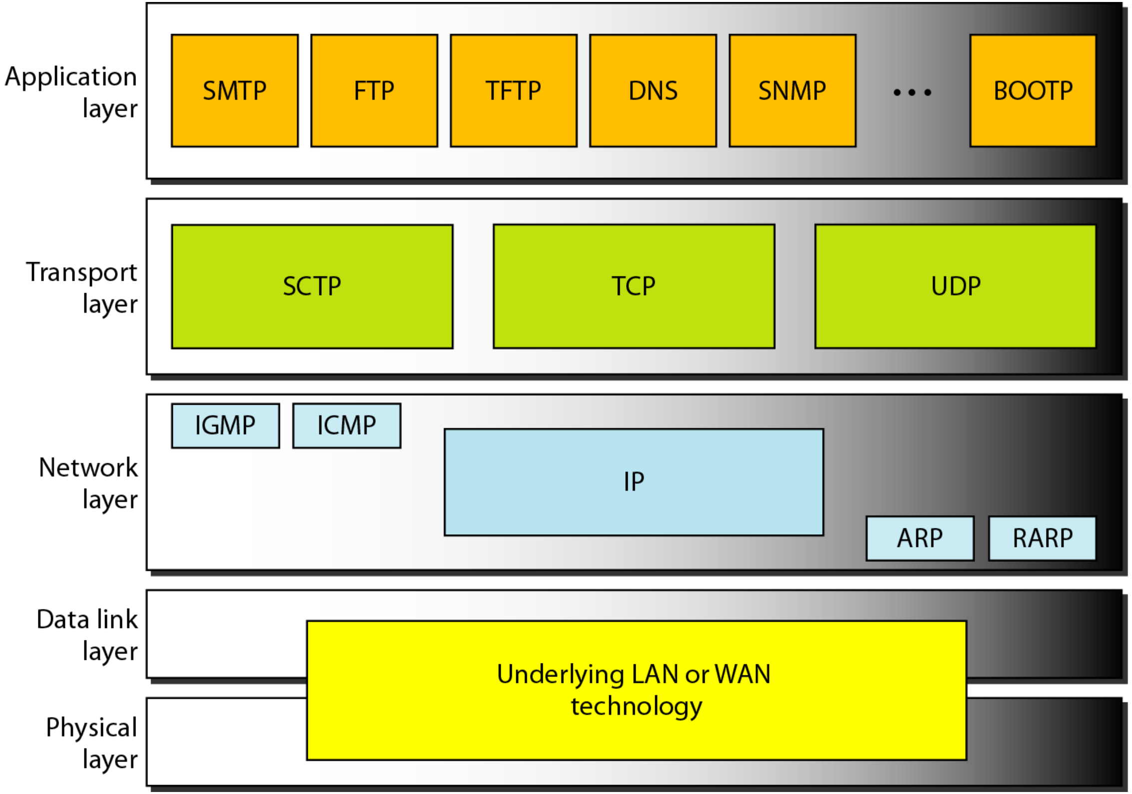
GLA UNIVERSITY

Computer Network

By:
Dr. Ankush Agarwal

TRANSPORT LAYER

Introduction



TCP

- TCP is a connection-oriented protocol
- It creates a virtual connection between two TCPs to send data
- In addition, TCP uses flow and error control mechanisms at the transport level

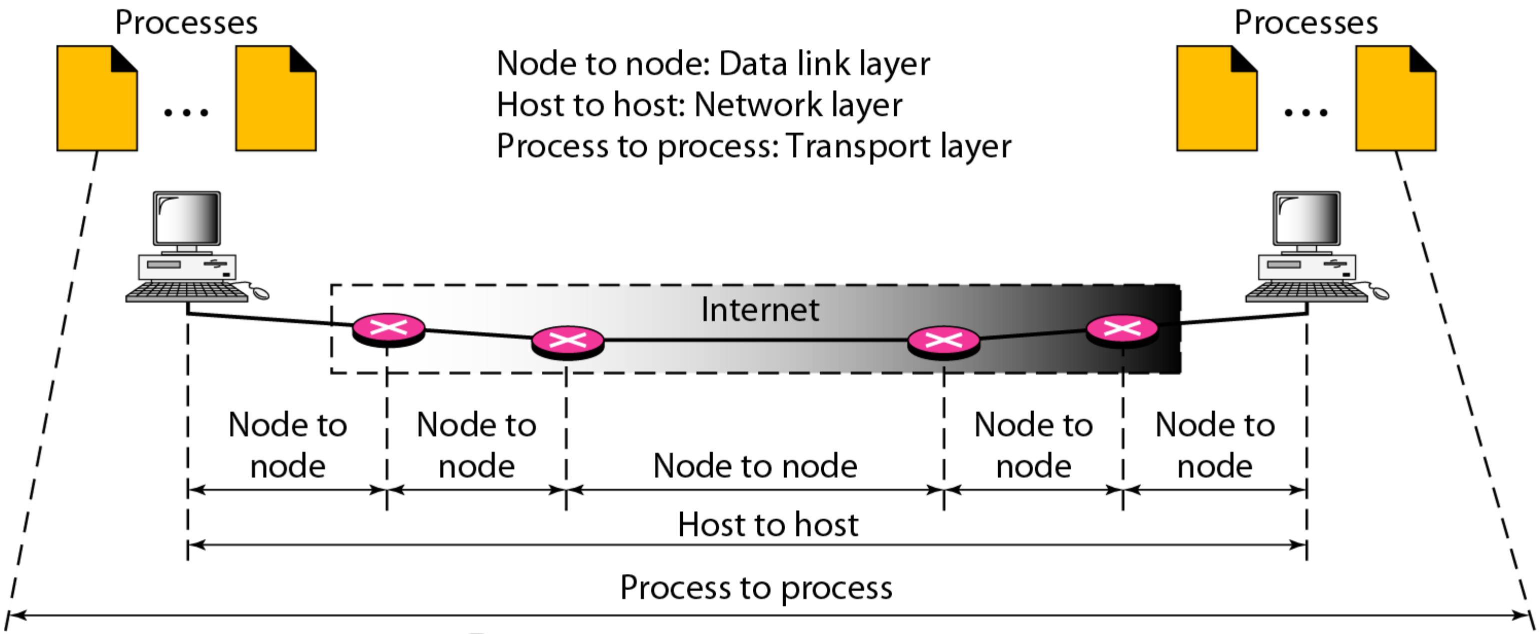
Why we need transport layer?

- Network layer is responsible for host to host communication (IP address)
- There are several network application running in OS
- How did NIC knows which packet belongs to which process/application?

Why we need transport layer?

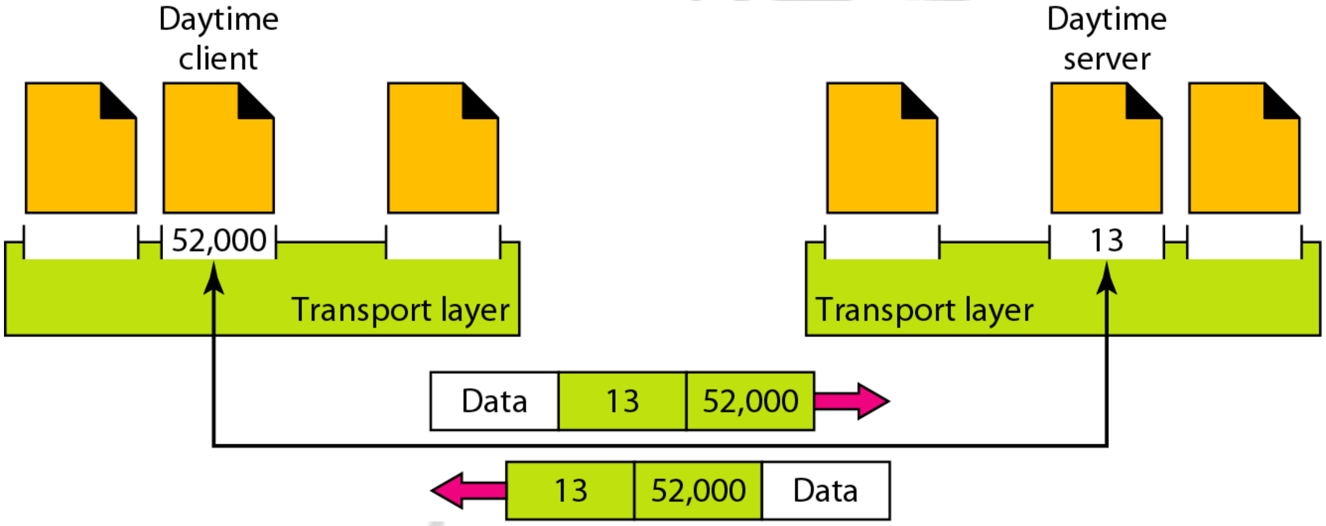
- The transport layer is responsible for process-to-process delivery
  - the delivery of a packet, part of a message, from one process to another
- It provides logical communication between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer

Types of data deliveries: Internet Stack

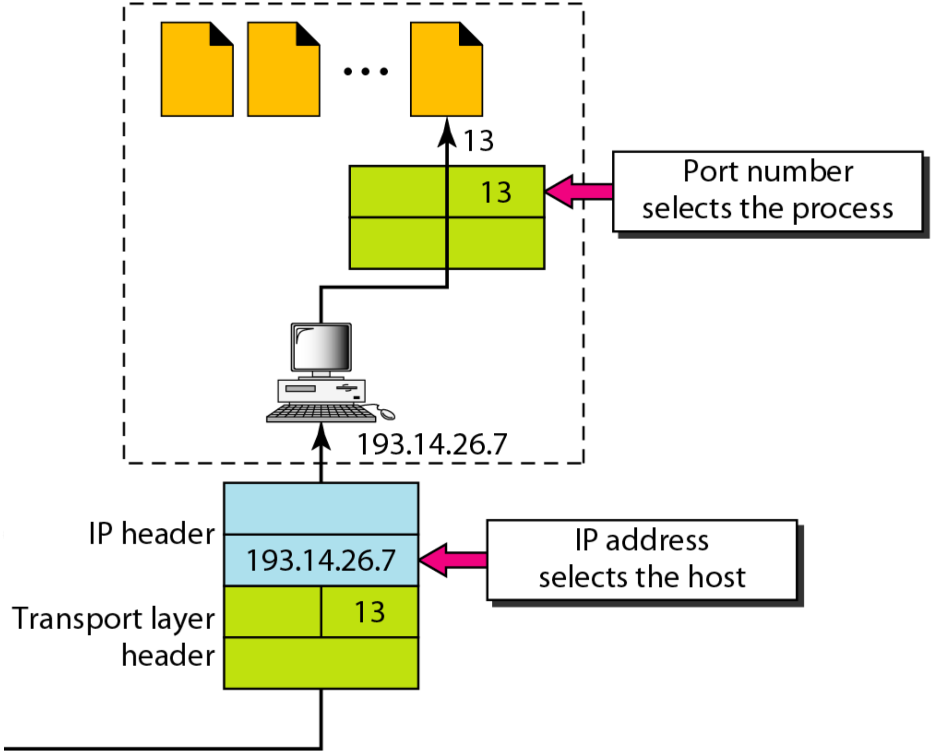


How it delivers messages to specific process

Using Port address (16 bit)
Ranges from 0 to 65,535



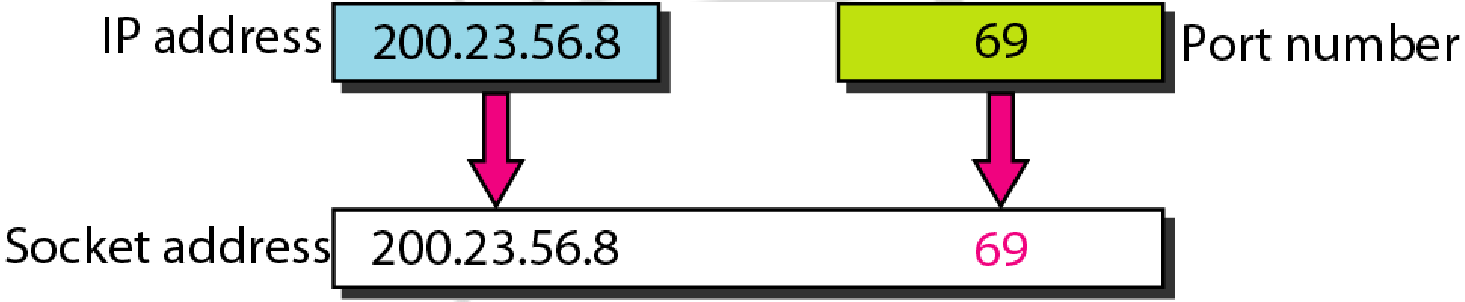
IP addresses v/s port numbers



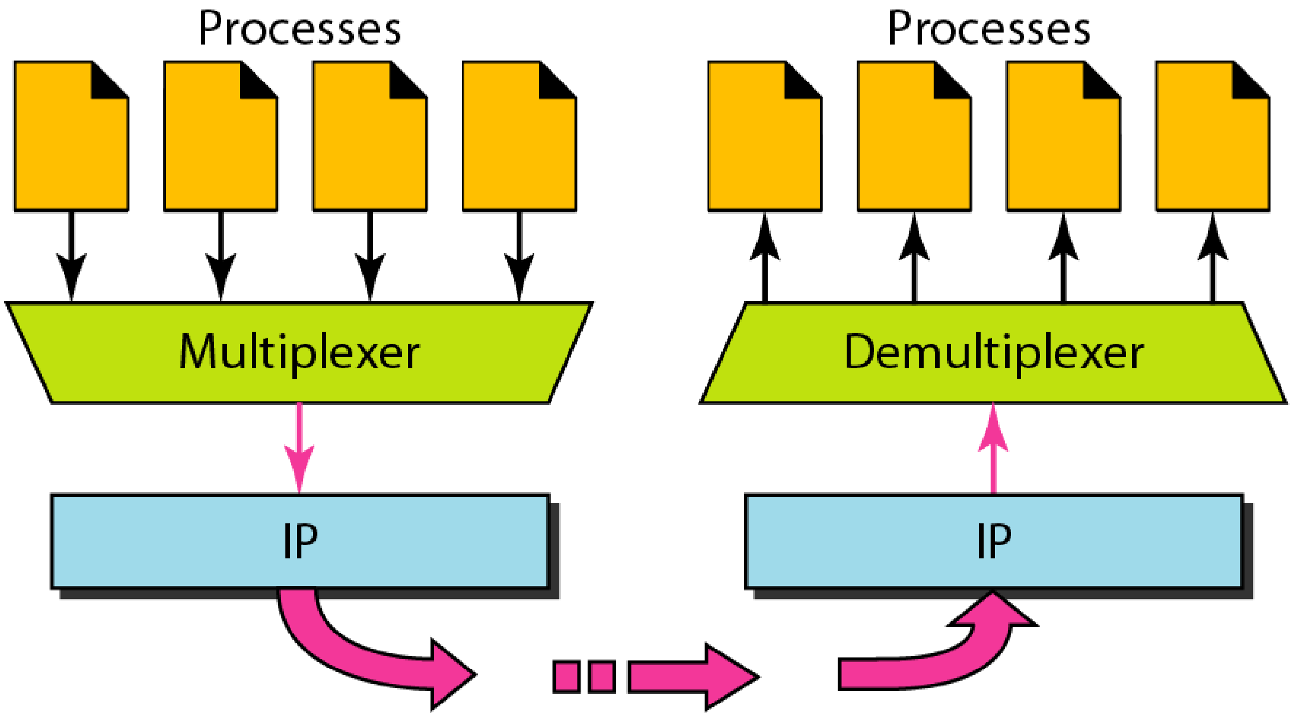
Socket address (IP address + Port Address)

What is the use of socket?

- The socket mechanism provides a means of inter-process communication (IPC)
- Socket is basically an API for enabling communication between two end points
- A socket is one endpoint of a two way communication link between two programs running on the network



Multiplexing and De-multiplexing



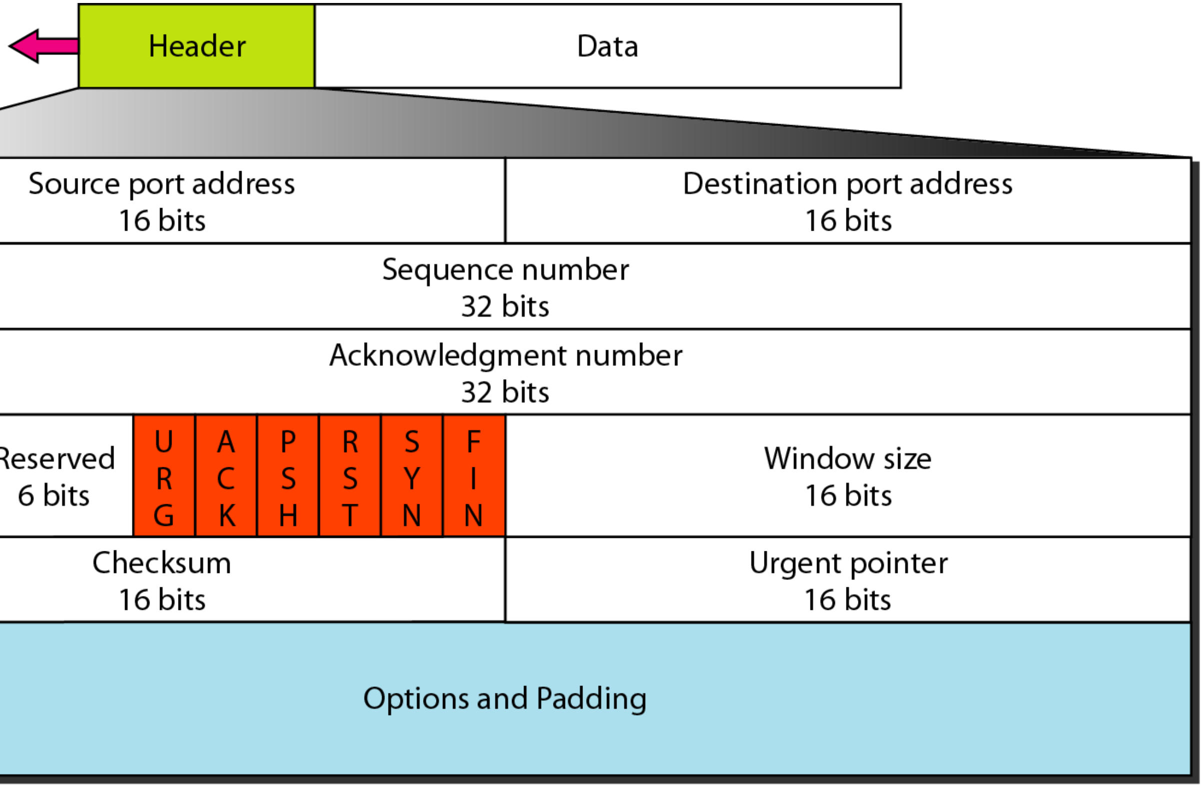
TCP segments

- The bytes of data being transferred in each connection are numbered by TCP
- The numbering starts with an arbitrarily generated number

Eg:

- Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

TCP segment format



Control field

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push

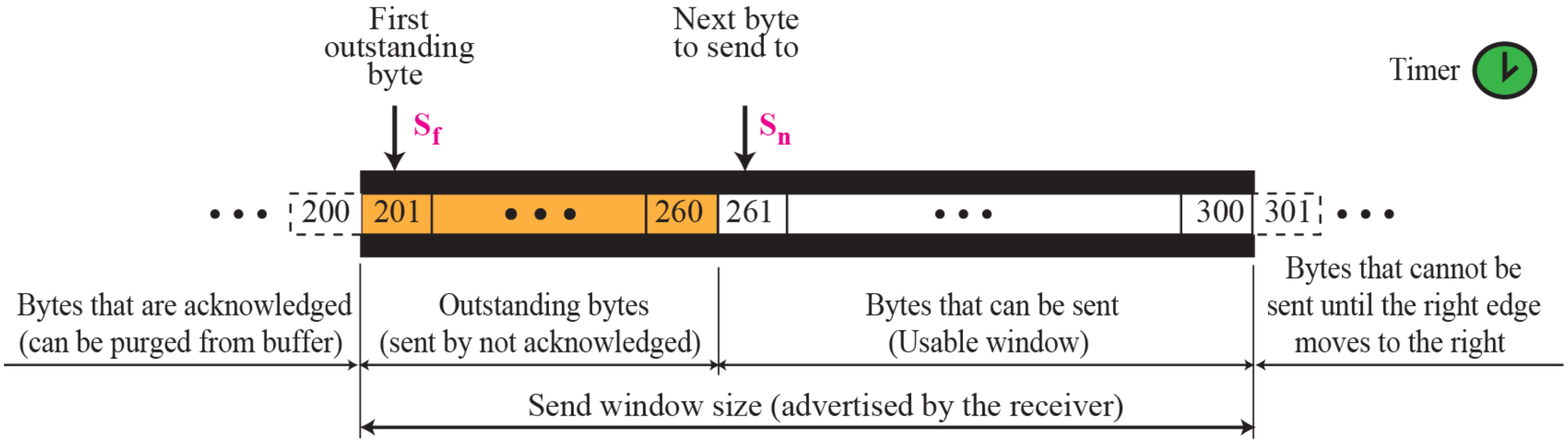
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection

URG	ACK	PSH	RST	SYN	FIN

TCP Window Management

- TCP uses two windows (send window and receive window) for each direction of data transfer, which means four windows for a bidirectional communication
- To make the discussion simple, we make an assumption that communication is only unidirectional
- The bidirectional communication can be inferred using two unidirectional communications with piggybacking

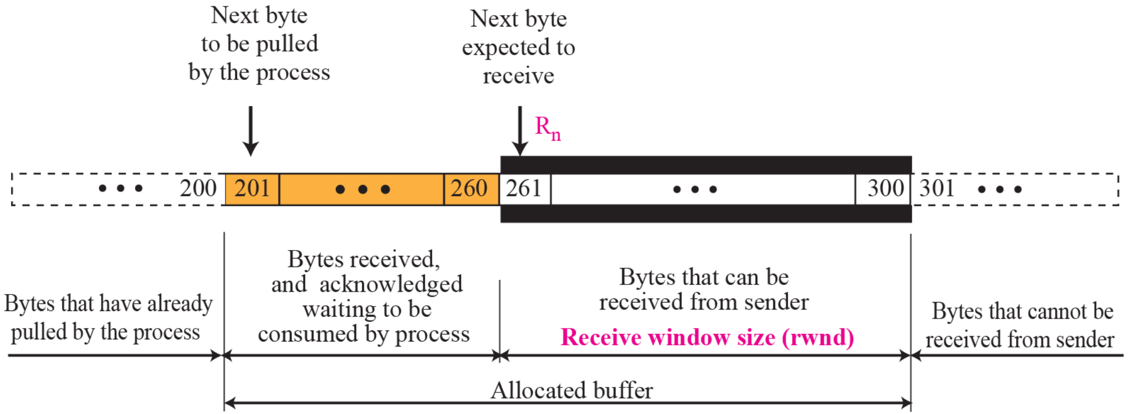
Send window in TCP



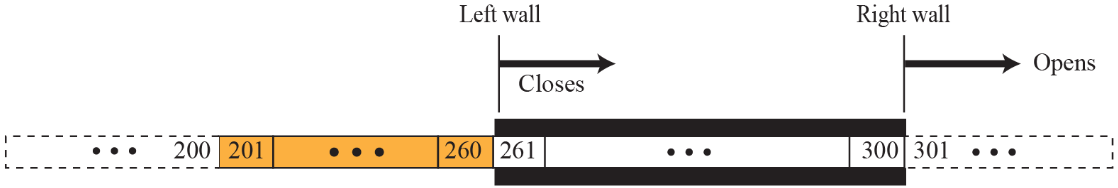
a. Send window

b. Opening, closing, and shrinking send window

Receive window in TCP



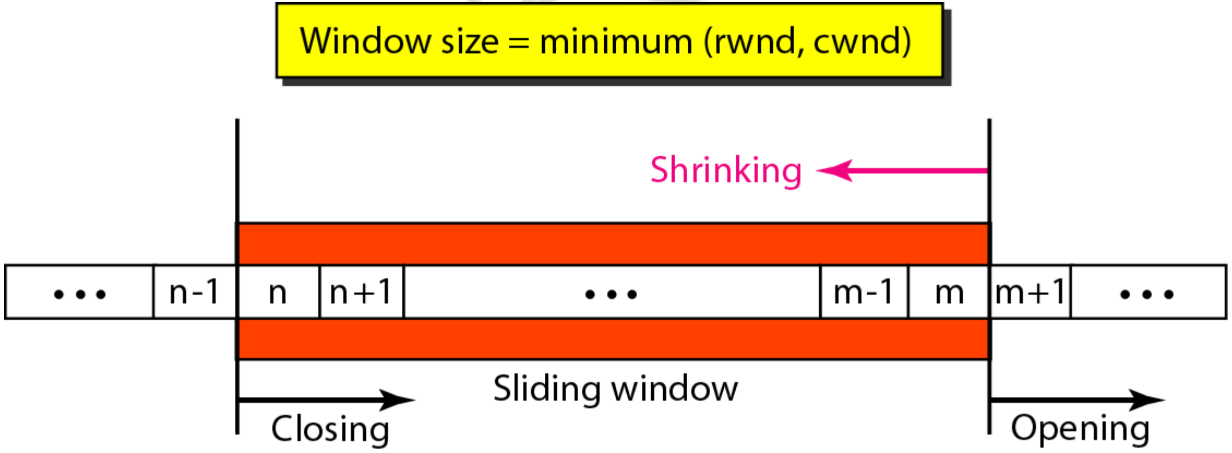
a. Receive window and allocated buffer



b. Opening and closing of receive window

Sliding window

- 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



Example

- What is the value of the receiver window (rwnd) for host A if the receiver host B has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?

Solution

- The value of rwnd = 5000 − 1000 = 4000
- Host B can receive only 4000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A

Example

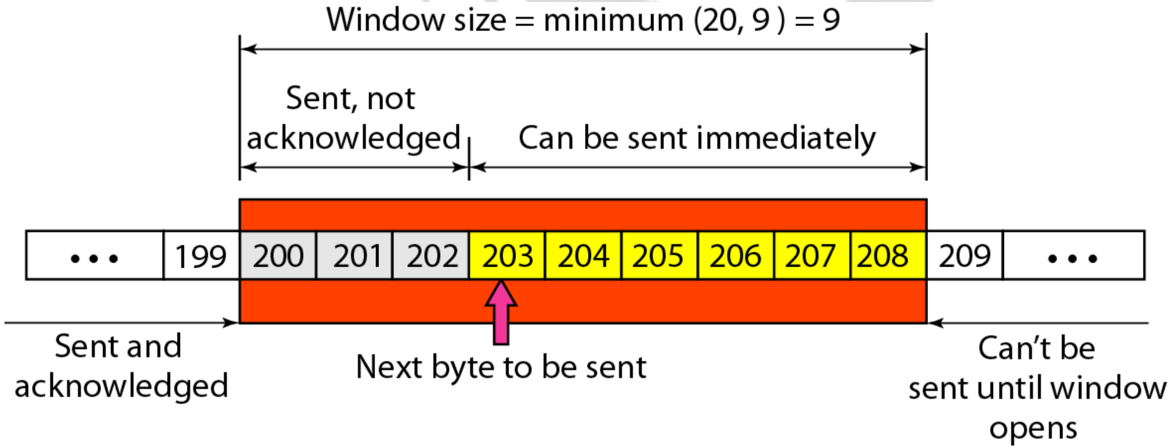
- What is the size of the window for host A if the value of rwnd is 3000 bytes and the value of cwnd is 3500 bytes?

Solution

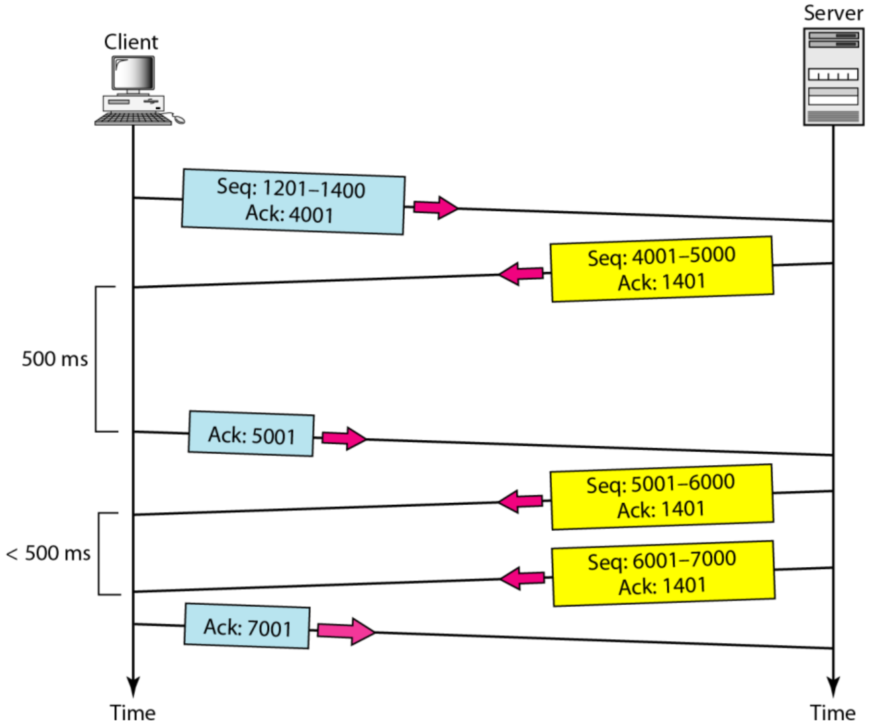
- The size of the window is the smaller of rwnd and cwnd, which is 3000 bytes

Example

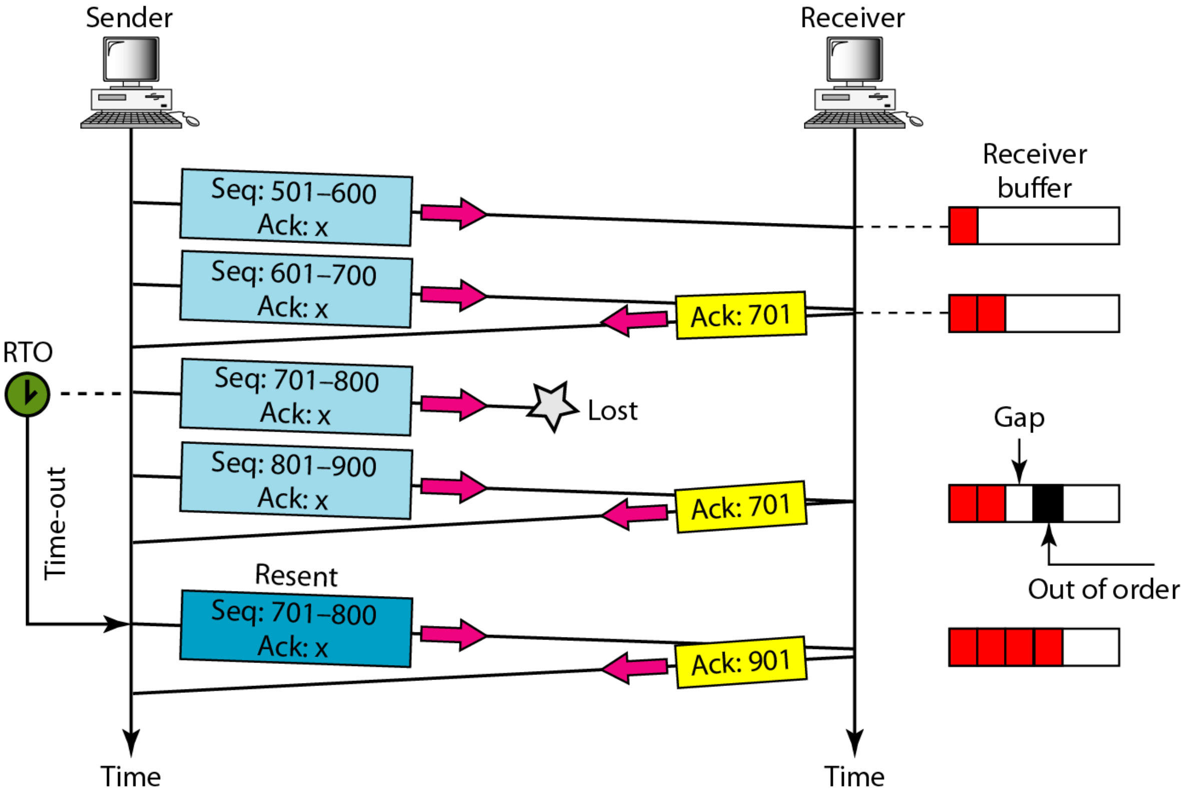
- The sender has sent bytes up to 202. We assume that cwnd is 20. The receiver has sent an acknowledgment number of 200 with an rwnd of 9 bytes. The size of the sender window is the minimum of rwnd and cwnd, or 9 bytes. Bytes 200 to 202 are sent, but not acknowledged. Bytes 203 to 208 can be sent without worrying about acknowledgment. Bytes 209 and above cannot be sent.



Normal operation

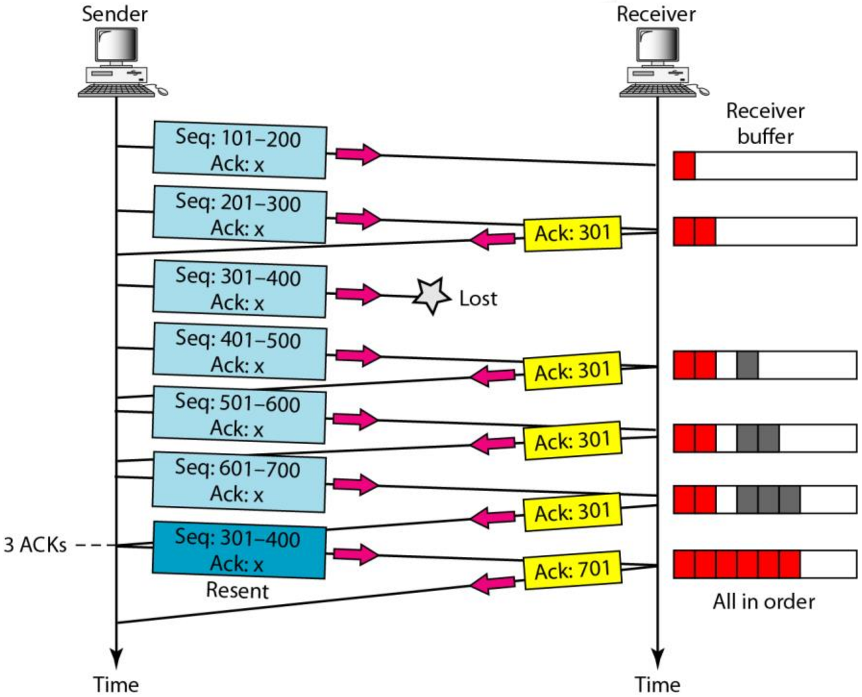


Lost segment



Fast retransmission

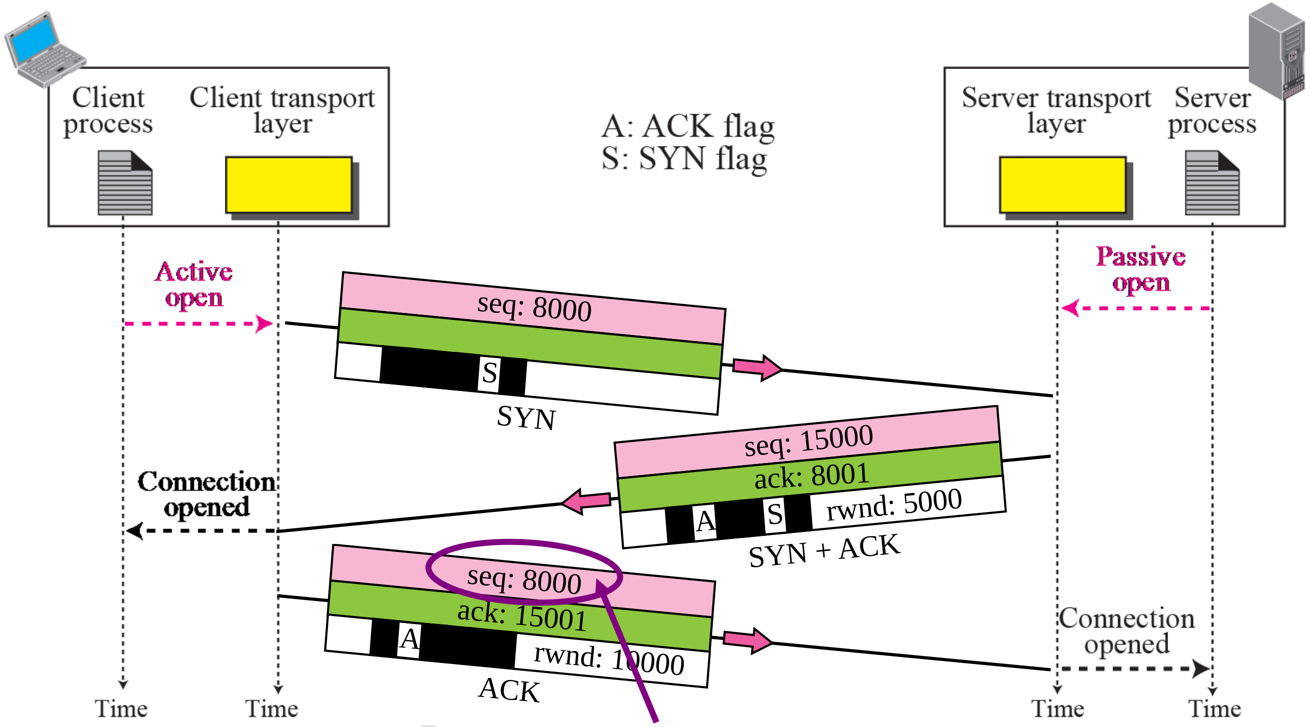
- The receiver TCP delivers only ordered data to the process



TCP connection

- TCP is connection-oriented and it establishes a virtual path between the source and destination
- TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented
- All of the segments belonging to a message are sent over this virtual path
- The point is that a TCP connection is virtual, not physical
- TCP operates at a higher level and uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself
- If a segment is lost or corrupted, it is retransmitted

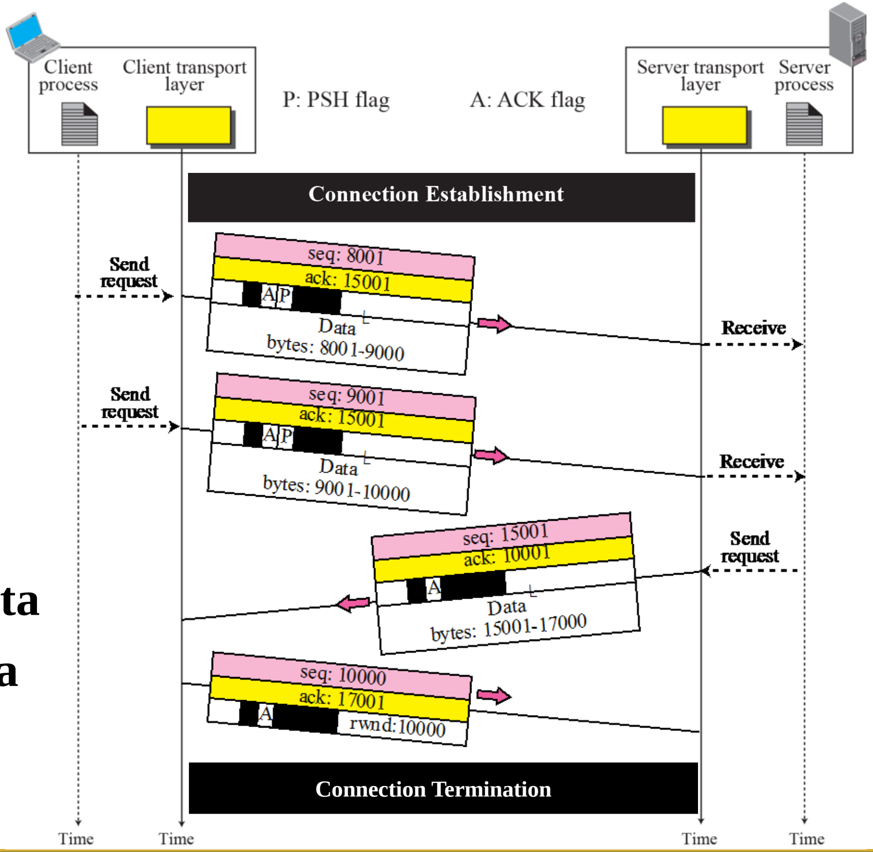
Connection establishment using three-way handshake



Means "no data" !

seq: 8001 if piggybacking

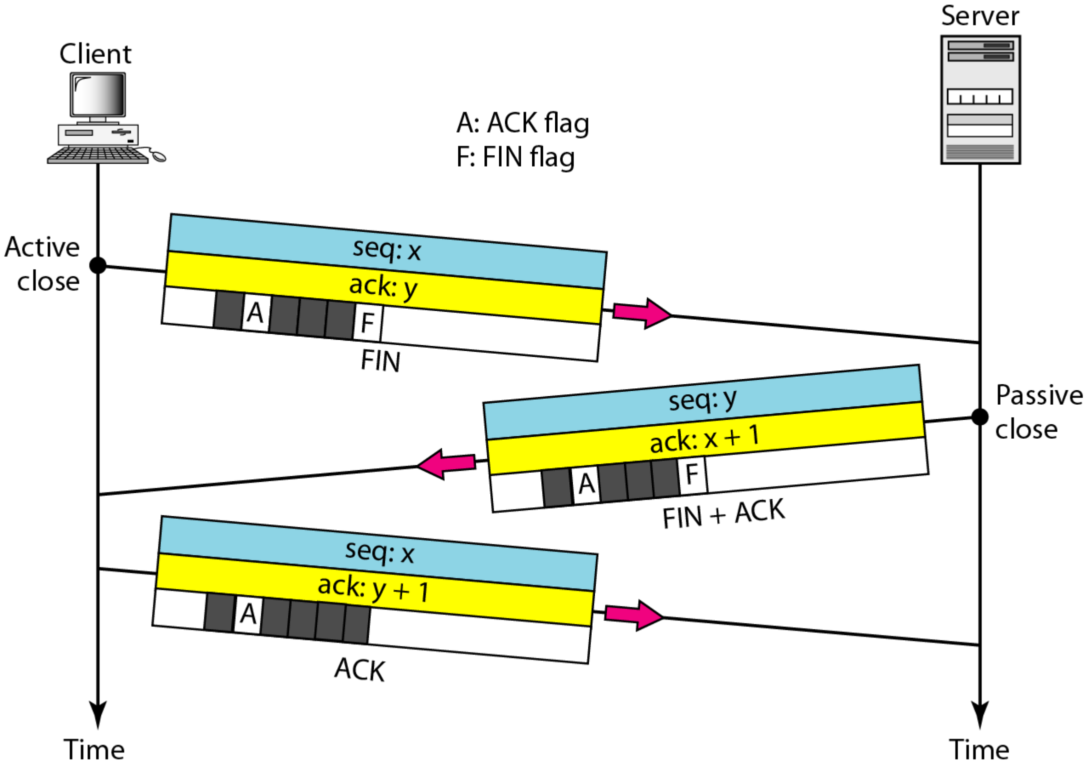
Data transfer



Pushing data

Urgent data

Connection termination using three-way handshaking



Congestion control

- Congestion control in TCP is based on both open loop and closed-loop mechanisms
- TCP uses a congestion window and a congestion policy that avoid congestion and detect

Congestion control

- When too many packets are present in the subnet, performance degrades, this situation is called congestion
- As traffic increases too far, the routers are no longer able to cope and they begin losing packets
- At very high traffic, performance collapses completely and almost no packets are delivered

- Reasons of Congestion:
  - Slow Processor
  - High stream of packets sent from one of the sender
  - Insufficient memory
  - Low bandwidth lines

- Congestion control: make sure the subnet is able to carry the offered traffic
- Congestion control and flow control are often confused but both helps reduce congestion

Congestion control

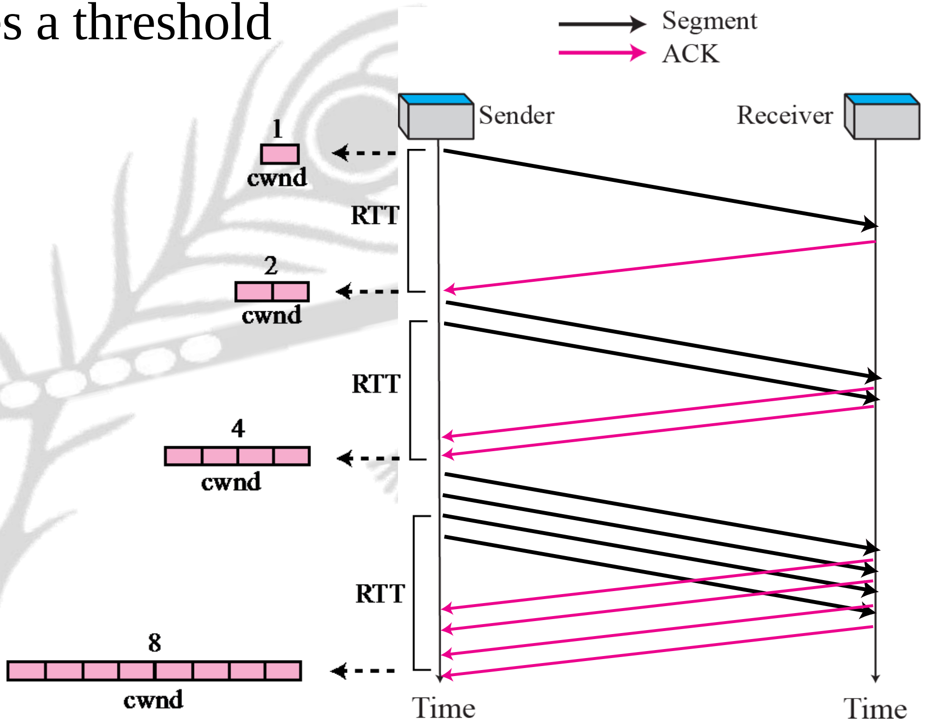
- Knowledge of congestion will cause the hosts to take appropriate action to reduce the congestion
- Dividing all algorithms into
  - open loop
    - They further divide the open loop algorithms into ones that act at the source versus the destination
  - closed loop
    - The closed loop algorithms are also divided into two subcategories:
      - In explicit feedback algorithms, packets are sent back from the point of congestion to warn the source
      - In implicit algorithms, the source deduces the existence of congestion by making local observations, such as the time needed for acknowledgements to come back
- The presence of congestion means that the load is (temporarily) greater than the resources can handle

Congestion control in TCP

- Slow Start
- Additive Increase (Congestion Avoidance)
- Multiplicative decrease

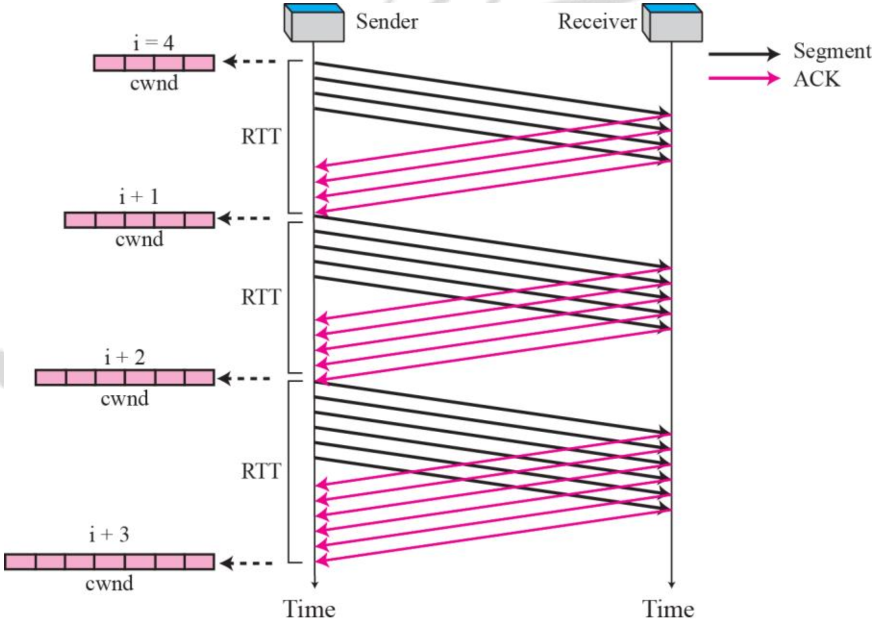
Slow start, exponential increase

- In the slow start algorithm, the size of the congestion window increases exponentially until it reaches a threshold

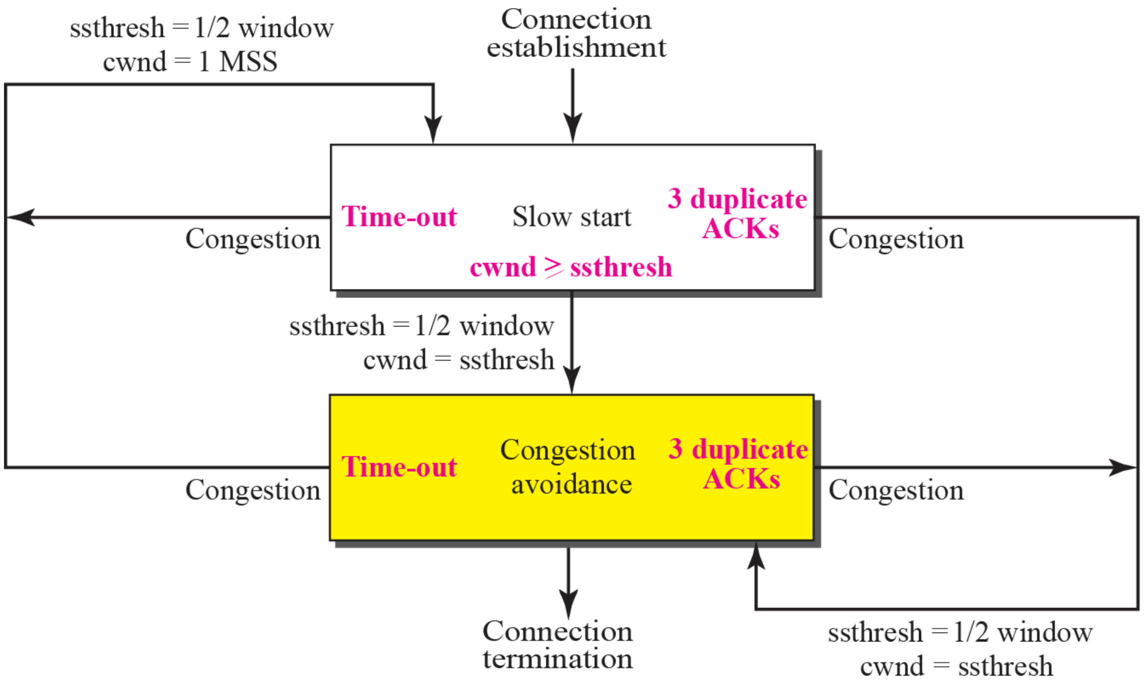


Congestion avoidance, additive increase

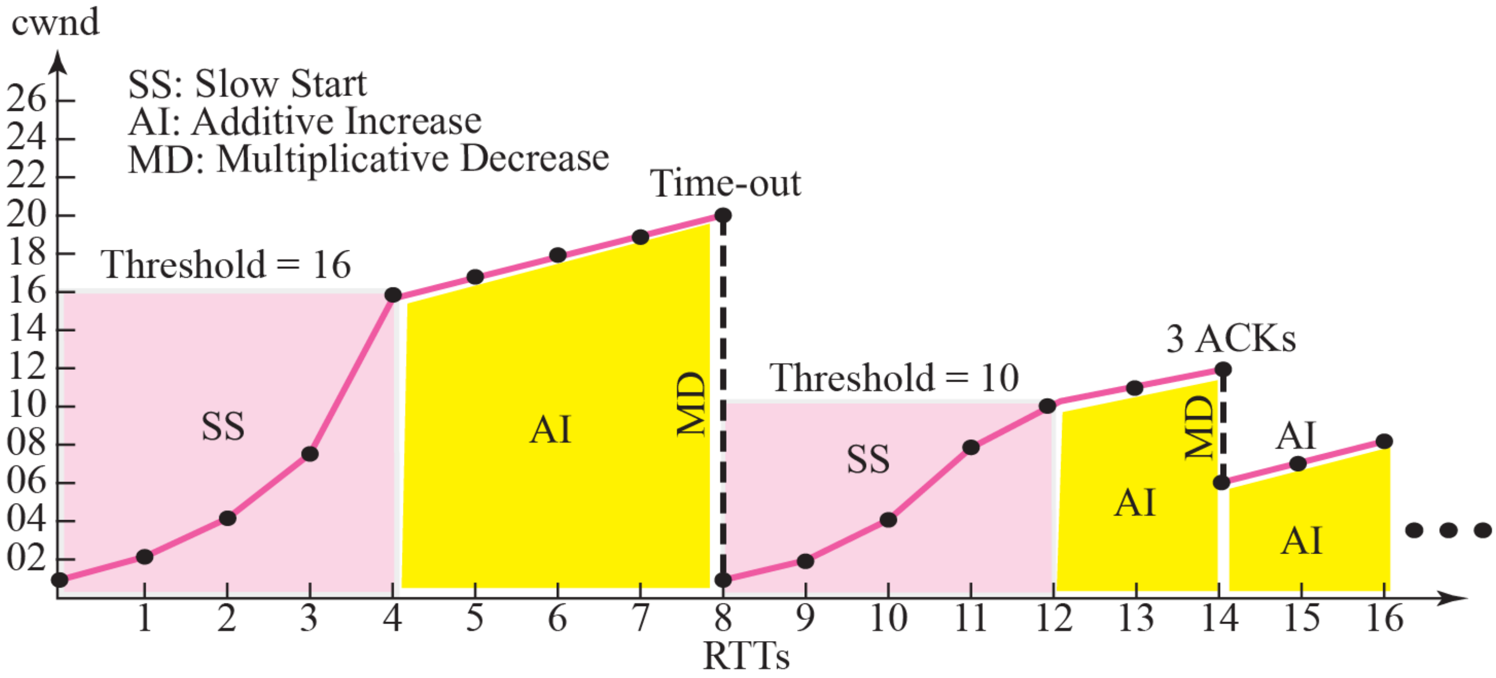
- In the congestion avoidance algorithm the size of the congestion window increases additively until congestion is detected



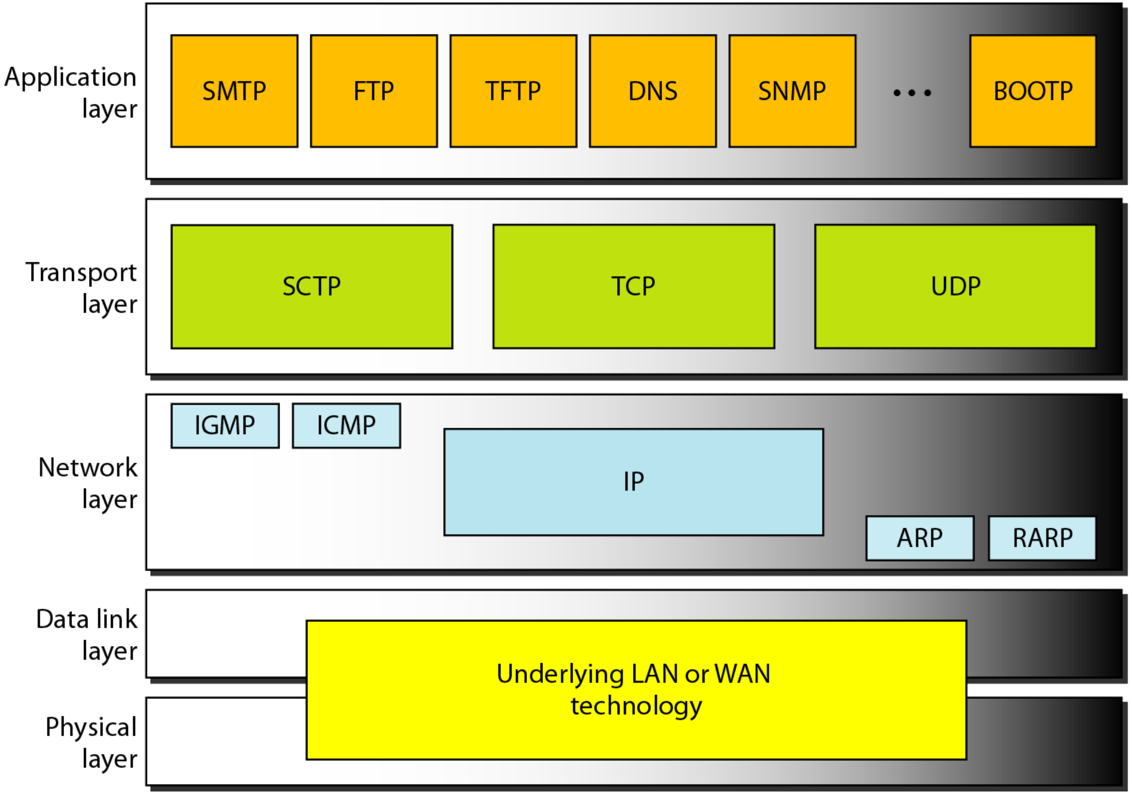
TCP Congestion policy summary



Congestion example



UDP


Application layer
SMTP
FTP
TFTP
DNS
SNMP
• • •
BOOTP
Transport layer
SCTP
TCP
UDP
Network layer
IGMP
ICMP
IP
ARP
RARP
Data link layer
Physical layer
Underlying LAN or WAN technology

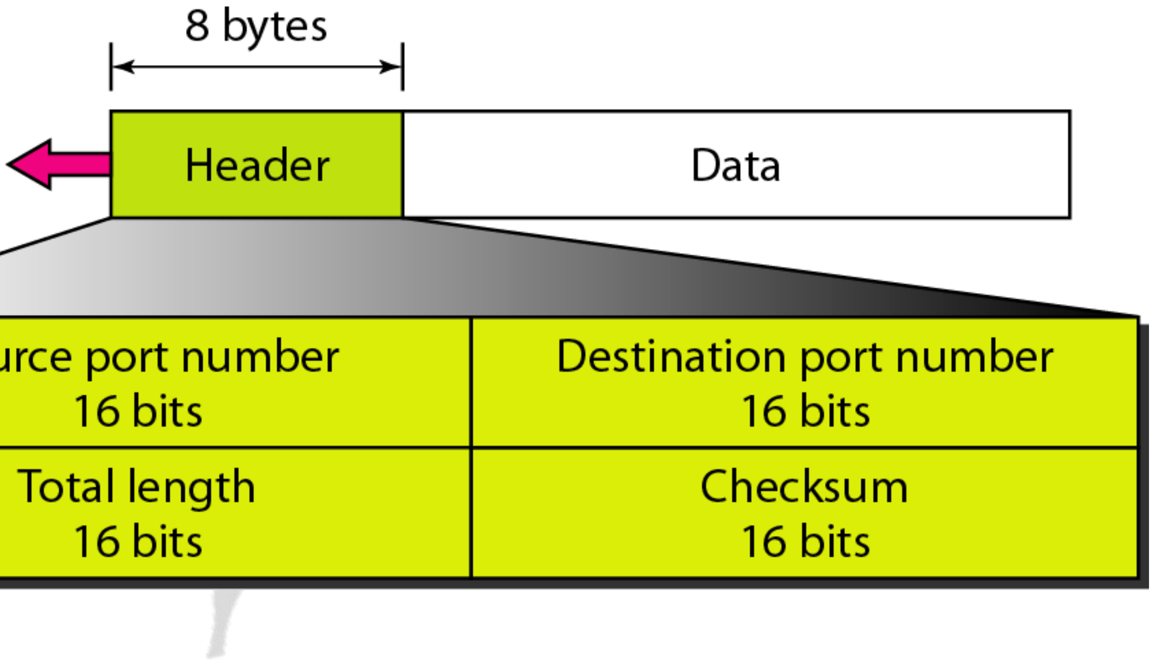
UDP

- UDP provides
  - best effort delivery
  - Connectionless
  - Unreliable
  - Out of order delivery
- TCP
  - Reliable
  - In-order delivery
  - Congestion control
  - Flow control
  - Connection setup

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

UDP format


8 bytes
Header
Data
Source port number
16 bits
Destination port number
16 bits
Total length
16 bits
Checksum
16 bits

Pseudoheader for checksum calculation