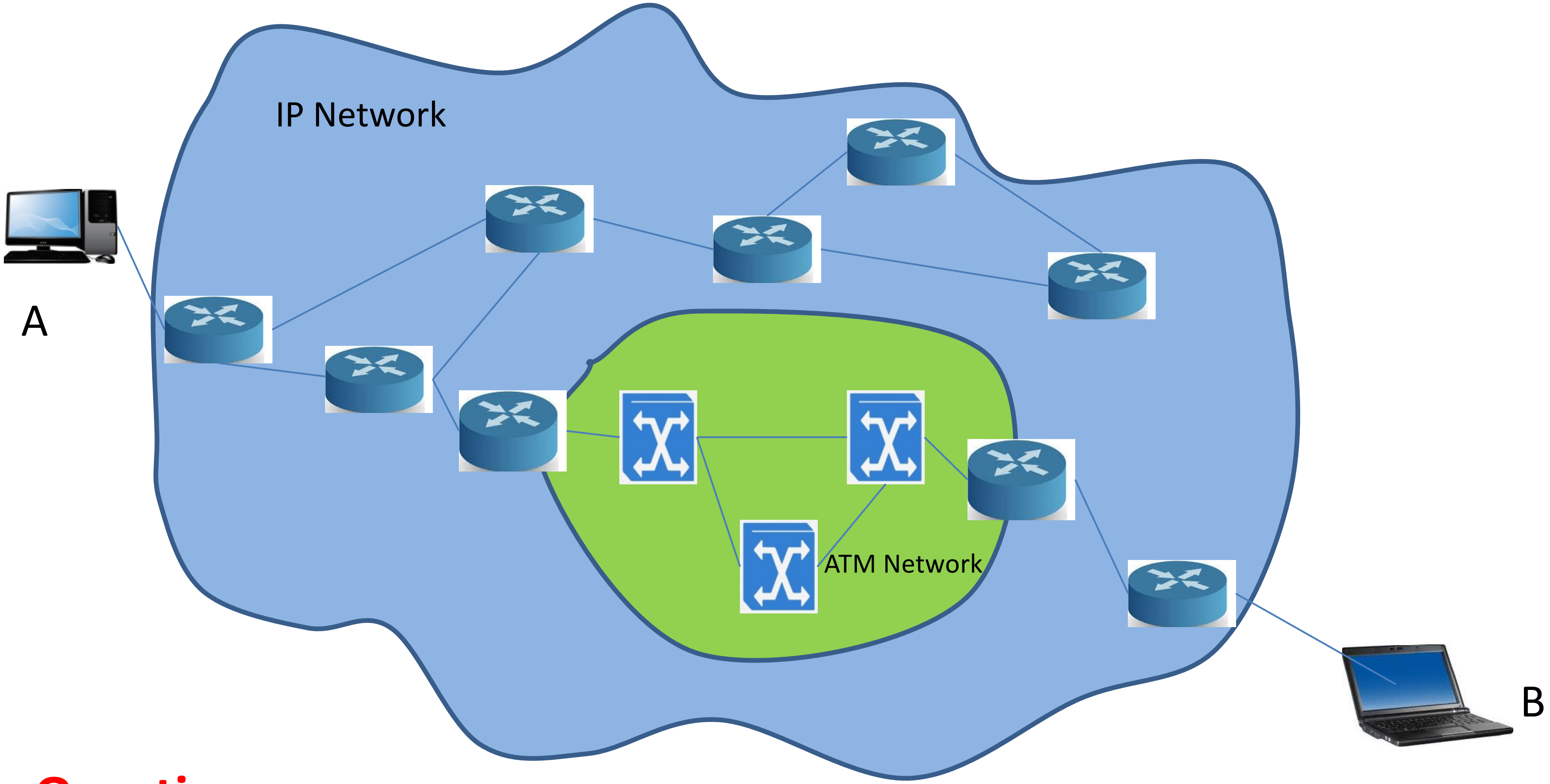
Computer Networks

Transport Layer – TCP/UDP

Nassim KOBEISSY

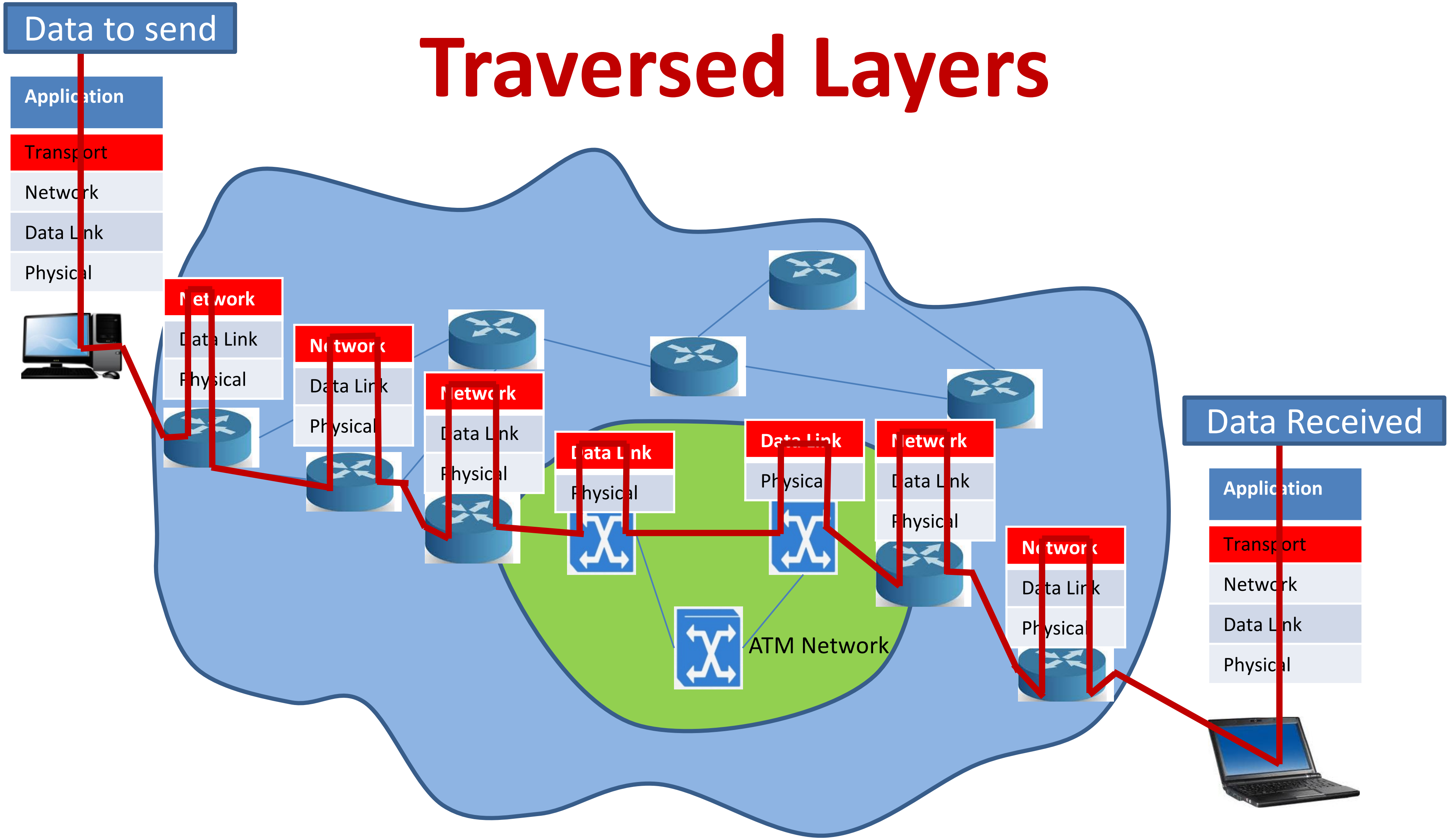
Recall: Networking Layers



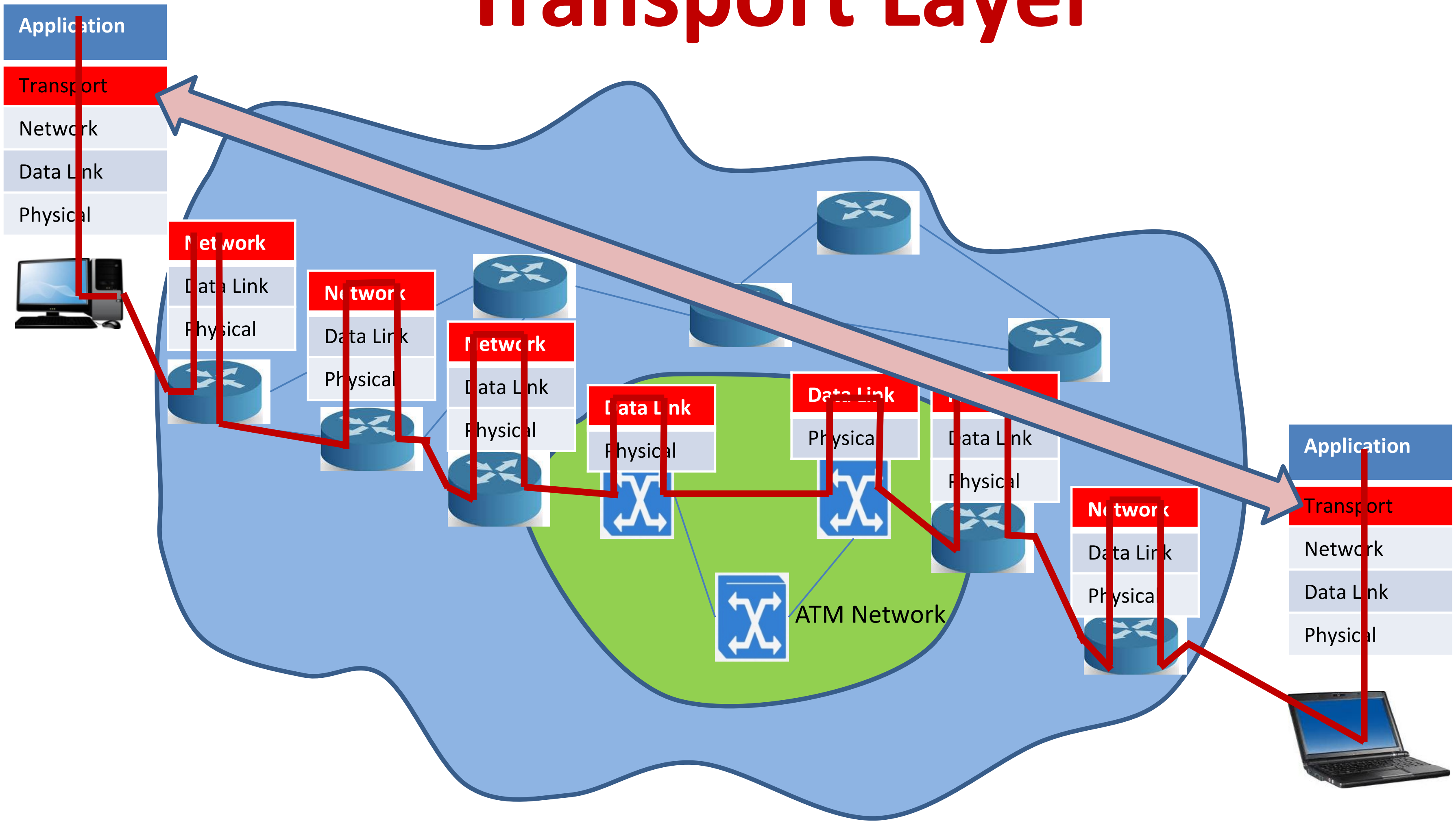
Question:

What are the involved layers on each networking device in the communication between A and B ?

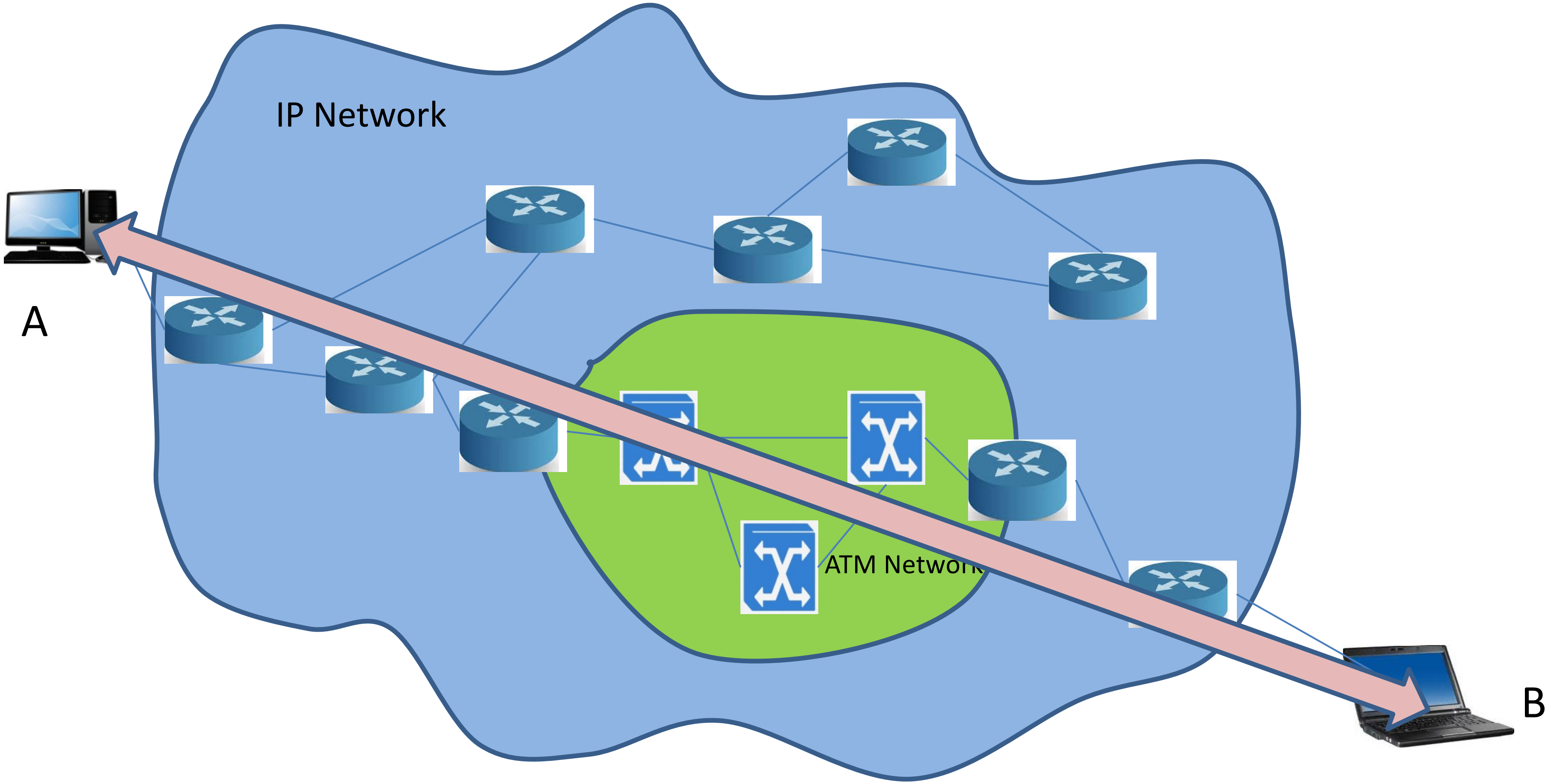
Traversed Layers



Transport Layer



Transport Layer

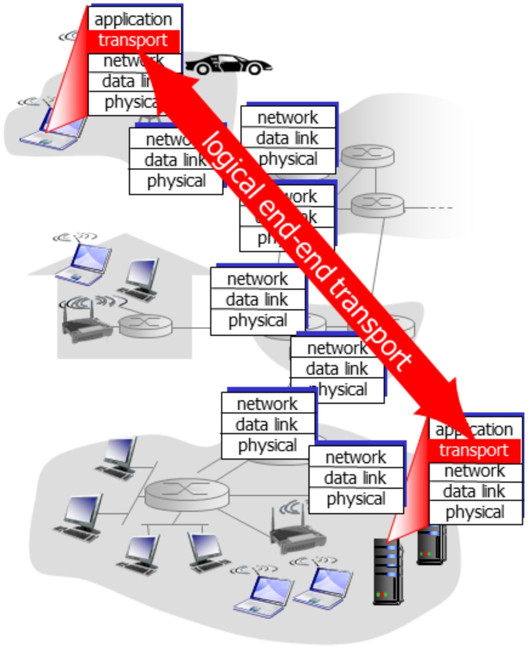


Provides data transport from a process on a source machine to a process on a destination machine.

The Transport Layer

Role

- provide *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
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - no-frills extension of “best-effort” IP

Services Provided to Application Layer

Like Network layer, Transport layer provides 2 services:

- Connectionless:
  - Just send datagrams (TPDU: Transport Protocol Data Unit) whenever data is received from applications
  - No acknowledgements, no connection establishment
  - Example: UDP
- Reliable Connection-Oriented
  - Connection establishment before communication
  - Reliable transport
  - Connection release
  - Example: TCP

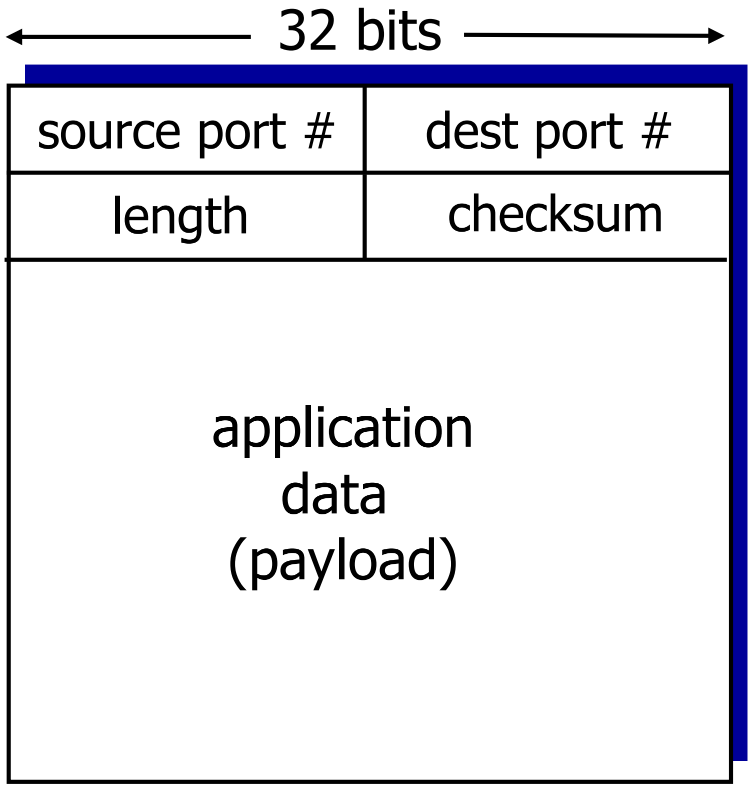
User Datagram Protocol (UDP)

UDP

- RFC 768
- Connectionless protocol
  - It provides no packet sequencing
  - May lose packets
  - Does not check for duplicates
- It does almost nothing beyond sending packets between applications
- 8 byte header - Small header
- Ports identify end-points (processes) in communicating machines
  - Port: think of it as a mailbox that the application rents to receive packets
  - When receiving a segment, the transport layer looks at the destination port field and learns to which process it shall deliver it.
- It is used by applications that do not need a high-reliable transport service:
  - real-time applications are sensible to delay and jitter (but can tolerate some losses)
  - Video streaming is sensible to jitter and bandwidth (but can tolerate some losses)
- It is also useful in client-server situations
  - Client sends a short request and expects a short replay back.
  - Timeouts are used, if the request or response are lost, just send back
  - Application Example: DNS

UDP: segment header

- Includes the 8B header and the data
- Maximum length is 65515 B (size limit on IP packet)



32 bits

source port # | dest port #

length | checksum

application data (payload)

UDP segment format

DNS With UDP

DNS: Domain Name System

- Client needs to learn the IP address of www.google.com
- It sends a UDP packet containing the host name to the DNS server.
- The server replies with a UDP packet.
- No setup is needed in advance, No release afterwards, no flow control …
- If the request or the response are lost, it is still less expensive to resend a new request than sending all the connection establishment and release packets of TCP

Transmission Control Protocol (TCP)

TCP

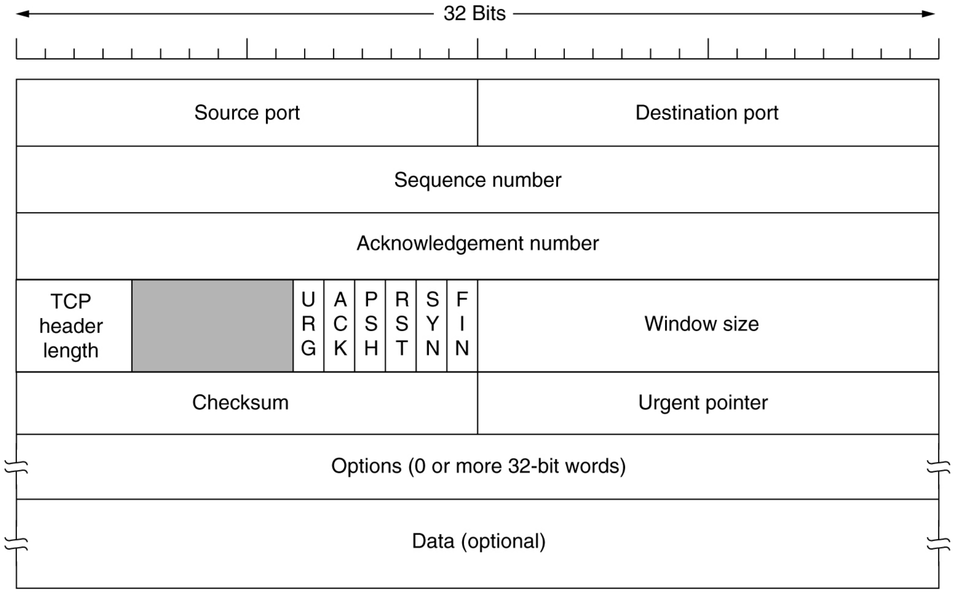
- Connection oriented
  - Connection establishment and release
- Stream-of-bytes service
  - Sends and receives a stream of bytes, not messages
- Reliable, in-order delivery
  - Checksums to detect corrupted data
  - Acknowledgments & retransmissions for reliable delivery
  - Sequence numbers to detect losses and reorder data
  - ***Each byte is numbered***
- Flow control
  - Prevent overflow of the receiver's buffer space
  - Exchange windows sizes during connection establishment
- Congestion control
  - Uses end-to-end mechanisms to discover and adapt to network state (congestion, break downs…..)

TCP Reliability

- **Checksums**
  - Used to detect corrupted data at the receiver
- **Sequence numbers**
  - Used to detect missing data
  - And for re-ordering segments to constitute original message
- **Retransmissions**
  - Sender retransmits lost or corrupted data
  - Timeouts are used to detect losses. They are calculated based on estimates of round-trip time
  - Fast retransmit algorithm is used for rapid retransmission

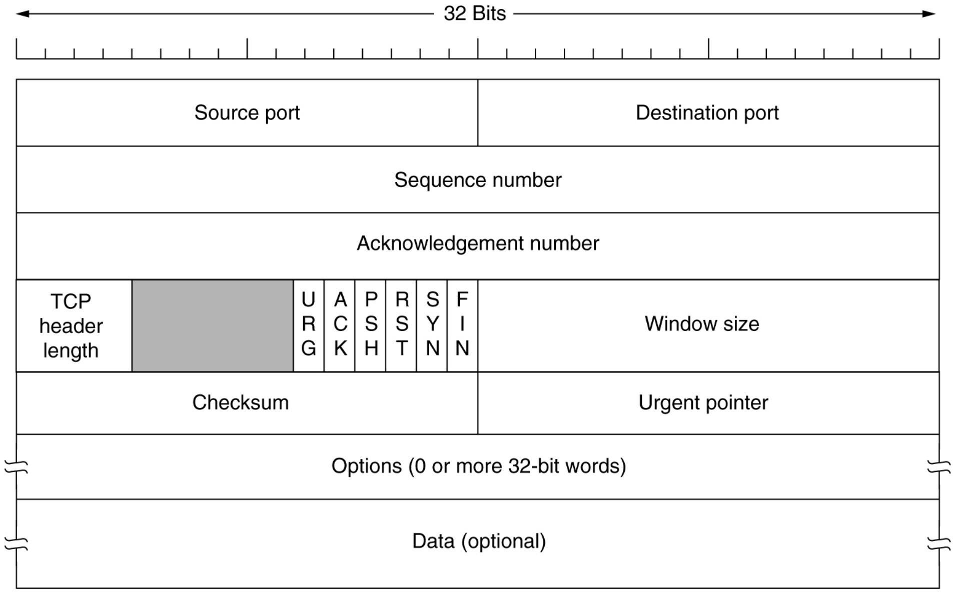
TCP Header

- **Ports:** 16 bits identify with IP addresses TCP connections.
  - Connection identifier is 5 tuple (TCP, src port, srt IP, dest port, dest IP)
  - rfc1700
  - Well-known ports range from 0 through 1023.
    - Ex 443 HTTPS, 80 HTTP
  - Registered ports are 1024 to 49151.
  - Dynamic ports (also called private ports) are 49152 to 65535.
- **Sequence number:** The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.
- **Acknowledgement Number:** Next expected TCP octet
  - ACK is cumulative



TCP Header

- **Header Length:** Number of 32-bit words in the header
- **URG** indicates that this segment contains an Urgent pointer field.
- **ACK** indicates that this segment contains an Acknowledgement field. 1 = Ack, 0 = No Ack.
- **PSH**
  - requests a Push (PSH=1). TCP software usually gathers enough data to fill the transmit buffer prior to transmitting the data. If an application requires data to be transmitted even though a buffer may not be full then a PUSH flag bit is set. At the receive side the PUSH makes the data available to the application without delay.
- **RST** The reset field will Reset the connection. When received, means that there is a problem.
- **SYN** is used to Synchronize sequence numbers to initiate a connection (used for connection establishment).
- **FIN** is used release a connection.
  - 1 = Fin, 0 = No Fin.
  - It specifies that sender has no more data to transmit

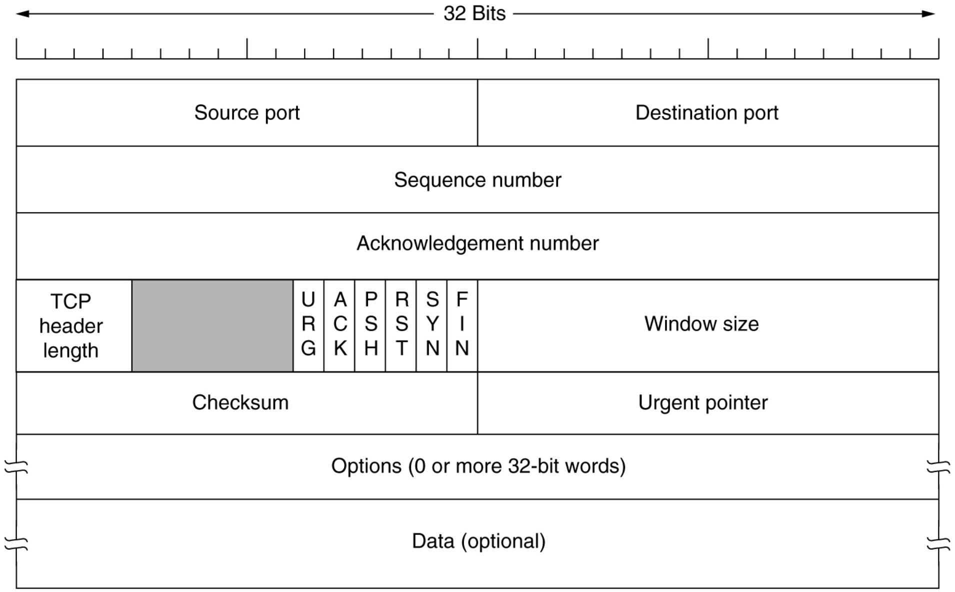


32 Bits

Source port								Destination port
Sequence number								
Acknowledgement number								
TCP header length		URG	ACK	PSH	RST	SYN	FIN	Window size
Checksum								Urgent pointer
Options (0 or more 32-bit words)								
Data (optional)								

TCP Header

- **Checksum**
  - ❑ provides extra reliability
  - ❑ Checksums the header , data and a pseudoheader.
- **Urgent Pointer**
  - ❑ Used by the sender to transmit emergency data to the receiver.
  - ❑ The URG flag must be set.
  - ❑ The Urgent Pointer is a 16 bit positive offset that is added to the sequence number field in the TCP header to obtain the sequence number of the last byte of the urgent data.
  - ❑ Practical use: indicate the pressing of an interrupt key during Telnet/Rlogin or a file transfer abort during FTP.
- **Window Size**
  - ❑ Tells how many bytes may be sent starting at the byte acknowledged. (States max block size that can be sent without ACK)
  - ❑ Used in flow control



32 Bits

Source port

Destination port

Sequence number

Acknowledgement number

TCP header length

URG

ACK

PSH

RST

SYN

FIN

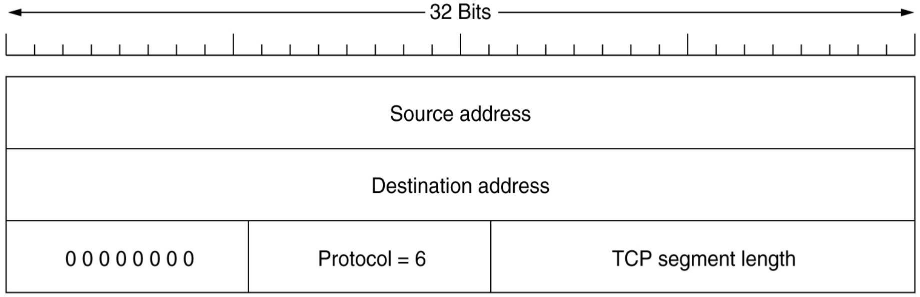
Window size

Checksum

Urgent pointer

Options (0 or more 32-bit words)

Data (optional)



32 Bits

Source address

Destination address

0 0 0 0 0 0 0 0

Protocol = 6

TCP segment length

The pseudoheader included in the TCP checksum.

TCP Connection Management

TCP Connection Management

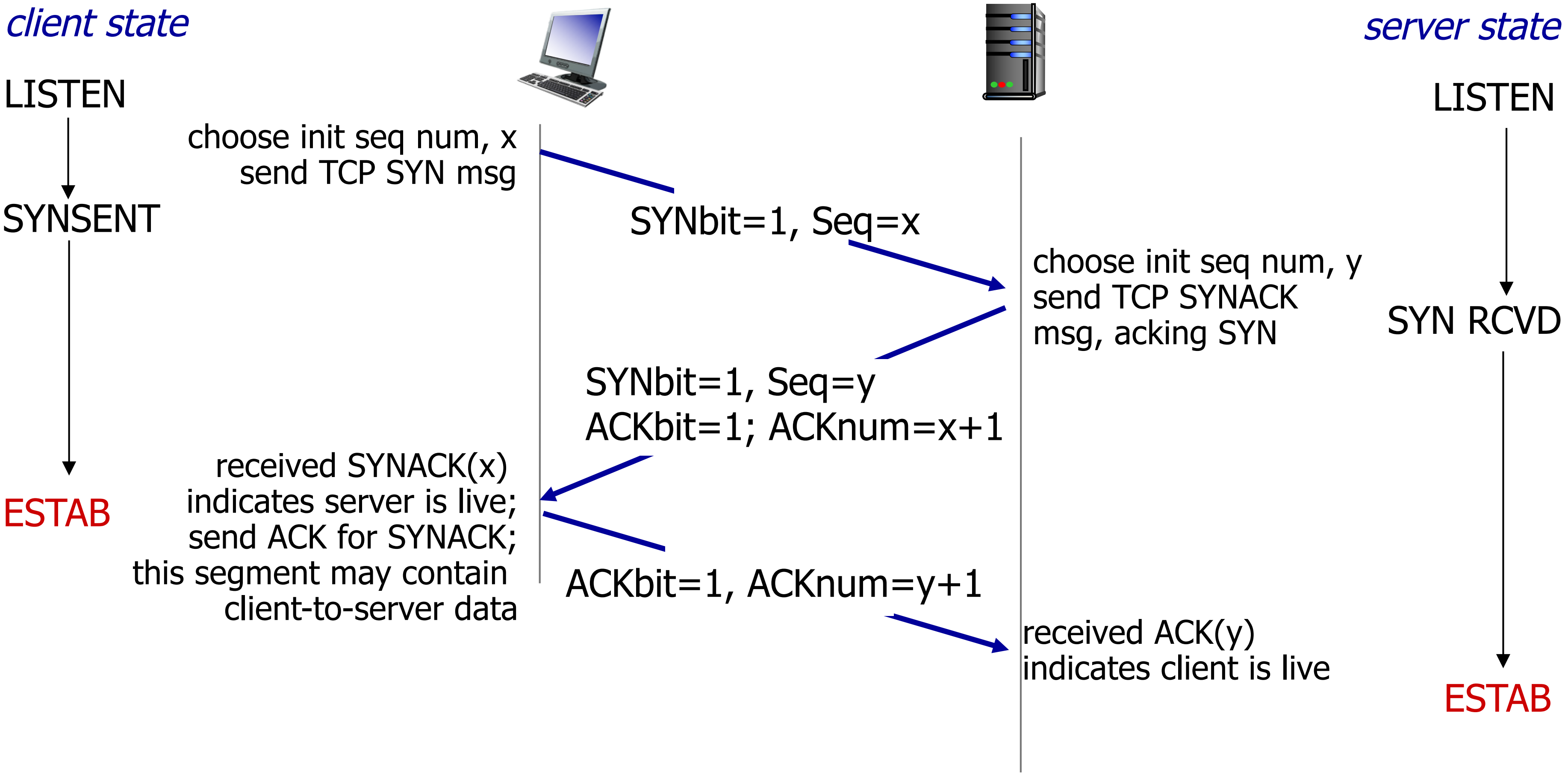
before exchanging data, sender/receiver "handshake":

- agree to establish connection (each knowing the other willing to establish connection)
- agree on connection parameters

closing a connection

- client, server each close their side of connection
  - send TCP segment with FIN bit = 1
- respond to received FIN with ACK
  - on receiving FIN, ACK can be combined with own FIN
- simultaneous FIN exchanges can be handled

Three-way handshake to establish connection



client state

LISTEN

choose init seq num, x
send TCP SYN msg

SYNSENT

SYNbit=1, Seq=x

server state

LISTEN

choose init seq num, y
send TCP SYNACK
msg, acking SYN

SYN RCVD

SYNbit=1, Seq=y
ACKbit=1; ACKnum=x+1

received SYNACK(x)
indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

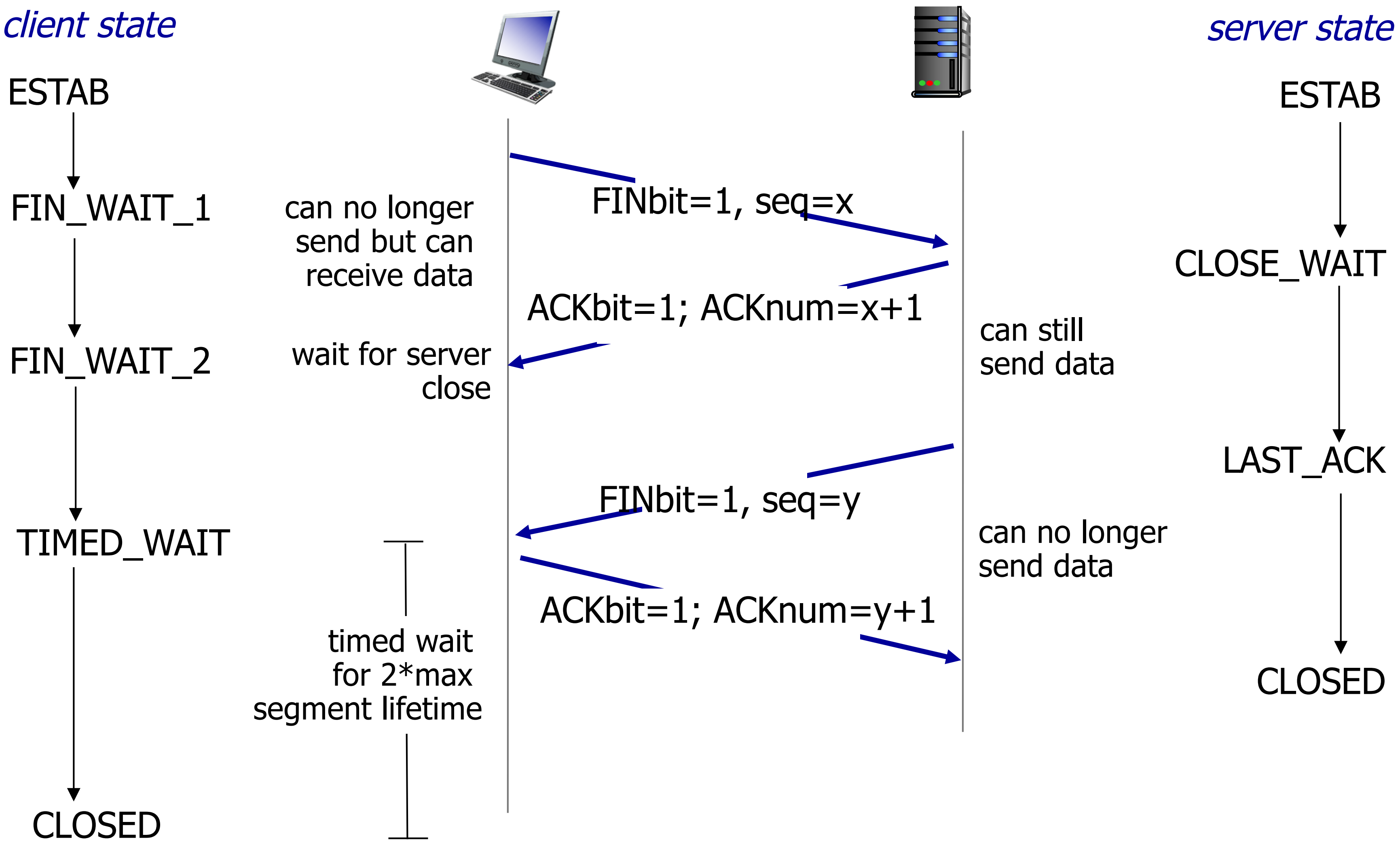
ESTAB

ACKbit=1, ACKnum=y+1

received ACK(y)
indicates client is live

ESTAB

TCP: closing a connection - Four-way handshake



client state

ESTAB

FIN_WAIT_1

can no longer send but can receive data

FINbit=1, seq=x

server state

ESTAB

CLOSE_WAIT

ACKbit=1; ACKnum=x+1

FIN_WAIT_2

wait for server close

can still send data

LAST_ACK

FINbit=1, seq=y

TIMED_WAIT

can no longer send data

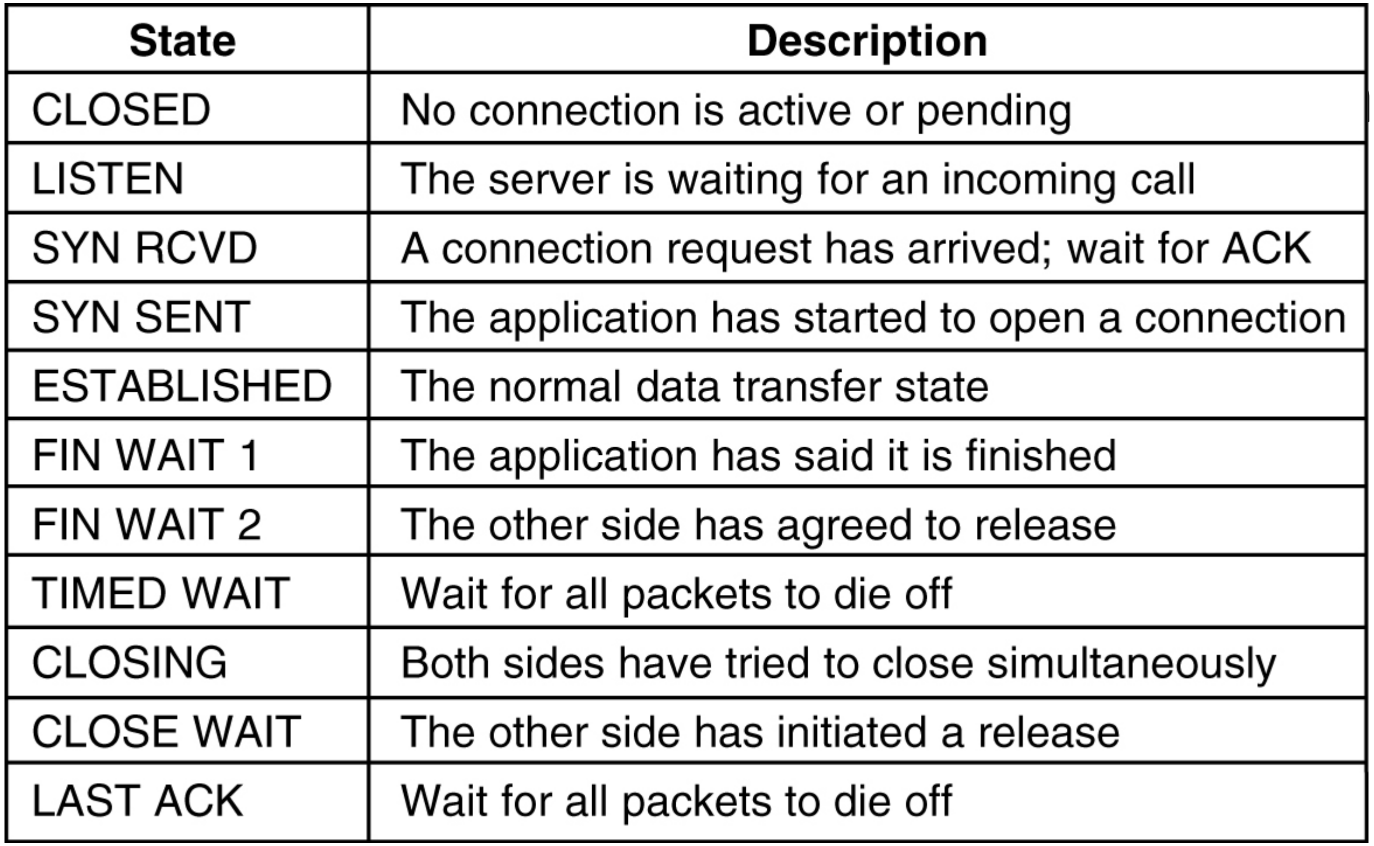
ACKbit=1; ACKnum=y+1

timed wait for 2*max segment lifetime

CLOSED

CLOSED

TCP Connection Management

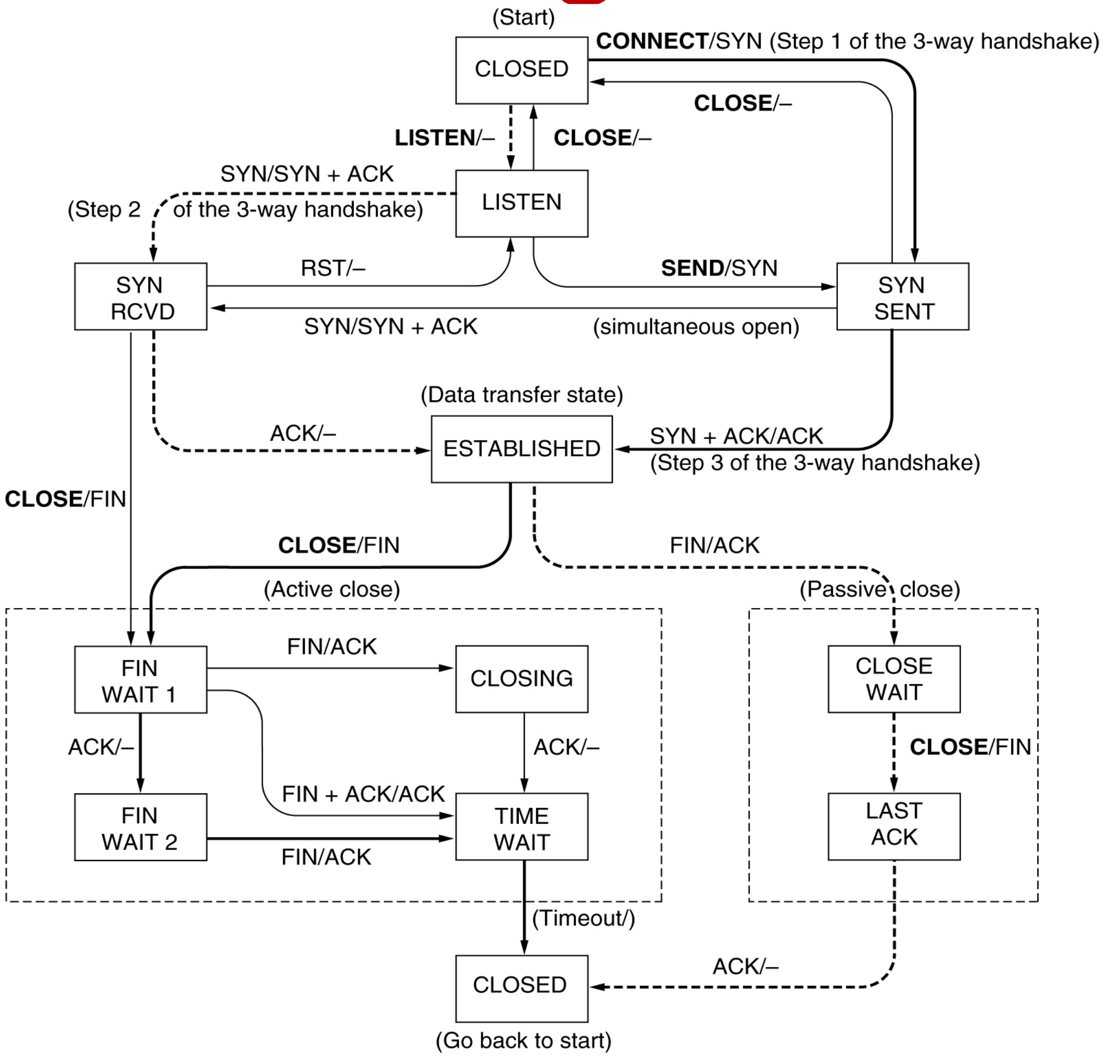


State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

The states used in the TCP connection management finite state machine[1]

TCP Connection Management

- The heavy solid line is the normal path for a client.
- The heavy dashed line is the normal path for a server.
- The light lines are unusual events.
- Each transition is labeled with the event causing it and the action resulting from it, separated by a slash.



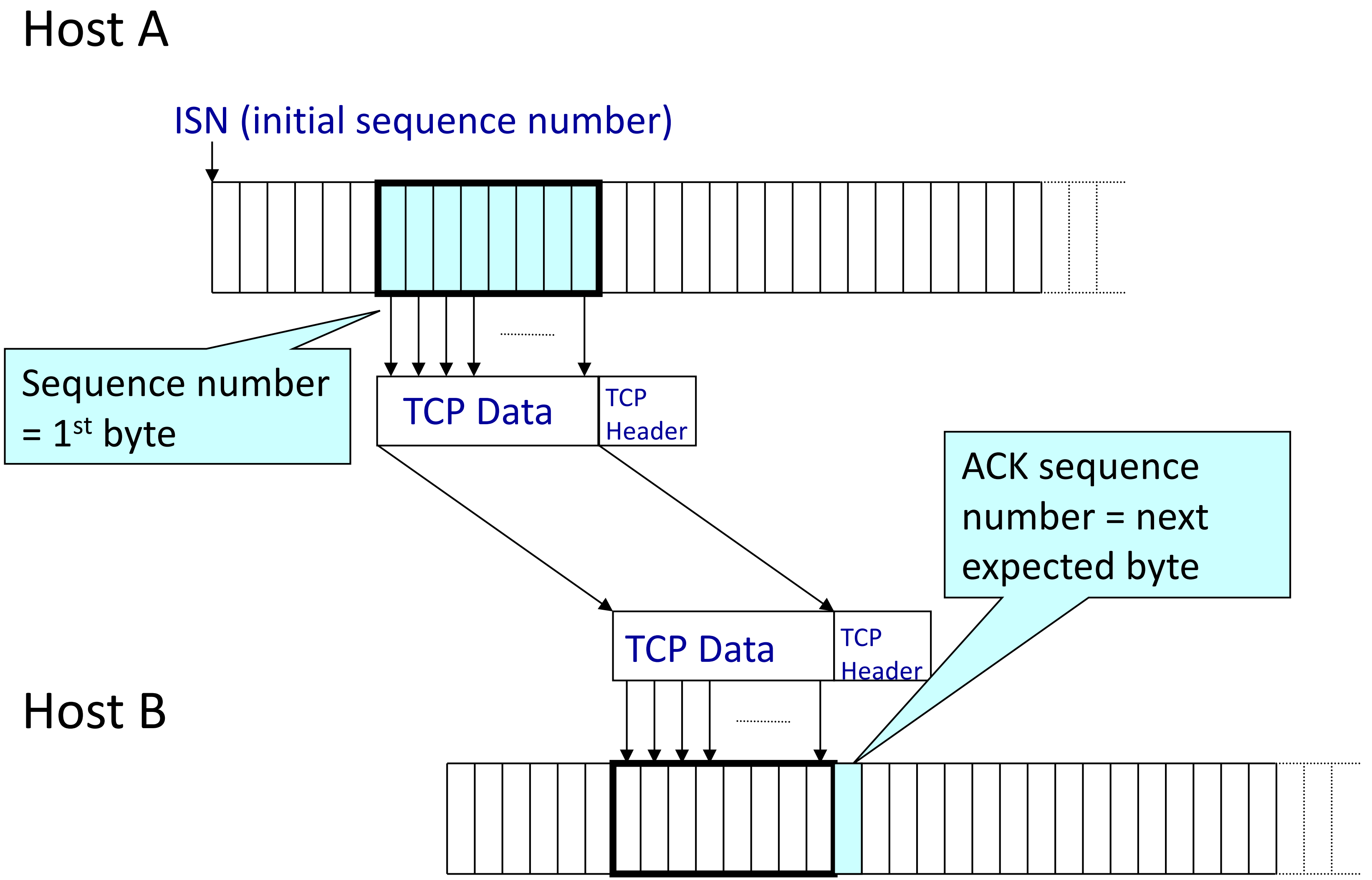
TCP connection management finite state machine[1]

TCP Flow Control

TCP Segments and Flow Control

- TCP splits large data messages into multiple segments.
- Hosts can specify maximum segment size (**MSS**) by using one of the header options.
- Two practical constraints are crucial:
  - The TCP header and data shall fit in the 65,515 byte IP
  - Each link has an **MTU** (Maximal Transfer Unit). **In practice, the 1500 B Ethernet payload defines the upper bound on segment size**.
- Each byte is attributed a sequence number
- TCP provides flow control through “windowing”
- Window size determines how much data (in number of bytes) is sent at a time (without receiving ack)
- Flow control is done using
  - Window size
  - Sequence numbers
  - ACK numbers

TCP Segments and Flow Control



Host A

ISN (initial sequence number)

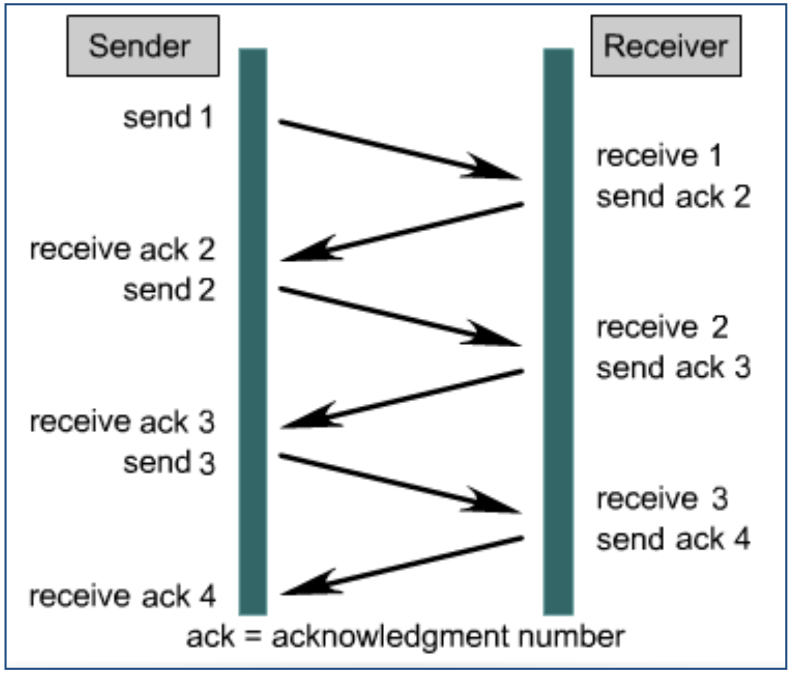
Sequence number = 1st byte

ACK sequence number = next expected byte

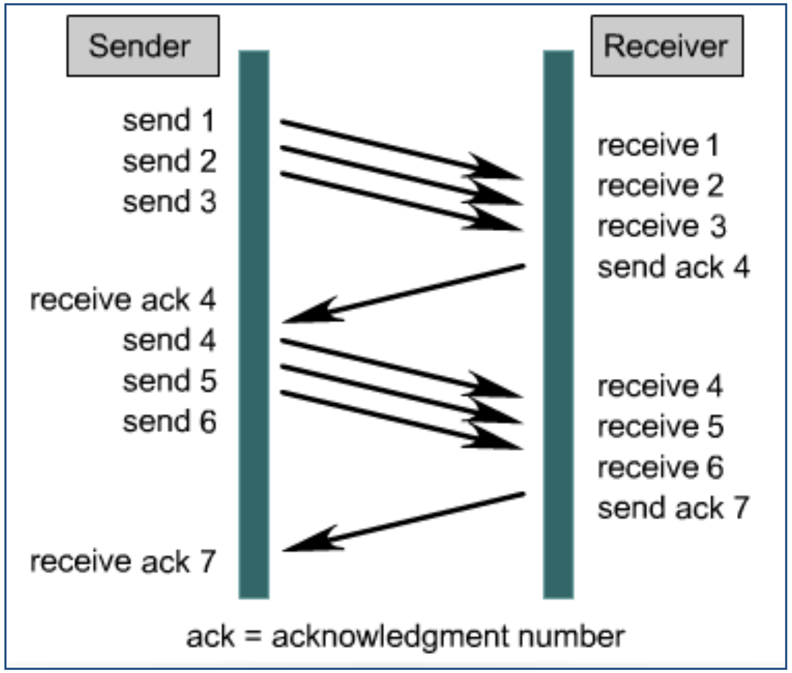
Host B

TCP Windows and Flow Control

Window Size = 1



Window Size = 3



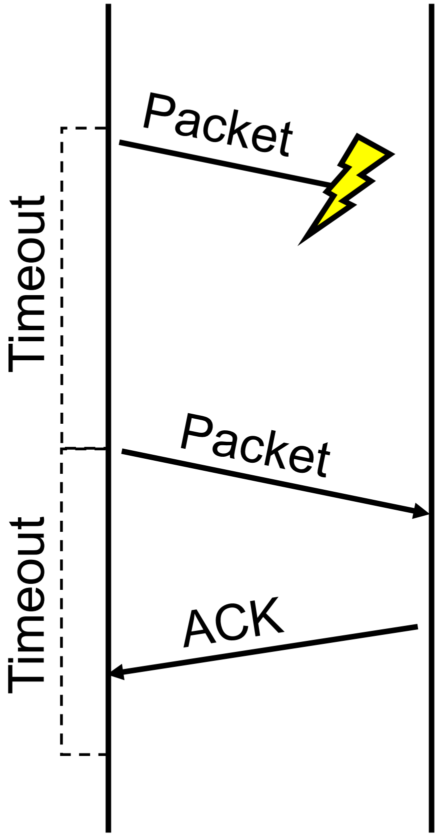
Question: What is the best Window size for controlling flows between senders and receivers?

- TCP goodput increases as window size increases and **as no losses are occurring**
- Finally window size depends on machines capacity (buffer memory, processing speed…)

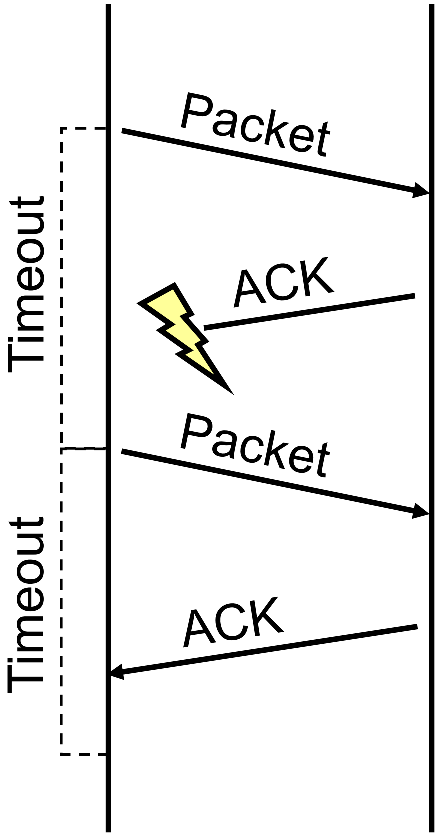
TCP Timer Management

TCP Timer Management

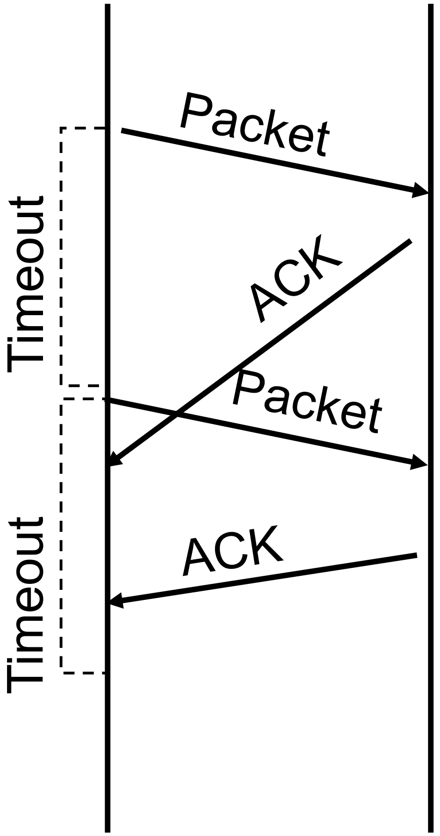
- •TCP senders use timeouts for each packet sent. It helps determining retransmissions
- •TCP tries to choose the best timer value (How ??)
- •Timer calculations depend essentially on the round trip time



Packet lost

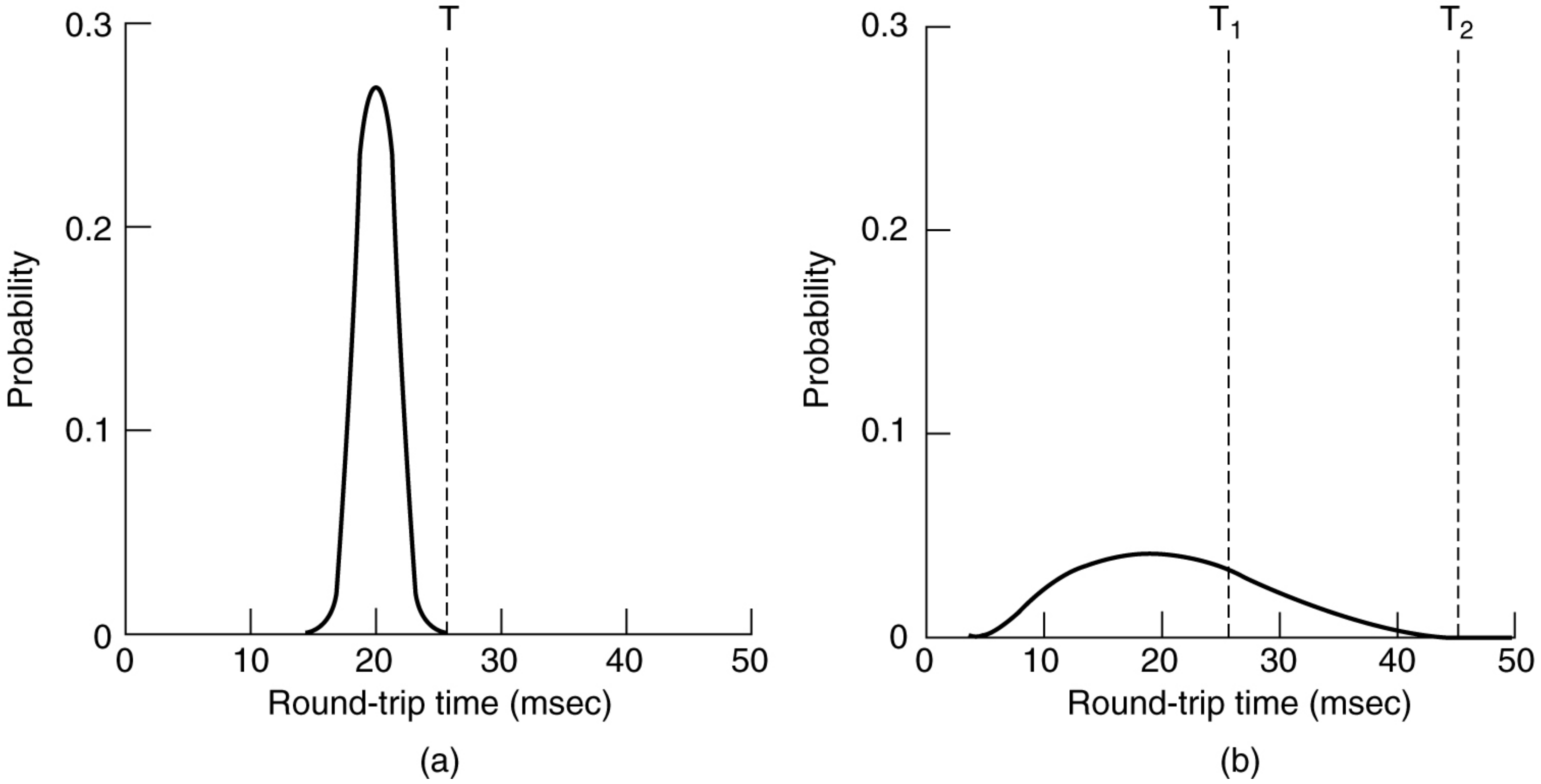


ACK lost



Early timeout

TCP Timer Management



(a) Probability density of ACK arrival times in the data link layer [1].

(b) Probability density of ACK arrival times for TCP[1].

TCP Timer Management

- But, how does the sender know the RTT?
  - TCP calculates RTT by measuring how long the ACKs took

Smooth estimate RTT (SRTT): keep a running average of the RTT

  - SRTT = a * SRTT + (1 –a) * RTT
  - Typical value of a = 7/8
  - SRTT initial Value between 1 and 3 seconds.
- Compute Re-Transmission Timeout **RTO= 2 * SRTT**

With Variance

- Experience showed that **fixed value for RTO is not flexible**
- Moreover average calculations are not sufficient and the variance shall be considered
  - RTTVAR = b * RTTVAR + (1 –b) * (SRTT - RTT)
  - Typical value for b = ¾
  - RTTVAR initial value : RTT/2
- Finally: **RTO= SRTT + 4 * RTTVAR**

TCP Congestion Control

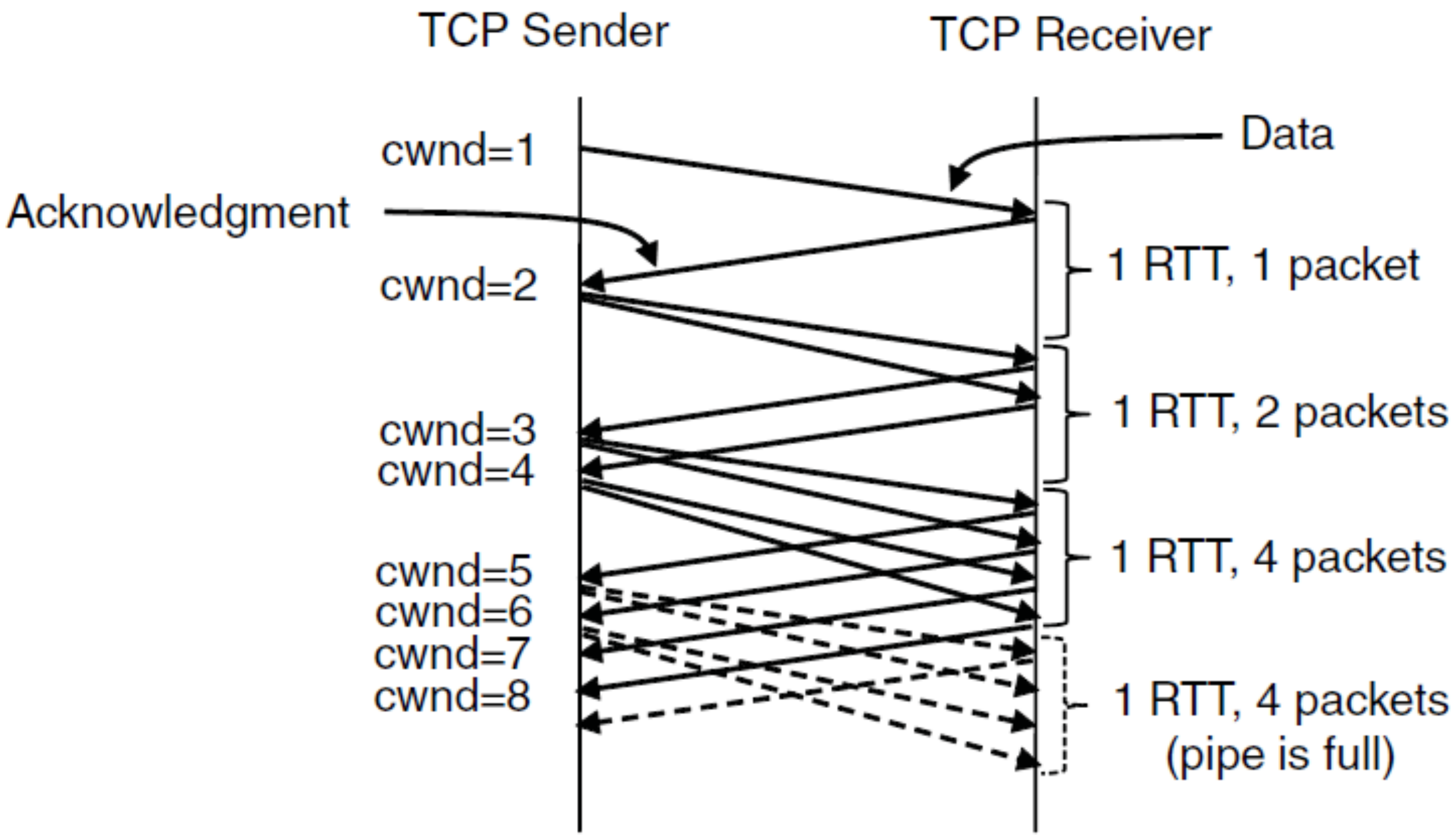
TCP Congestion Control

- TCP controls congestion of the network for improving performance and at the same time ensuring reliability
- Congestion is a network (especially core network) property
- When routers become full, they just drop packets
- **TCP assumes a good network transmission quality**
  - BER (Bit Error Rate) is very low in cables and in fiber
  - In wireless mediums, retransmissions are performed at the data link level.
- **When losses occur, TCP assumes that there is a congestion in the network ➔ it shall control it**
- The timeouts may also be small they must be managed dynamically (as we see previously)

TCP Congestion Control

- TCP adapts the transmission rate to the congestion state of the network
- It starts slowly and then accelerates transmission rate until a packet loss occurs. It then slows down.
- Three parameters are tracked for controlling congestion:
  - Congestion window size: **CWND**
  - **Sequence numbers**
  - **ACK numbers**
- **CWND** is calculated with AIMD rule.
- **AIMD**: Additive Increase Multiple Decrease.

TCP Congestion Control: Slow Start



Slow start from an initial congestion window of 1 segment [1]

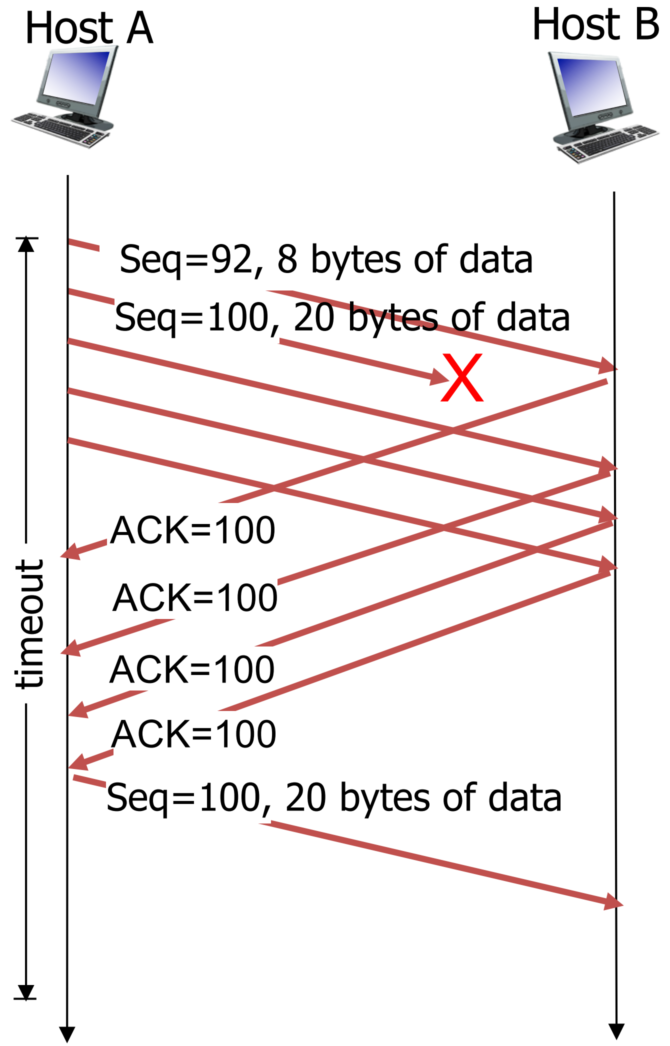
RTT: Round Trip Time

TCP Congestion Control

- **Slow start: Is it really slow ???**
- Finally, slow start technique is not slow at all.
- The congestion window size grows exponentially.
- Each packet acknowledged increases CWND
- In presence of segments to send on the TCP connection, **CWND CAN BE doubled each RTT**
- In fact, **CWND ← CWND + MSS (for each ACK received)**

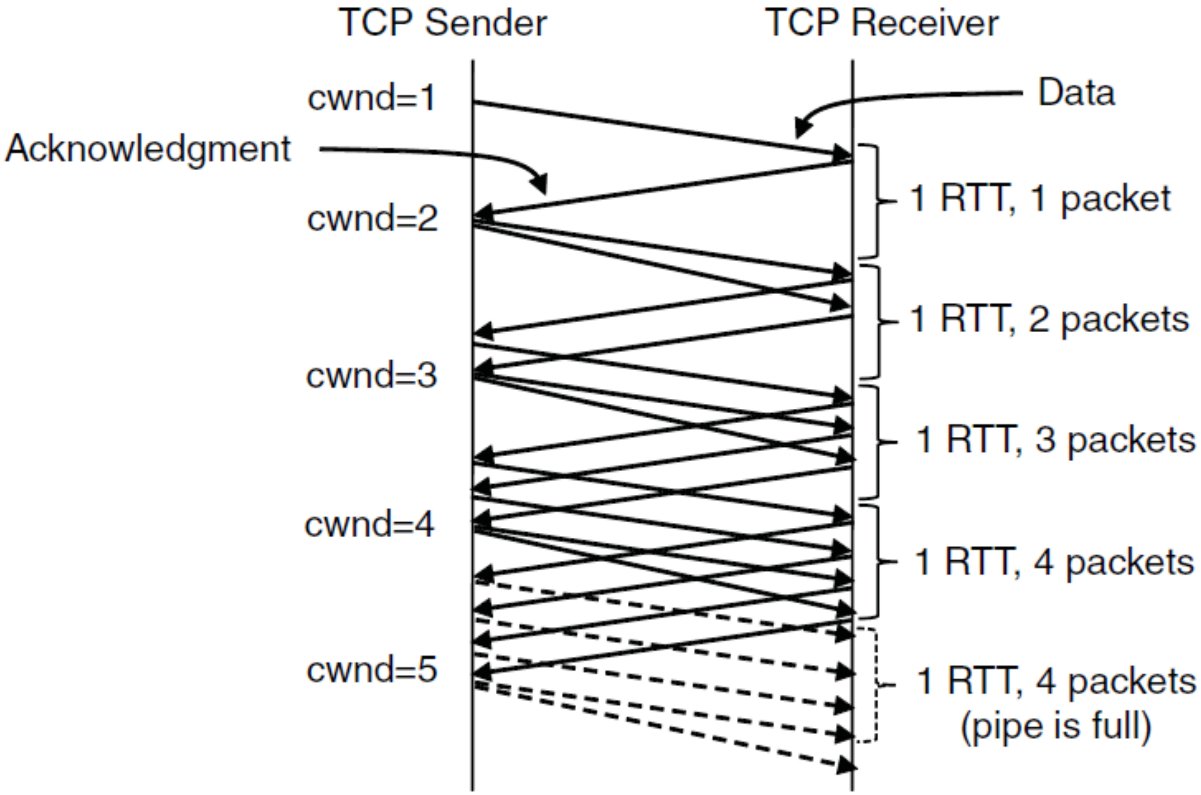
TCP fast retransmit

fast retransmit after sender receipt of triple duplicate ACK



Source : Adapted from Computer Networking: A Top Down Approach

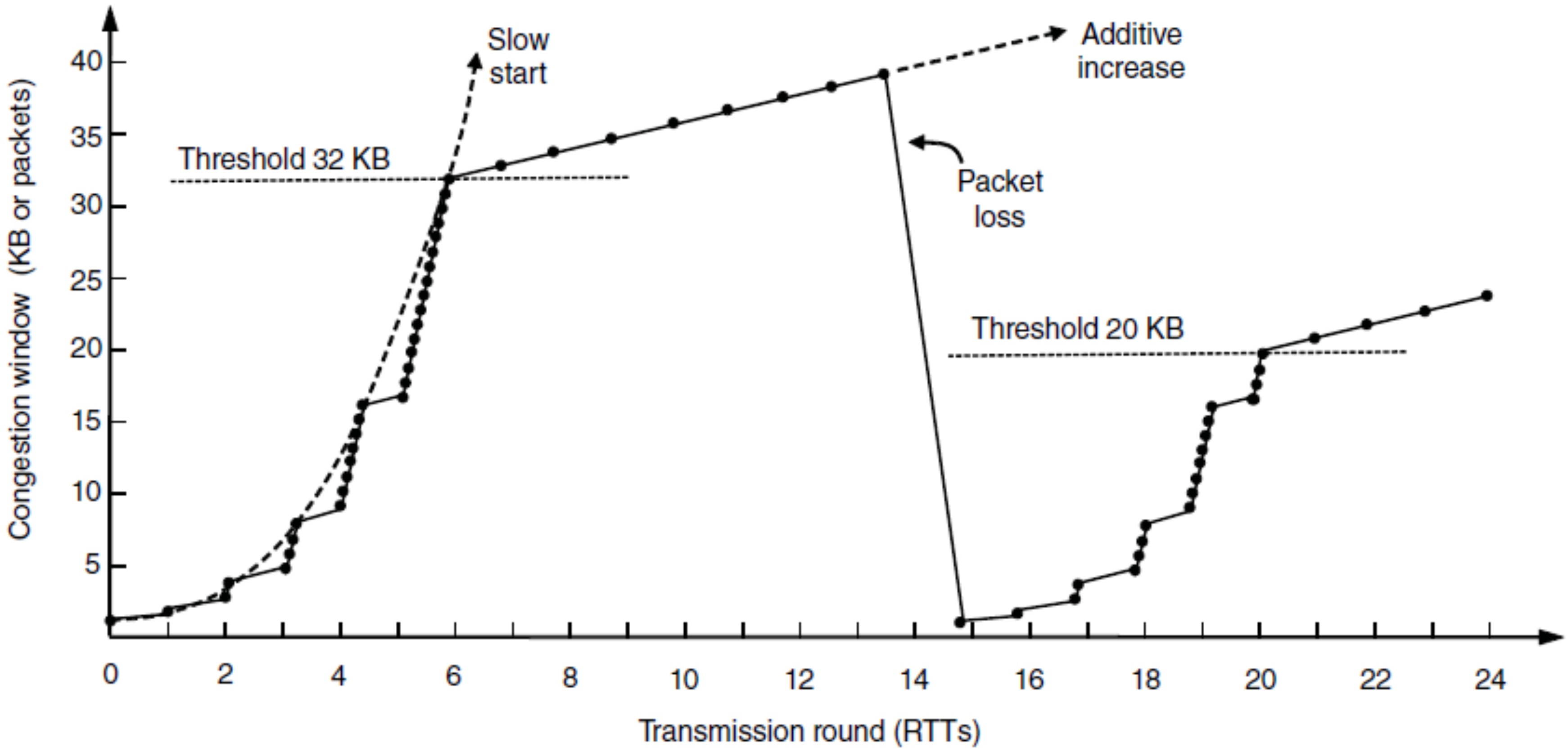
TCP Congestion Control: Additive Increase



Additive increase from an initial congestion window of 1 segment [1]

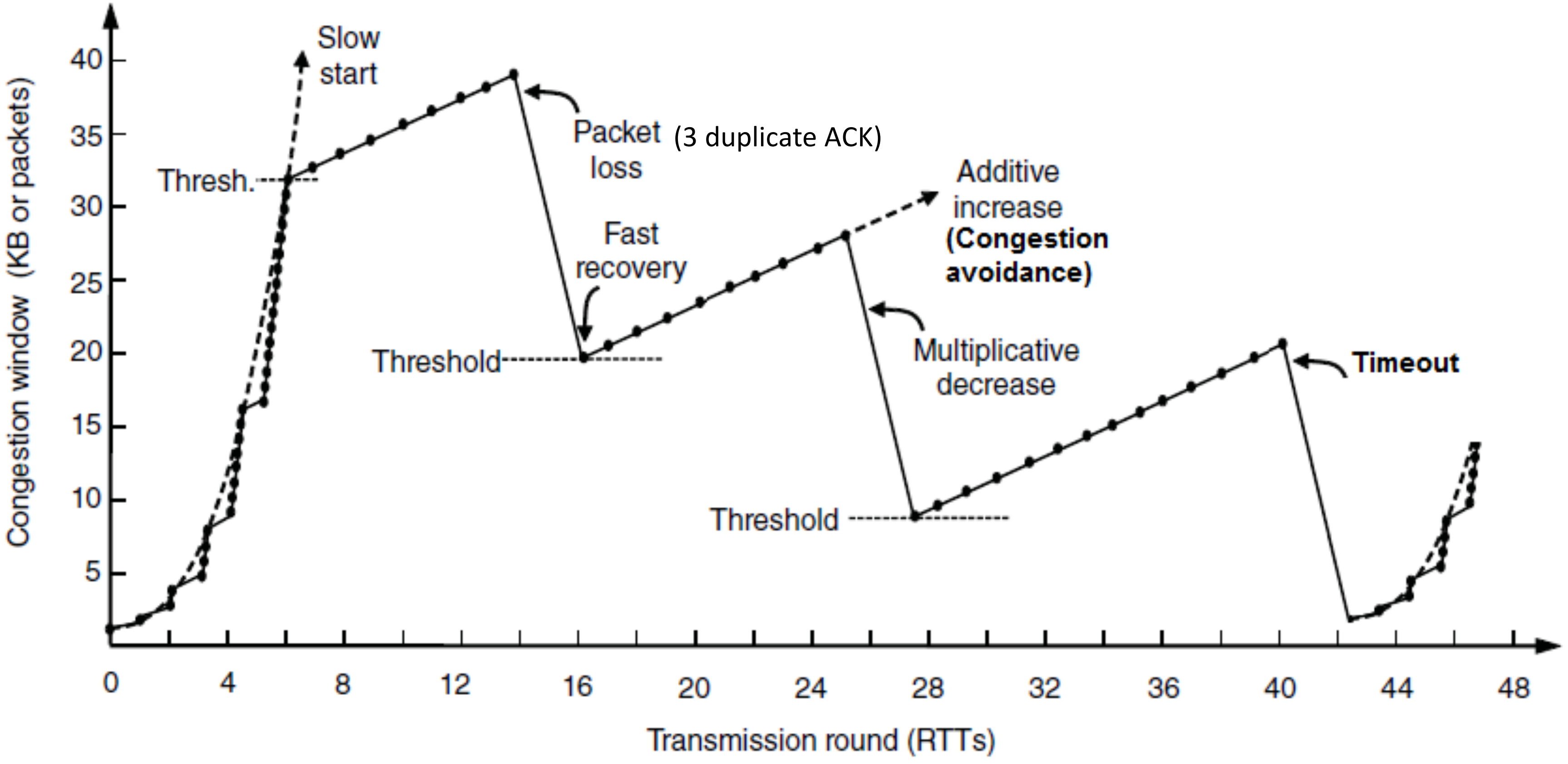
With this method, cwnd CAN be incremented (+1) each RTT ➔ really slow

TCP Congestion Control



Slow start followed by additive increase in TCP Tahoe [1].

TCP Congestion Control



Slow start, Congestion avoidance and Fast recovery of TCP Reno [1].

Some Transport Known Ports

Port	Transport Protocol	Application Protocol	Use
21	TCP	FTP	**File Transfer Protocol**
23	TCP/UDP	Telnet	Remote login
25	TCP	SMTP	**Simple Mail Transfer Protocol:** E-mail routing between email servers
53	TCP/UDP	DNS	**Domain Name System**
69	UDP	TFTP	**Trivial File Transfer Protocol**
79	TCP	Finger	Lookup info about a user
80	TCP	HTTP	**Hyper Text Transfer Protocol:** World Wide Web
110	TCP	POP-3	**Post Office Protocol v3:** Remote e-mail access
119	TCP	NNTP	**Network News Transfer Protocol:** Retrieval of newsgroup messages

References

- [1]: Computer Networks, Fifth edition: A.S. Tanenbaum and D. J. Wetherall
- [2]: Computer Networks, A Systems Approach: L.L. Peterson and B.S. Davie