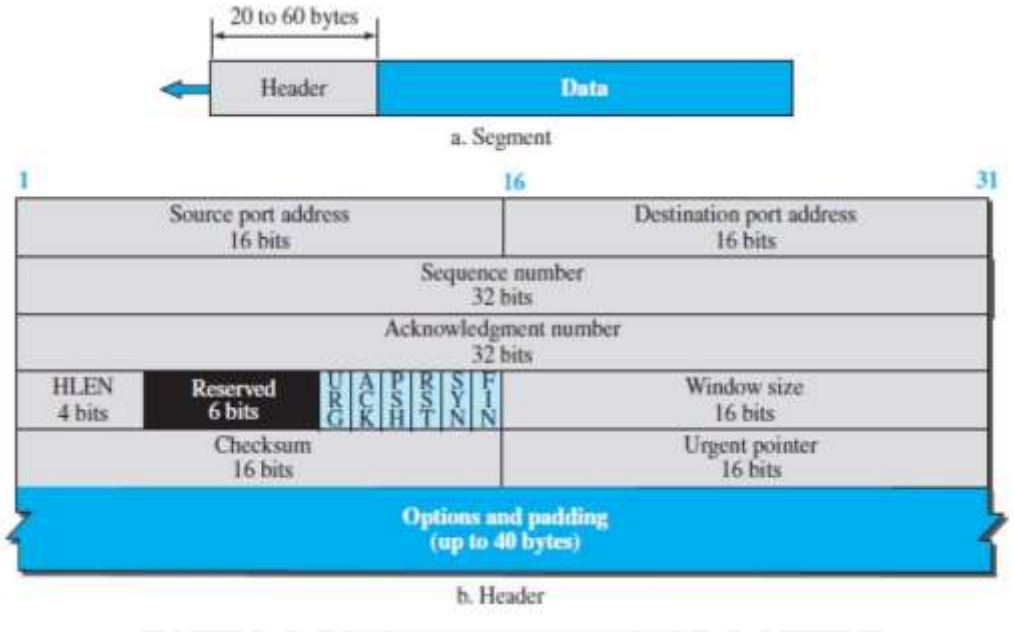

Fig4.2.1: TCP segment.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-748]

TCP is a stream transport protocol.

To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence is the first byte in the segment. During connection establishment each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.

Acknowledgment number. This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x from the other party, it returns x +1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.

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.

Control. This field defines 6 different control bits or flags.These bits enable flow control, connection establishment and termination, and the mode of data transfer in TCP.

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

This value is referred to as the receiving window(rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

Checksum. This 16-bit field contains the checksum (error detection).The calculation of the checksum for TCP is important. The same pseudo header, serving the same purpose, is added to the segment.

Urgent pointer. This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.

Options. There can be up to 40 bytes of optional information in the TCP header.

Encapsulation

A TCP segment encapsulates the data received from the application layer.

UNIT4 TRANSPORT LAYER

4.1 TRANSPORT LAYER PROTOCOLS

Duty of transport layer is process-to-process communication.

These protocols use port numbers to accomplish this. Port numbers provide end-to end addresses at the transport layer and allow multiplexing and demultiplexing at this layer.

Stop-and-Wait Protocol

This is a connection-oriented protocol called the Stop-and-Wait protocol, which uses both flow and error control. Both the sender and the receiver use a sliding window of size 1. The sender sends one packet at a time and waits for an acknowledgment before sending the next one. To detect corrupted packets, we need to add a checksum to each data packet. When a packet arrives at the receiver site, it is checked. If its checksum is incorrect, the packet is corrupted and silently discarded.

Every time the sender sends a packet, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next packet (if it has one to send). If the timer expires, the sender resends the previous packet, assuming that the packet was either lost or corrupted. This means that the sender needs to keep a copy of the packet until its acknowledgment arrives

Figure 4.1.1 shows the outline for the Stop-and-Wait protocol.

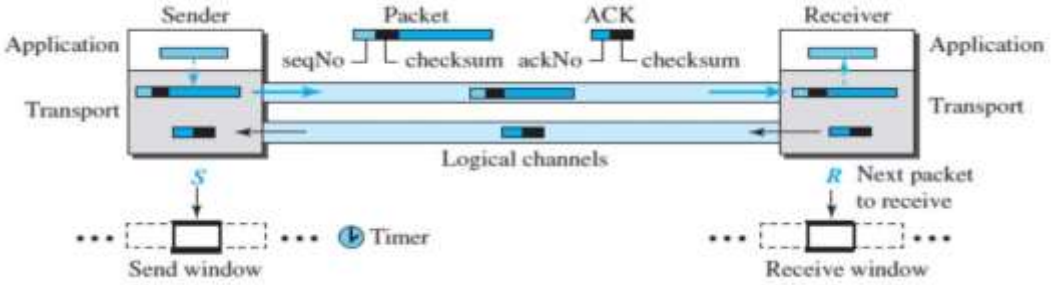

Fig4.1.1:Stop and Wait Protocol

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-709]

Sequence Numbers

To prevent duplicate packets, the protocol uses sequence numbers and acknowledgment numbers.

A field is added to the packet header to hold the sequence number of that packet.

Assume that the sender has sent the packet with sequence number x. Three things can happen.

1. The packet arrives safe and sound at the receiver site; the receiver sends an acknowledgment. The acknowledgment arrives at the sender site, causing the sender to send the next packet numbered x [?]1.

2. The packet is corrupted or never arrives at the receiver site; the sender resends the packet (numbered x) after the time-out. The receiver returns an acknowledgment.

3. The packet arrives safe and sound at the receiver site; the receiver sends an acknowledgment, but the acknowledgment is corrupted or lost. The sender resends the packet (numbered x) after the time-out. Note that the packet here is a duplicate. The receiver can recognize this fact because it expects packet x [?]1 but packet x was received.

We can see that there is a need for sequence numbers *x* and *x* [?]1 because the receiver needs to distinguish between case 1 and case 3. But there is no need for a packet to be numbered *x* [?]2. In case 1, the packet can be numbered *x* again because packets *x* and *x* [?]1 are acknowledged and there is no ambiguity at either site. In cases 2 and 3, the new packet is *x* [?]1, not *x* [?]2. If only *x* and *x* [?]1 are needed, we can let *x* [?]0 and *x* [?]1 [?]1. This means that the sequence is 0, 1, 0, 1, 0, and so on. This is referred to as modulo 2 arithmetic.

Acknowledgment Numbers

The acknowledgment numbers always announce the sequence number of the next packet expected by the receiver. For example, if packet 0 has arrived safe and sound, the receiver sends an ACK with acknowledgment 1 (meaning packet 1 is expected next). If packet 1 has arrived safe and sound, the receiver sends an ACK with acknowledgment 0 (meaning packet 0 is expected).

Go-Back-N Protocol (GBN)

To improve the efficiency of transmission (to fill the pipe), multiple packets must be in transition while the sender is waiting for acknowledgment. In other words, we need tolet more than one packet be outstanding to keep the channel busy while the sender is waiting for acknowledgment. In this section, we discuss one protocol that can achieve this goal; in the next section, we discuss a second. The first is called Go-Back-N (GBN) (the rationale for the name will become clear later).

The key to Go-back-N is that we can send several packets before receiving acknowledgments, but the receiver can only buffer one packet. We keep a copy of the sent packets until the acknowledgments arrive. Figure 4.1.2 shows the outline of the protocol. Note that several data packets and acknowledgments can be in the channel at the same time.

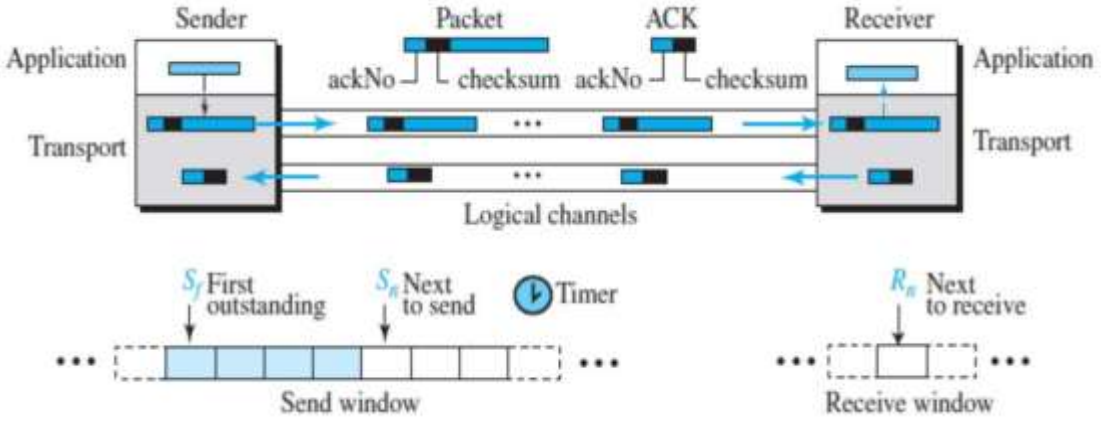

Fig4.1.2:Go-Back- N Protocol

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-713]

Sequence Numbers

The sequence numbers are modulo 2m, where m is the size of the sequence number field in bits.

Acknowledgment Numbers

An acknowledgment number in this protocol is cumulative and defines the sequence number of the next packet expected. For example, if the acknowledgment number (ackNo) is 7, it means all packets with sequence number up to 6 have arrived, safe and sound, and the receiver is expecting the packet with sequence number 7.

USER DATAGRAM PROTOCOL

The User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol as shown in figure 4.1.3. UDP is a very simple protocol using a minimum of overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP.

User Datagram

UDP packets are called user datagrams. The first two fields define the source and destination port numbers. The third field 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.

Connectionless Services

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.

The user datagrams travel on a different path on the way to destination.

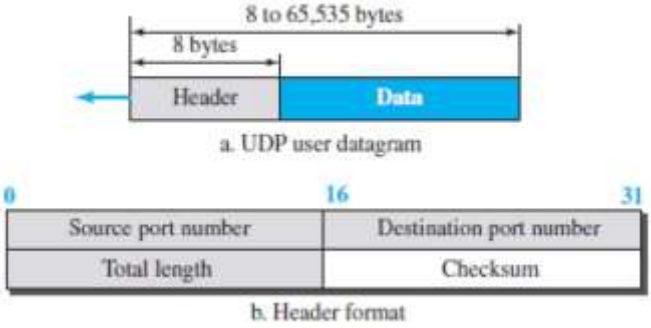

Fig4.1.3: UDP format.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-738]

Flow Control

UDP is a very simple protocol. There is no flow control.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 has three sections: a pseudo header as shown in figure 4.1.4, the UDP header, and the data coming from the application layer.The pseudo header is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s.If the checksum does not include the pseudo header, a user datagram may arrive safe and sound. If the IP header is corrupted, it may be delivered to the wrong host.

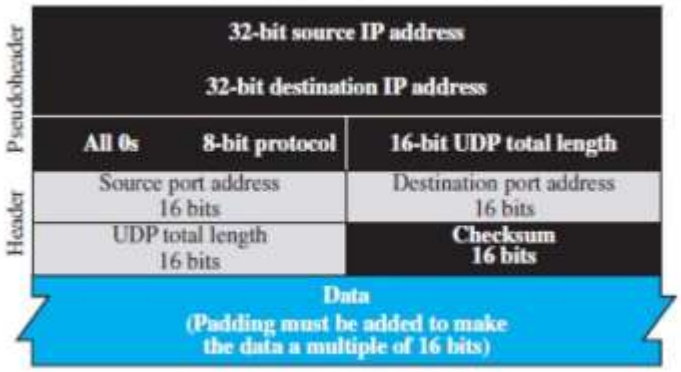

Fig4.1.4:Pseudo header for checksum.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-739]

Congestion Control

Since UDP is a connectionless protocol, it does not provide congestion control.

Encapsulation and Decapsulation

To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages.

Multiplexing and Demultiplexing

In a TCP/IP protocol suite, there is only one UDP but several processes that may want to use the services of UDP. To handle this situation, UDP multiplexes and demultiplexes.

UDP features

Connectionless Service

UDP is a connectionless protocol. Each UDP packet is independent from other packets sent by the same application program. It is an advantage.

For example, a client application needs to send a short request to a server and to receive a short response. If the request and response can each fit in a single user datagram, a connectionless service may be preferable.In the connection oriented service, to send and receive short message , at least 9 packets are exchanged between the client and the server; in connectionless service only 2 packets are exchanged.

UDP Applications

UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.UDP is suitable for a process with internal flow- and error-control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.UDP is a suitable transport protocol for multicasting.

Multicasting capability is embedded in the UDP software but not in the TCP software. UDP is used for management processes such as SNMP. UDP is used for some route updating protocols such as Routing Information Protocol(RIP).
