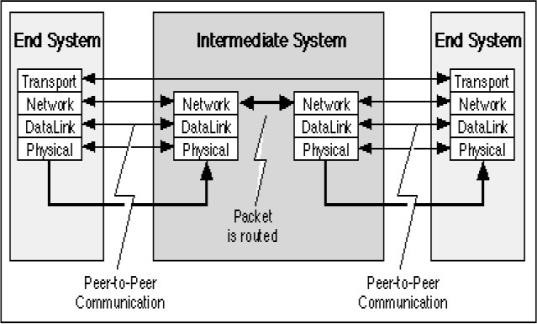
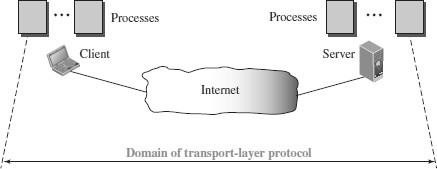
**1. INTRODUCTION**

* The transport layer is the fourth layer of the OSI model and is the core of the Internet model.
* It responds to service requests from the session layer and issues service requests to the network Layer.
* The transport layer provides transparent transfer of data between hosts.
* It provides end-to-end control and information transfer with the quality of service needed by the application program.
* It is the first true end-to-end layer, implemented in all End Systems (ES).



## TRANSPORT LAYER FUNCTIONS / SERVICES

* The transport layer is located between the network layer and the application layer.
* The transport layer is responsible for providing services to the application layer; it receives services from the network layer.
* The services that can be provided by the transport layer are
  1. Process-to-Process Communication
  2. Addressing : Port Numbers
  3. Encapsulation and Decapsulation
  4. Multiplexing and Demultiplexing
  5. Flow Control
  6. Error Control
  7. Congestion Control

## Process-to-Process Communication

* The Transport Layer is responsible for delivering data to the appropriate application process on the host computers.
* This involves multiplexing of data from different application processes, i.e. forming data packets, and adding source and destination port numbers in the header of each Transport Layer data packet.
* Together with the source and destination IP address, the port numbers constitutes a network socket, i.e. an identification address of the process-to-process communication.

## Addressing: Port Numbers

* Ports are the essential ways to address multiple entities in the same location.
* Using port addressing it is possible to use more than one network-based application at the same time.
* Three types of Port numbers are used :
* ***Well-known ports*** *-* These are permanent port numbers. They range between 0 to 1023.These port numbers are used by Server Process.
* ***Registered ports -*** The ports ranging from 1024 to 49,151 are not assigned or controlled.
* ***Ephemeral ports (Dynamic Ports) –*** These are temporary port numbers. They range between 49152–65535.These port numbers are used by Client Process**.**

## Encapsulation and Decapsulation

* To send a message from one process to another, the transport-layer protocol encapsulates and decapsulates messages.
* Encapsulation happens at the sender site. The transport layer receives the data and adds the transport-layer header.
* Decapsulation happens at the receiver site. When the message arrives at the destination transport layer, the header is dropped and the transport layer delivers the message to the process running at the application layer.

## Multiplexing and Demultiplexing

* Whenever an entity accepts items from more than one source, this is referred to as

***multiplexing*** (many to one).

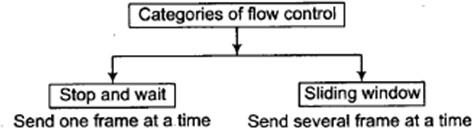
* Whenever an entity delivers items to more than one source, this is referred to as

***demultiplexing*** (one to many).

* The transport layer at the source performs multiplexing
* The transport layer at the destination performs demultiplexing

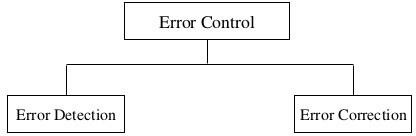
## Flow Control

* Flow Control is the process of managing the rate of data transmission between two nodes to prevent a fast sender from overwhelming a slow receiver.
* It provides a mechanism for the receiver to control the transmission speed, so that the receiving node is not overwhelmed with data from transmitting node.



## Error Control

* Error control at the transport layer is responsible for
  + 1. Detecting and discarding corrupted packets.
    2. Keeping track of lost and discarded packets and resending them.
    3. Recognizing duplicate packets and discarding them.
    4. Buffering out-of-order packets until the missing packets arrive.
* Error Control involves Error Detection and Error Correction

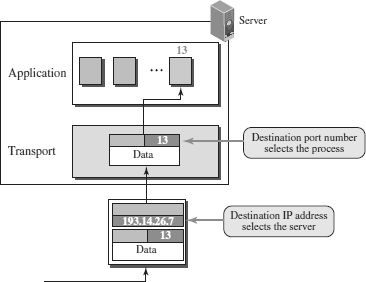


## Congestion Control

* 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 that control the congestion and keep the load below the capacity.
* Congestion Control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened
* Congestion control mechanisms are divided into two categories,

1. Open loop - prevent the congestion before it happens.
2. Closed loop - remove the congestion after it happens.

**2. PORT NUMBERS**

* A transport-layer protocol usually has several responsibilities.
* One is to create a process-to-process communication.
* Processes are programs that run on hosts. It could be either *server* or *client*.
* A process on the local host, called a *client,* needs services from a process usually on the remote host, called a *server.*
* Processes are assigned a unique 16-bit *port number* on that host.
* Port numbers provide end-to-end addresses at the transport layer
* They also provide multiplexing and demultiplexing at this layer.
* The port numbers are integers between 0 and 65,535 .

ICANN (Internet Corporation for Assigned Names and Numbers) has divided the port numbers into three ranges:

#### Well-known ports

* + **Registered**

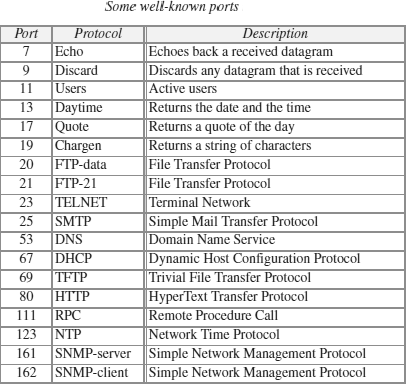
#### Ephemeral ports (Dynamic Ports)





#### WELL-KNOWN PORTS

* These are permanent port numbers used by the servers.
* They range between 0 to 1023.
* This port number cannot be chosen randomly.
* These port numbers are universal port numbers for servers.
* Every client process knows the well-known port number of the corresponding server process.
* For example, while the daytime client process, a well-known client program, can use an ephemeral (temporary) port number, 52,000, to identify itself, the daytime server process must use the well-known (permanent) port number 13.



#### EPHEMERAL PORTS (DYNAMIC PORTS)

* The client program defines itself with a port number, called the ***ephemeral port number.***
* The word *ephemeral* means “short-lived” and is used because the life of a client is normally short.
* An ephemeral port number is recommended to be greater than 1023.
* These port number ranges from 49,152 to 65,535 .
* They are neither controlled nor registered. They can be used as temporary or private port numbers.

#### REGISTERED PORTS

* The ports ranging from 1024 to 49,151 are not assigned or controlled.

**3. TRANSPORT LAYER PROTOCOLS**

* Three protocols are associated with the Transport layer.
* They are

#### UDP –User Datagram Protocol

1. **TCP – Transmission Control Protocol**

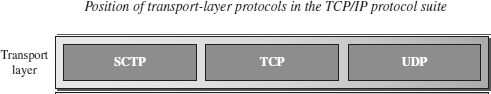
#### SCTP - Stream Control Transmission Protocol

* Each protocol provides a different type of service and should be used appropriately.

**UDP -** UDP is an unreliable connectionless transport-layer protocol used for its simplicity and efficiency in applications where error control can be provided by the application-layer process.

**TCP -** TCP is a reliable connection-oriented protocol that can be used in any application where reliability is important.

**SCTP -** SCTP is a new transport-layer protocol designed to combine some features of UDP and TCP in an effort to create a better protocol for multimedia communication.

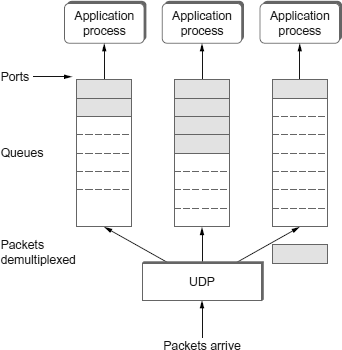


# 4. USER DATAGRAM PROTOCOL (UDP)

* User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol.
* UDP adds process-to-process communication to best-effort service provided by IP.
* UDP is a very simple protocol using a minimum of overhead.
* UDP is a simple demultiplexer, which allows multiple processes on each host to communicate.
* UDP does not provide flow control , reliable or ordered delivery.
* UDP can be used to send small message where reliability is not expected.
* Sending a small message using UDP takes much less interaction between the sender and receiver.
* UDP allow processes to indirectly identify each other using an abstract locator called port or mailbox

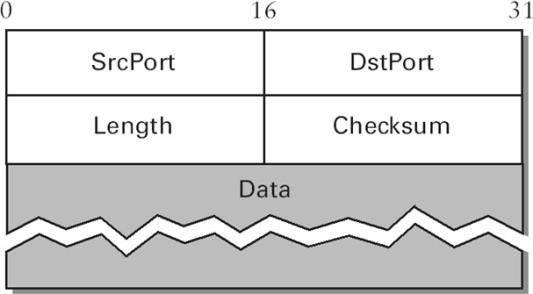
## UDP PORTS

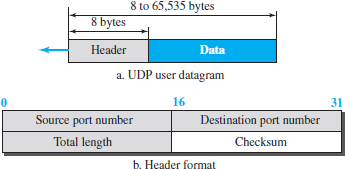
* Processes (server/client) are identified by an abstract locator known as port.
* Server accepts message at *well known port*.
* Some well-known UDP ports are 7–Echo, 53–DNS, 111–RPC, 161–SNMP, etc.
* < *port*, *host* > pair is used as key for demultiplexing.
* Ports are implemented as a *message queue*.
* When a message arrives, UDP *appends* it to end of the queue.
* When queue is *full*, the message is discarded.
* When a message is *read*, it is removed from the queue.
* When an application process wants to receive a message, one is removed from the front of the queue.
* If the queue is empty, the process blocks until a message becomes available.



## UDP DATAGRAM (PACKET) FORMAT

* UDP packets are known as user *datagrams* .
* These ***user datagrams,*** have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).





## Source Port Number

* Port number used by process on source host with 16 bits long.
* If the source host is client (sending request) then the port number is an temporary one requested by the process and chosen by UDP.
* If the source is server (sending response) then it is well known port number.

## Destination Port Number

* Port number used by process on Destination host with 16 bits long.
* If the destination host is the server (a client sending request) then the port number is a well known port number.
* If the destination host is client (a server sending response) then port number is an temporary one copied by server from the request packet.

## Length

* This field denotes the total length of the UDP Packet (Header plus data)
* The total length of any UDP datagram can be from 0 to 65,535 bytes.

## Checksum

* UDP computes its checksum over the UDP header, the contents of the message body, and something called the pseudoheader.
* The pseudoheader consists of three fields from the IP header—protocol number, source IP address, destination IP address plus the UDP length field.

## Data

* Data field defines tha actual payload to be transmitted.
* Its size is variable**.**

## UDP SERVICES

**Process-to-Process Communication**

* + UDP provides process-to-process communication using socket addresses**,** a combination of IP addresses and port numbers.

## Connectionless Services

* + UDP provides a connectionless service.
  + There is no connection establishment and no connection termination .
  + Each user datagram sent by UDP is an independent datagram.
  + There is no relationship between the different user datagrams even if they are
  + coming from the same source process and going to the same destination program.
  + The user datagrams are not numbered.
  + Each user datagram can travel on a different path.

## Flow Control

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

## Error Control

* + There is no error control mechanism in UDP except for the checksum.
  + This means that the sender does not know if a message has been lost or duplicated.
  + When the receiver detects an error through the checksum, the user datagram is silently discarded.
  + The lack of error control means that the process using UDP should provide for this service, if needed.

## Checksum

* + UDP checksum calculation includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer.
  + The pseudoheader is the part of the header in which the user datagram is to be encapsulated with some fields filled with 0s.

##### Optional Inclusion of Checksum

* + - The sender of a UDP packet can choose not to calculate the checksum.
    - In this case, the checksum field is filled with all 0s before being sent.
    - In the situation where the sender decides to calculate the checksum, but it happens that the result is all 0s, the checksum is changed to all 1s before the packet is sent.
    - In other words, the sender complements the sum two times.

## Congestion Control

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

## Encapsulation and Decapsulation

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

## Queuing

* + In UDP, queues are associated with ports.
  + At the client site, when a process starts, it requests a port number from the operating system.
  + Some implementations create both an incoming and an outgoing queue associated with each process.
  + Other implementations create only an incoming queue associated with each process.

## Multiplexing and Demultiplexing

* + In a host running a transport protocol suite, there is only one UDP but possibly several processes that may want to use the services of UDP.
  + To handle this situation, UDP multiplexes and demultiplexes.

## APPLICATIONS OF UDP

* + UDP is used for management processes such as SNMP.
  + UDP is used for route updating protocols such as RIP.
  + UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software
  + UDP is suitable for a process with internal flow and error control mechanisms such as Trivial File Transfer Protocol (TFTP).
  + UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
  + UDP is normally used for interactive real-time applications that cannot tolerate uneven delay between sections of a received message.

# 5. TRANSMISSION CONTROL PROTOCOL (TCP)

* TCP is a reliable, connection-oriented, byte-stream protocol.
* TCP guarantees the reliable, in-order delivery of a stream of bytes. It is a full-duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction.
* TCP includes a flow-control mechanism for each of these byte streams that allow the receiver to limit how much data the sender can transmit at a given time.
* TCP supports a demultiplexing mechanism that allows multiple application programs on any given host to simultaneously carry on a conversation with their peers.
* TCP also implements congestion-control mechanism. The idea of this mechanism is to prevent sender from overloading the network.
* Flow control is an end to end issue, whereas congestion control is concerned with how host and network interact.

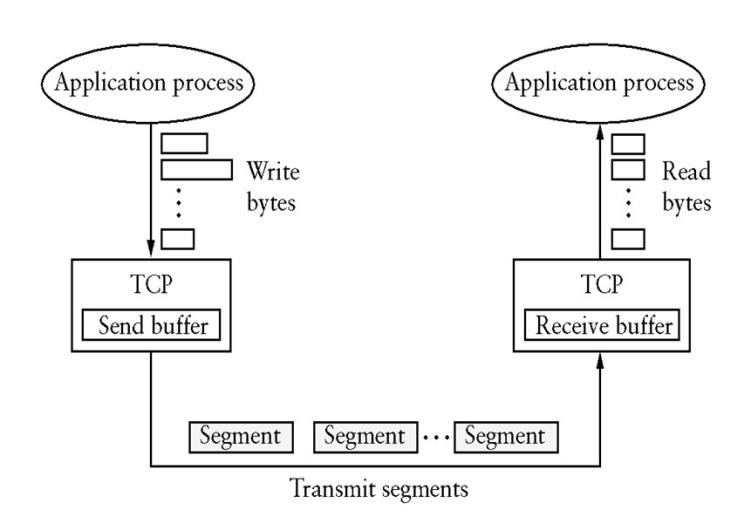
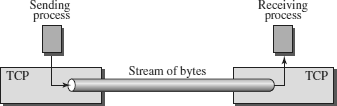
## TCP SERVICES

#### Process-to-Process Communication

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

#### Stream Delivery Service

* + TCP is a stream-oriented protocol.
  + TCP allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.
  + TCP creates an environment in which the two processes seem to be connected by an imaginary “tube” that carries their bytes across the Internet.
  + The sending process produces (writes to) the stream and the receiving process consumes (reads from) it.



#### Full-Duplex Communication

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

#### Multiplexing and Demultiplexing

TCP performs multiplexing at the sender and demultiplexing at the receiver.

#### Connection-Oriented Service

* + TCP is a connection-oriented protocol.
  + A connection needs to be established for each pair of processes.
  + When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:

1. The two TCP’s establish a logical connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

#### Reliable Service

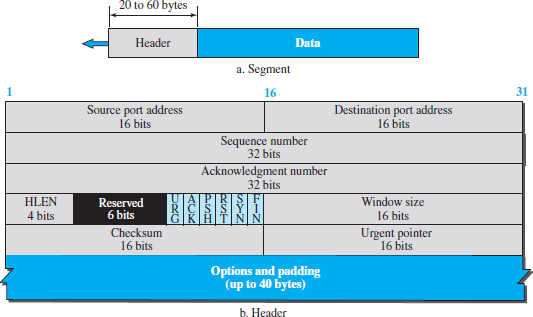
* + TCP is a reliable transport protocol.
  + It uses an acknowledgment mechanism to check the safe and sound arrival of data.

## TCP SEGMENT

* + A packet in TCP is called a segment.
  + Data unit exchanged between TCP peers are called ***segments*.**
  + A TCP segment encapsulates the data received from the application layer.
  + The TCP segment is encapsulated in an IP datagram, which in turn is encapsulated in a frame at the data-link layer.

11

* + TCP is a byte-oriented protocol, which means that the sender writes bytes into a TCP connection and the receiver reads bytes out of the TCP connection.
  + TCP does not, itself, transmit individual bytes over the Internet.
  + TCP on the source host buffers enough bytes from the sending process to fill a reasonably sized packet and then sends this packet to its peer on the destination host.
  + TCP on the destination host then empties the contents of the packet into a receive buffer, and the receiving process reads from this buffer at its leisure.
  + TCP connection supports byte streams flowing in both directions.
  + The packets exchanged between TCP peers are called segments, since each one carries a segment of the byte stream.



## TCP PACKET FORMAT

* + Each TCP segment contains the header plus the data.
  + The segment consists of a header of 20 to 60 bytes, followed by data from the application program.

The header is 20 bytes if there are no options and up to 60 bytes if it contains options

**Sorce Port and Destination Port Addresses**―port number of source and destination process.

**Sequence Number**―contains sequence number, i.e. first byte of data segment.

**Acknowledgment**― byte number of segment, the receiver expects next.

**HLen**―Length of TCP header as 4-byte words.

**Flags**― contains ***six*** control bits known as flags. o **URG(Urgent)** — segment contains urgent data.

* **ACK9)Acknowledgement)** — value of acknowledgment field is valid.
* **PSH(Push)** — sender has invoked the push operation.
* **RST(Reset)** — receiver wants to abort the connection.
* **SYN(Sync)** — synchronize sequence numbers during connection establishment.
* **FIN(Finish)** — terminates the TCP connection.

**Advertised Window**―defines receiver’s window size and acts as flow control.

**Checksum**―It is computed over TCP header, Data, and pseudo header containing IP fields

(Length, SourceAddr & DestinationAddr).

**UrgPtr** ― used when the segment contains urgent data. It defines a value that must be added to the sequence number.

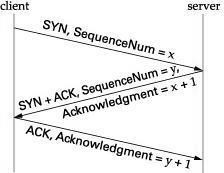
**Options -** There can be up to 40 bytes of optional information in the TCP header.

## TCP CONNECTION MANAGEMENT

* + TCP is connection-oriented.
  + A connection-oriented transport protocol establishes a logical path between the source and destination.
  + All of the segments belonging to a message are then sent over this logical path.
  + In TCP, connection-oriented transmission requires three phases: Connection Establishment, Data Transfer and Connection Termination.

## Connection Establishment

* While opening a TCP connection the two nodes(client and server) want to agree on a set of parameters.
* The parameters are the starting sequence numbers that is to be used for their respective byte streams.
* Connection establishment in TCP is a *three-way handshaking*.



1. Client sends a SYN segment to the server containing its initial sequence number (Flags

**=** SYN, SequenceNum **=** *x*)

1. Server responds with a segment that acknowledges client’s segment and specifies its initial sequence number (Flags **=** SYN **+** ACK, ACK **=** *x* **+** 1 SequenceNum **=** *y*).
2. Finally, client responds with a segment that acknowledges server’s sequence number

(Flags **=** ACK, ACK **=** *y* **+** 1).

* + The reason that each side acknowledges a sequence number that is one larger than the one sent is that the Acknowledgment field actually identifies the “next sequence number expected,”
  + A timer is scheduled for each of the first two segments, and if the expected response is not received, the segment is retransmitted.

## Data Transfer

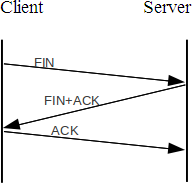
* + After connection is established, bidirectional data transfer can take place.
  + The client and server can send data and acknowledgments in both directions.
  + The data traveling in the same direction as an acknowledgment are carried on the same segment.
  + The acknowledgment is piggybacked with the data.

## Connection Termination

* Connection termination or teardown can be done in two ways :

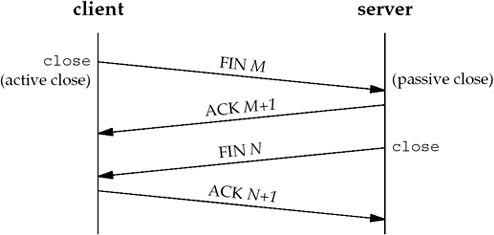
#### Three-way Shake and Four Way Handshake

***Three-way Shake****—*Both client and server close *simultaneously.*

**

* + Client sends a FIN segment.
  + The FIN segment can include last chunk of data.
  + Server responds with FIN + ACK segment to inform its closing.
  + Finally, client sends an ACK segment

***Four Way Hand Shaking****—*Client stops sending but receives data.

* + Client half-closes the connection by sending a FIN segment.
  + Server sends an ACK segment.
  + Data transfer from client to the server *stops*.
  + After sending all data, server sends FIN segment to client, which is acknowledged by the client.

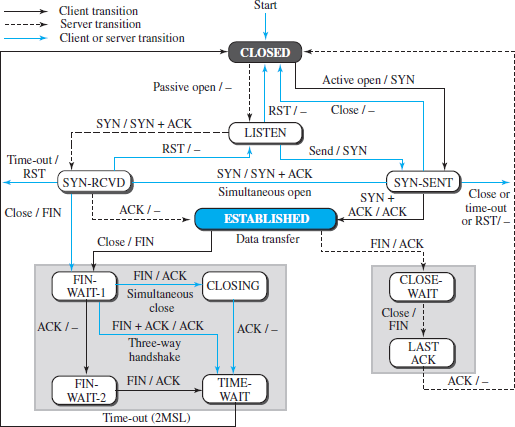
# State Transition Diagram

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

Here two FSMs used by the TCP client and server combined in one diagram.

The rounded-corner rectangles represent the states. The transition fromone state to another is shown using directed lines. Each line has two strings separatedby a slash.

**The first string** is the input, what TCP receives. **The second** is the output,what TCP sends.



**Fig: State transition diagram..[Source : Data Communications and Networking by Behrouz A. Forouzan].**

.

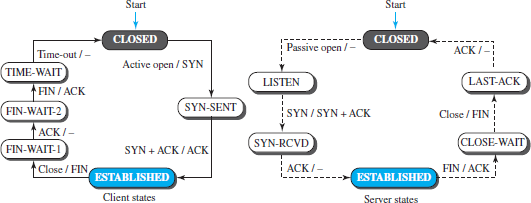
The dotted black lines in the figure represent the transition that aserver normally goes through; the solid black lines show the transitions that a client

normally goes through.

In some situations, a server transitions through asolid line or a client transitions through a dotted line. The colored lines show specialsituations.

The rounded-corner rectangle marked ESTABLISHED has twosets of states, a set for the client and another for the server, that are used for flow anderror control.

**Consider the scenario**. Figure (below) shows the state transition diagram for this scenario.



**Fig: State transition diagram .[Source : Data Communications and Networking by Behrouz A. Forouzan].**

**The client process issues** an active open command to its TCP to request a connectionto a specific socket address. TCP sends a SYN segment and moves to the **SYN-SENT**state.

After receiving the SYN +ACK segment, TCP sends an ACK segment and goes tothe **ESTABLISHED** state. Data are transferred, possibly in both directions, and acknowledged. When the client process has no more data to send, it issues a commandcalled an *active close*. The TCP sends a FIN segment and goes to the **FIN- WAIT-1** state.

When it receives the ACK segment, it goes to the **FIN-WAIT-2** state. When the clientreceives a FIN segment, it sends an ACK segment and goes to the **TIME-WAIT** state.

The client remains in this state for 2 MSL .**MSL** is the maximum time a **TCP** segment is expected to live, or stay in the network.

It is an advertisement from the **TCP** to the other side about the maximum size of a

**TCP** segment it is prepared to receive.

When the corresponding timer expires, the client goes to the **CLOSED** state.

**The server process** issues a passive open command. The server TCP goes to the**LISTEN** state and remains there passively until it receives a SYN segment.

The TCPthen sends a SYN +ACK segment and goes to the **SYN-RCVD** state, waiting for theclient to send an ACK segment. After receiving the ACK segment, TCP goes to the**ESTABLISHED** state, where data transfer can take place. TCP remains in this state until it receives a FIN segment from the client signifying that there are no more data tobe exchanged and the connection can be closed.

The server, upon receiving the FINsegment, sends all queued data to the server with a virtual EOF marker, which meansthat the connection must be closed.

It sends an ACK segment and goes to the **CLOSEWAIT**state, but postpones acknowledging the FIN segment received from the clientuntil it receives a *passive close* command from its process.

After receiving the passiveclose command, the server sends a FIN segment to the client and goes to the **LASTACK** state, waiting for the final ACK. When the ACK segment is received from the client,the server goes to the **CLOSE** state.

# Windows in TCP

TCP uses two windows (send windowand receive window) for each direction of data transfer, which means fourwindows for a bidirectional communication.

# Send Window

. The send window in TCP is like Selective-Repeat protocol,but with some differences:

1. The window size inSR is the number of packets, but the window size in TCP is the number of bytes.
2. TCP can store datareceived from the process and send them later, but we assume that the sendingTCP is capable of sending segments of data as soon as it receives them from itsprocess.
3. The TCP protocol uses only one timer.

# Receive window

The receive window size decide the number of bytes that the receive window can accept from the sender beforebeing crowded (flow control).

The acknowledgment mechanism in TCP isa cumulative acknowledgment announcing the next expected byte to receive.

# Congestion Avoidance

TCP impose some methods to control congestiononce it happens, instead of trying to avoid congestion.

It is a prevention mechanism while congestion control is a recovery mechanism.

# DECbit

DECbit means destination experiencing congestion bit.

This mechanism was developed for use on the **Digital Network Architecture**

(DNA), a connectionless network with a connection-oriented transport protocol.

This mechanism could, therefore, also be applied to

TCP and IP.It split theresponsibility for congestion control between the routers and the endnodes.

Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur. This notification is implemented **by setting a binary congestion bit** in the packets that flow through the router, hence the name DECbit.

The destination host thencopies this congestion bit into the ACK it sends back to the source. Finally,the source adjusts its sending rate so as to avoid congestion.

A router setsthis bit in a packet if its average queue length is greater than or equalto 1 at the time the packet arrives.

This average queue length is measuredover a time interval as , the last busy+idle cycle, plus thecurrent busy cycle.

Thesource records how many of its packets has set the congestion bit.

If less than 50% of the packetshad the bit set, then the source increases its congestion window byone packet. If 50% or more of the last window’of packets had thecongestion bit set, then the source decreases its congestion window to0.875 times the previous value.

# Random Early Detection (RED)

REDprovide congestion control at the router for TCP flows.

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

Properties of RED

1. RED drops packets before queue is full, in the hope of reducing the rates of some flows.
2. Drops packet for each flow roughly in proportion to its rate. 3.Red maintains average queue length.
3. Random drops desynchronize the TCP sources.
4. RED calculates the average que length using a weighted running average.

**The Formula** is as follows.

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

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

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

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

ifAvgLen≤ MinThreshold Then queue the packet

ifMinThreshold<AvgLen<MaxThreshold calculate probability P

drop the arriving packet with probability P ifMaxThreshold≤AvgLength

!drop the arriving packet

If the average queue length is smaller than the lower threshold, no action is taken, and if the average queue length is larger than the upper threshold, then the packet is always dropped.

If the average queue length isbetween the two thresholds, then the newly arriving packet is droppedwith some probability P.

# Congestion control in TCP

Congestion in a network may occur if the load on thenetwork—the number of packets sent to the networkis greater than the capacity of the network.

Congestioncontrol refers to the mechanisms and techniques tocontrol the congestion.

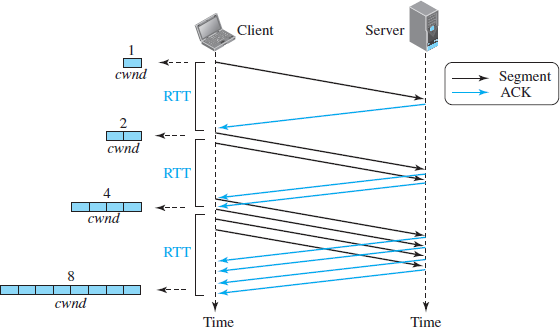
TCP uses a congestion window and a congestion policy that avoid congestion.

If the network cannot able to deliver the data as fast as it is created by the sender, it must tell the sender to slow down.

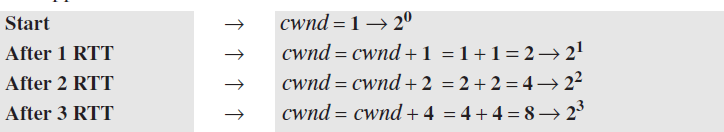
# Congestion policy in TCP

* 1. Slow Start Phase: starts slowly increment is exponential to threshold
  2. Congestion Avoidance Phase: After reaching the threshold increment is by 1.
  3. Congestion Detection Phase: Sender goes back to Slow start phase or Congestion avoidance phase.

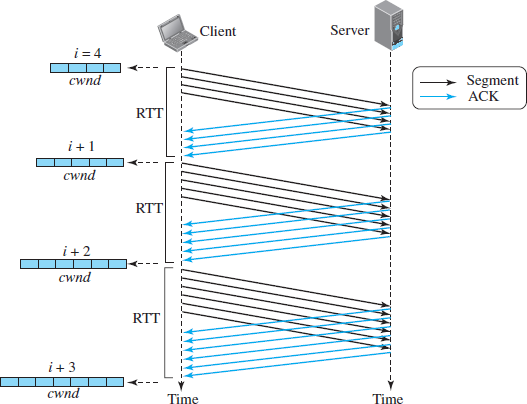
**Slow Start , Exponential Increase –** In this phase after every RTT( round trip time) the congestion window size increments exponentially.



**Fig: Slow start congestion control .[Source : Data Communications and Networking by Behrouz A. Forouzan].**



# Congestion Avoidance Phase : additive increase



**Fig:TCP congestion avoidance .[Source : Data Communications and Networking by Behrouz A. Forouzan].**

To avoid congestion before it happens.

This phase starts after the threshold value is denoted *as ssthresh. The size of cwnd*(congestion window) increases additive.

When the size ofthe congestion window reaches the slow-start threshold in the case where*cwnd =i*, the slow-start phase stops and the additive phase begins. In this algorithm, each time the whole “window” of segments is acknowledged, the size of the congestion windowis increased by one.

After each RTT cwnd = cwnd + 1.

**Congestion Detection Phase : multiplicative decrement –** If congestion occurs, the congestion window size is decreased.

The only way a sender knows that congestion has occurred is the need to retransmit a segment. Retransmission is needed to recover a missing packet which is dropped by a router due to congestion.

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

* **Case 1 : Retransmission due to Timeout .** In this case congestion possibility is high.
  + 1. ssthresh ( Slow start threshold) is reduced to half of the current window size.
    2. setcwnd = 1
    3. start with slow start phase again.

# Case 2 : Retransmission due to 3 Acknowledgement Duplicates .

In this case congestion possibility is less.

* + - 1. ssthresh value reduces to half of the current window size.
      2. set cwnd= ssthresh
      3. start with congestion avoidance phase