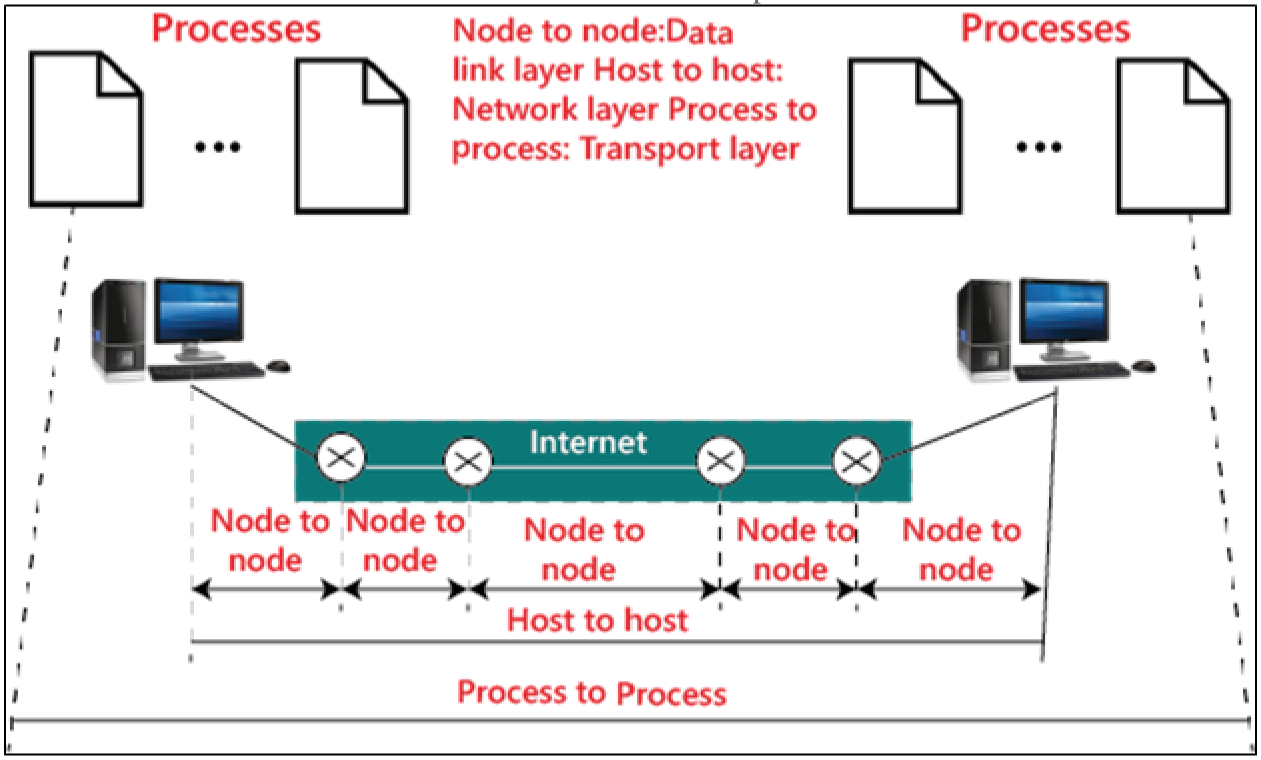
Process-to-Process Delivery: A transport-layer protocol's first task is to perform process-to-process delivery. A process is an entity of the application layer which uses the services of the transport layer. Two processes can be communicated between the client/server relationships.



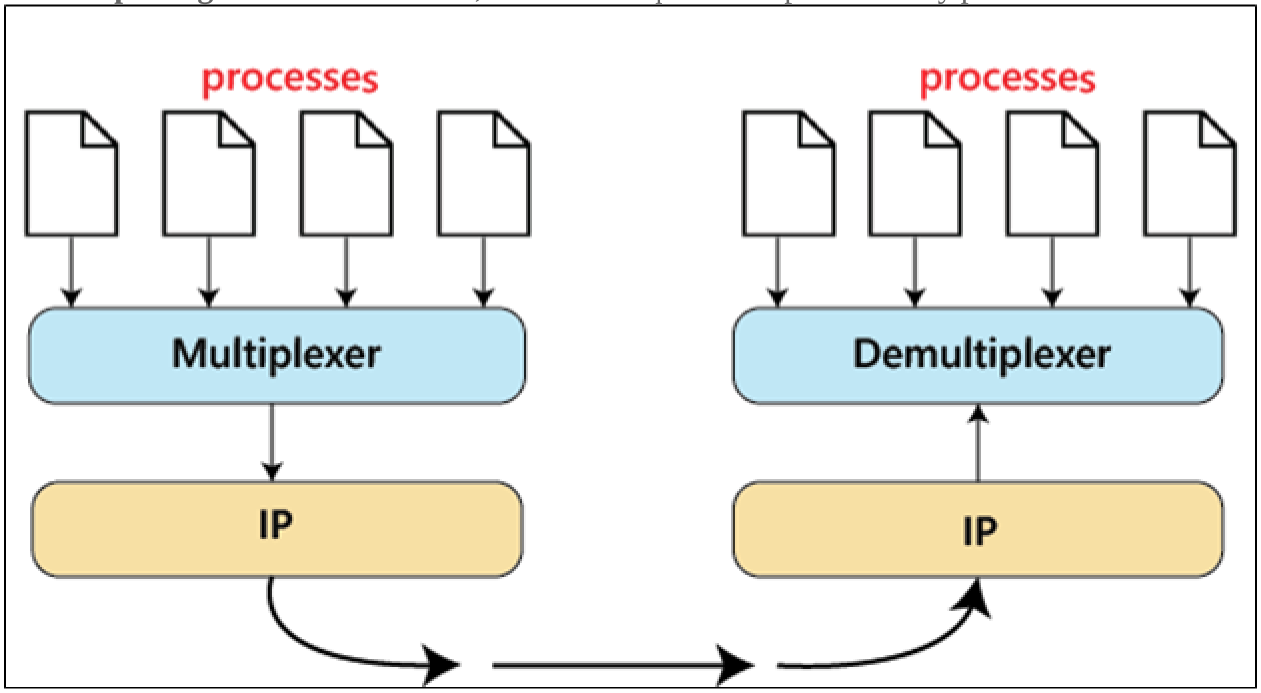
Client/Server Paradigm

There are many ways to obtain the process-to-process communication, and the most common way is through the client/server paradigm. A process is called a client on the local-host. Usually, the remote host is needed services on the processes, that is called server. The same name applies to both processes (client and server). IP address and port number combination are called socket address, and that address defines a process and a host.

Multiplexing and Demultiplexing

Multiplexing: At the sender site, multiple processes can occur, and those processes are required to send packets. It is a technique that combines multiple processes into one process.

Demultiplexing: At the receiver site, it is a technique that separates many processes.



UDP (User Datagram Protocol)

UDP was developed by David P Reed in 1980. It is a connection-less and unreliable protocol. This means when the data transfer occurs; this protocol does not establish the connection between the sender and receiver. The receiver does not send any acknowledgment of the receiving data. It sends the data directly. In the UDP, the data packet is called datagram. UDP does not guarantee your data that will reach its destination or not. It does not require that the data reach the receiver in the same order in which the sender has sent the data.

Transmission Control Protocol

TCP stands for Transmission Control Protocol. It was introduced in 1974. It is a connection-oriented and reliable protocol. It establishes a connection between the source and destination device before starting the communication. It detects whether the destination device has received the data sent from the source device or not. If the data received is not in the proper format, it sends the data again. TCP is highly reliable because it uses a handshake and traffic control mechanism. In TCP protocol, data is received in the same sequencer in which the sender sends the data. We use the TCP protocol services in our daily life, such as HTTP, HTTPS, Telnet, FTP, SMTP, etc.

Difference between UDP and TCP

UDP	TCP
UDP stands for User Datagram Protocol.	TCP stands for Transmission Control Protocol.
UDP sends the data directly to the destination device.	TCP establishes a connection between the devices before transmitting the data.
It is a connection-less protocol.	It is a connection-oriented protocol.
UDP is faster than the TCP protocol.	TCP is slower than the UDP protocol.
It is an unreliable protocol.	It is a reliable protocol.
It has not a sequence number of each byte.	It has a sequence number of each byte.

User Datagram Protocol: UDP stands for the User Datagram Protocol. UDP was developed by David P Reed in 1980. It is a connection-less and unreliable protocol. In this protocol, when the data transfer occurs, it does not establish the connection between the sender and receiver. It sends the data directly. The receiver does not send any acknowledgment of the receiving data in this protocol. In the UDP, the data packet is called datagram.

UDP does not guarantee any user data, whether it will reach its destination or not. In this protocol, it is not necessary that the data reach the receiver in the same sequencer in which the sender has sent the data.

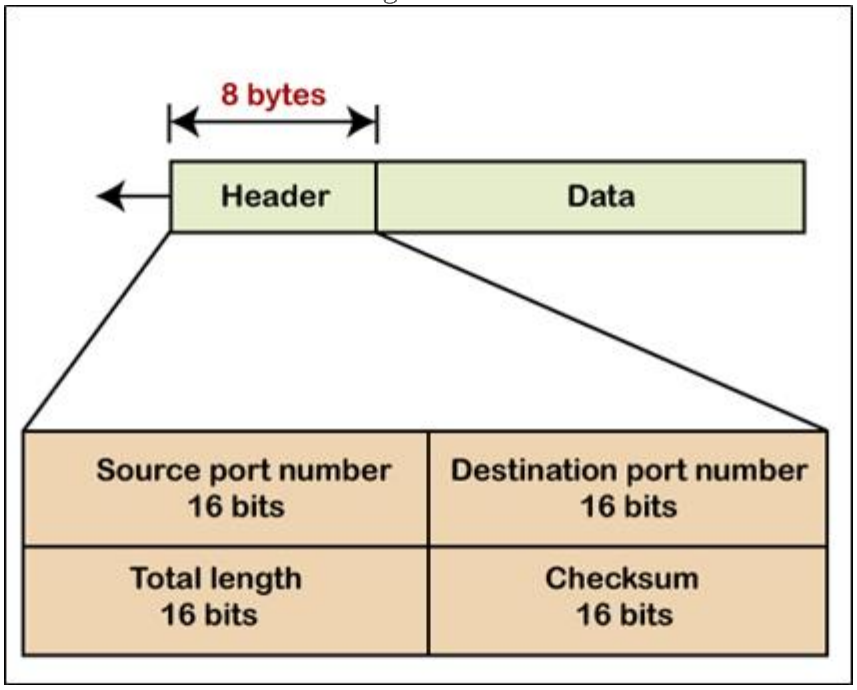
Importance of UDP

- The UDP protocol is used to transfer data where you need a higher speed than accuracy and reliability.
- If the data flow is in the same direction, UDP is used.
- It is also used for streaming applications, for example, YouTube and online gaming.

- It provides faster data transfer speed than the TCP protocol.

User Datagram Protocol Format

The UDP format is very simple. The header size of the UDP is 8 bytes (8 bytes mean 64 bits). The format of UDP is shown below in the figure.



It has four fields shown below:

- **Source Port Number:** The size of the source port is 16 bits. It is used to identify the process of the sender.
- **Destination Port Number:** The size of the destination port is 16 bits. It is used to identify the process of the receiver.
- **Total length:** The size of the total length is 16 bits. It defines the total length of the UDP and also stores the length of the data and header.

  UDP length = IP length – IP header's length
- **Checksum:** The size of the checksum port is 16 bits. It is used to detect errors across the entire user datagram.

Advantages of UDP

1. You can easily broadcast and multicast transmission through the UDP.
2. It is faster than TCP.
3. It uses the small size of the header (8 bytes).
4. It takes less memory than other protocols.
5. Whenever data packets need to be transmitted, then UDP is used.

Disadvantages of UDP

1. It is an unreliable protocol for transmission.
2. There is no such function in it to know that the data has been received or not.
3. The handshake method is not used in UDP.
4. It does not control the congestion.
5. The main disadvantage of using routers with UDP is that once transmission failure occurs, the routers do not transmit the datagram again.

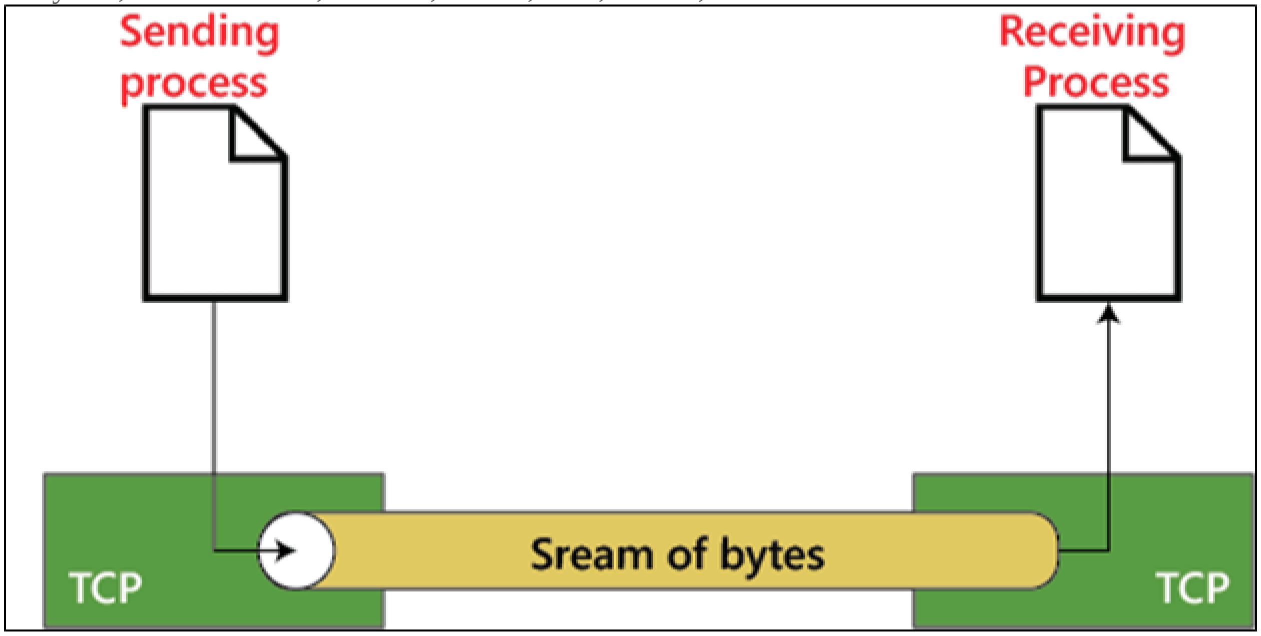
UDP is used in the following applications.

1. Domain name system.
2. Simple network management protocol.
3. Routing information protocol.
4. Trivial file transfer protocol.

Services of UDP and TCP

Services	**UDP**	**TCP**
Sequence data delivery	No	Yes
Multi-Streaming	No	No
Multi-Homing	No	No
Connection-Oriented	No	Yes
Connection-less	Yes	No
Allows half-closed connection	N/A	Yes
Application PDU bundling	No	Yes
Congestion Control	No	Yes
Application PDU fragmentation	No	Yes
Preserve message boundaries	Yes	No
Partial reliable data transfer	No	No
Selective Acknowledgements	No	Optional

Transmission Control Protocol: TCP stands for Transmission Control Protocol. It was introduced in 1974. It is a connection-oriented and reliable protocol. It establishes a connection between the source and destination device before starting the communication. It detects whether the destination device has received the data sent from the source device or not. If the received data is not in the proper format, it sends the data again. TCP is highly reliable because it uses a handshake and traffic control mechanism. In the TCP protocol, the receiver receives the data in the same sequence in which the sender sends it. We use the TCP protocol services in our daily life, such as HTTP, HTTPS, Telnet, FTP, SMTP, etc.

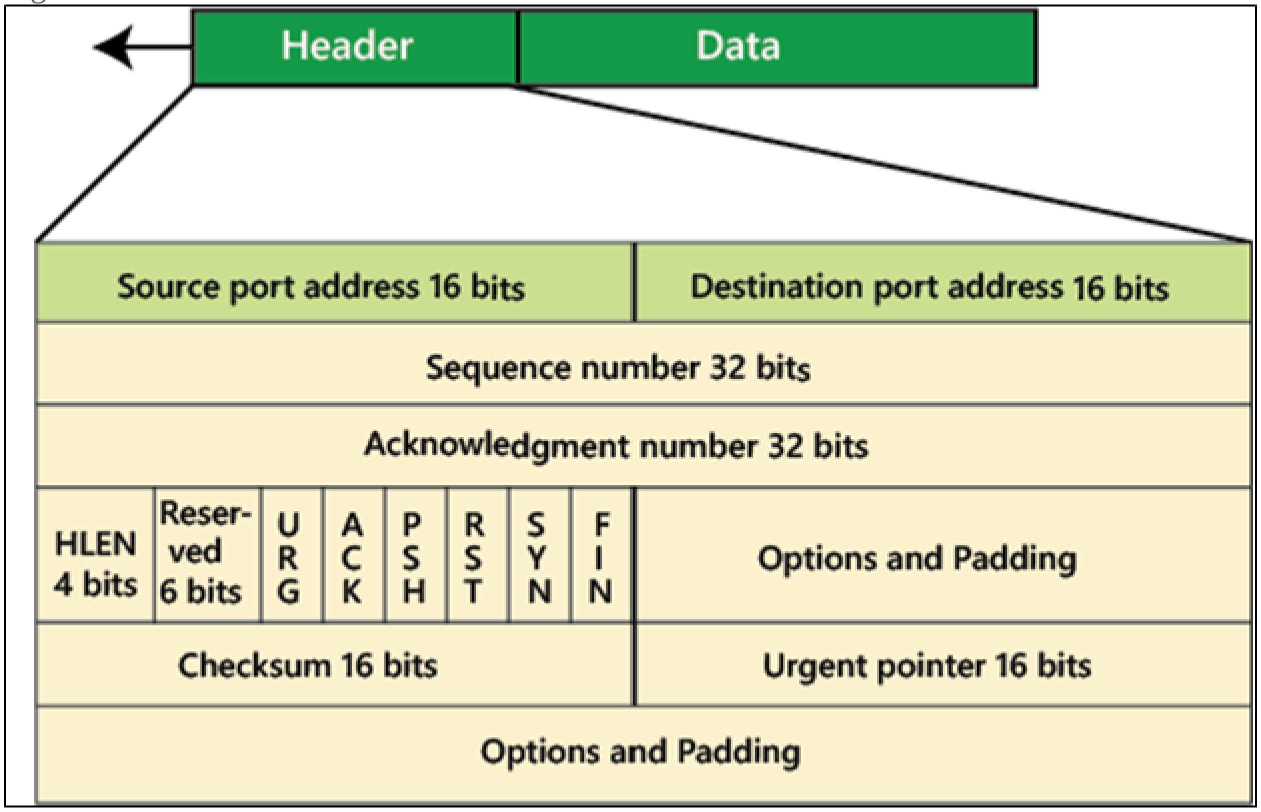


Importance of UDP

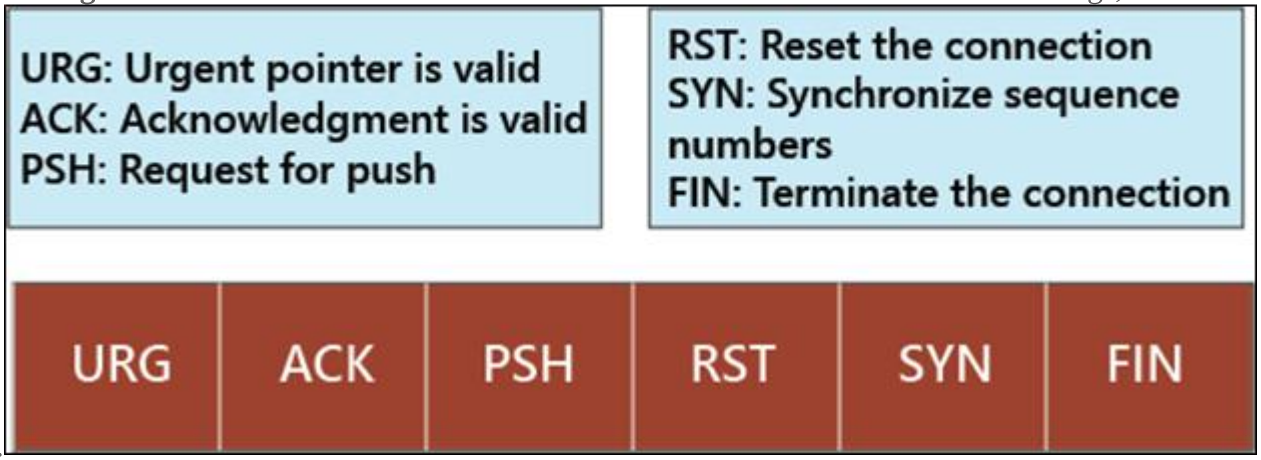
- The TCP protocol is used to transfer data where, you need accuracy and reliability rather than the speed.
- Both TCP and UDP can check for errors, but only TCP can fix the error because it can control the traffic and flow of data.

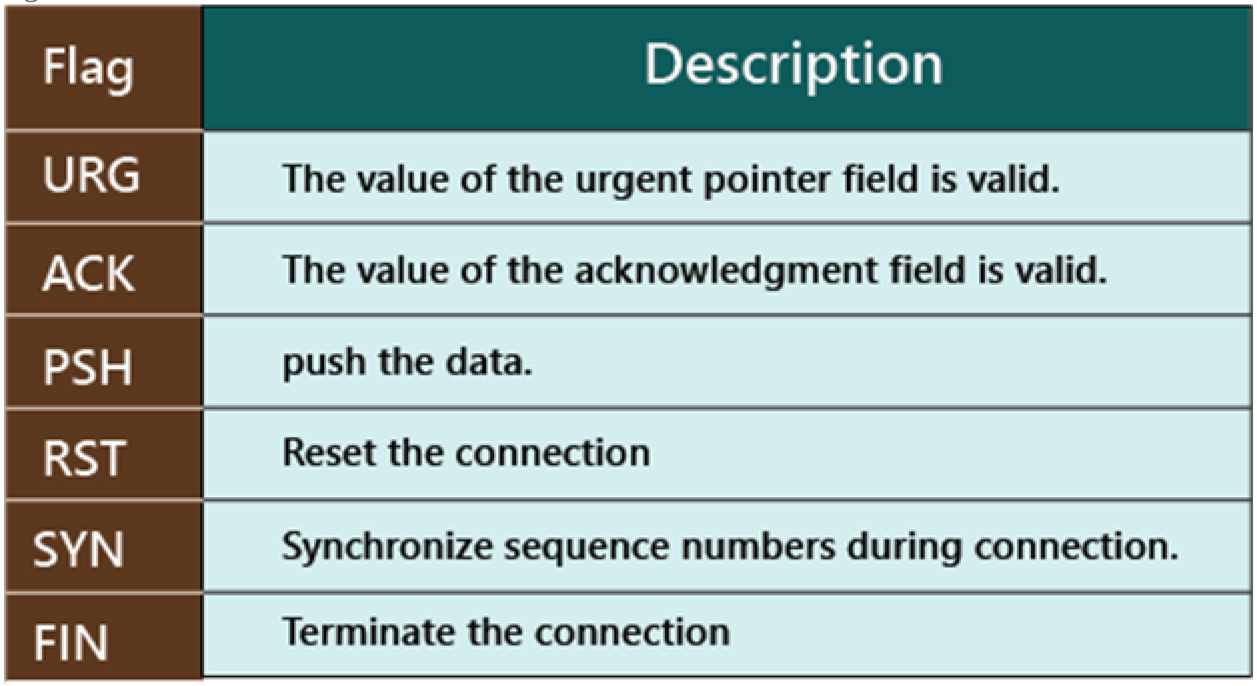
TCP Segment Format

In the TCP, the data packet is called a segment. The size of the header in the segment is 20 to 60 bytes. The segment format of TCP is shown below the figure.



- **Source Port Address:** The size of the source port is 16 bits. It is used to define the port address of the application that sends the segment.
- **Destination Port Address:** The size of the destination port is 16 bits. The destination port is used to define the port address of the application that receives the segment.
- **Sequence Number:** The size of the sequence number is 32 bits. It defines the unique number of the data in the segment.
- **Acknowledgment number:** The size of the acknowledgment number is 32 bits.
- **Header Length:** The size of the header length is 4 bits. It indicates the header of the application. The header length can be lies between 20 and 60 bytes. Therefore, the value of this field is 5 (5 × 4 = 20) and 15 (15 × 4 = 60).
- **Reserved:** The size of this field is 6 bits. This field is for future uses.

- **Control Flag:** The size of this field is 6 bits. It defines the six different control bits or flags, as shown in the figure.
URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection
URG
ACK
PSH
RST
SYN
FIN



Flag	Description
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	push the data.
RST	Reset the connection
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection

- **Window Size:** The size of the window field is 16 bits. It defines the size of the sending window of the sender.
- **Checksum:** The size of the checksum field is 16 bits. The checksum field is used for error control. It is mandatory in TCP.
- **Urgent Pointer:** The size of the urgent pointer field is 16 bits, which is only required when the URG flag is set. It is used for urgent data in the segment.
- **Options and Padding:** The size of options and padding field vary from 0 to 40 bytes.

Advantages of TCP

1. **Retransmission:** In the TCP, when a data fail, this protocol sends that data again after a specific time.
2. TCP can fix the error because it can control the flow-control and congestion control.
3. TCP can easily detect the errors.
4. In the TCP protocol, the receiver receives the data in the same sequencer in which the sender sends it.

Disadvantages of TCP

1. The data transfer speed of the TCP protocol is less than the UDP protocol.
2. TCP protocol cannot broadcast and multicast the message.

TCP vs UDP:

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

4.1.1 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 data across the Internet. This imaginary environment is depicted in Figure 4.13. The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.

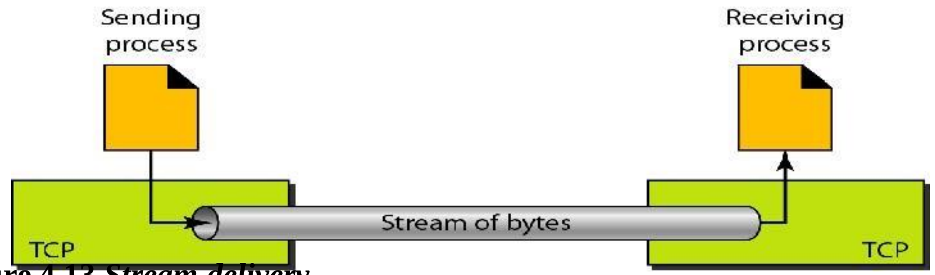


Figure 4.13 ***Stream delivery***

Sending and Receiving Buffers:

Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receivingbuffer, one for each direction. One way to implement a buffer is to use a circular array of I-byte locations as shown in Figure 4.14. For simplicity, we have shown two buffers of 20 bytes each; normally the buffers are hundreds or thousands of bytes, depending on the implementation. We also show the buffers as the same size, which is not always the case.

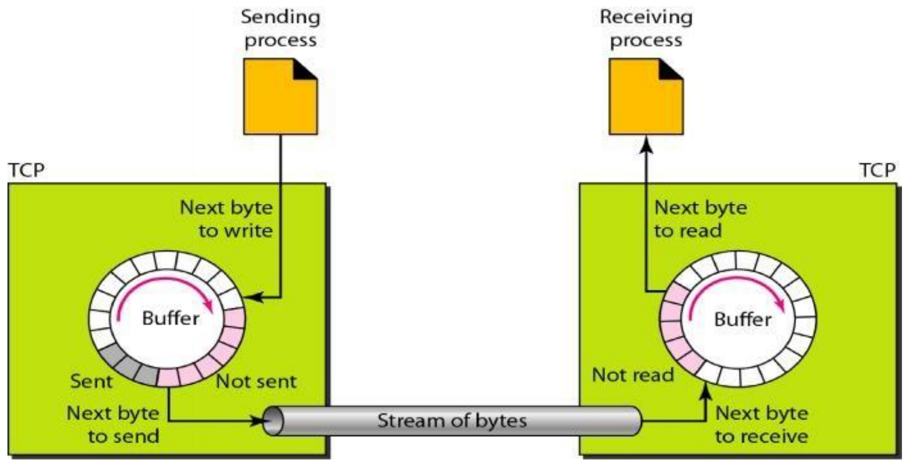


Figure 4.14 Sending and Receiving Buffers

Figure 4.14 shows the movement of the data in one direction. At the sending site, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process (producer). The gray area holds bytes that have been sent but not yet acknowledged. TCP keeps these bytes in the buffer until it receives an acknowledgment. The colored area contains bytes to be sent by the sending TCP.

Segments

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. Figure 4.15 shows how segments are created from the bytes in the buffers.

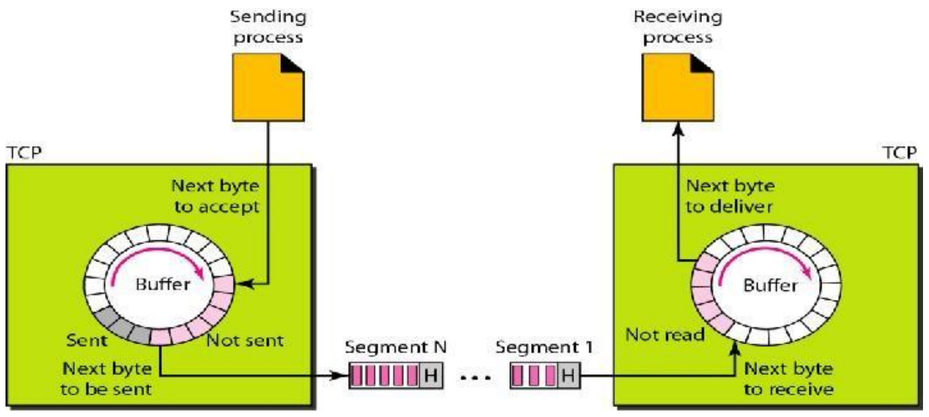


Figure 4.15 TCP Segments

4.1.2 Full-Duplex Communication

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

4.1.3 Connection-Oriented Service

TCP is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:

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

Reliable Service

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

4.1.4 TCP Features

TCP has several 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 the byte number and not the segment number.

Byte Number

TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them.

The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.

Sequence Number

After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.

The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.

When a segment carries a combination of data and control information (piggybacking), it uses a sequence number. If a segment does not carry user data, it does not logically define a sequence number. The field is there, but the value is not valid. However, some segments, when carrying only control information, need a sequence number to allow an acknowledgment from the receiver. These segments are used for connection establishment, termination, or abortion.

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 also uses an acknowledgment number to confirm the bytes it has received. However, the acknowledgment number defines the number of the next byte that the party expects to receive. In addition, the acknowledgment number is cumulative, which means that the party takes the number of the last byte that it has received, safe and sound, adds I to it, and announces this sum as the acknowledgment number.

4.1.5 Flow Control

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

Error Control

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

Congestion Control

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

4.1.6 Segment

A packet in TCP is called a segment.

4.1.7 Format

The format of a segment is shown in Figure 4.16

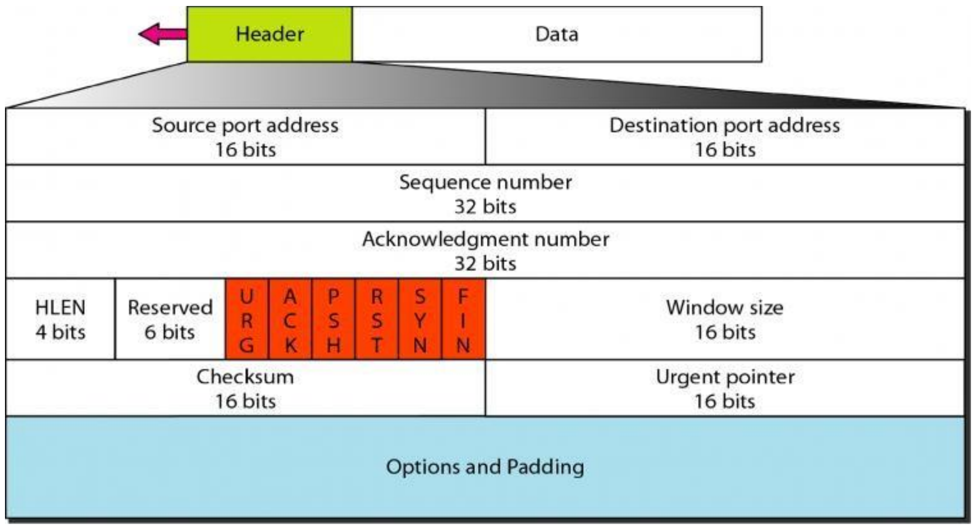


Figure 4.16 TCP Segment format

The segment consists of a 20- to 60-byte header, 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. This serves the same purpose as the source port address in the UDP header.
- **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. This serves the same purpose as the destination port address in the UDP header.
- **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. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment.
- **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. Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).
- **Reserved**. This is a 6-bit field reserved for future use.
- **Control**. This field defines 6 different control bits or flags as shown in Figure 4.17. One or more of these bits can be set at a time.

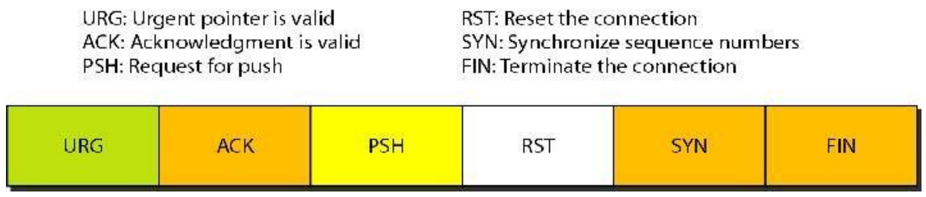

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection
URG
ACK
PSH
RST
SYN
FIN

Figure 4.17 Control Field

These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP. A brief description of each bit is shown in Table 4.3

Table 4.3 Description of flags in the control field

Flag	Description
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledge field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection
FIN	Terminate the connection.

- **Window size**. This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally 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. The inclusion of the checksum for TCP is mandatory. For the TCP pseudoheader, the value for the protocol field is 6.
- **Urgent pointer**. This l6-bit field, which is valid, only if the urgent flag is set, is used when the segment contains urgent data. It defines the number 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.

4.1.8 A TCP Connection

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtualpath between the source and destination. All 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.

a. 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. In our 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 is called a request for a passive open. Although the server TCP is readyto accept any connection from any machine in the world, it cannot make the connection itself.

The client program issues a request for an active open. A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process as shown in Figure 4.18.

To show the process, we use two time lines: one at each site. Each segment has values for all its header fields and perhaps for some of its option fields, too. However, we show only the few fields necessary to understand each phase. We show the sequence number, the acknowledgment number, the control flags (only those that are set), and the window size, if not empty. The three steps in this phase are as follows.

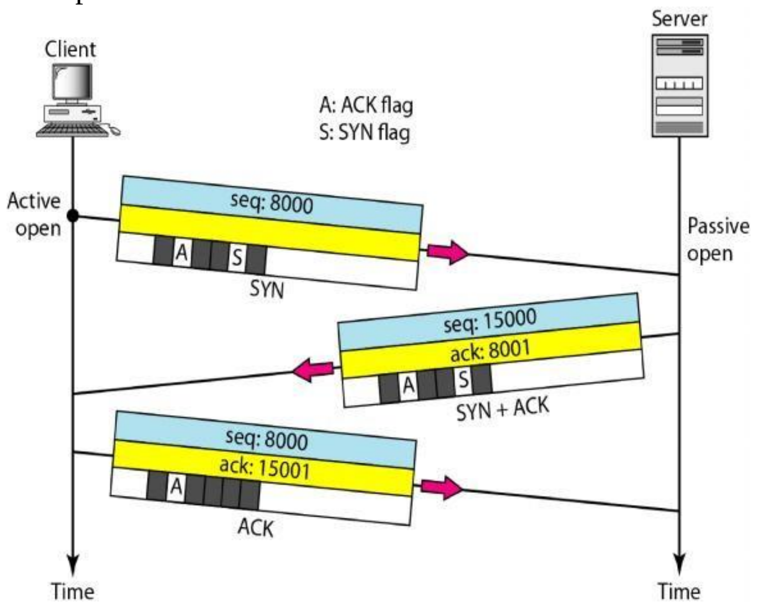


Figure 4.18 Connection establishment using three-way handshaking

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. It consumes one sequence number. When the data transfer start, the sequence number is incremented by 1. We can say that the SYN segment carries no real data, but we can think of it as containing 1 imaginary byte.

A SYN segment cannot carry data, but it consumes one sequence number.

2. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

A SYN +ACK segment cannot carry data, but does consume one sequence number.

3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

An ACK segment, if carrying no data, consumes no sequence number.

Simultaneous Open

A rare situation, called a simultaneous open, may occur when both processes issue an active open. In this case, both TCPs transmit a SYN + ACK segment to each other, and one single connection is established between them.

SYN Flooding Attack

The connection establishment procedure in TCP is susceptible to a serious security problem called the SYN flooding attack. This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is corning from a different client by faking the source IP addresses in the datagrams.

b. Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments. The acknowledgment is piggybacked with the data. Figure 4.19 shows an example.

In this example, after connection is established (not shown in the figure), the client sends 2000 bytes of data in two segments. The server then sends 2000 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 are 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. The segment from the server, on the other hand, does not set the push flag. Most TCP implementations have the option to set or not set this flag.

Pushing Data

The sending TCP uses a buffer to store the stream of data coming from the sending application program. The sending TCP can select the segment size. The receiving TCP also buffers the data when they arrive and delivers them to the application program when theapplication program is ready or when it is convenient for the receiving TCP. This type of flexibility increases the efficiency of TCP. However, on occasion the application program has noneed for this flexibility.

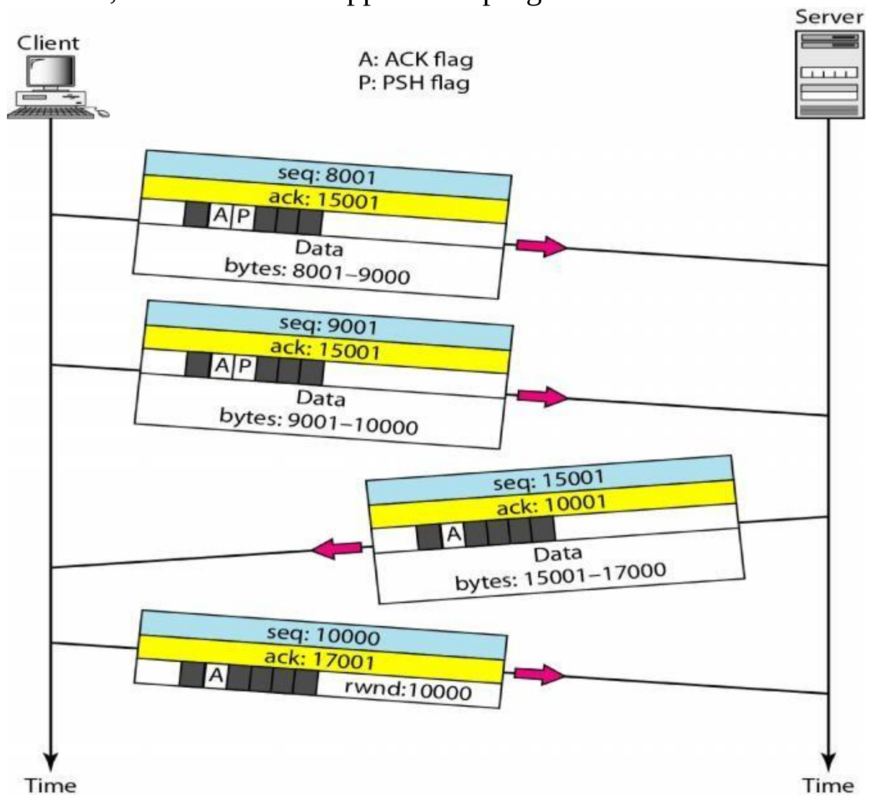


Figure 4.19 Data transfer

TCP can handle such a situation. The application program at the sending site 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 must also 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.

Although the push operation can be requested by the application program, most current implementations ignore such requests. TCP can choose whether or not to use this feature.

Urgent Data

TCP is a stream-oriented protocol. This means that the data are presented from the application program to TCP as a stream of bytes. Each byte of data has a position in the stream. However, on occasion an application program needs to send urgent bytes. This means that the sending application program wants a piece of data to be read out of order by the receiving application program. As an example, suppose that the sending application program is sending data to be processed by the receiving application program. When the result of processing comes back, the sending application program finds that everything is wrong. It wants to abort the process, but it has already sent a huge amount of data. If it issues an abort command, these two characters will be stored at the end of the receiving TCP buffer. It will be delivered to the receiving application program after all the data have been processed.

c. Connection Termination

Any of the two parties involved in exchanging data (client or server) can close theconnection, although it is usually initiated by the client. Most implementations today allow two Options for connection termination: three-way handshaking and four-way handshaking with ahalf-close option.

Three-Way Handshaking

Most implementations today allow three-way handshaking for connection termination as shown in Figure 4.20.

1. In a normal 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 Figure 4.20. If it is only a control segment, it consumes only one sequence number.

The FIN segment consumes one sequence number if it does not carry data.

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.

The FIN +ACK segment consumes one sequence number if it does not carry data.

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 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

Half-Close

In TCP, one end can stop sending data while still receiving data. This is called a half- close. Although either end can issue a half-close, it is normally initiated by the client. It can occur when the server needs all the data before processing can begin. A good example is sorting. When the client sends data to the server to be sorted, the server needs to receive all the data before sorting can start. This means the client, after sending all the data, can close the connectionin the outbound direction. However, the inbound direction must remain open to receive the sorted data. The server, after receiving the data, still needs time for sorting; its outbound direction must remain open.

Figure 4.21 shows an example of a half-close. The client half-closes the connection by sending a FIN segment. The server accepts the half-close by sending the ACK segment. The data transfer from the client to the server stops. The server, however, can still send data. When the server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.

After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server. The client cannot send any more data to the server. Note the sequence numbers we have used. The second segment (ACK) consumes no sequence number. Although the client has received sequence number y - 1 and is expecting y, the server sequence number is still y - 1. When the connection finally closes, the sequence number of the last ACK segment is still x, because no sequence numbers are consumed during data transfer in that direction.

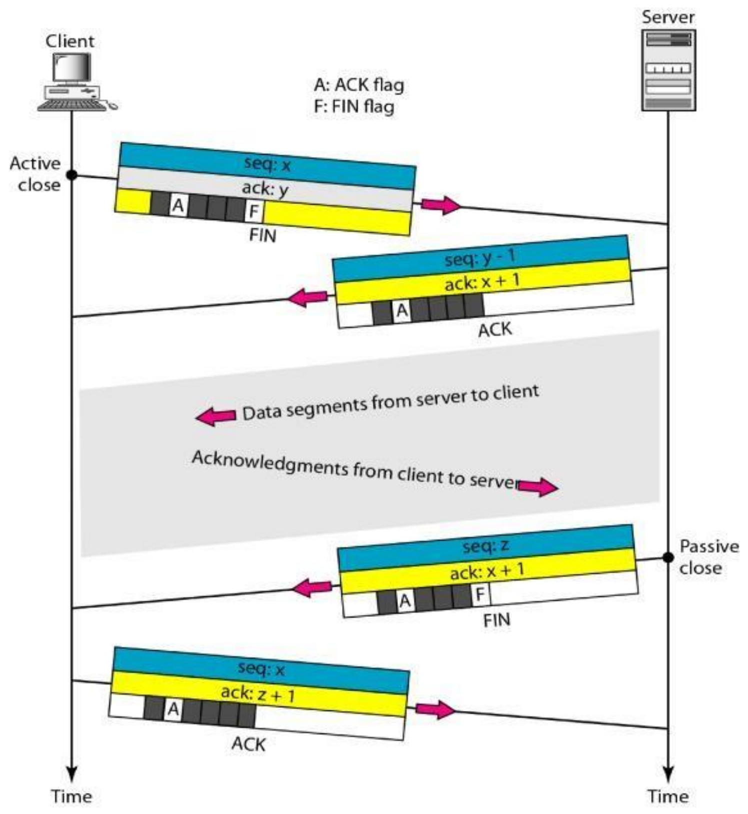


Figure 4.21 Half close

4.1.9 Flow Control

TCP uses a sliding window to handle flow control. The sliding window protocol used by TCP, however, is something between the Go-Back-N and Selective Repeat sliding window. The sliding window protocol in TCP looks like the Go-Back-N protocol because it does not use NAKs; it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.

The window is opened, closed, or shrunk. These three activities, as we will see, are in the control of the receiver (and depend on congestion in the network), not the sender. The sender must obey the commands of the receiver in this matter.

Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending. Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore. Shrinking the window means moving the right wall to the left. This is strongly discouraged and not allowed in some implementations because it means revoking the eligibility of some bytes for sending. This is a problem if the sender has already sent these bytes. Note that the left wall cannot move to the left because this would revoke some of the previously sent acknowledgments.

A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data. TCP sliding windows are byte-oriented.

The size of the window at one end is determined by the lesser of two values: receiver window (rwnd) or congestion window (cwnd). The receiver window is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded. The congestion window is a value determined by the network to avoid congestion.

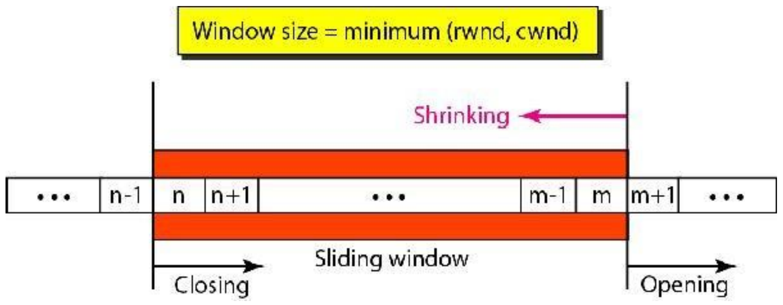


Figure 4.22 Sliding Window Some

points about TCP sliding windows:

- The size of the window is the lesser of *rwnd* and *cwnd.*
- The source does not have to send a full window's worth of data.
- The window can be opened or closed by the receiver, but should not be shrunk.
- The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.

4.1.10 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 corrupted segments, lost segments, out-of-order segments, and duplicated segments. Error control also includes a mechanism for correcting errors after they are detected. Error detection and correction in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.

a. Checksum

Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment. The 16-bit checksum is considered inadequate for the new transport layer, SCTP. However, it cannot be changed for TCP because this would involve reconfiguration of the entire header format.

b. Acknowledgment

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.

ACK segments do not consume sequence numbers and are not acknowledged.

c. Retransmission

The heart of the error control mechanism is the retransmission of segments. When a segment is corrupted, lost, or delayed, it is retransmitted. In modern implementations, a segment is retransmitted on two occasions: when a retransmission timer expires or when the sender receives three duplicate ACKs.

In modern implementations, a retransmission occurs if the retransmission timer expiresor three duplicate ACK segments have arrived.

Note that no retransmission occurs for segments that do not consume sequence numbers. In particular, there is no transmission for an ACK segment.

No retransmission timer is set for an ACK segment.

i) Retransmission After RTO

A recent implementation of TCP maintains one retransmission time-out (RTO) timer for all outstanding (sent, but not acknowledged) segments. When the timer matures, the earliest outstanding segment is retransmitted even though lack of a received ACK can be due to a delayed segment, a delayed ACK, or a lost acknowledgment. Note that no time-out timer is set for a segment that carries only an acknowledgment, which means that no such segment is resent.

ii) Retransmission After Three Duplicate ACK Segments

The previous rule about retransmission of a segment is sufficient if the value of RTO is not very large. Sometimes, however, one segment is lost and the receiver receives so many out- of-order segments that they cannot be saved (limited buffer size).

iii) Out-of-Order Segments

When a segment is delayed, lost, or discarded, the segments following that segment arrive out of order. Originally, TCP was designed to discard all out-of-order segments, resulting in the retransmission of the missing segment and the following segments. Most implementations today do not discard the out-of-order segments. They store them temporarily and flag them as out-of-order segments until the missing segment arrives.

Some Scenarios

In these scenarios, we show a segment by a rectangle. If the segment carries data, we show the range of byte numbers and the value of the acknowledgment field. If it carries only an acknowledgment, we show only the acknowledgment number in a smaller box.

a. Normal Operation

The first scenario shows bidirectional data transfer between two systems, as in Figure 4.23. The client TCP sends one segment; the server TCP sends three. The figure shows which rule applies to each acknowledgment. There are data to be sent, so the segment displays the next byte expected. When the client receives the first segment from the server, it does not have any more data to send; it sends only an ACK segment.

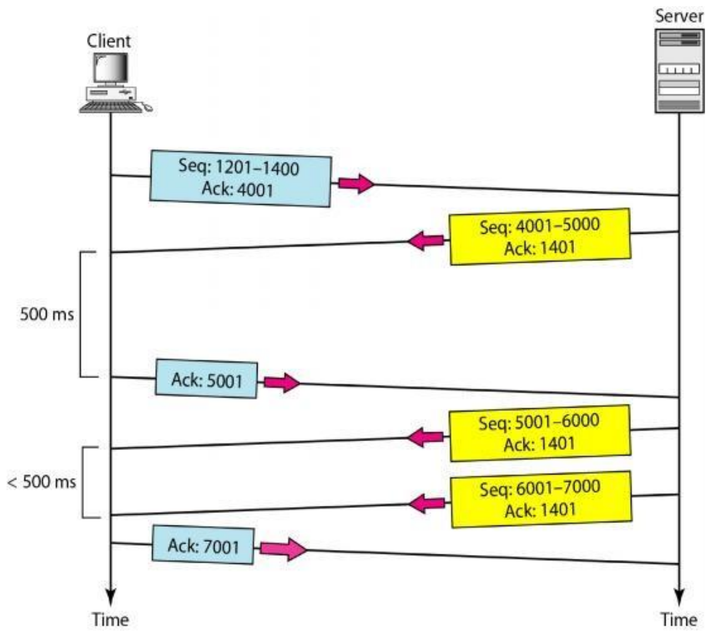


Figure 4.23 Normal operation

b. Lost Segment

In this scenario, we show what happens when a segment is lost or corrupted. A lost segment and a corrupted segment are treated the same way by the receiver. A lost segment is discarded somewhere in the network; a corrupted segment is discarded by the receiver itself. Both are considered lost. Figure 4.24 shows a situation in which a segment is lost and discarded by some router in the network, perhaps due to congestion.

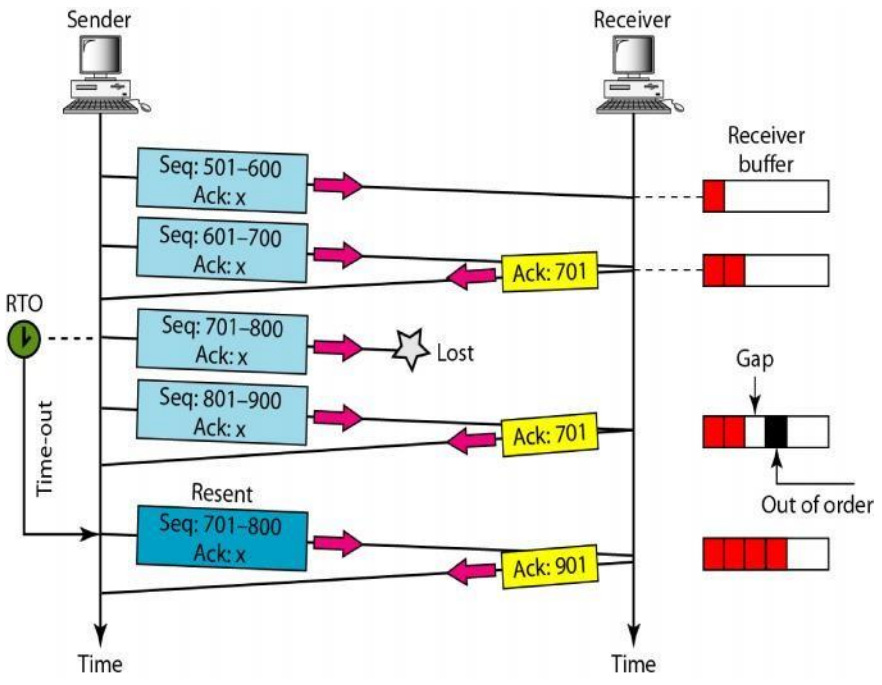


Figure 4.24 Lost segment.

We are assuming that data transfer is unidirectional: one site is sending and the other is receiving. In our scenario, the sender sends segments 1 and 2, which are acknowledged immediately by an ACK. Segment 3, however, are lost. The receiver receives segment 4, which
is out of order. The receiver stores the data in the segment in its buffer but leaves a gap to indicate that there is no continuity in the data. The receiver immediately sends an acknowledgment to the sender, displaying the next byte it expects. Note that the receiver stores bytes 801 to 900, but never delivers these bytes to the application until the gap is filled.

The receiver TCP delivers only ordered data to the process.

c. Fast Retransmission

In this scenario, we want to show the idea of fast retransmission. Our scenario is the sameas the second except that the RTO has a higher value (see Figure 4.25).

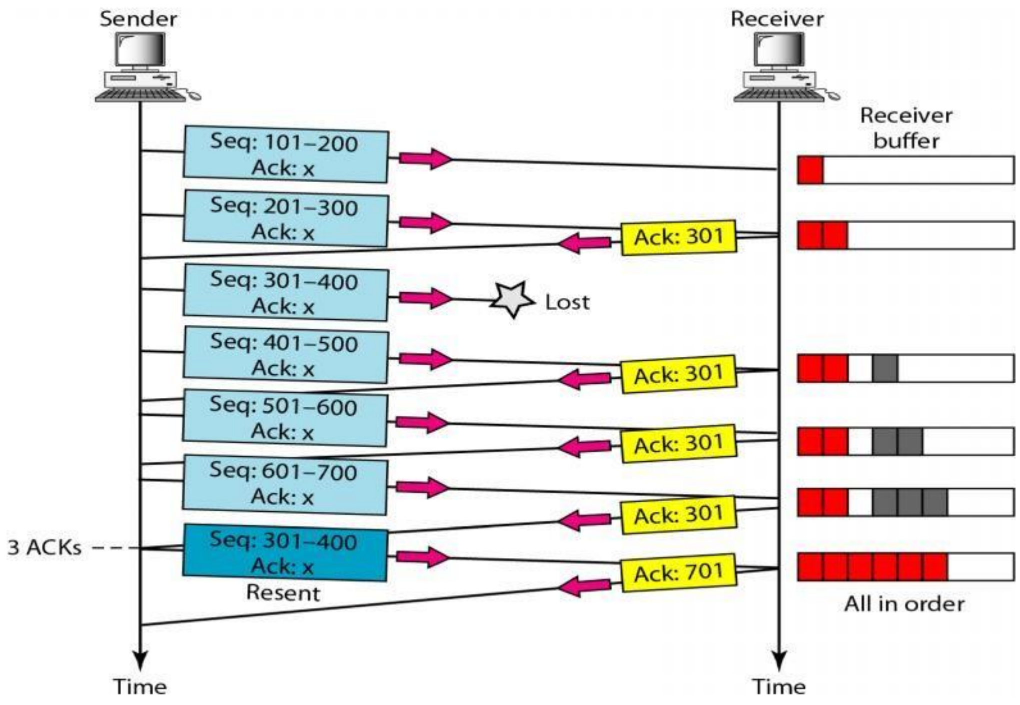


Figure 4.25 Fast Retransmission

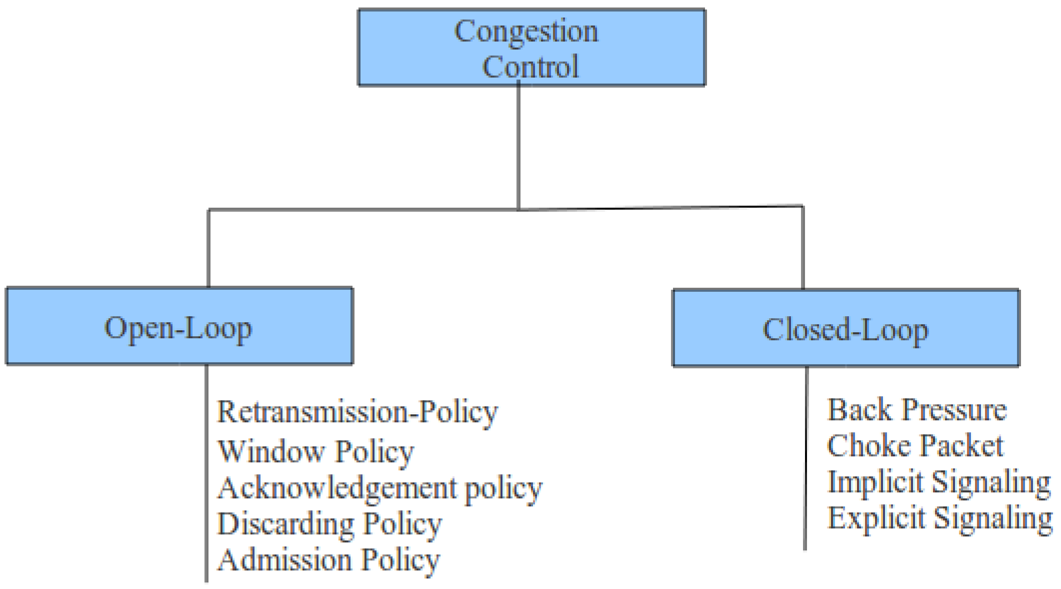
When the receiver receives the fourth, fifth, and sixth segments, it triggers an acknowledgment. The sender receives four acknowledgments with the same value (three duplicates). Although the timer for segment 3 has not matured yet, the fast transmission requires that segment 3, the segment that is expected by all these acknowledgments, be resent immediately.

Congestion control:

Congestion in a network may occur if the load on the network(the number of packets sent to the network) is greater than the capacity of the network(the number of packets a network can handle). Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity .

- When too many packets are pumped into the system, congestion occur leading into degradation ofperformance.
- Congestion tends to feed upon itself and back ups.
- Congestion shows lack of balance between various networking equipments.
- It is a global issue.

In general, we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal) as shown in Figure



Open Loop Congestion Control:

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

Retransmission Policy

Retransmission is sometimes unavoidable. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. Retransmission in general may increase congestion in the network. However, a good retransmission policy can prevent congestion. The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion. For example, the retransmission policy used by TCP is designed to prevent or alleviate congestion.

Window Policy

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

Acknowledgment Policy :

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

Discarding Policy :

A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrityof the transmission. For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.

Admission Policy :

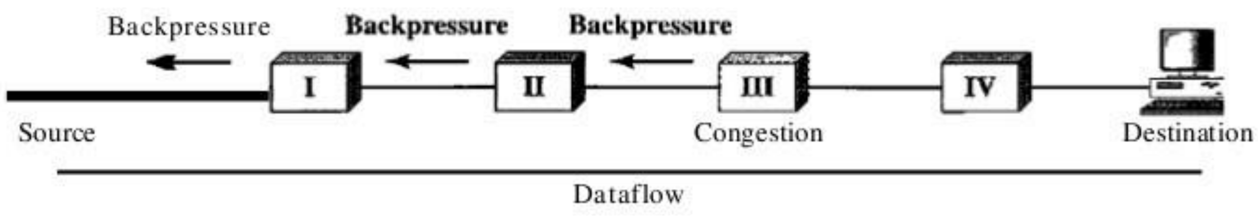
An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual- circuit connection if there is congestion in the network or if there is a possibility of future congestion.

Closed-Loop Congestion Control

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

Back-pressure:

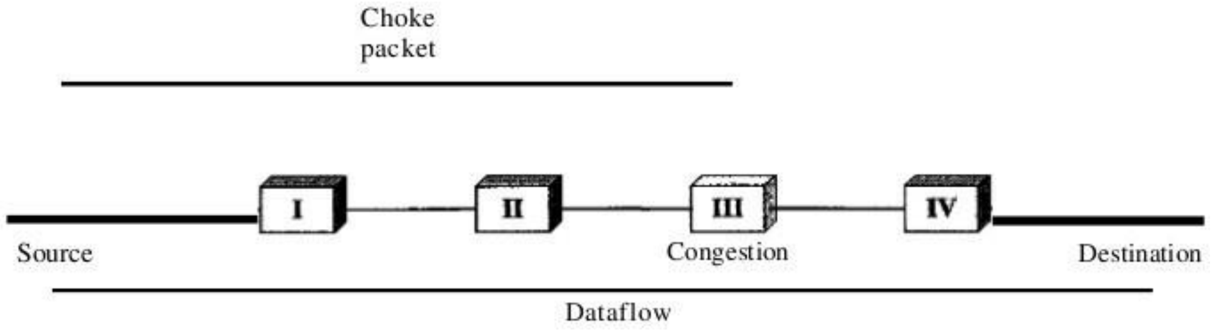
The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes. And so on. Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source. The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is corning.



Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down. Node II, in turn, may be congested because it is slowing down the output flow of data. If node II is congested, it informs node I to slow down, which in turn may create congestion. If so, node I informs the source of data to slow down. This, in time, alleviates the congestion. Note that the pressure on node III is moved backward to the source to remove the congestion. None of the virtual-circuit networks we studied inthis book use backpressure. It was, however, implemented in the first virtual-circuit network, X.25. The technique cannot be implemented in a datagram network because in this type of network, a node (router) does not have the slightest knowledge of the upstream router.

Choke Packet

A choke packet is a packet sent by a node to the source to inform it of congestion. Note the difference between the backpressure and choke packet methods. In backpresure, the warning is from one node to its upstream node, although the warning may eventually reach the source station. In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly. The intermediate nodes through which the packet has traveled are not warned. We have seen an example of this type of control in ICMP. When a router in the Internet is overwhelmed datagrams, it may discard some of them; but it informs the source . host, using a source quench ICMP message. The warning message goes directly to the source station; the intermediate routers, and does not take any action. Figure shows the idea of a choke packet.



Implicit Signaling

In implicit signaling, there is no communication between the congested node or nodes and the source. The source guesses that there is a congestion somewhere in the network from other symptoms. For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested. The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down. We will see this type of signaling when we discuss TCP congestion control later in the chapter.

Explicit Signaling

The node that experiences congestion can explicitly send a signal to the source or destination. The explicit signaling method, however, is different from the choke packet method. In the choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data. Explicit signaling, as we will see in Frame Relay congestion control, can occur in either the forward or the backward direction.

Backward Signaling A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

Forward Signaling A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

Traffic Shaping

Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic: leaky bucket and token bucket.

i) 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 to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate. Figure 4.34 shows a leaky bucket and its effects.

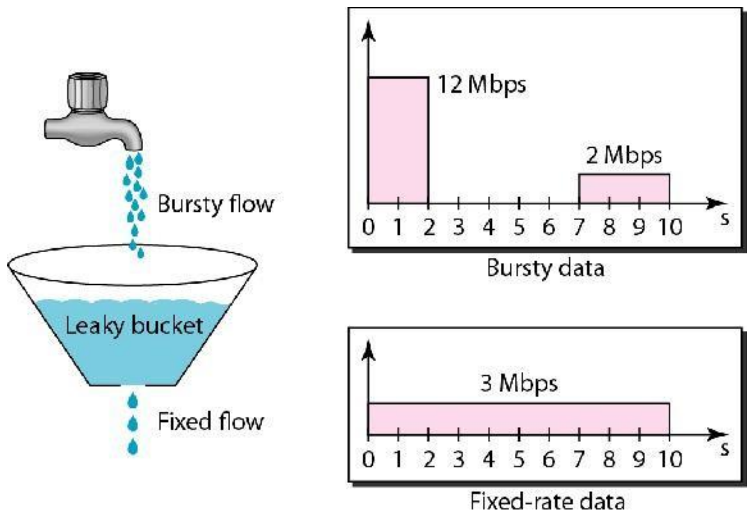


Figure 4.34 Leaky Bucket

In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure 4.34 the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits ofdata. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in lOs. The leaky bucket smooth's the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

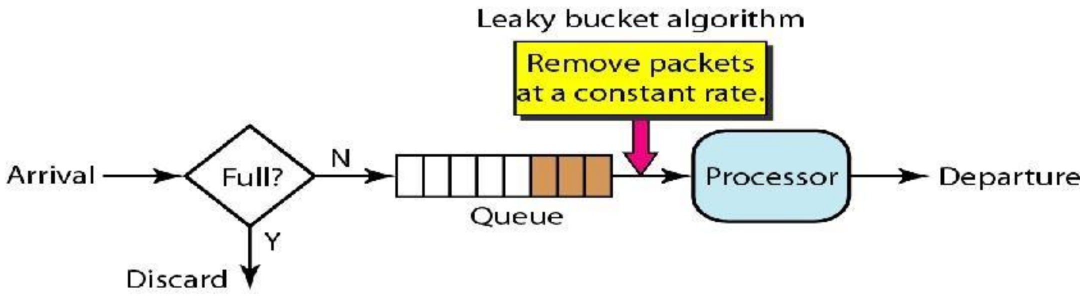


Figure 4.35 Leaky bucket implementation

A simple leaky bucket implementation is shown in Figure 4.35. 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 consistsof variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.
2. If n is greater than the size of the packet, send the packet and decrement the counter bythe packet size. Repeat this step until n is smaller than the packet size.
3. Reset the counter and go to step 1.

A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

ii) Token Bucket

The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens. For each tick of the clock, the system sends n tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if n is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens.

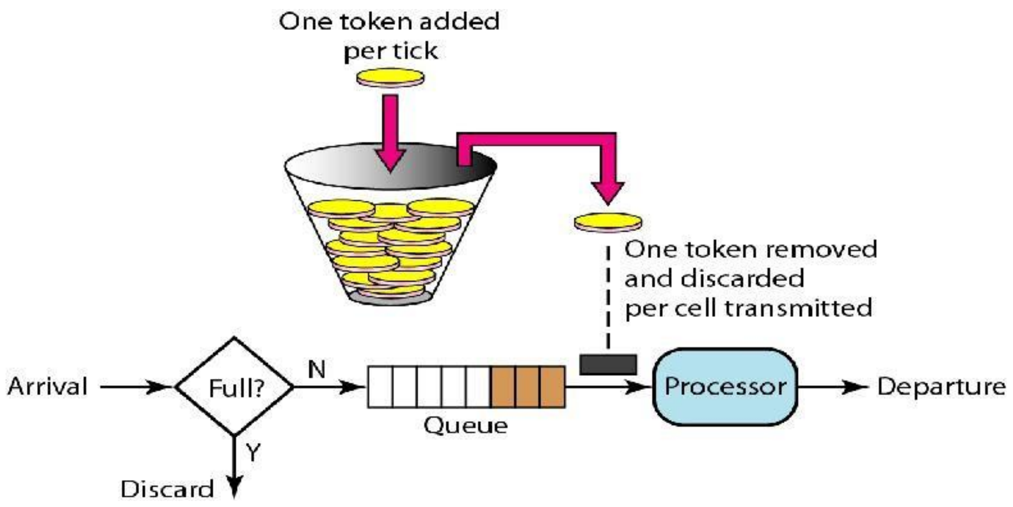


Figure 4.36 Token bucket

The token bucket can easily be implemented with a counter. The token is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.

The token bucket allows bursty traffic at a regulated maximum rate.

Combining Token Bucket and Leaky Bucket

The two techniques can be combined to credit an idle host and at the same time regulate the traffic. The leaky bucket is applied after the token bucket; the rate of the leaky bucket needs to be higher than the rate of tokens dropped in the bucket.

a. Resource Reservation

A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on. The quality of service is improved if these resources are reserved beforehand. We discuss in this section one QoS model called Integrated Services, which depends heavily on resource reservation to improve the quality of service.

b. 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 (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

7 Quality of services:
QoS is an overall performance measure of the computer network.

Important flow characteristics of the QoS are given below:

1. Reliability
If a packet gets lost or acknowledgement is not received (at sender), the re-transmission of data will be needed. This decreases the reliability.
The importance of the reliability can differ according to the application.
For example:
E- mail and file transfer need to have a reliable transmission as compared to that of an audio conferencing.

2. Delay
Delay of a message from source to destination is a very important characteristic. However, delay can be tolerated differently by the different applications.
For example:
The time delay cannot be tolerated in audio conferencing (needs a minimum time delay), while the time delay in the e-mail or file transfer has less importance.

3. Jitter
The jitter is the variation in the packet delay.
If the difference between delays is large, then it is called as **high jitter.** On the contrary, if the difference between delays is small, it is known as **low jitter.**
Example:
Case1: If 3 packets are sent at times 0, 1, 2 and received at 10, 11, 12. Here, the delay is same for all packets and it is acceptable for the telephonic conversation.
Case2: If 3 packets 0, 1, 2 are sent and received at 31, 34, 39, so the delay is different for all packets. In this case, the time delay is not acceptable for the telephonic conversation.

4. Bandwidth
Different applications need the different bandwidth.
For example:
Video conferencing needs more bandwidth in comparison to that of sending an e-mail.

Integrated Services and Differentiated Service

These two models are designed to provide Quality of Service (QoS) in the network.

1. Integrated Services(IntServ)
Integrated service is flow-based QoS model and designed for IP.
In integrated services, user needs to create a flow in the network, from source to destination and needs to inform all routers (every router in the system implements IntServ) of the resource requirement.

Following are the steps to understand how integrated services works.
I) Resource Reservation Protocol (RSVP)
An IP is connectionless, datagram, packet-switching protocol. To implement a flow-based model, a signaling protocol is used to run over IP, which provides the signaling mechanism to make reservation (every applications need assurance to make reservation), this protocol is called as RSVP.

ii) Flow Specification
While making reservation, resource needs to define the flow specification. The flow specification has two parts:
a) Resource specification
It defines the resources that the flow needs to reserve. For example: Buffer, bandwidth, etc.
b) Traffic specification
It defines the traffic categorization of the flow.

iii) Admit or deny
After receiving the flow specification from an application, the router decides to admit or deny the service and the decision can be taken based on the previous commitments of the router and current availability of the resource.
Classification of services
The two classes of services to define Integrated Services are:

a) Guaranteed Service Class
This service guarantees that the packets arrive within a specific delivery time and not discarded, if the traffic flow maintains the traffic specification boundary.
This type of service is designed for real time traffic, which needs a guaranty of minimum end to end delay.
For example: Audio conferencing.

b) Controlled Load Service Class
This type of service is designed for the applications, which can accept some delays, but are sensitive to overload network and to the possibility to lose packets.
For example: E-mail or file transfer.
Problems with Integrated Services.

The two problems with the Integrated services are:

i) Scalability
In Integrated Services, it is necessary for each router to keep information of each flow. But, this is not always possible due to growing network.

ii) Service- Type Limitation
The integrated services model provides only two types of services, guaranteed and control-load.
2. Differentiated Services (DS or Diffserv):
DS is a computer networking model, which is designed to achieve the scalability by managing the network traffic.
DS is a class based QoS model specially designed for IP.
DS was designed by IETF (Internet Engineering Task Force) to handle the problems of Integrated Services.
The solutions to handle the problems of Integrated Services are explained below:

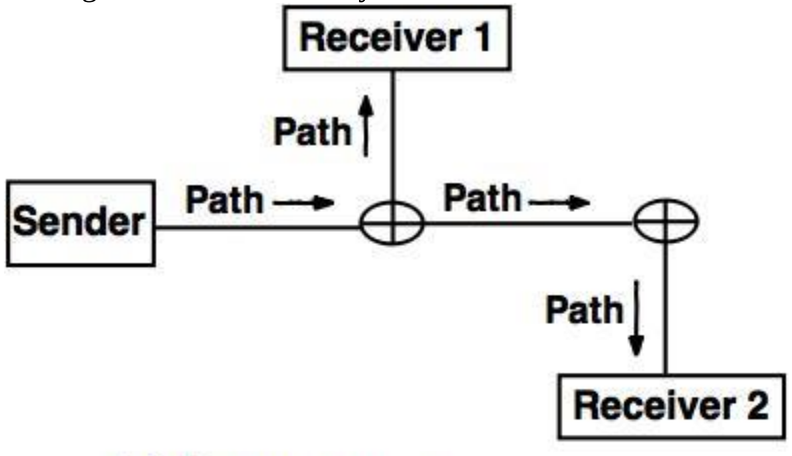
1. Scalability
The main processing unit can be moved from central place to the edge of the network to achieve

the scalability. The router does not need to store the information about the flows and the applications (or the hosts) define the type of services they want every time while sending the packets.

2. Service Type Limitation
The routers, route the packets on the basis of class of services define in the packet and not by the flow. This method is applied by defining the classes based on the requirement of the applications.
Resource Reservation Protocol (RSVP)

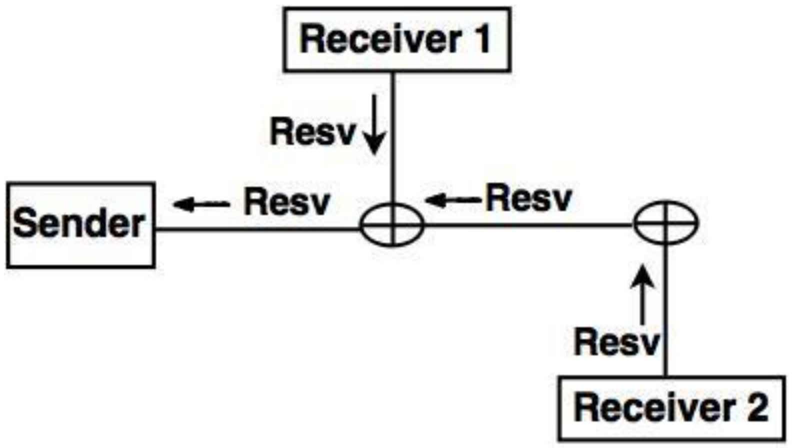
The RSVP is a signaling protocol, which helps IP to create a flow and to make resource reservation.
It is an independent protocol and also can be used in other different model.
RSVP helps to design multicasting (one to many or many to many distribution), where a data can be sent to group of destination computers simultaneously.
For example: The IP multicast is technique for one to many communication through an IP infrastructure in the network.
RSVP can be also used for unicasting (transmitting a data to all possible destination) to provide resource reservation for all types of traffic.
The two important types of RSVP messages are:

1. Path messages:
The receivers in a flow make the reservation in RSVP, but the receivers do not know the path traveled by the packets before the reservation. The path is required for reservation To solve this problem the RSVP uses the path messages.
A path message travels from the sender and reaches to all receivers by multicasting and path message stores the necessary information for the receivers.



Path messages

2. Resv messages:
After receiving path message, the receiver sends a Resv message. The Resv message travels to the sender and makes a resource reservation on the routers which supports for RSVP.



Resv Messages