Unit 5: The Transport Layer

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

Functions of Transport Layer:

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

Elements of Transport Protocols:

Addressing:

Transport layer must figure out which process to send the data to. To make this possible, an additional addressing element is necessary. This address allows a more specific location—a software process—to be identified within a particular IP address. In TCP/IP, this transport layer address is called a port address.

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

The network layer uses the IP packet information to transport the packet across the network (routing). At the transport layer, we need a transport layer address, called a port number, to choose among multiple processes running on the destination host. The destination port number is needed for delivery; the source port number is needed for the reply.

In the Internet model, the port numbers are 16-bit integers between 0 and 65,535. The IANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well-known, registered, and dynamic (or private).

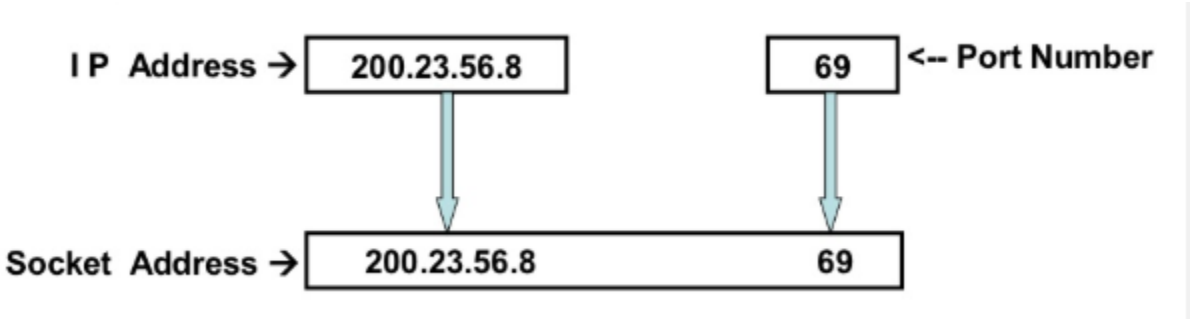
- Well-known ports. The ports ranging from 0 to 1023 are assigned and controlled by IANA. These are the well-known ports.
- Registered ports. The ports ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication.
- Dynamic ports. The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process.

Some of the well-known port addresses associated with network applications are:

Port Number	Assignment
21	File Transfer Protocol (FTP)
23	TELNET remote login

25	SMTP
80	HTTP
53	DNS

Socket is the program that is associated with every process. It is the endpoint of a two-way communication link between two processes running on the network. A socket is bound to a port number so that the transport layer can identify the application that data is destined to be sent. Socket address is the combination of IP address and a Port number.



Establishing and Releasing Connection:

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

In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either.

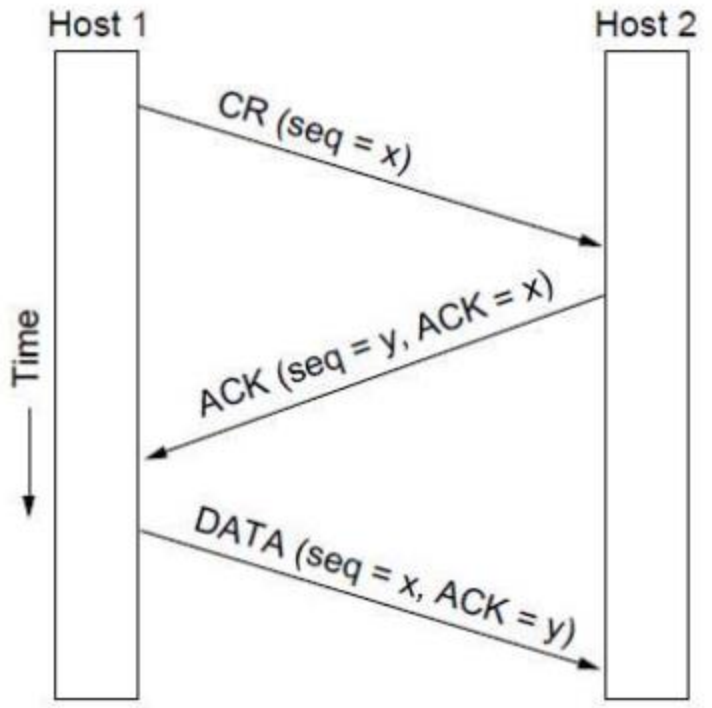
In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released. This constitutes three phases of communication for connection-oriented mechanism:

- Connection is established.
- Information is sent.
- Connection is released.

Connection Establishment:

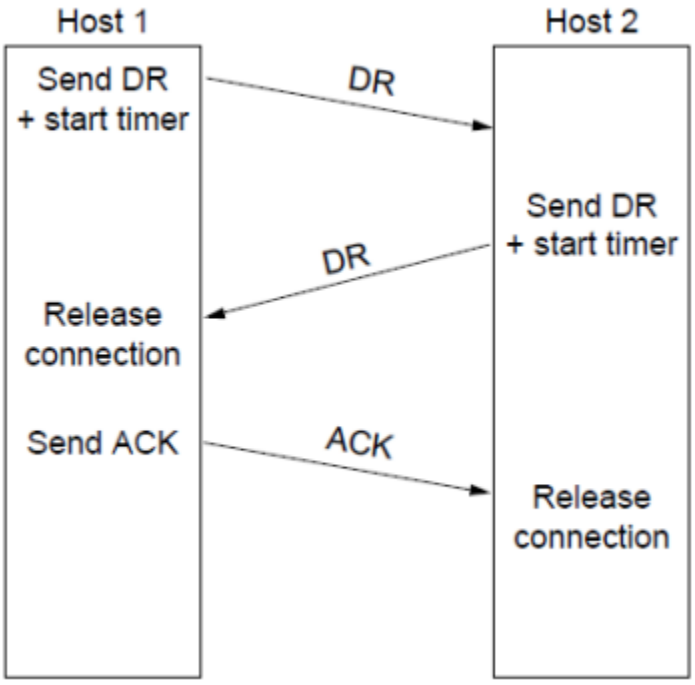
The connection establishment is done using three-way handshaking. The figure below shows the handshaking process.

CR: Connection Request



Connection Release:

Three-way handshaking is also used to release the connection.



DR: Disconnect Request

Flow Control and Buffering:

Flow control is done by having a sliding window on each connection to keep a fast transmitter over a slow receiver. Buffering must be done by the sender, if the network service is unreliable. The sender buffers all the TPDUs sent to the receiver. The buffer size varies for different TPDUs (Transport Layer Processing Data Unit).

They are:

- Chained Fixed-size Buffers
- Chained Variable-size Buffers
- One large Circular Buffer per Connection

Chained Fixed-size Buffers:

If most TPDUs are nearly the same size, the buffers are organized as a pool of identical size buffers, with one TPDU per buffer.

Chained Variable-size Buffers:

This is an approach to the buffer-size problem. i.e., if there is wide variation in TPDU size, from a few characters typed at a terminal to thousands of characters from file transfers, some problems may occur:

- If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short TPDU arrives.
- If the buffer size is chosen less than the maximum TPDU size, multiple buffers will be needed for long TPDUs.

To overcome these problems, we employ variable-size buffers.

One large Circular Buffer per Connection:

A single large circular buffer per connection is dedicated when all connections are heavily loaded.

- Source Buffering is used for low band width bursty traffic
- Destination Buffering is used for high band width smooth traffic.
- Dynamic Buffering is used if the traffic pattern changes randomly.

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 is achieved through the use of three simple tools: *checksum, acknowledgment, and retransmission*.

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 and is considered as lost.

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.

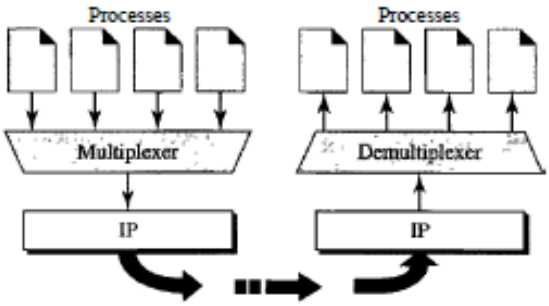
Retransmission

The heart of the error control mechanism is the retransmission of segments. When a segment is corrupted, lost, or delayed, it is retransmitted.

Multiplexing and Demultiplexing:

Multiplexing at the transport layer refers to multiple applications on the same host sharing the common medium. All applications at a host create one signal stream. The physical medium multiplexes signal streams from multiple hosts.

Multiplexing and demultiplexing



Multiplexing

At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing. The protocol accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, the transport layer passes the packet to the network layer.

Demultiplexing

At the receiver site, the relationship is one-to-many and requires demultiplexing. The transport layer receives datagrams from the network layer. After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.

Crash Recovery:

If hosts and routers are subject to crashes, recovery from these crashes becomes an issue.

If the transport entity is entirely within the hosts, it is difficult to recover from host crashes.

But recovery from network and router crashes is straightforward since transport entities are alive at the host.

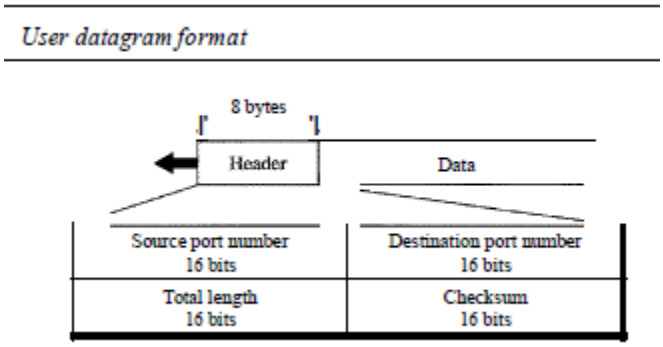
- If the network layer provides datagram service, the transport entities expect lost TPDUs all the time and know how to cope with them.

- If the network layer provides connection-oriented service, then loss of a virtual circuit is handled by establishing a new one and then probing the remote transport entity to ask it which TPDUs it has received and which ones it has not received. The latter ones can be retransmitted.

User Datagram Protocol (UDP):

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

The UDP packet structure is as follows:



UDP Operations:

UDP uses concepts common to the transport layer.

Connectionless Services: UDP provides connectionless service which means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination process. The user datagrams are not numbered and there is no connection establishment and no connection termination, which means each user datagram can travel on a different path. Stream of data cannot be sent; it should get fragmented.

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

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

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

Uses of UDP:

- UDP allows very simple data transmission without any error detection. So, it can be used in networking applications like VOIP, streaming media, etc. where loss of few packets can be tolerated and still function appropriately.

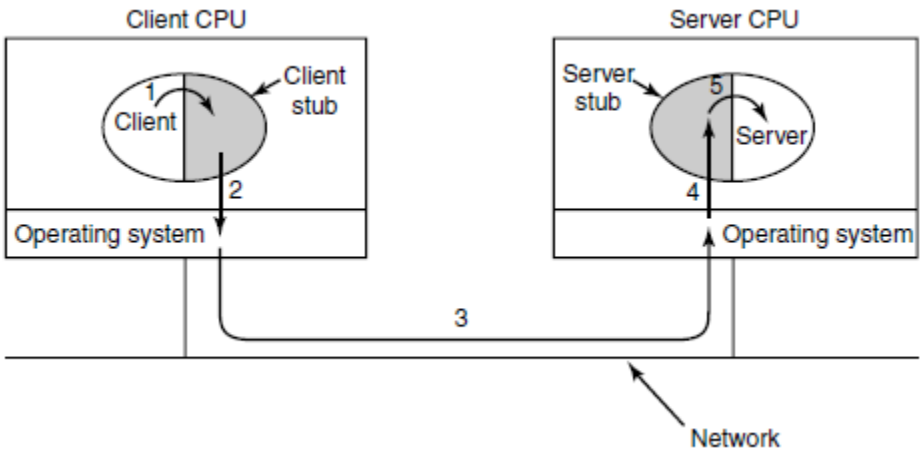
- UDP is suitable for multicasting since multicasting capability is embedded in UDP software but not in TCP software.
- UDP is used for management processes such as SNMP.
- UDP is used for some route updating protocols such as RIP.
- UDP is used by Dynamic Host Configuration Protocol (DHCP) to assign IP addresses to systems dynamically.
- UDP is an ideal protocol for network applications where latency is critical such as gaming and voice and video communications.

Remote Procedural Call (RPC):

In a certain sense, sending a message to a remote host and getting a reply back is a lot like making a function call in a programming language. In both cases, you start with one or more parameters and you get back a result. This observation has led people to try to arrange request-reply interactions on networks to be cast in the form of procedure calls. Such an arrangement makes network applications much easier to program and more familiar to deal with. For example, just imagine a procedure named get IP address (host name) that works by sending a UDP packet to a DNS server and waiting for the reply, timing out and trying again if one is not forthcoming quickly enough. In this way, all the details of networking can be hidden from the programmer.

When a process on machine 1 calls a procedure on machine 2, the calling process on 1 is suspended and execution of the called procedure takes place on 2. Information can be transported from the caller to the callee in the parameters and can come back in the procedure result. No message passing is visible to the application programmer. This technique is known as RPC (Remote Procedure Call) and has become the basis for many networking applications. Traditionally, the calling procedure is known as the client and the called procedure is known as the server.

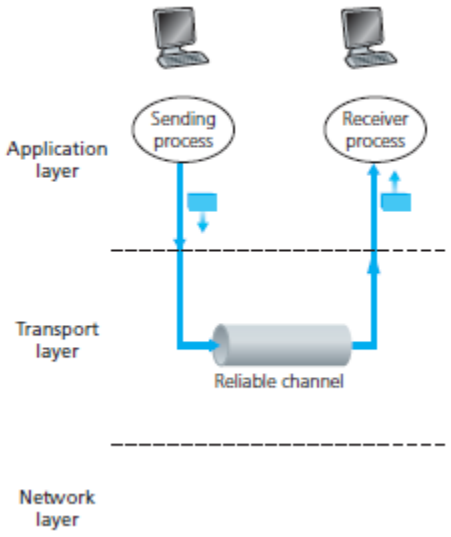
The idea behind RPC is to make a remote procedure call look as much as possible like a local one. In the simplest form, to call a remote procedure, the client program must be bound with a small library procedure, called the client stub, that represents the server procedure in the client's address space. Similarly, the server is bound with a procedure called the server stub. These procedures hide the fact that the procedure call from the client to the server is not local.



Principles of Reliable Data Transfer:

The problem of implementing reliable data transfer occurs not only at the transport layer, but also at the link layer and the application layer as well. The general problem is thus of central importance to networking. The service abstraction provided to the upper-layer entities is that of a reliable channel through which data can be transferred. With a reliable channel, no transferred data bits are corrupted (flipped from 0 to 1, or vice versa) or lost, and all are delivered in the order in which they were sent. This is precisely the service model offered by TCP to the Internet applications that invoke it.

It is the responsibility of a reliable data transfer protocol to implement this service abstraction. This task is made difficult by the fact that the layer below the reliable data transfer protocol may be unreliable. For example, TCP is a reliable data transfer protocol that is implemented on top of an unreliable (IP) end-to-end network layer. More generally, the layer beneath the two reliably communicating end points might consist of a single physical link (as in the case of a link-level data transfer protocol) or a global internetwork (as in the case of a transport-level protocol). However, we can view this lower layer simply as an unreliable point-to-point channel.



Building a Reliable Data Transfer Protocol:

The internet network layer provides only best effort service with no guarantee that packets arrive at their destination. Also, since each packet is routed individually it is possible that packets are received out of order. For connection-oriented service provided by TCP, it is necessary to have a reliable data transfer (RDT) protocol to ensure delivery of all packets and to enable the receiver to deliver the packets in order to its application layer.

A simple alternating bit RDT protocol can be designed using some basic tools. This protocol is also known as a stop-and-wait protocol: after sending each packet the sender stops and waits for feedback from the receiver indicating that the packet has been received.

Stop-and-wait RDT protocols have poor performance in a long-distance connection. At best, the sender can only transmit one packet per round-trip time. For a 1000-mile connection this amounts to approximately 1 packet (about 1500 bytes) every 20 ms. That results in a pathetic 75 KB per second rate.

To improve transmission rates, a realistic RDT protocol must use pipelining. This allows the sender to have a large number of packets "in the pipeline". This phrase refers to packets that have been sent but whose receipt has not yet verified by the receiver.

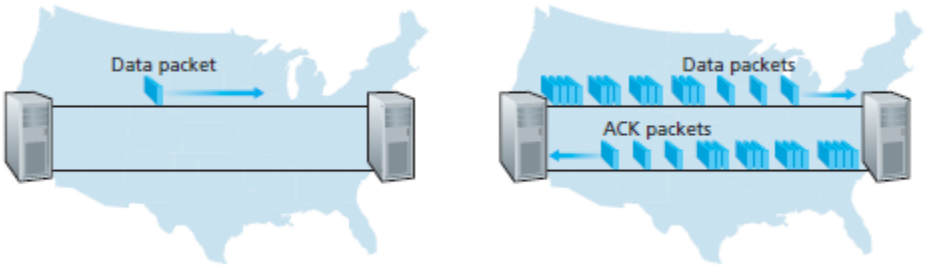
The following tools are essential for any RDT protocol implemented on top of an unreliable network.

- Error detection
- Sequence numbering
- Feedback/Acknowledgement
- Timers

Pipelined Reliable Data Transfer Protocol:

Pipelining allows a sender to send out multiple packets without waiting for acknowledgment. But all packets that have been sent must be retained until they are acknowledged in case, they need to be resent. To limit the number of packets that need to be retained, pipelined protocols need to set a bound on the number of packets in the pipeline (sent but not yet acknowledged). This bound imposes a "window" on the sequence numbers for packets in the pipeline.

Pipelined protocols can be distinguished based on how they use acknowledgments and timers. Two basic pipelined protocols, "Go back n" and "Selective Repeat" acknowledge individual packets, but differ in the number of timers that they use.



a. A stop-and-wait protocol in operation

b. A pipelined protocol in operation

Go-Back-N (GBN) Protocol:

In a Go-Back-N (GBN) protocol, the sender is allowed to transmit multiple packets (when available) without waiting for an acknowledgment, but is constrained to have no more than some maximum allowable number, N, of unacknowledged packets in the pipeline.

The go back n protocol uses individual acknowledgments. A go back n sender uses a single timer. When this timer fires the sender does not know which packet failed to be received so the sender resends all packets in the window.

Selective Repeat (SR):

The selective repeat protocol, like the go back n protocol, uses individual acknowledgments. Unlike the go back n protocol, a selective repeat sender uses a timer for each packet in the pipeline. This makes it possible to be selective about handling timer firing. The selective repeat sender only resends the packet associated with the timer that fired.

Transmission Control Protocol (TCP):

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

TCP Services:

Various services and operations of TCP are as follows:

- Process to Process Communication: Like UDP, TCP provides process-to-process communication using port numbers.
- Stream Delivery Service: TCP is a stream-oriented protocol. It allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. TCP creates an environment in which the two processes seem to have a dedicated connection that carries their data across the internet. For flow control, TCP uses sending and receiving buffer that provides some storage for data packets in case of overflow. This prevents the loss of packets by synchronizing the flow rate.
- Full-Duplex Communication: TCP offers full-duplex services in which data can flow in both directions at the same time. Each TCP then has sending and receiving buffer, and segments move in both directions.
- Connection-Oriented Service: When a process at site A wants to send and receive data from another process at site B, the following occurs:
    - The two TCPs establish a connection between them.
    - Data are exchanged in both directions.
    - The connection is terminated.

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

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

TCP Features and Characteristics:

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

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

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

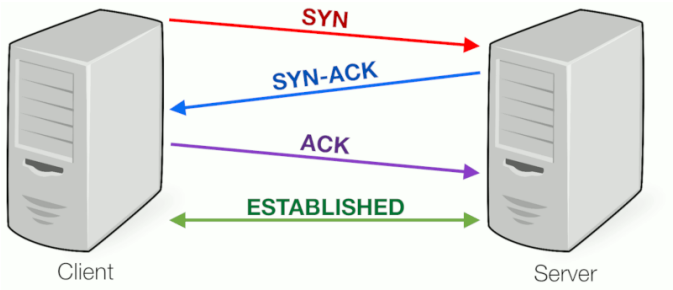
Error Control: To provide reliable service, TCP implements an error control mechanism. Although error control mechanism considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.

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

Three-Way Handshaking:

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

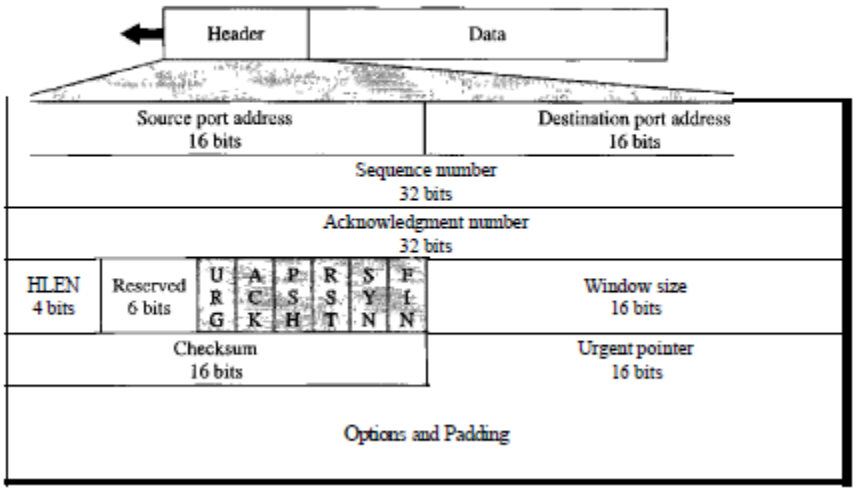
The connection establishment in TCP is called three-way handshaking. The figure below shows the handshaking process.



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

TCP Segment Structure:

Figure 23.16 *TCP segment format*



- Source Port Address:

  16-bit field that holds the port address of the application that is sending the data segment.

- Destination Port Address:

  16-bit field that holds the port address of the application in the host that is receiving the data segment.

- Sequence Number:

  32-bit field that holds the sequence number, i.e, the byte number of the first byte that is sent in that particular segment. It is used to reassemble the message at the receiving end if the segments are received out of order.

- Acknowledgement Number:

  32-bit field that holds the acknowledgement number, i.e, the byte number that the receiver expects to receive next. It is an acknowledgement for the previous bytes being received successfully.

- Header Length (HLEN):

  This is a 4-bit field that indicates the length of the TCP header by number of 4-byte words in the header, i.e, if the header is of 20 bytes (min length of TCP header), then this field will hold 5 (because 5 x 4 = 20) and the maximum length: 60 bytes, then it'll hold the value 15(because 15 x 4 = 60). Hence, the value of this field is always between 5 and 15.

- Control flags:

  These are 6 1-bit control bits that control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc. Their function is:

  - URG: Urgent pointer is valid
  - ACK: Acknowledgement number is valid (used in case of cumulative acknowledgement)
  - PSH: Request for push
  - RST: Reset the connection
  - SYN: Synchronize sequence numbers
  - FIN: Terminate the connection

- Window size: This field tells the window size of the sending TCP in bytes.
- Reserved: This is a 6-bit field reserved for future use.
- Checksum: This field holds the checksum for error control. It is mandatory in TCP as opposed to UDP.
- Urgent pointer:
  This field (valid only if the URG control flag is set) is used to point to data that is urgently required that needs to reach the receiving process at the earliest. The value of this field is added to the sequence number to get the byte number of the last urgent byte.

Key Differences Between TCP and UDP:

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

Congestion:

Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called congestion. In other words, congestion in a network may occur if the load on the network, the number of packets sent to the network, is greater than the capacity of the network (the number of packets a network can handle). The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets. However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

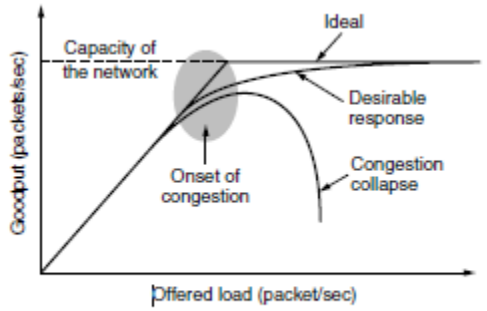
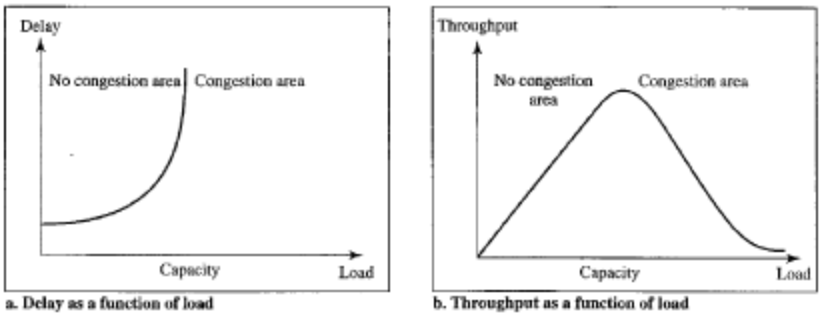


Figure 5-21. With too much traffic, performance drops sharply.

Figure 24.4 *Packet delay and throughput as functions of load*



a. Delay as a function of load

b. Throughput as a function of load

Effects of Congestion:

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

Congestion Control:

Congestion, in the context of networks, refers to a network state where a node or link carries so much data that it may deteriorate network service quality, resulting in queuing delay, frame or data packet loss and the blocking of new connections.

Congestion in data networking and queuing theory is the reduced quality of service that occurs when a network node is carrying more data than it can handle. Typical effects include queuing, delay, packet loss or the blocking of new connections.

Congestion in a network or internetwork occurs because routers and switches have queues – buffers that hold the packets before and after processing.

A router, for example, has an input queue and an output queue for each interface. When a packet arrives at the incoming interface, it undergoes three steps before departing.

- The packet is put at the end of the input queue while waiting to be checked.
- The processing module of the router removes the packet from the input queue once it reaches the front of the queue and uses its routing table and the destination address to find the route.
- The packet is put in the appropriate output queue and waits its turn to be sent.

General Principles of Congestion Control:

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

The General Principles of Congestion Control are as follows:

Open Loop Principle: solve problem by good design

- attempt to prevent congestion from happening

- after system is running, no corrections made

Closed Loop Principle: concept of feedback loop

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

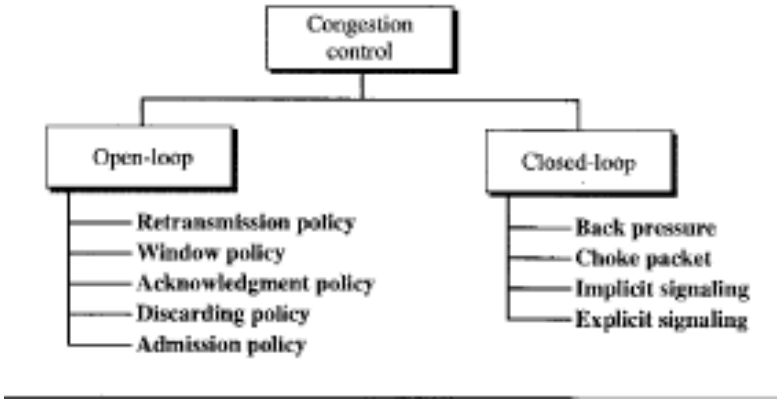
Congestion Prevention Policies:

Congestion prevention policies can be broadly divided into two broad categories.

Open loop congestion control

Closed loop congestion control

Congestion control categories



Open Loop Congestion Control

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

Retransmission Policy:

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

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

Window Policy

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

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

Acknowledgement Policy

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

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

Discarding Policy

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

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

Admission Policy

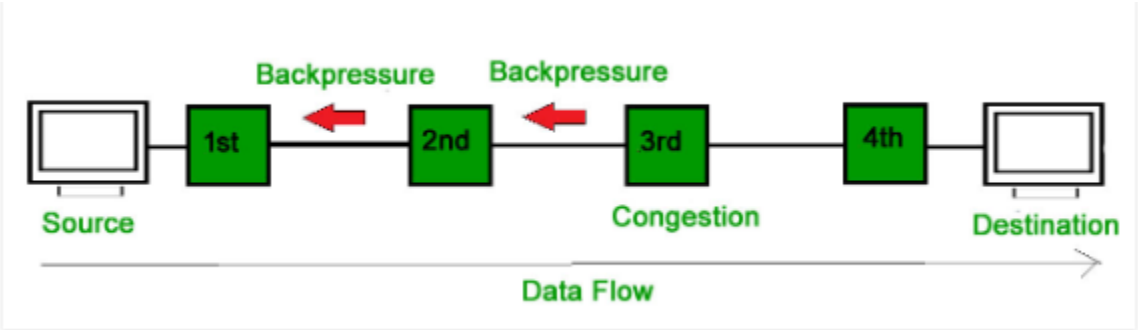
In admission policy a mechanism should be used to prevent congestion. Switches in a flow should first check the resource requirement of a network flow before transmitting it further. If there is a chance of a congestion or there is a congestion in the network, router should deny establishing a virtual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

Closed Loop Congestion Control

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

Back Pressure:

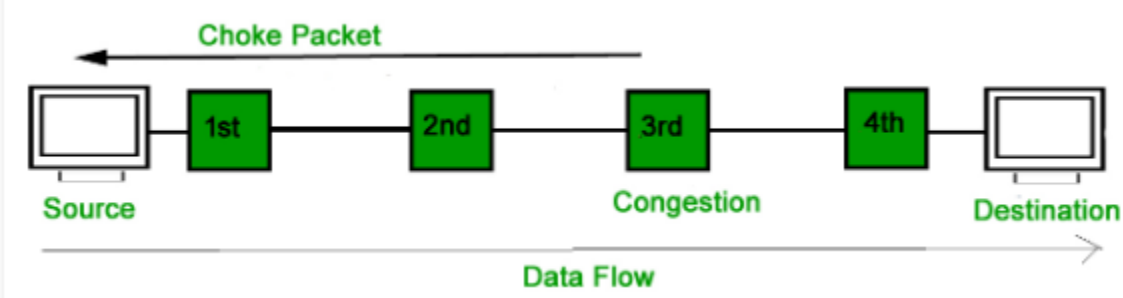
Backpressure is a technique in which a congested node stops receiving packet from upstream node. This may cause the upstream node or nodes to become congested and rejects receiving data from above nodes. Backpressure is a node-to-node congestion control technique that propagate in the opposite direction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



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

Choke Packet:

Choke packet technique is applicable to both virtual networks as well as datagram subnets. A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitors its resources and the utilization at each of its output lines. whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets have traveled are not warned about congestion.



Implicit Signaling:

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

Explicit Signaling:

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

Explicit signaling can occur in either forward or backward direction.

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

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