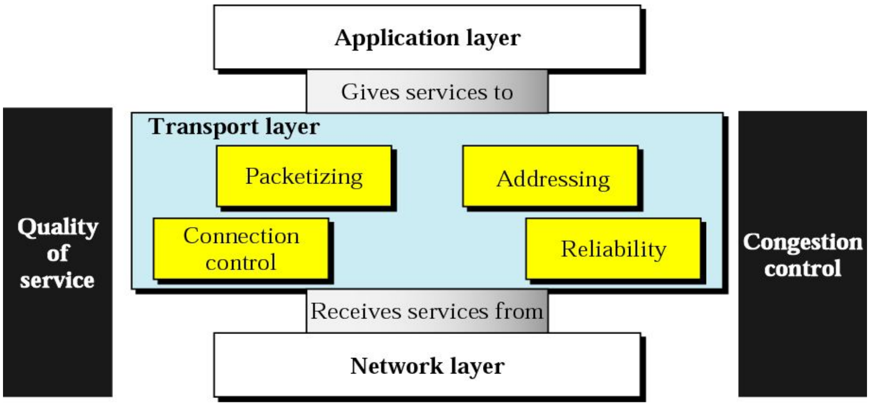
Data Communication & Computer Networks

10. Transport Layer Protocols

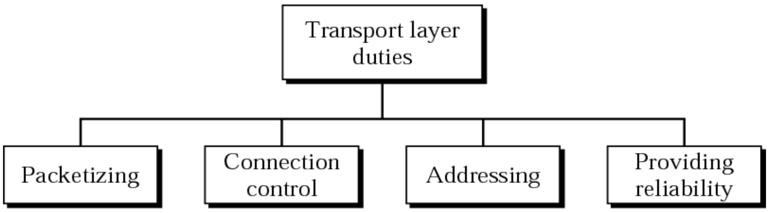
Transport layer



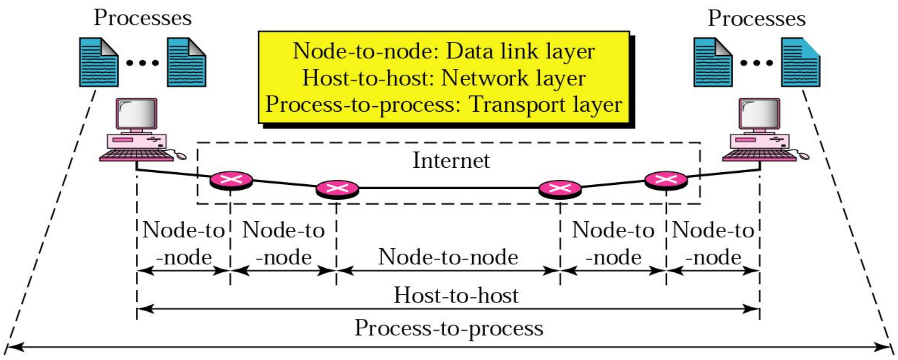
Transport Protocols

- end-to-end data transfer service
- shield upper layers from network details
- reliable, connection oriented
  - has greater complexity
  - eg. TCP
- best effort, connectionless
  - datagram
  - eg. UDP

Transport layer duties



Process-to-process delivery



Node-to-node: Data link layer
Host-to-host: Network layer
Process-to-process: Transport layer

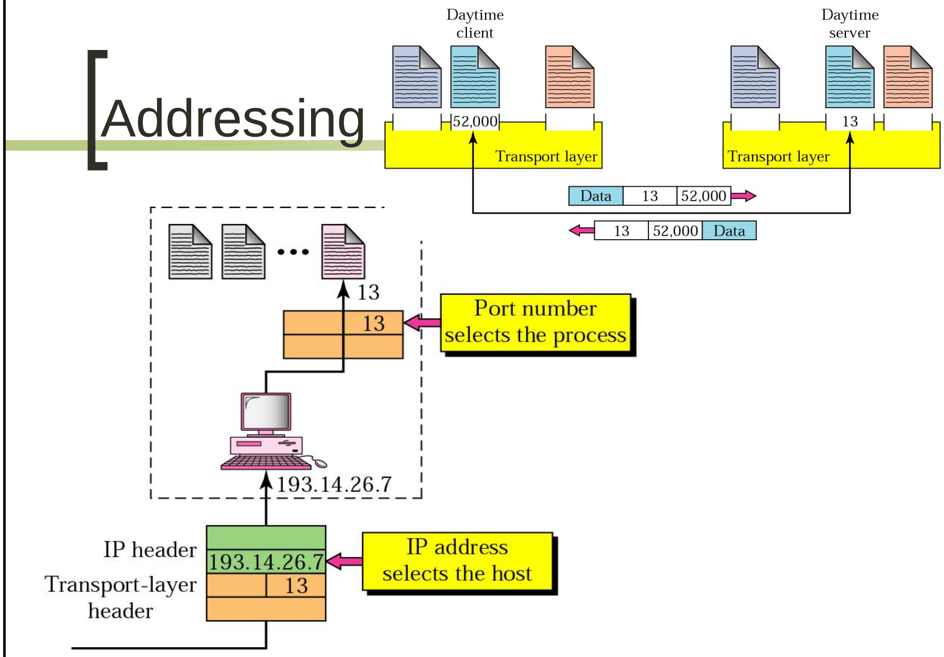
Client–Server Computing

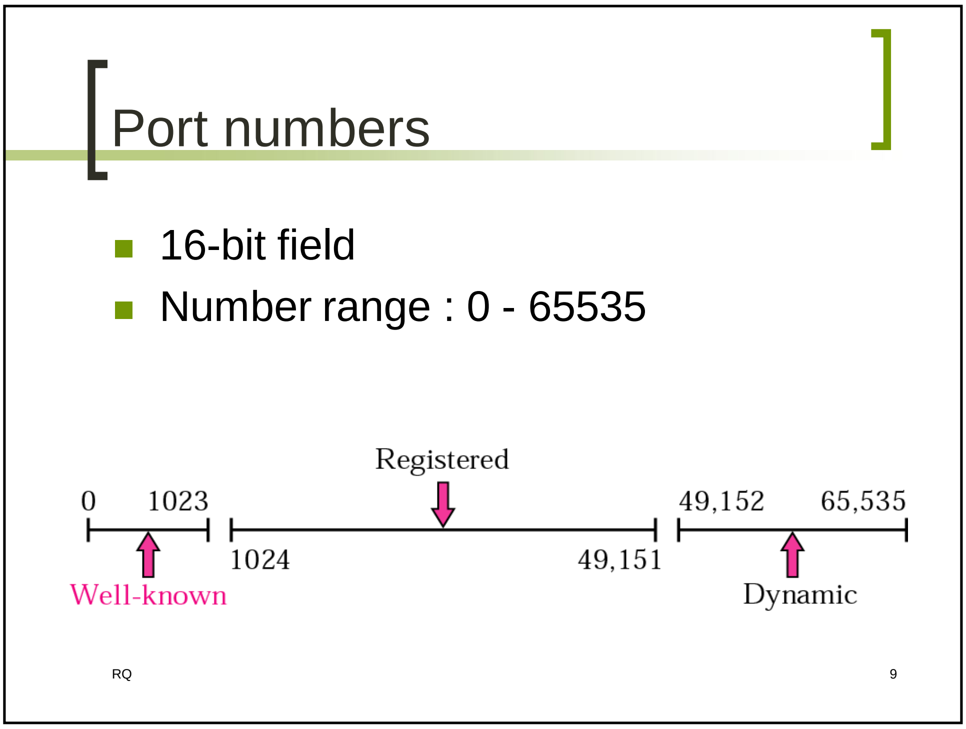
- Process takes place
  - on the server and
  - on the client
- Servers
  - Store and protect data
  - Process requests from clients
- Clients
  - Make requests
  - Format data on the desktop

Types of Servers

- Application Servers
- Audio/Video Servers
- Chat Servers
- FTP Servers
- IRC Servers
- Mail Servers
- News Servers
- Proxy Servers
- Telnet Servers
- Web Servers

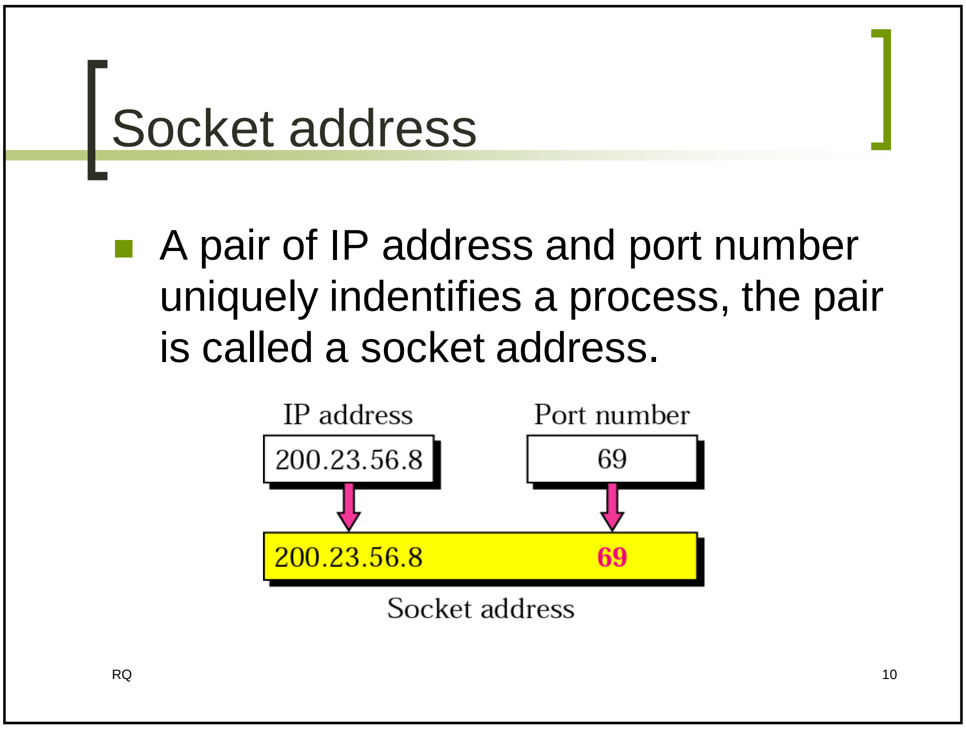
Addressing





Port numbers

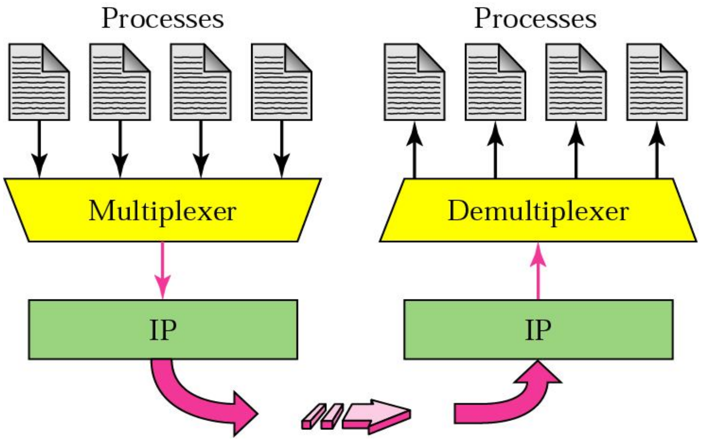
- 16-bit field
- Number range : 0 - 65535



Socket address

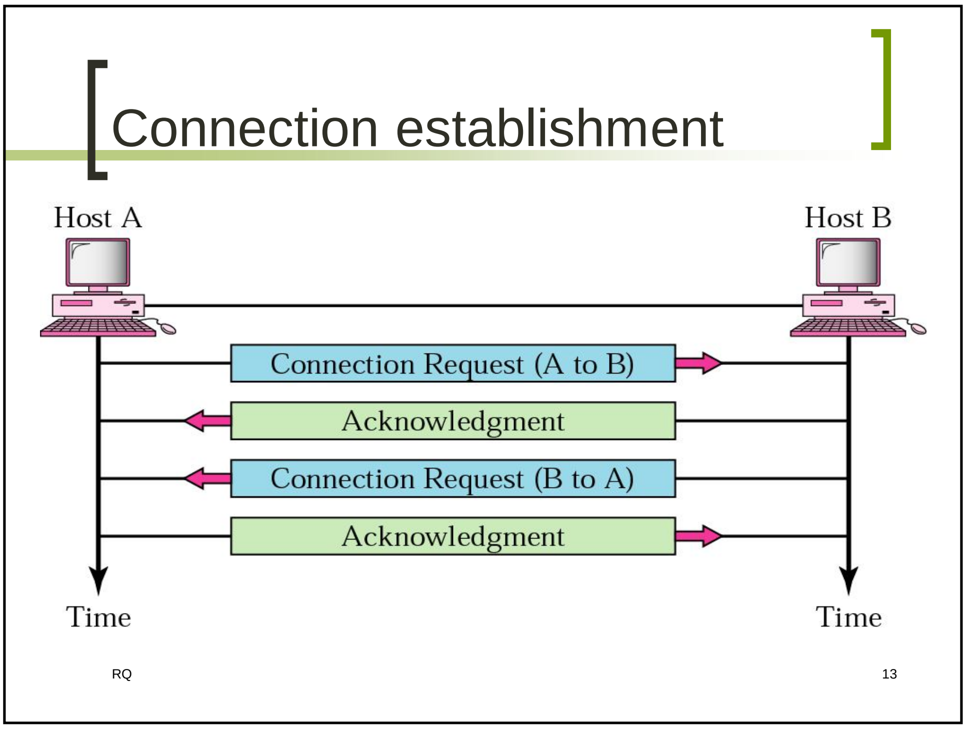
- A pair of IP address and port number uniquely indentifies a process, the pair is called a socket address.

Multiplexing and Demultiplexing



Connectionless vs. Connection–Oriented Service

- Connectionless service
  - No connection established
  - Packets may be delayed, lost or arrive out of order
- Connection-oriented service
  - Communication begins after establishing a connection



Connection establishment

Host A

Host B

Connection Request (A to B)

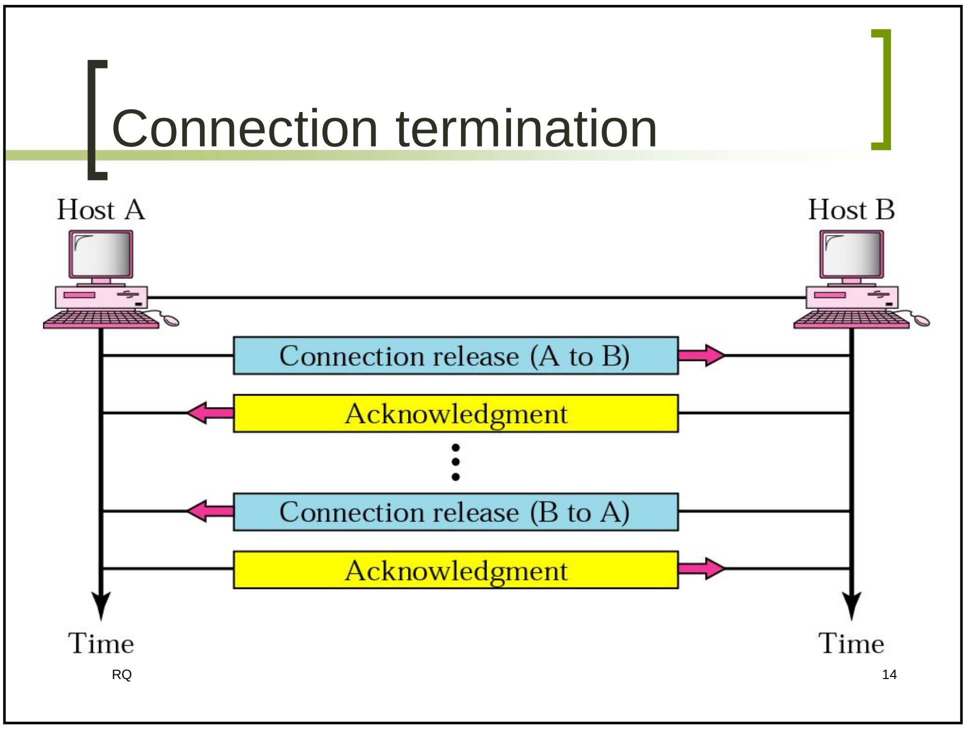
Acknowledgment

Connection Request (B to A)

Acknowledgment

Time

Time



Connection termination

Host A

Host B

Connection release (A to B)

Acknowledgment

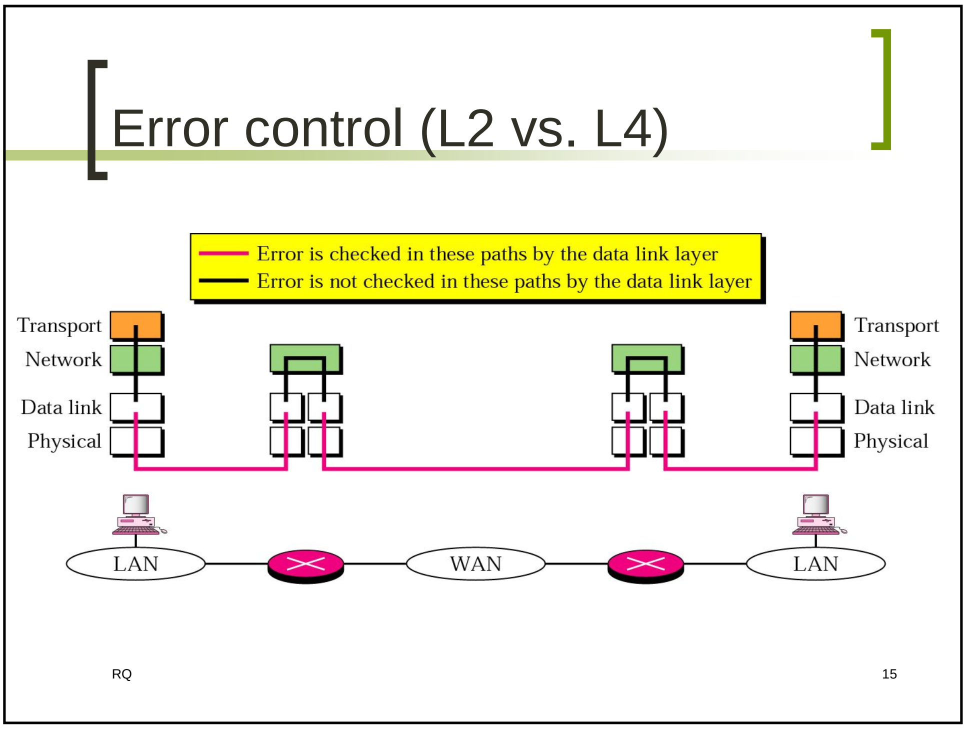
⋮

Connection release (B to A)

Acknowledgment

Time

Time

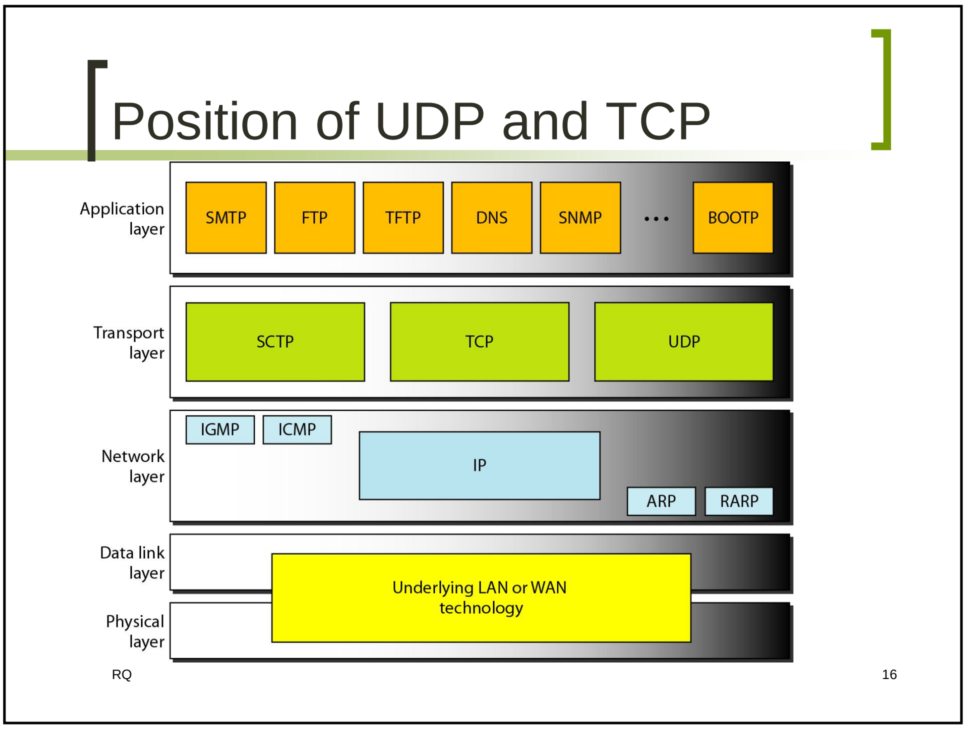


Error control (L2 vs. L4)

Error is checked in these paths by the data link layer

Error is not checked in these paths by the data link layer

RQ 15



Position of UDP and TCP

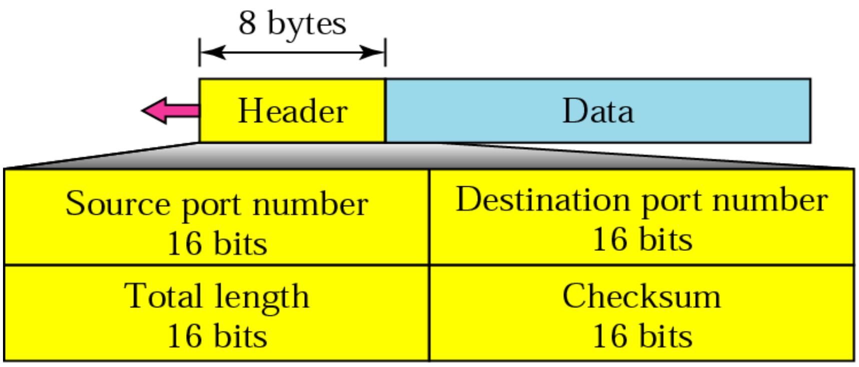
RQ 16

Impo !

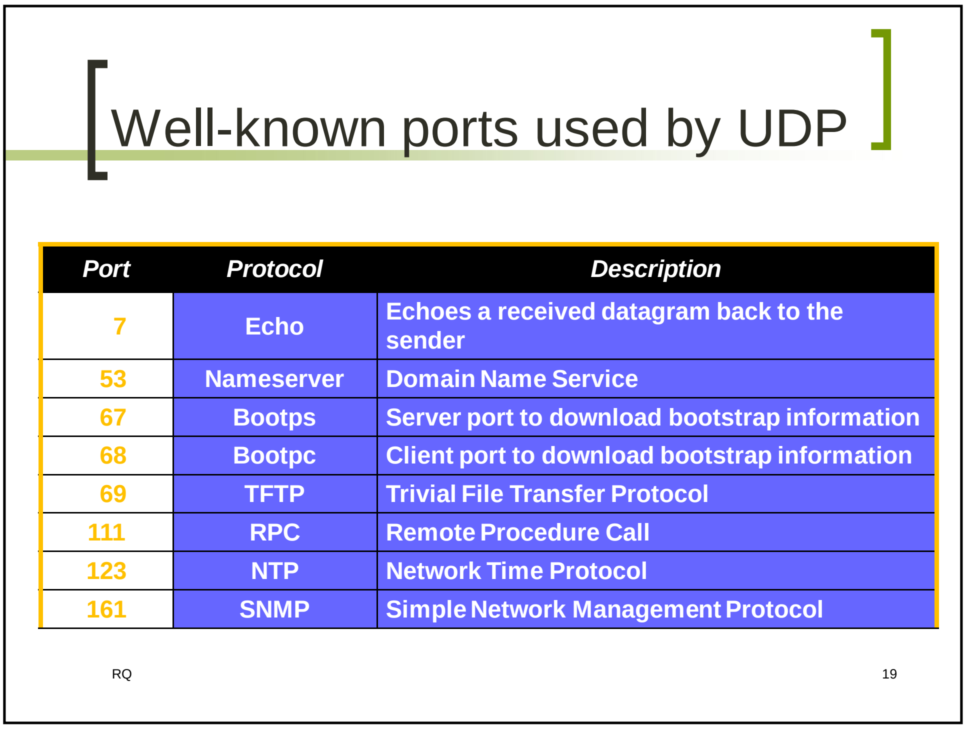
User Datagram Protocol (UDP)

- Connectionless and unreliable protocol
- No flow and error control
- A convenient Layer 4 protocol for applications that provide their own flow and error control.
- Commonly used by multimedia applications.

UDP header format

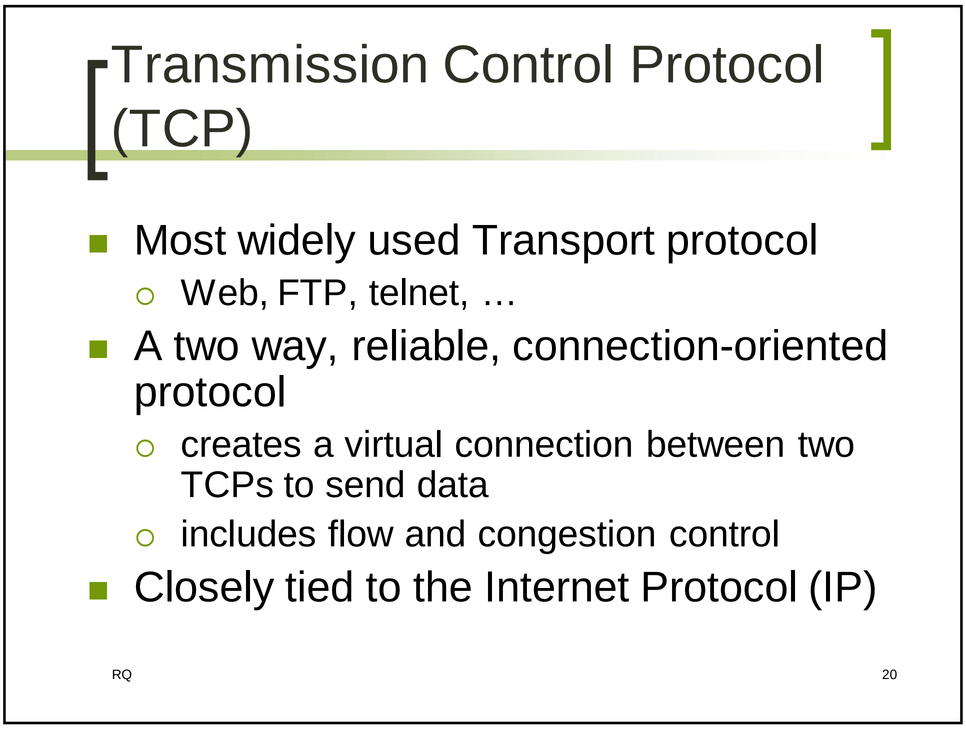


Source port number 16 bits	Destination port number 16 bits
Total length 16 bits	Checksum 16 bits



Well-known ports used by UDP

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
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

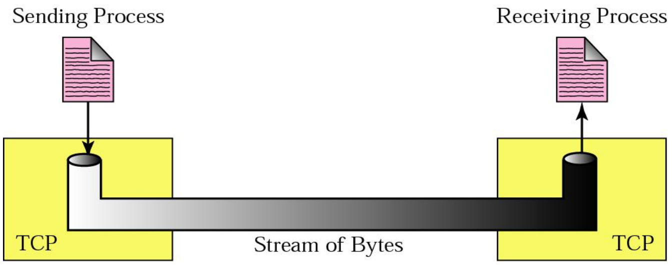


Transmission Control Protocol (TCP)

- Most widely used Transport protocol
  - Web, FTP, telnet, …
- A two way, reliable, connection-oriented protocol
  - creates a virtual connection between two TCPs to send data
  - includes flow and congestion control
- Closely tied to the Internet Protocol (IP)

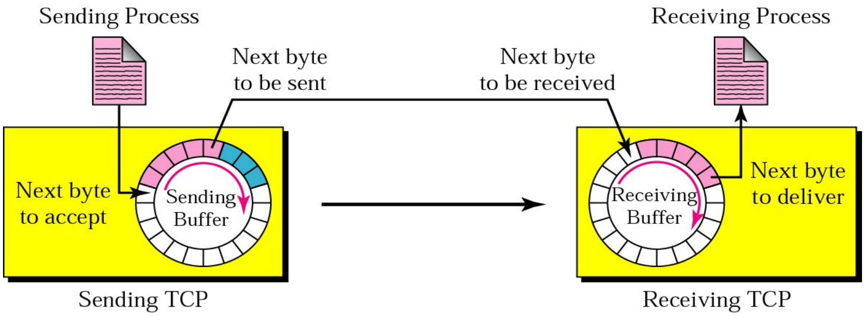
Stream delivery service

- TCP allows processes to send and receive data in a stream of bytes (unlike UDP which deals in chunks of data)

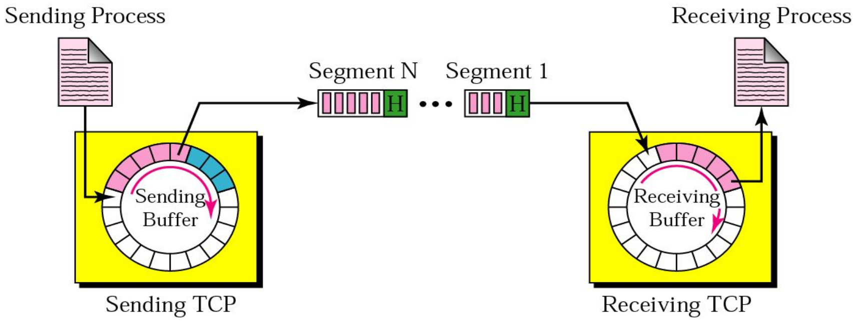


Buffers

- Sending and receiving processes may not work at the same speed, buffers handle this problem



TCP segments



Other TCP services

- Full-duplex service
- Connection-oriented service
- Reliable service

Well-known ports used by TCP

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
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
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Byte, Sequence and Acknowledgement numbers

- TCP numbers the bytes of data being transferred in each connection
  - numbering starts with a random number
- The sequence number of a segment is the number of the first data byte contained in that segment
- The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive.

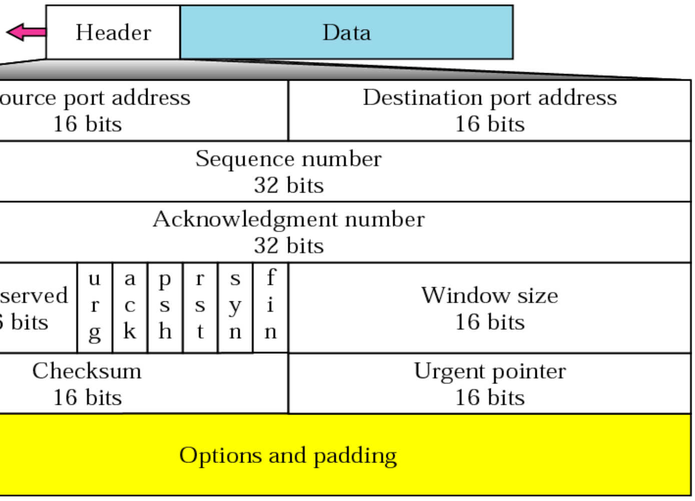
Example

Q. Imagine a TCP connection is transferring a file of 6000 bytes. The first byte is numbered 10010. What are the sequence numbers for each segment if data are sent in five segments with the first four segments carrying 1000 bytes and the last segment carrying 2000 bytes?

The following shows the sequence number for each segment:

Segment 1 ==>	sequence number: 10,010 (range: 10,010 to 11,009)
Segment 2 ==>	sequence number: 11,010 (range: 11,010 to 12,009)
Segment 3 ==>	sequence number: 12,010 (range: 12,010 to 13,009)
Segment 4 ==>	sequence number: 13,010 (range: 13,010 to 14,009)
Segment 5 ==>	sequence number: 14,010 (range: 14,010 to 16,009)

TCP header (segment) format



Header | Data

Source port address 16 bits | Destination port address 16 bits

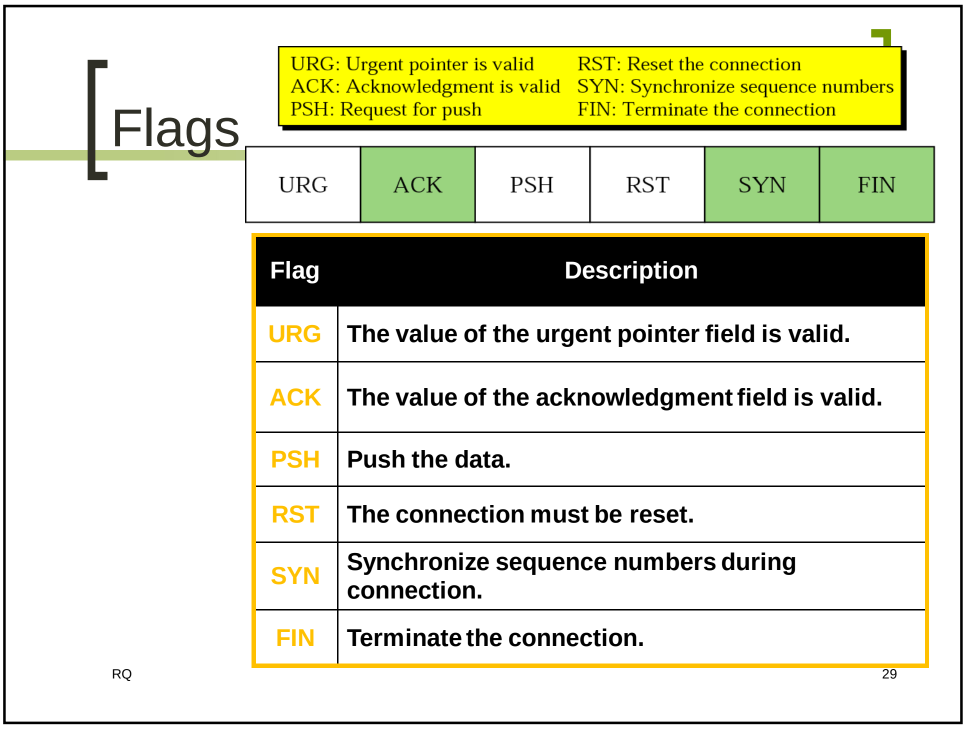
Sequence number 32 bits

Acknowledgment number 32 bits

HLEN 4 bits | Reserved 6 bits | urg | ack | psh | rst | syn | fin | Window size 16 bits

Checksum 16 bits | Urgent pointer 16 bits

Options and padding



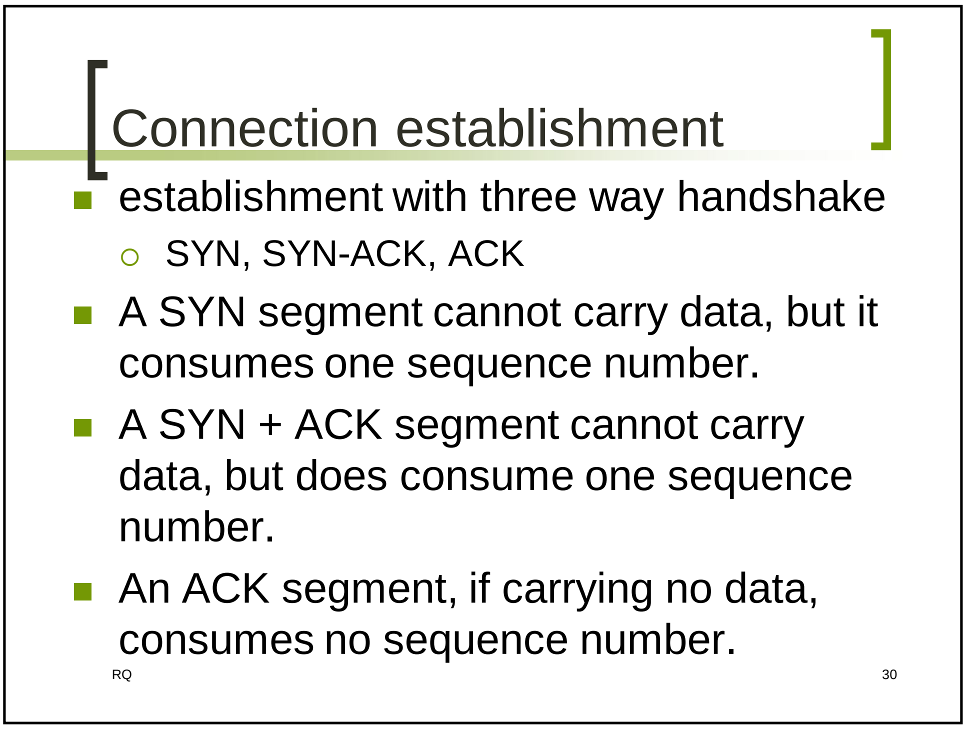
Flags

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

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

URG	ACK	PSH	RST	SYN	FIN

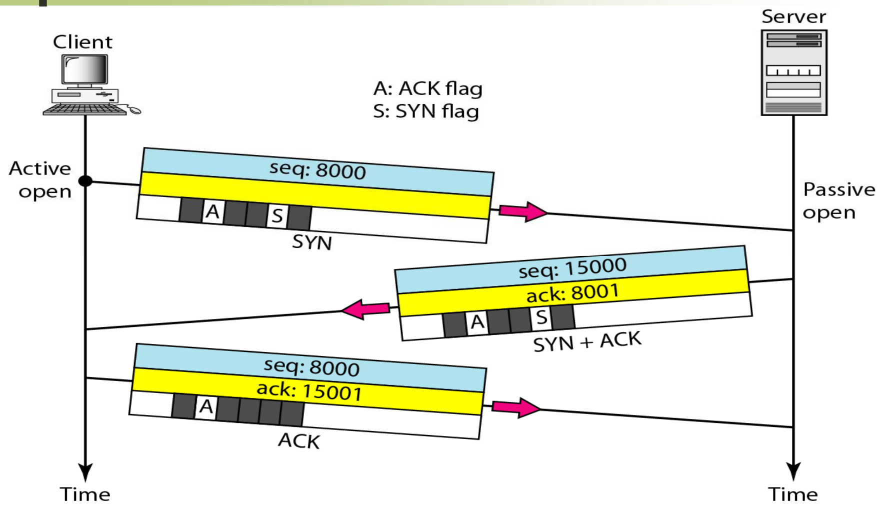
Flag	Description
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	The connection must be reset.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.



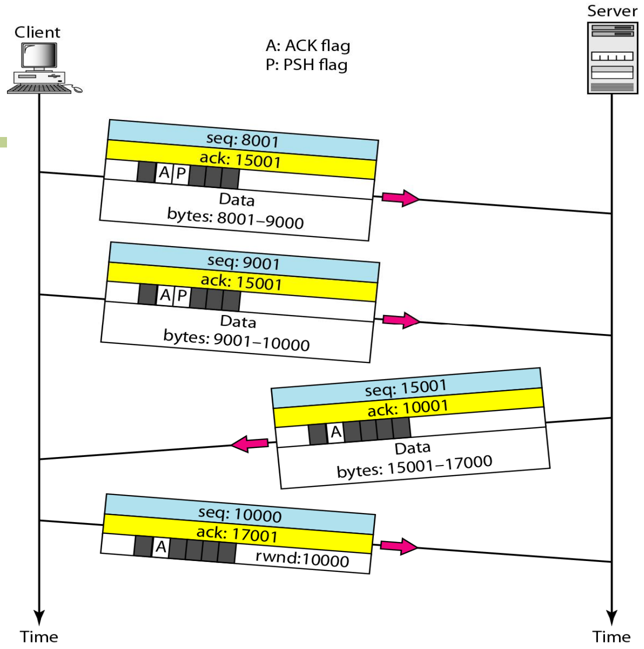
Connection establishment

- establishment with three way handshake
  - SYN, SYN-ACK, ACK
- 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.

Connection establishment



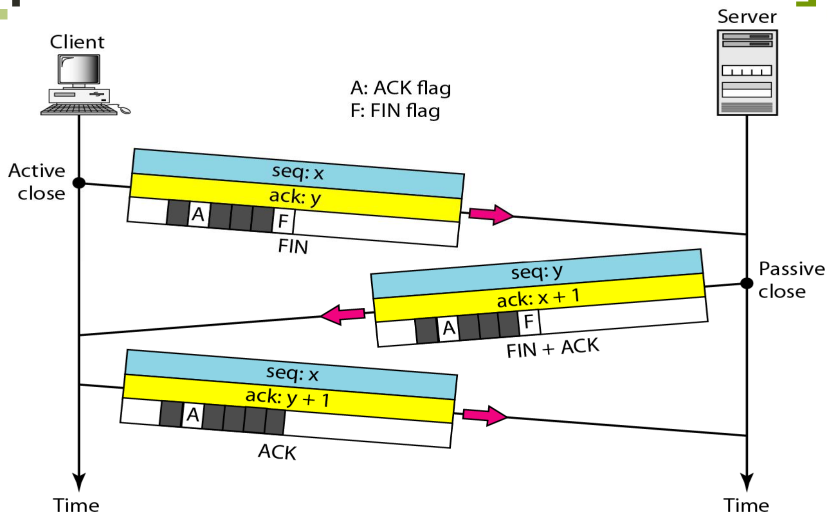
Data transfer



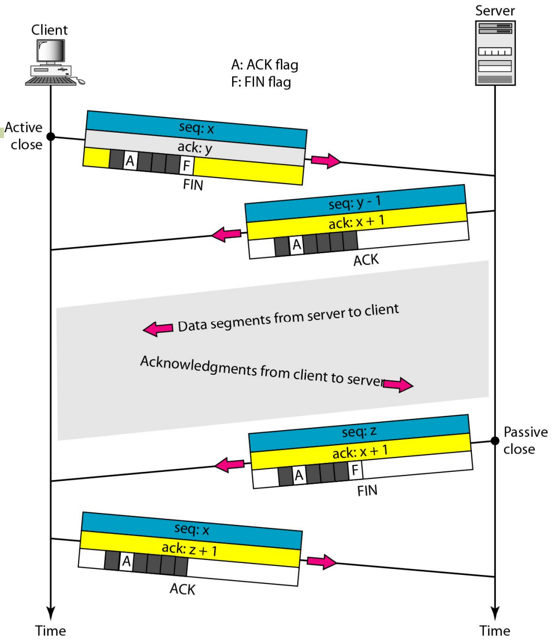
Connection termination

- graceful close (FIN)
  - transport entity sets FIN flag on last segment sent with last of data
  - 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.
- abrupt termination (RST)
  - entity abandons all attempts to send or receive data

Connection termination



Connection termination



Flow Control

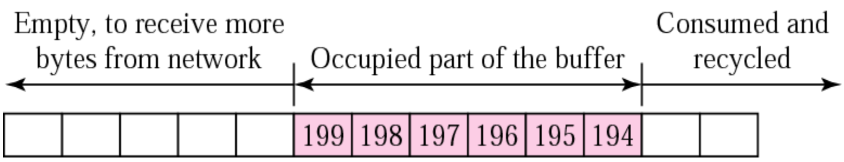
- issues:
  - Longer and variable transmission delays
- want Transport layer flow control because:
  - receiving user can not keep up
  - receiving transport entity can not keep up
- which can result in buffer overflowing
- managing flow difficult because of gap between sender and receiver

Flow control with Sliding Window

- A sliding window is used to control the flow of data so that the destination does not become overwhelmed with data.
- each transport segment has seq number (SN), ack number (AN) and window size (W) in header
- sends seq number of first octet in segment
- ACK includes (AN=*i*, W=*j*) which means
  - all octets through SN=*i-1* acknowledged, want *i* next
  - permission to send additional window of W=*j* octets

Receiver window

- If total size of receiving buffer is *N* and *M* locations are already occupied, then the size of receiver window is (*N - M)*



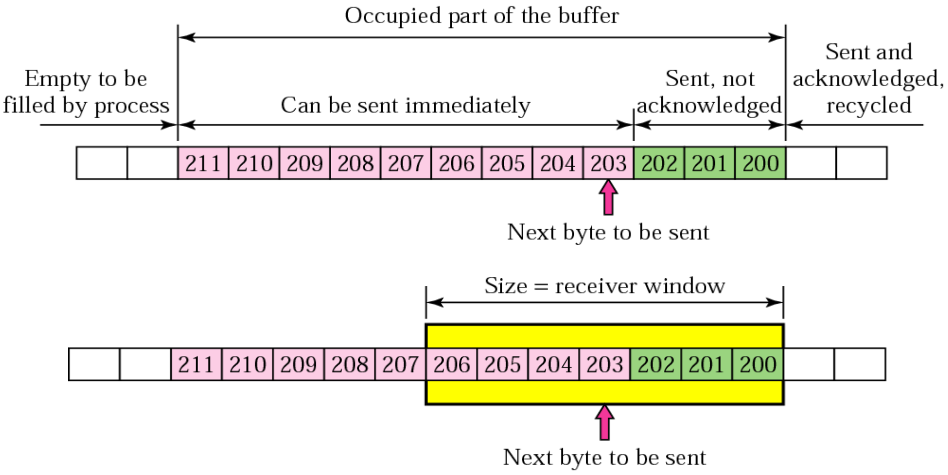
Empty, to receive more bytes from network

Occupied part of the buffer

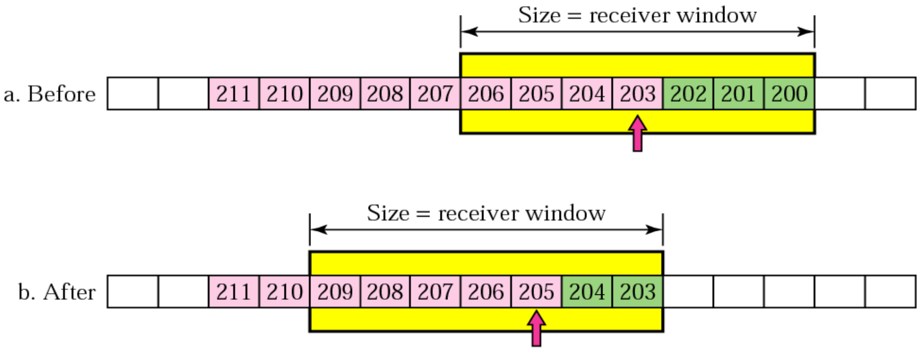
Consumed and recycled

					199	198	197	196	195	194		

Sender buffer and window



Sliding the sender window

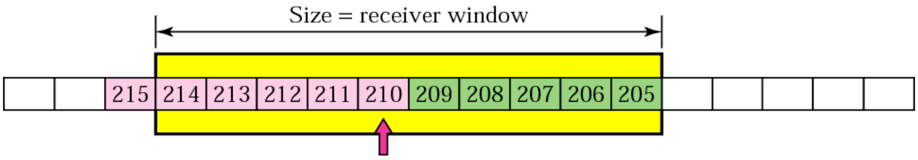


Changing window size

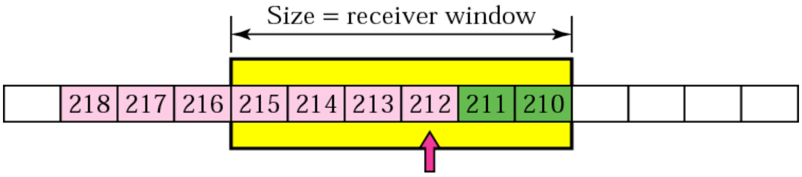
- In TCP, the sender window size is controlled by the receiver window value (the number of empty locations in the receiver buffer).
- However, the source does not have to send a full window’s worth of data.
- The size of the window can be increased or decreased by the destination.

Changing sender window size

- Expanding



- Shrinking

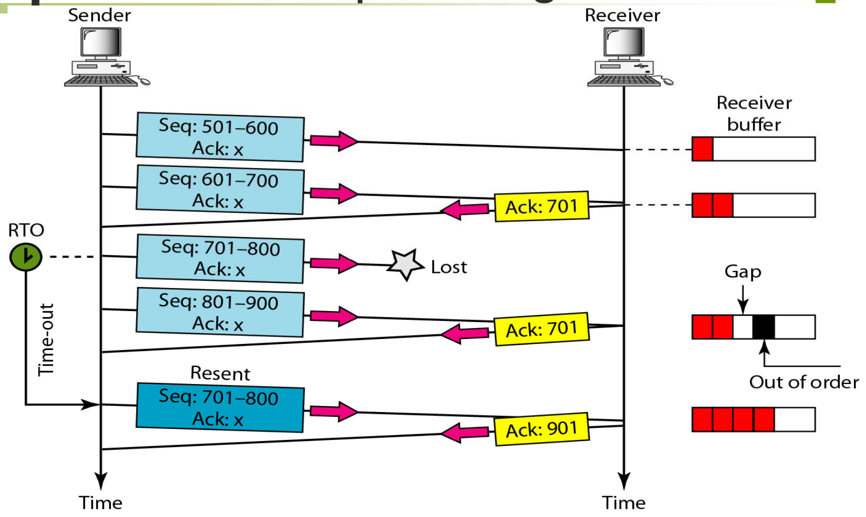


Error control

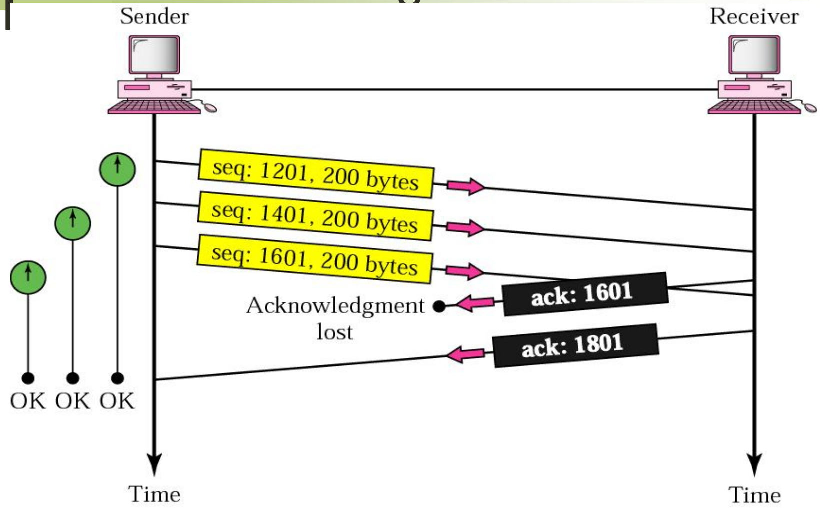
- TCP uses three simple tools
  - Checksum
  - Acknowledgment
  - Time-out

- There is no negative acknowledgment in TCP

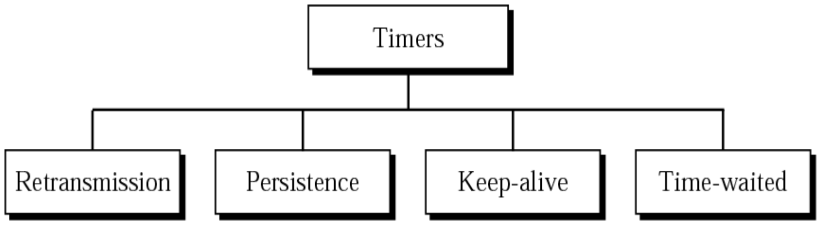
Lost or corrupted segment

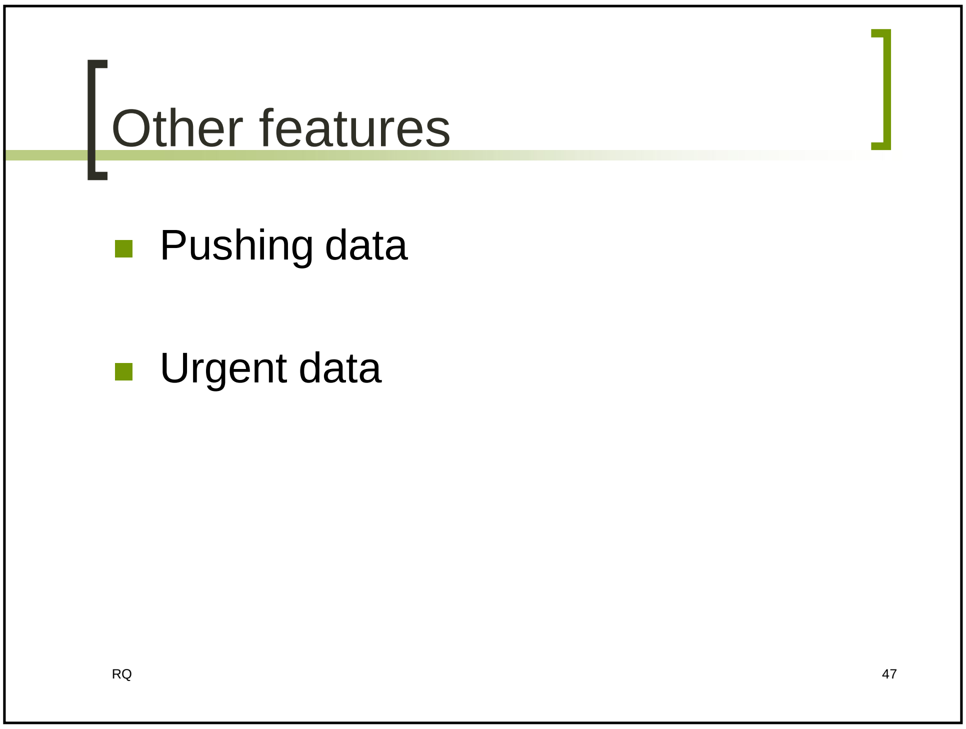


Lost acknowledgment



TCP timers





Other features

- Pushing data
- Urgent data