Transport Protocols

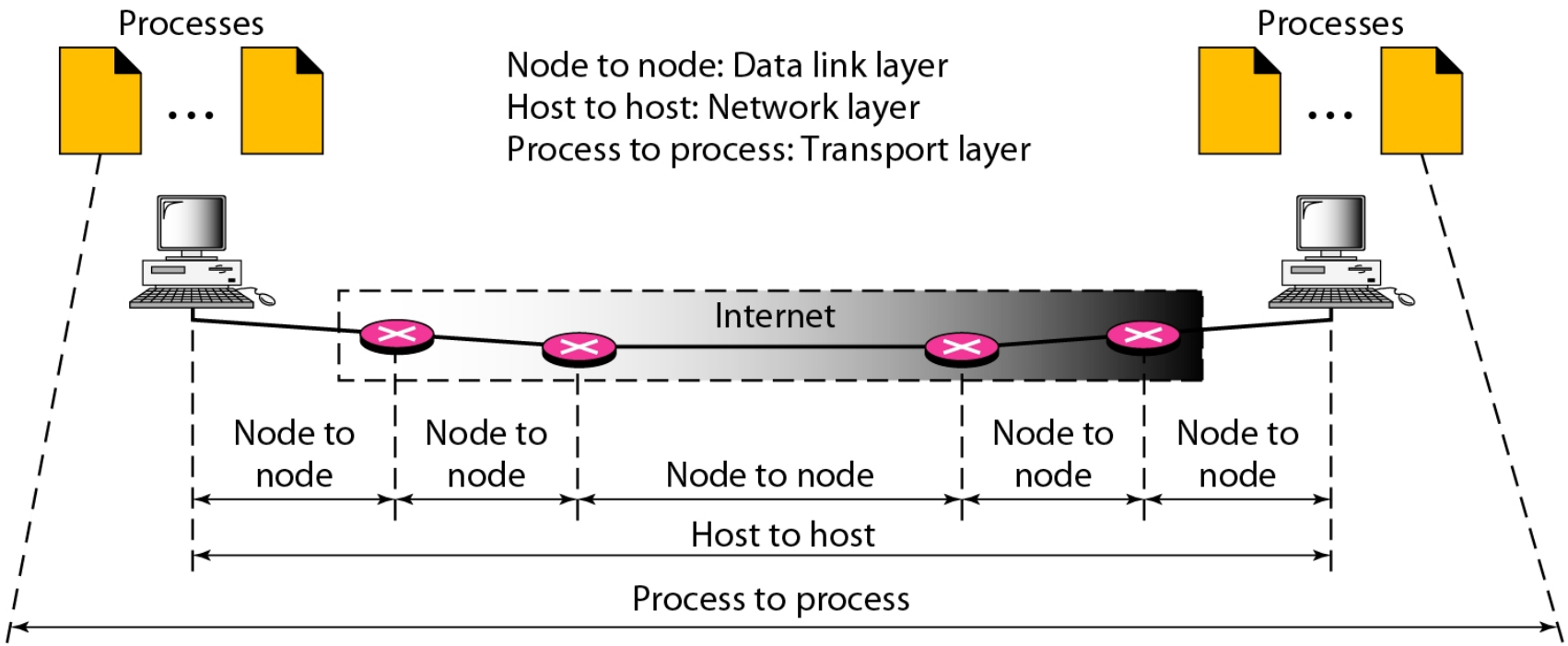
Mridul Sankar Barik

Jadavpur University

2023

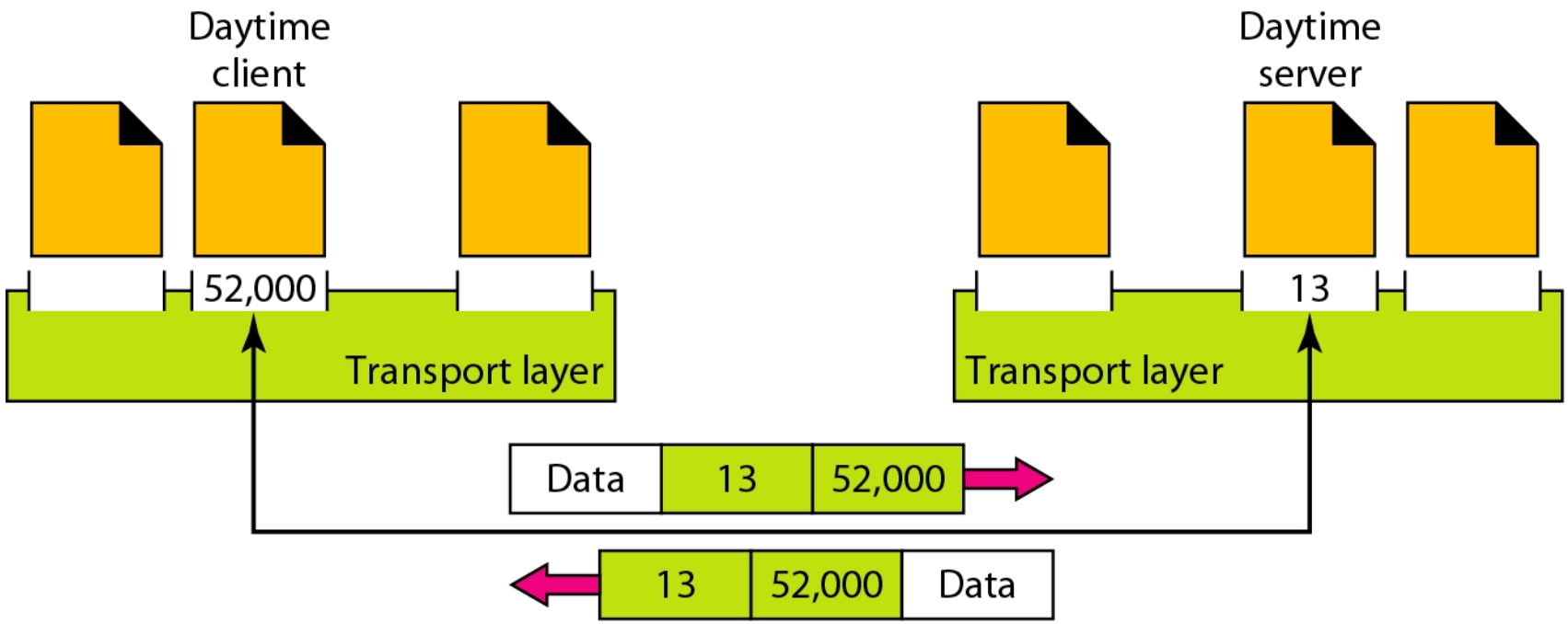
Outline

Transport Layer Delivery



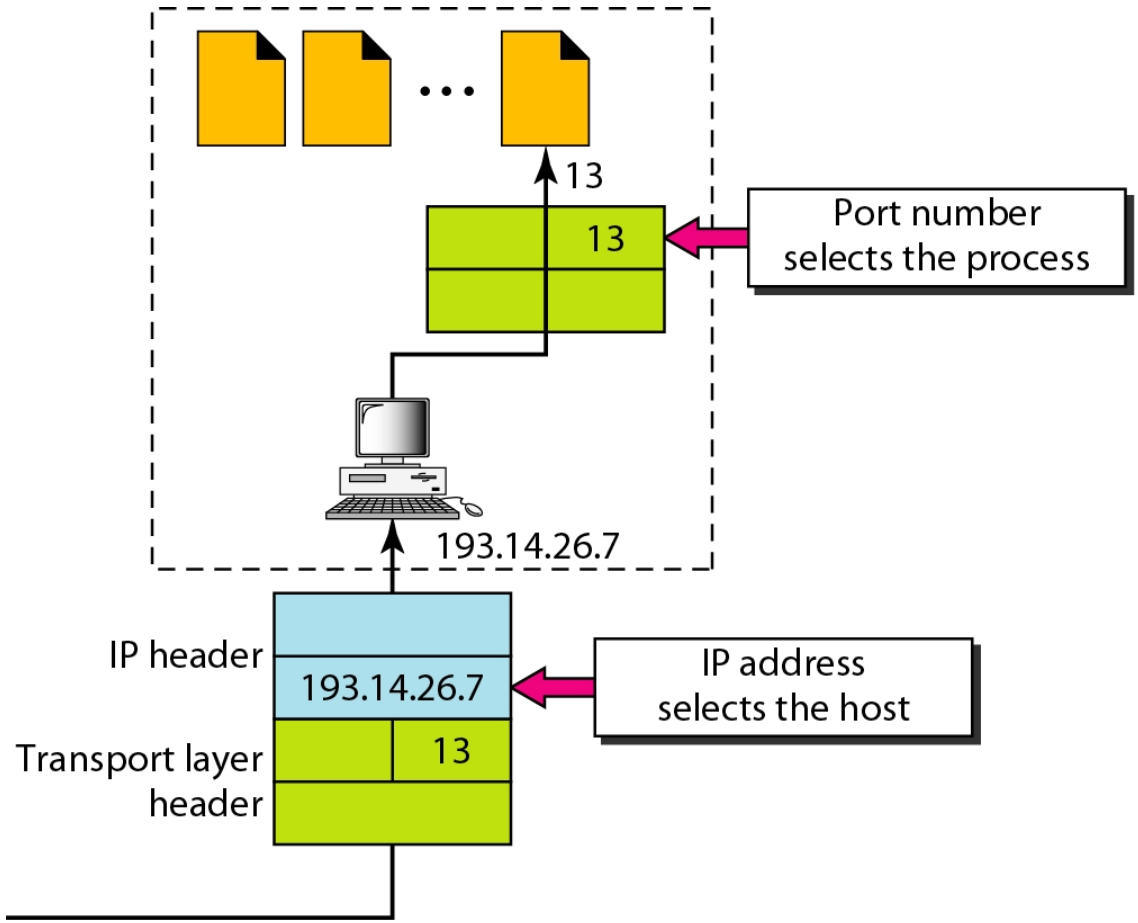
- Transport layer is responsible for process-to-process delivery

Transport Layer Addressing - Port Numbers



- 16 bit integer values

IP Addres vs. Port Number



- Network layer address vs. Transport layer address

Multiplexing De-multiplexing

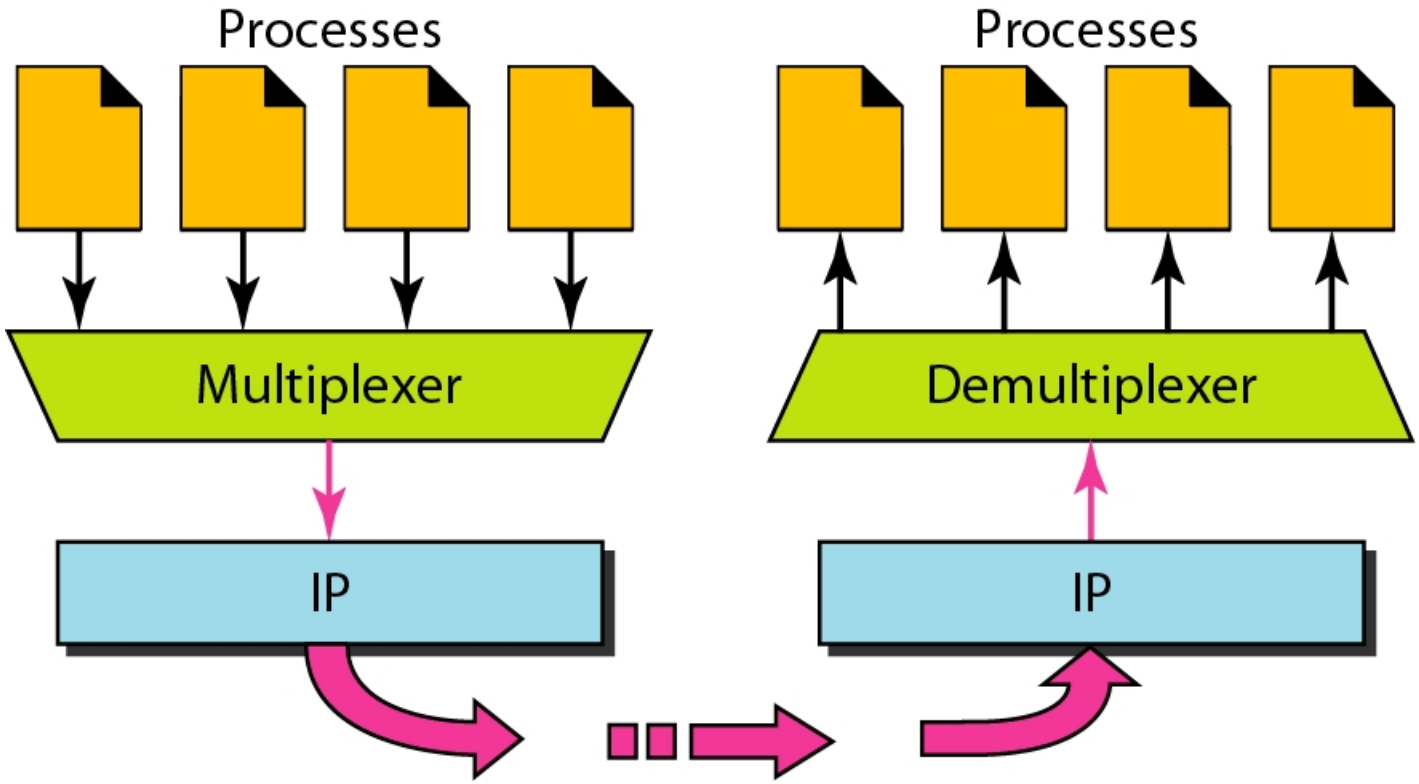
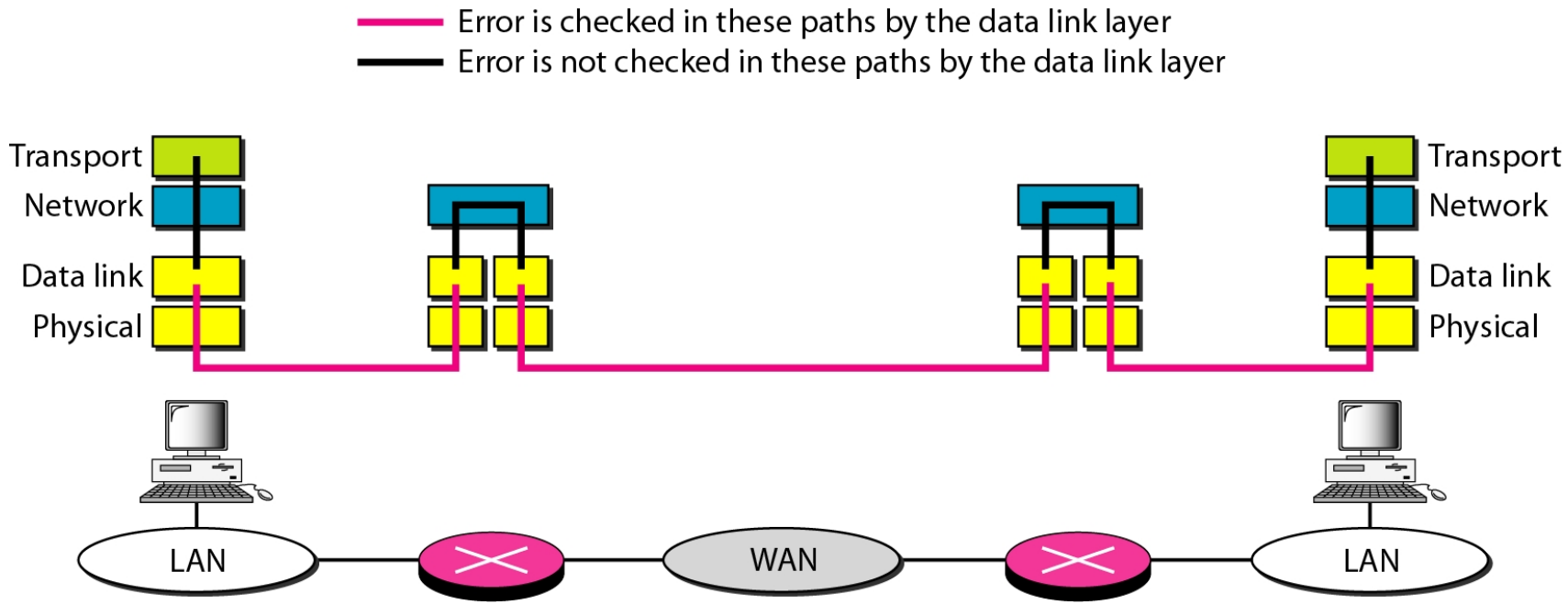


Figure 1: Process to Process Delivery

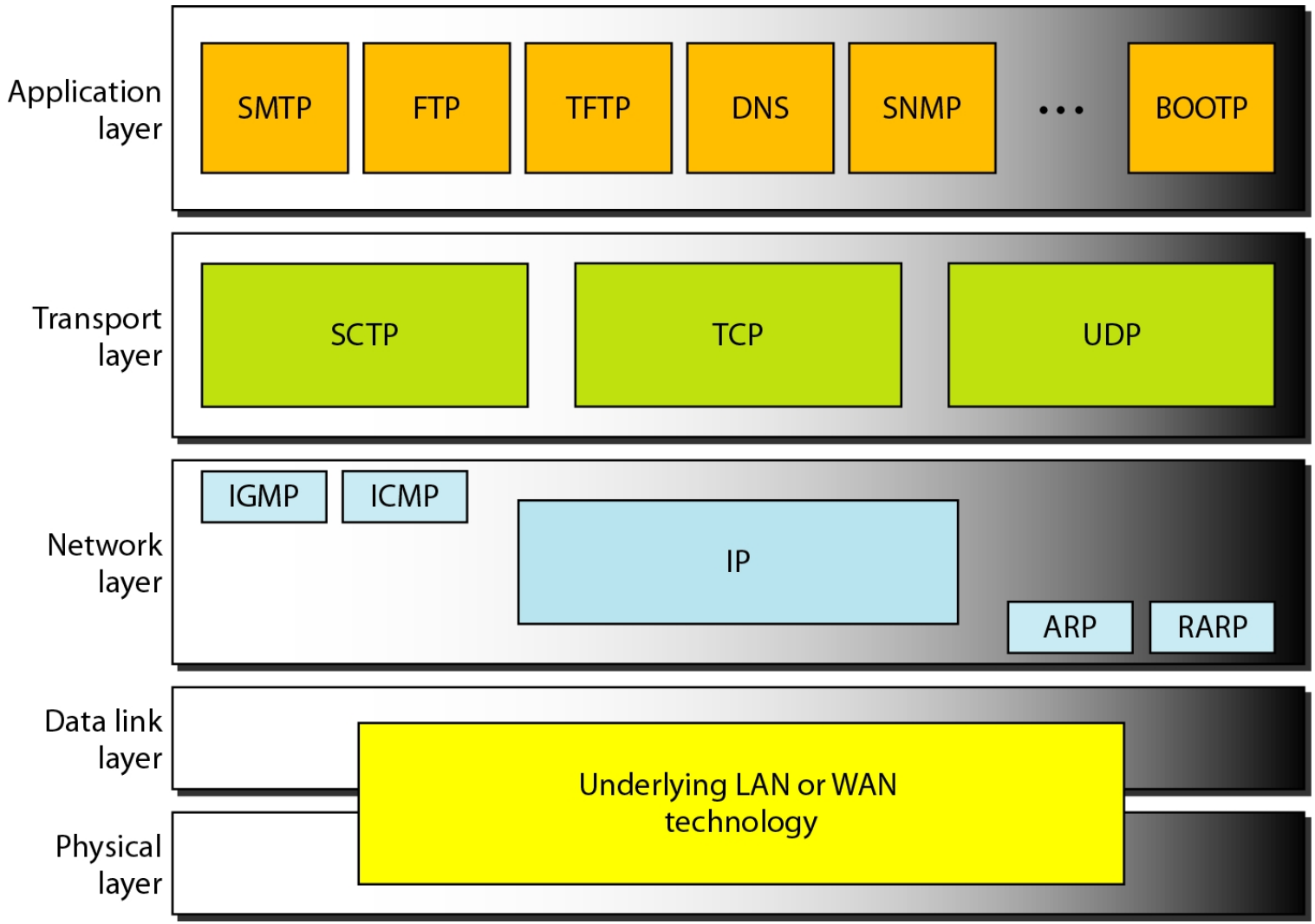
- Sender: same network layer service is multiplexed among multiple application processes
- Receiver: reverse

Error Control



- Why error control at transport layer? Data link layer does it already.
- Data link error control cannot detect errors introduced at the network layer and above in intermediate forwarding elements.

Protocols in TCP/IP Stack



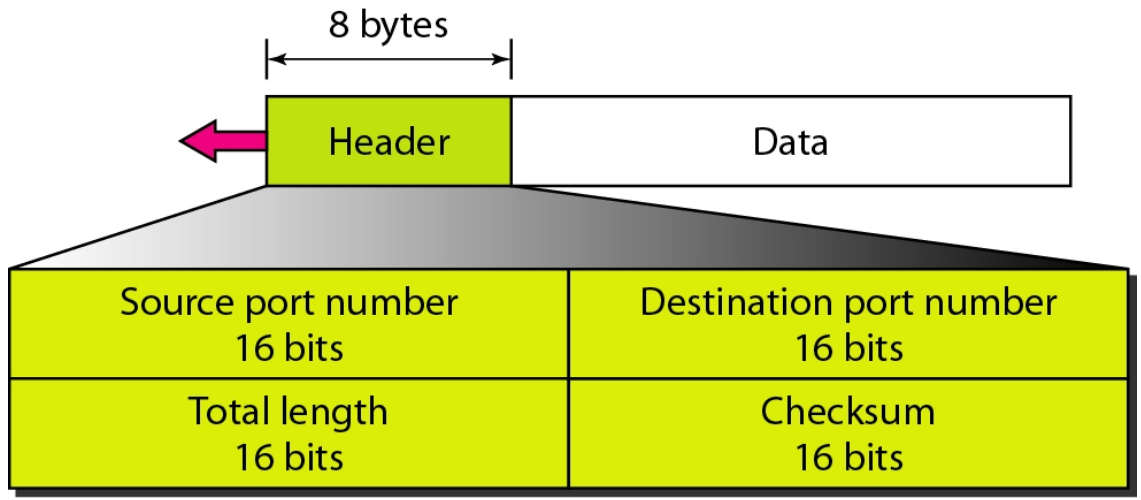
User Datagram Protocol

- 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

Well-known Ports used with UDP

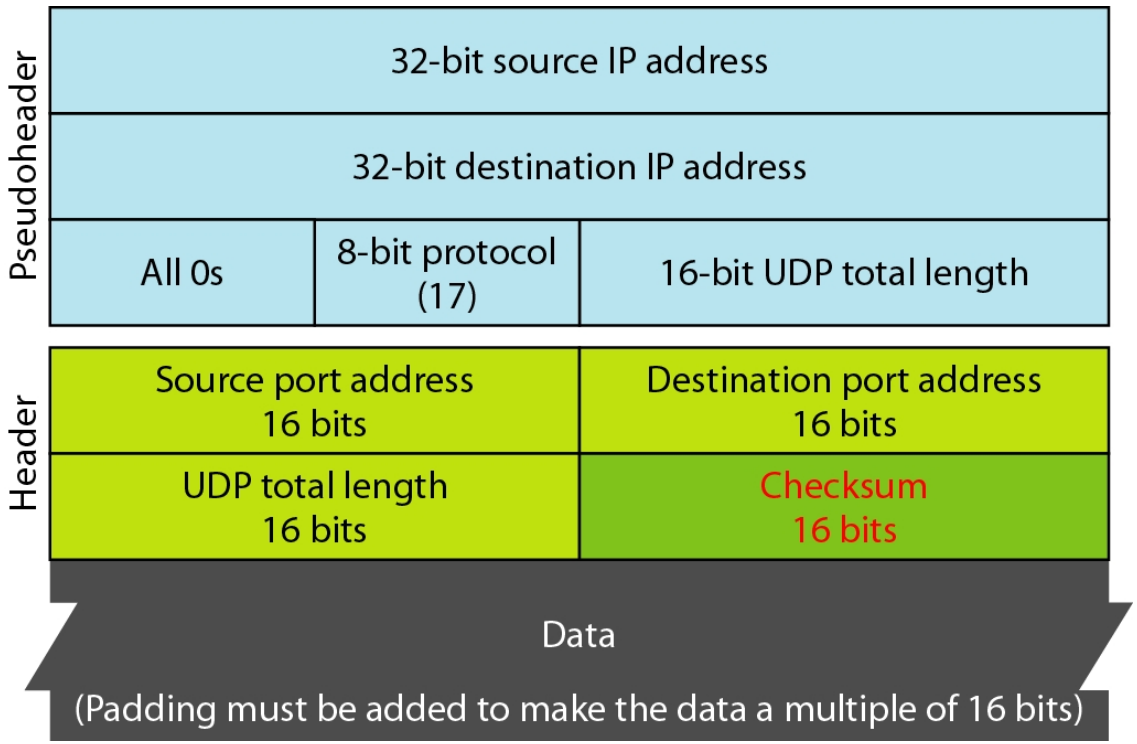
Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

UDP Segments



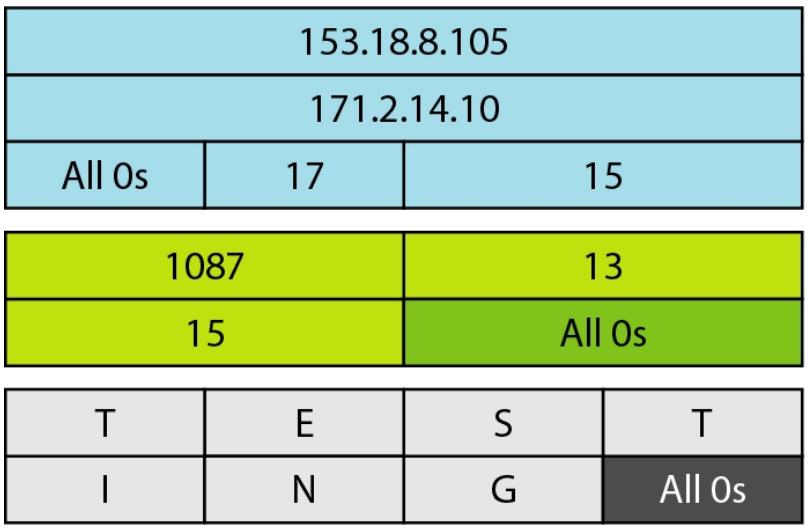
- 16 bit UDP length field:
  - Maximum up to $2^{16} - 1 = 65535$ bytes
  - Includes 8 bytes UDP header (max data $= 65527$)
  - But max IP packet size is also 65535
  - Minus 20 bytes IP header, minus 8 bytes UDP header
  - Max UDP data $= 65507$ bytes

UDP Pseudoheader for Checksum Calculation



- Intention: double check that packet has arrived at correct destination and transport protocol

Checksum Calculation of a Simple UDP User Datagram



10011001 00010010	153.18
00001000 01101001	8.105
10101011 00000010	171.2
00001110 00001010	14.10
00000000 00010001	0 and 17
00000000 00001111	15
00000100 00111111	1087
00000000 00001101	13
00000000 00001111	15
00000000 00000000	0 (checksum)
01010100 01000101	T and E
01010011 01010100	S and T
01001001 01001110	I and N
01000111 00000000	G and 0 (padding)
10010110 11101011	Sum
01101001 00010100	Checksum

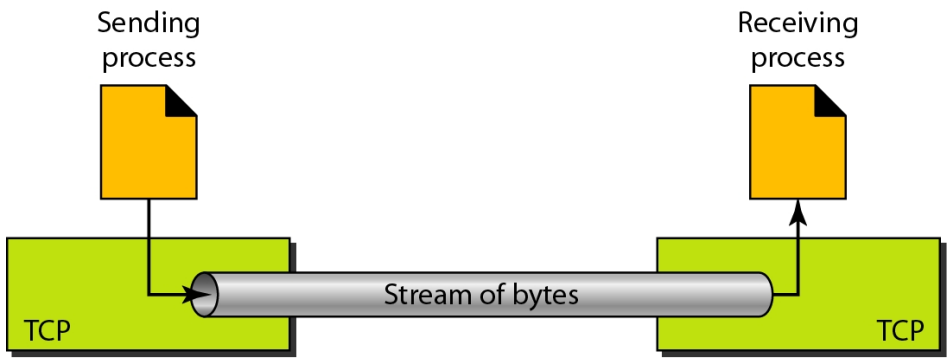
Transmission Control Protocol

- 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

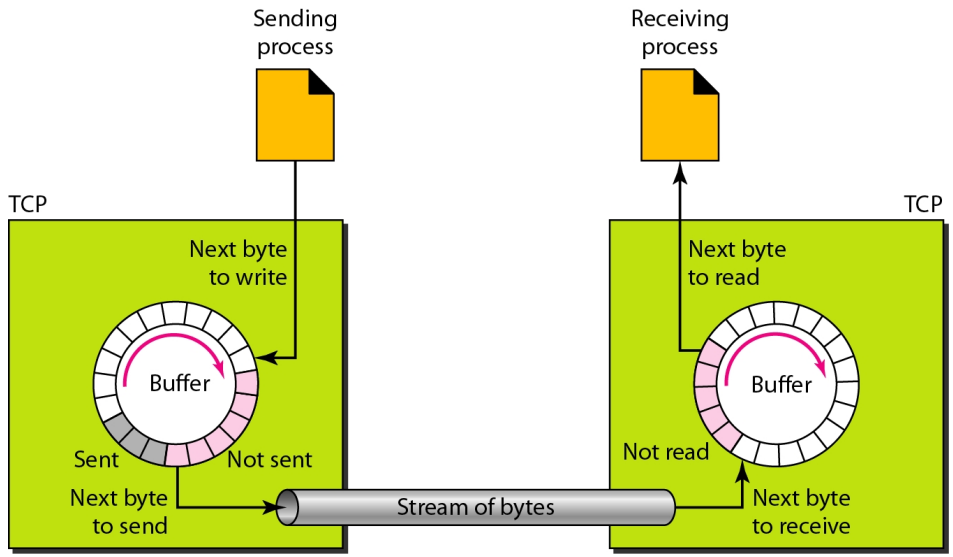
Well-known Ports used by TCP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

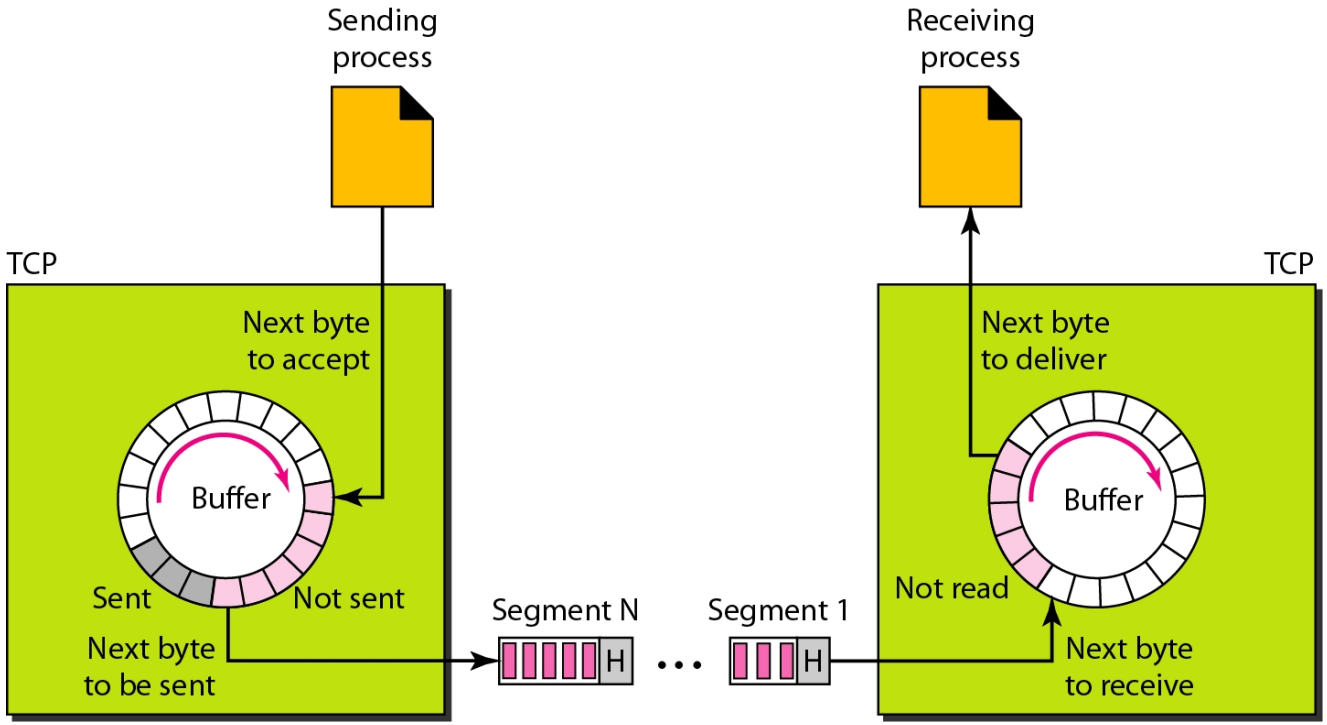
TCP Stream Delivery I



TCP Stream Delivery II

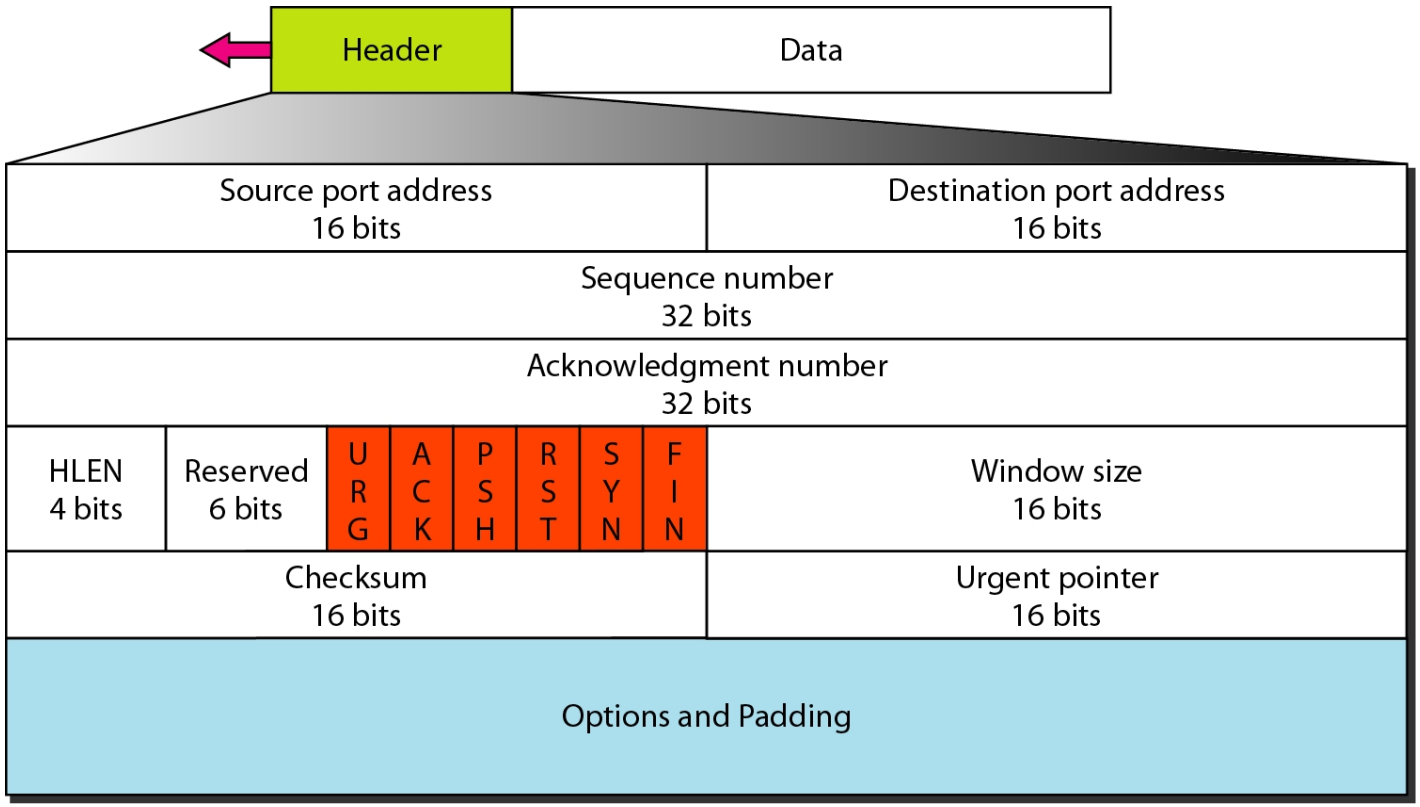


TCP Segments I



- Data bytes transferred in each connection are numbered $\Rightarrow$ numbering starts with a randomly generated number
- Sequence number field of a segment defines the number of the first data byte contained in that segment
- Value of the acknowledgment field in a segment defines the number of the next byte a host expects to receive
- Acknowledgment number is cumulative

TCP Segments II

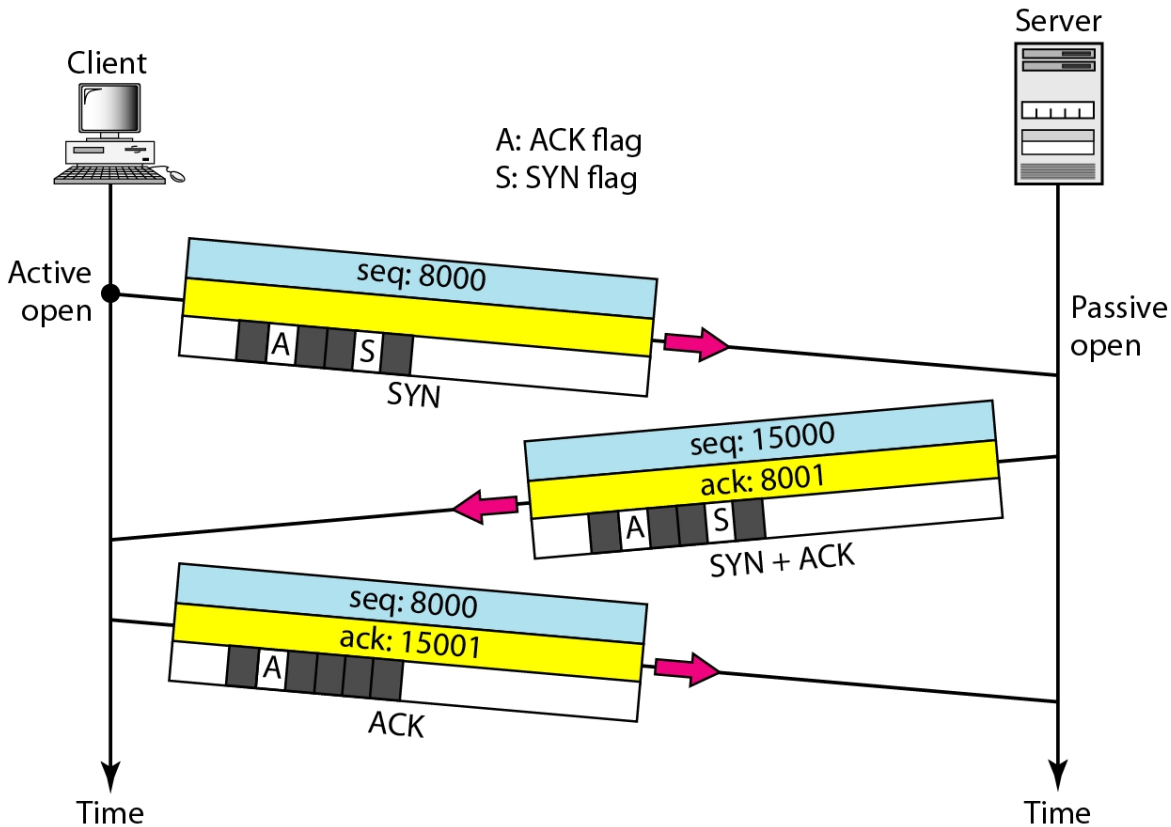


TCP Flags

- **SYN (Synchronize)**: used during session setup to agree on initial sequence numbers. Sequence numbers are random.
- **FIN (Finish)**: used during a graceful session close to show that the sender has no more data to send.
- **ACK (Acknowledge)**: receiver will send an ACK that equals the senders sequence number plus the Len, or amount of data, at the TCP layer.
  - **SYN, and FIN** flags count as 1 byte. The ACK can also be thought of as the sequence number of the next octet the receiver expects to receive.
- **RST (Reset)**: an instantaneous abort in both directions (abnormal session disconnection).
- **PSH (Push)**: forces data delivery without waiting for buffers to fill. This is used for interactive traffic. The data will also be delivered to the application on the receiving end with out buffering.
- **URG (Urgent)**: Data is sent out of band.

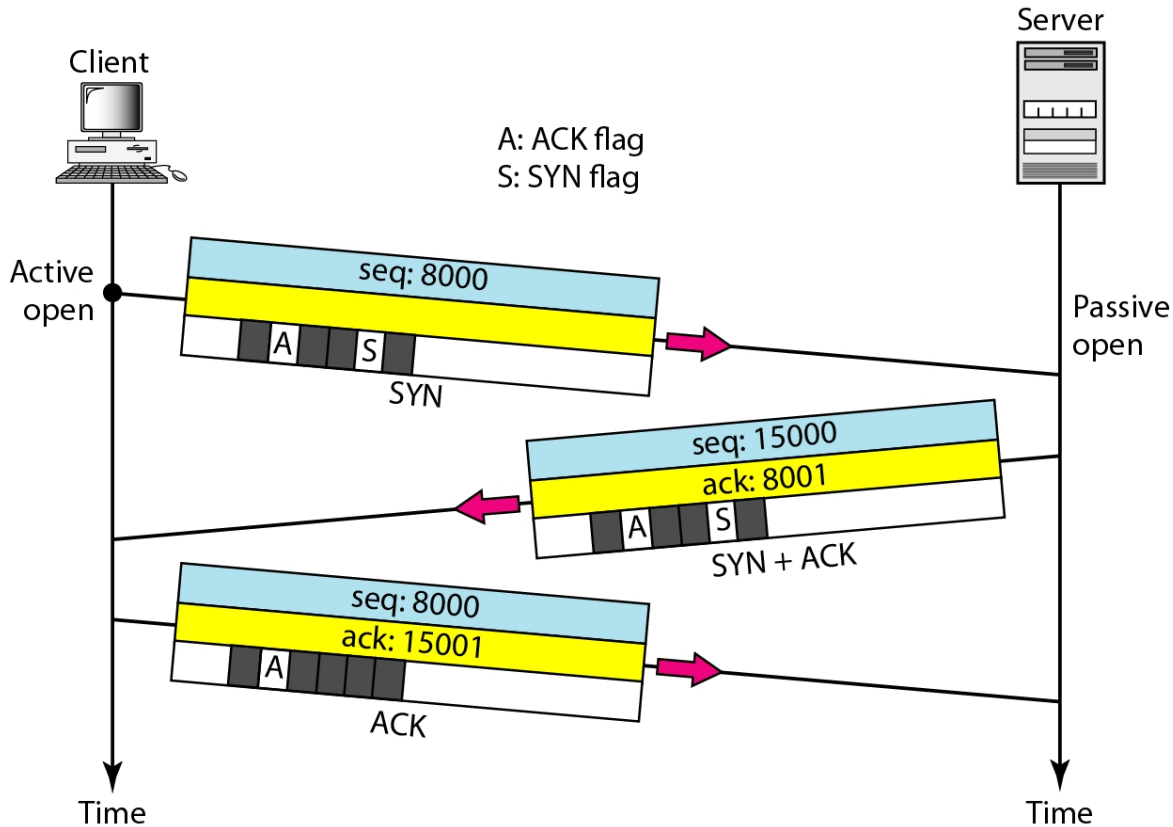
TCP Connection Establishment I

- Three way handshake
- A SYN segment cannot carry data, but it consumes one sequence number
- A SYN + ACK segment cannot carry data, but does consume one sequence number
- An ACK segment, if carrying no data, consumes no sequence number



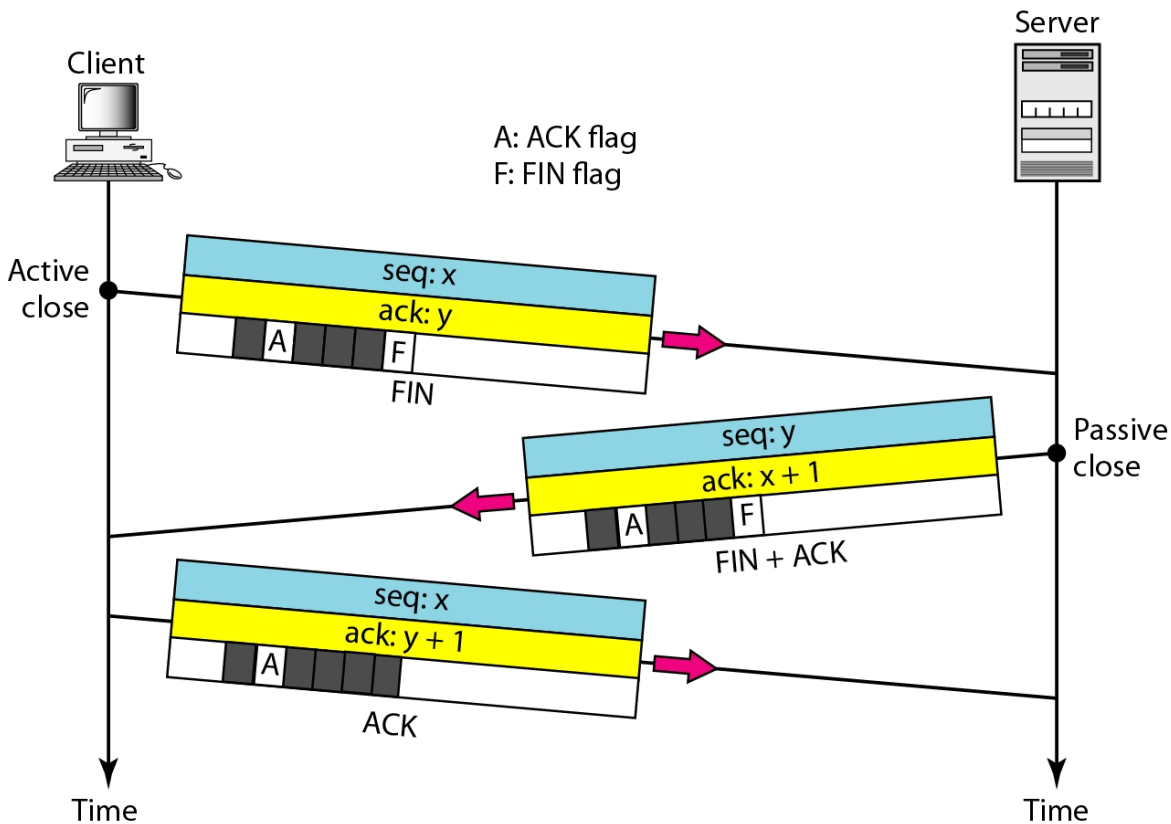
TCP Connection Establishment II

- Step 1: client host sends TCP SYN segment to server
  - specifies initial seq. no.
  - no data
- Step 2: server host receives SYN, replies with SYNACK segment
  - server allocates buffers
  - specifies server initial seq. no.
- Step 3: client receives SYNACK, replies with ACK segment, which may contain data



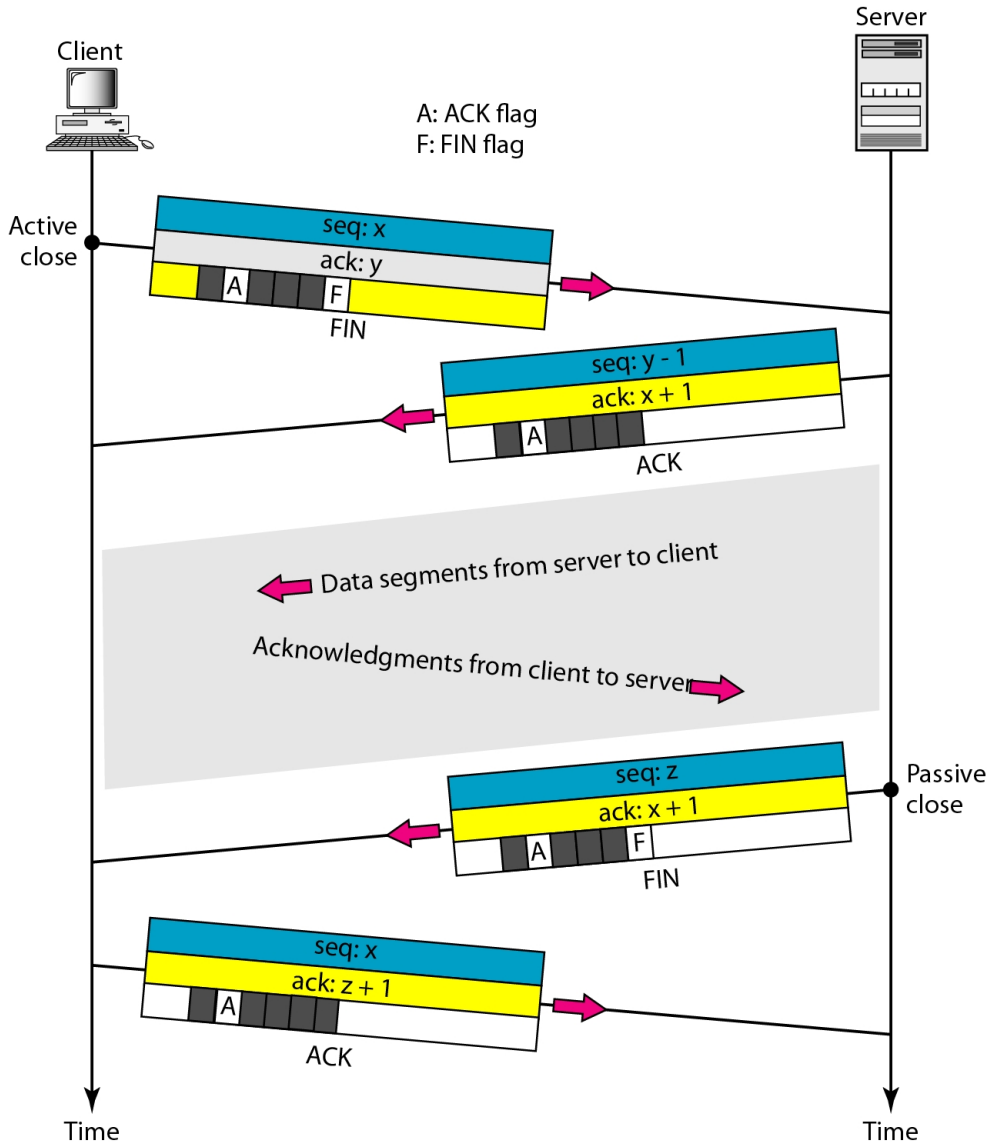
TCP Connection Termination I

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



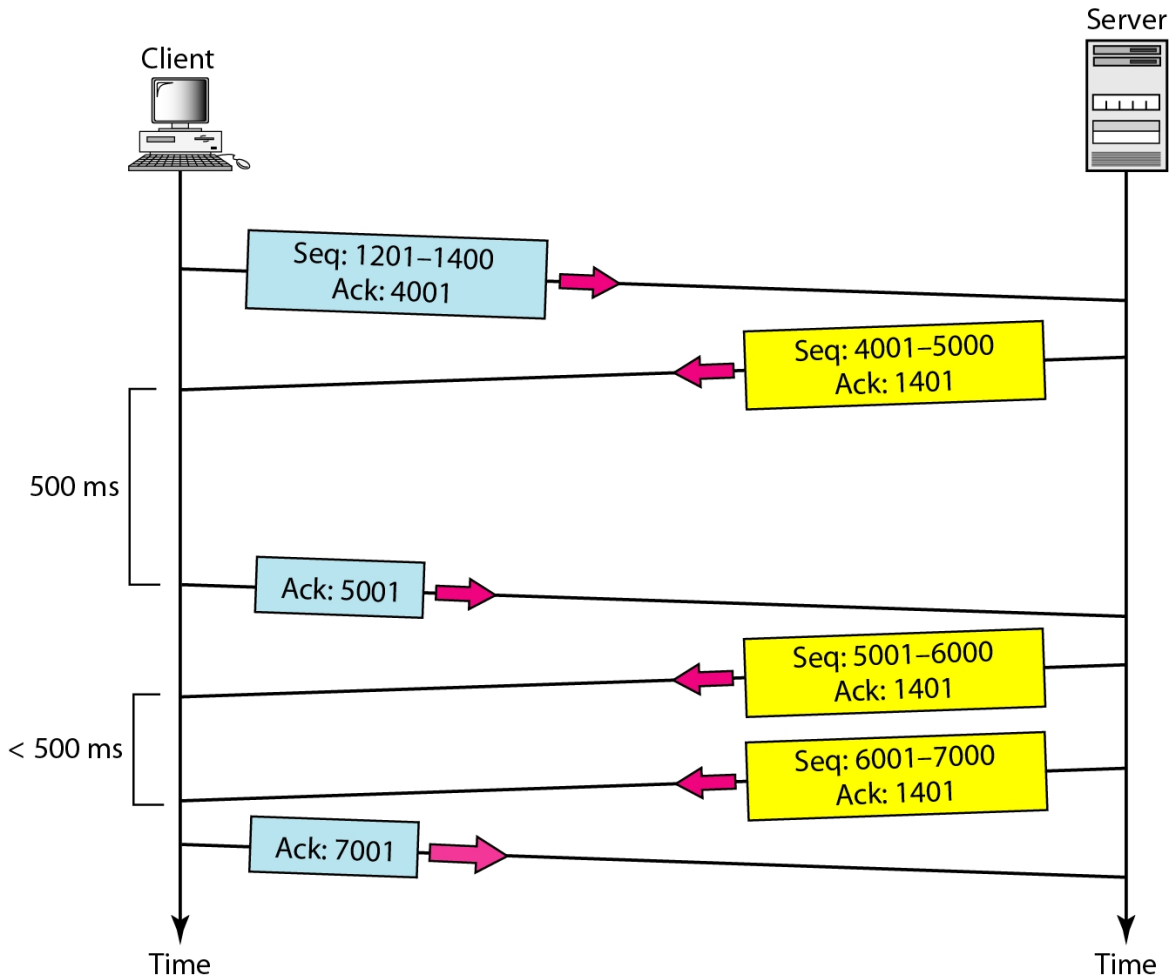
TCP Connection Termination II

- Step 1: client end system sends TCP FIN control segment to server
- Step 2: server receives FIN, replies with ACK. Closes connection, sends FIN.
- Step 3: client receives FIN, replies with ACK.
  - Enters "timed wait" - will respond with ACK to received FINs
- Step 4: server, receives ACK. Connection closed.



TCP Normal Operation

- Data may arrive out of order and be temporarily stored by the receiving TCP, but TCP guarantees that no out-of-order segment is delivered to the process



TCP Lost Segments

- The receiver TCP delivers only ordered data to the process