TRANSPORT LAYER PROTOCOLS

TRANSPORT LAYER PROTOCOLS

Layer 4 protocol (in OSI model)

Provides Process-to-Process communication service

Also called as end-to-end prototcol

Performs Multiplexing and Demultiplexing

Transport Layer Protocols

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

Runs over IP

- TCP and UDP packets are encapsulated into IP packets

Use their own control information, stored in packet headers

- Port numbers (indicate consuming program in the destination host)

TRANSMISSION CONTROL PROTOCOL (TCP)

Connection Oriented

Reliable Byte Stream service

Guaranteed in-order delivery

Full-duplex

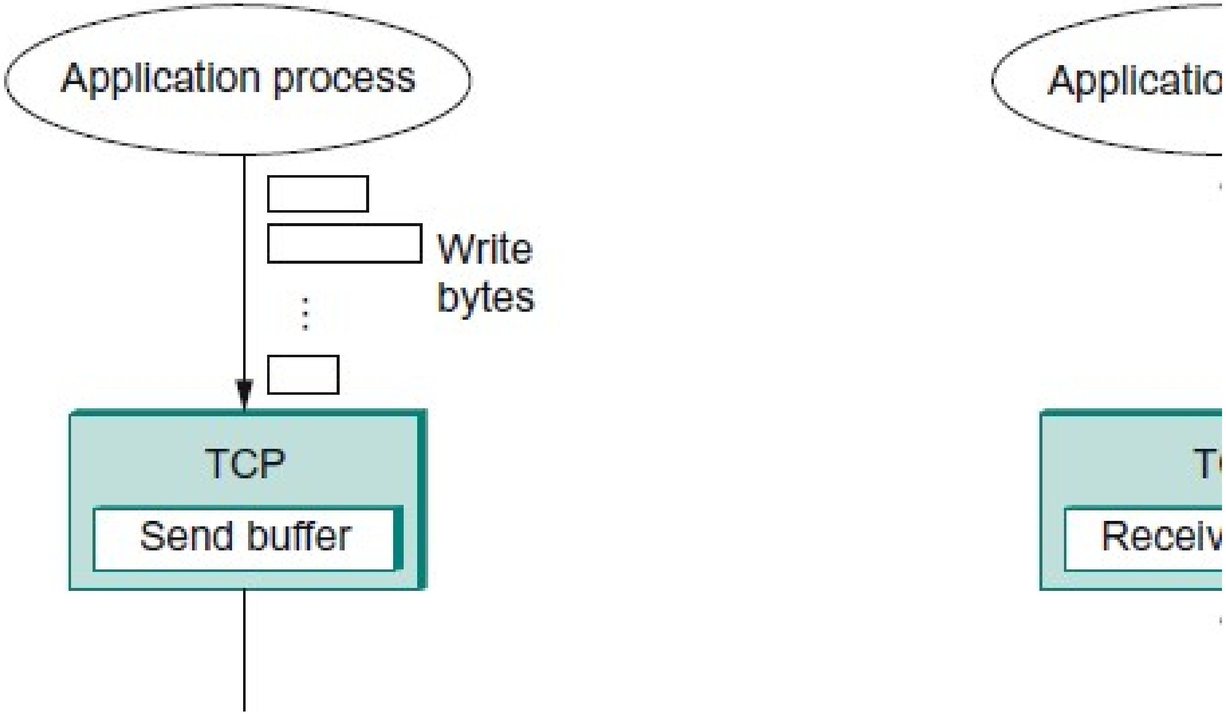
Includes a flow-control mechanism

Implements congestion-control mechanism

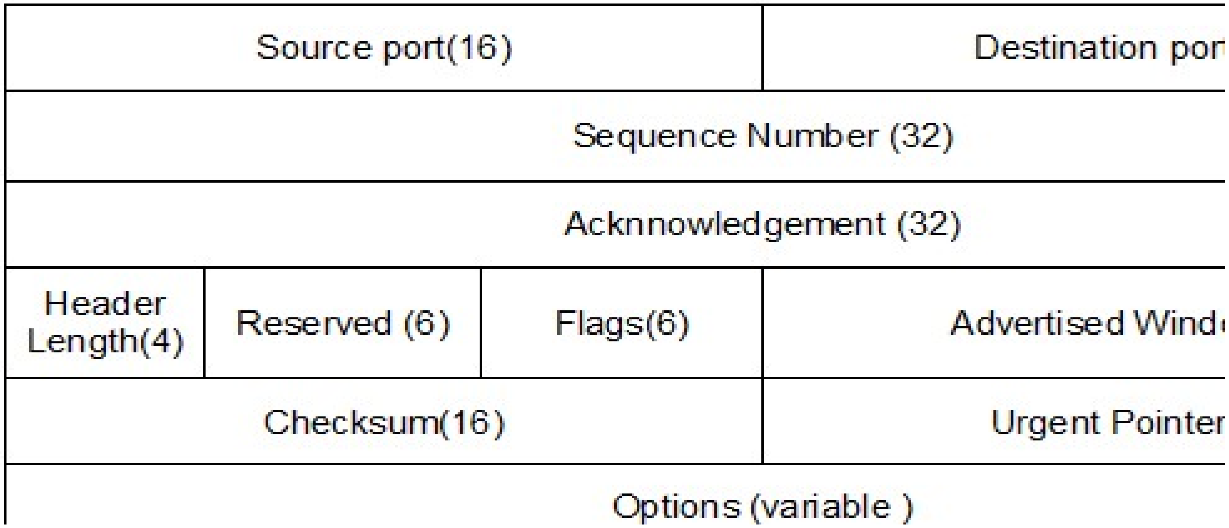
Packets exchanges between TCP peers – Segments

Each Segment has a header

TRANSMISSION CONTROL PROTOCOL (TCP)



TCP HEADER FORMAT



<table>
<tr><td colspan="3">Source port(16)</td><td>Destination por</td></tr>
<tr><td colspan="4">Sequence Number (32)</td></tr>
<tr><td colspan="4">Acknnowledgement (32)</td></tr>
<tr><td>Header Length(4)</td><td>Reserved (6)</td><td>Flags(6)</td><td>Advertised Wind</td></tr>
<tr><td colspan="3">Checksum(16)</td><td>Urgent Pointer</td></tr>
<tr><td colspan="4">Options (variable)</td></tr>
</table>

- Both the TCP header and data must have a length in bits multiple of 32
- {Source port, Source IP, Destination port, Destination IP} – uniquely identifies a TCP Connetion

TCP HEADER

Flags:

- SYN - establishing a TCP connection,
- FIN - terminating a TCP connection
- RST - close the connection
- ACK – acknowledgment
- PSH - push function
- URG - signifies that this segment contains urgent data. If set, the urgent pointer field indicates the starting location of the nonurgent data

USER DATAGRAM PROTOCOL (UDP)

Connectionless

Unreliable

Processes indirectly identify each other using port number (or mailbox)

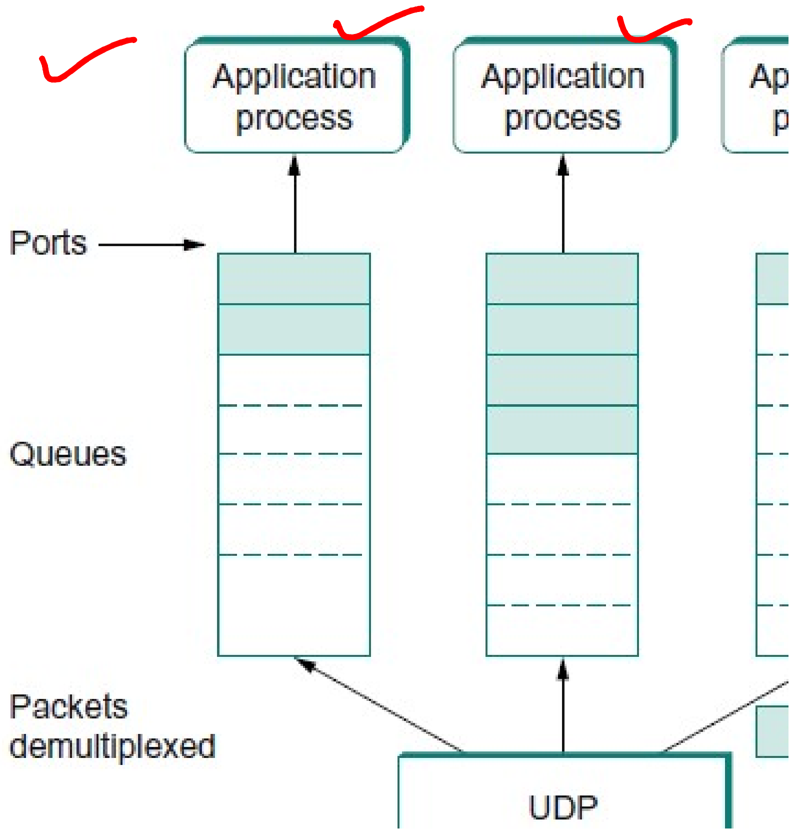
UDP port field is only 16 bits long - there are up to 64K possible ports

Source process sends a message to a port and destination process receives the message from a port

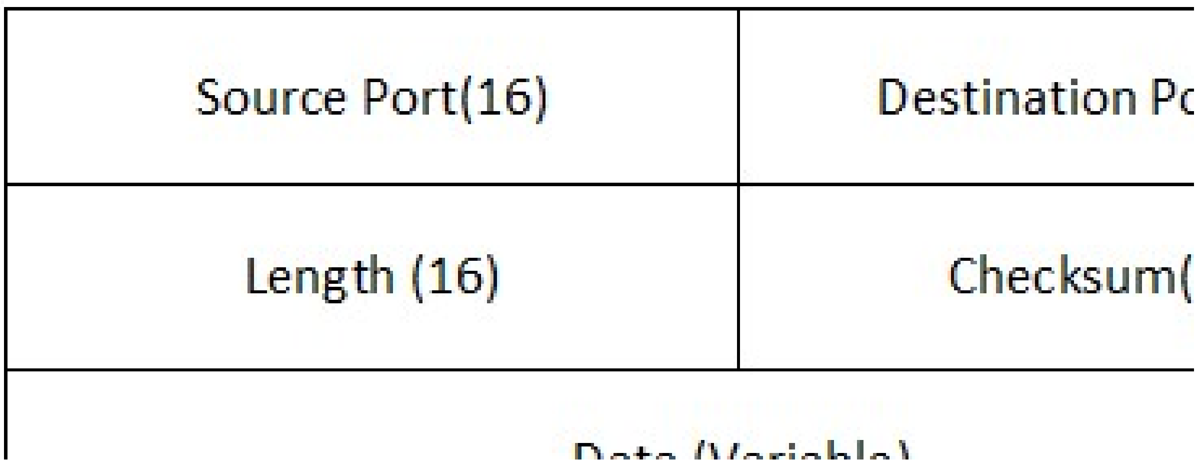
Client know about the server's port either through

- Well known ports
- Port mapper

UDP MESSAGE QUEUE



UDP HEADER



Source Port(16)	Destination Pc
Length (16)	Checksum(
Data (Variable)	

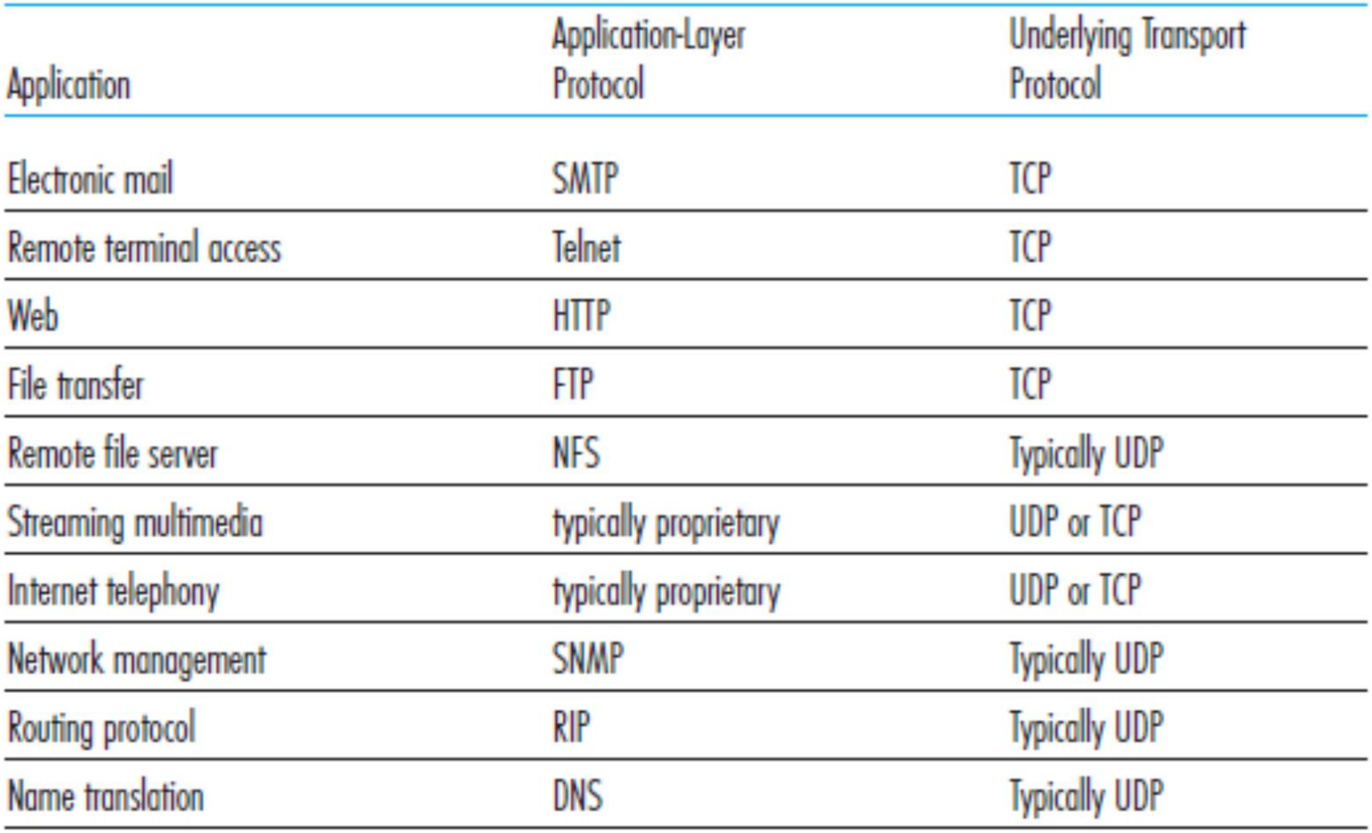
UDP packet: No state information for the communication session.

UDP is a stateless protocol, without re-transmission of loss data or protecting against data recording

DIFFERENCE BETWEEN TCP & UDP

TCP	UDP
Connection Oriented	Connectionless
Byte stream Service	Datagram Service
Reliable	Unreliable
Inorder delivery of data	Unordered
Guaranteed delivery	Best effort service (no guarantee)
Implements Flow Control	No flow control mechanism
Implements Congestion Control	No such specific mechanism

POPULAR INTERNET APPLICATIONS AND THEIR UNDERLYING TRANSPORT PROTOCOLS



Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

TCP CONNECTION ESTABLISHMENT

Steps involved in establishing TCP connection are

The server must be ready to accept an incoming connection by calling socket, bind, and listen - passive open

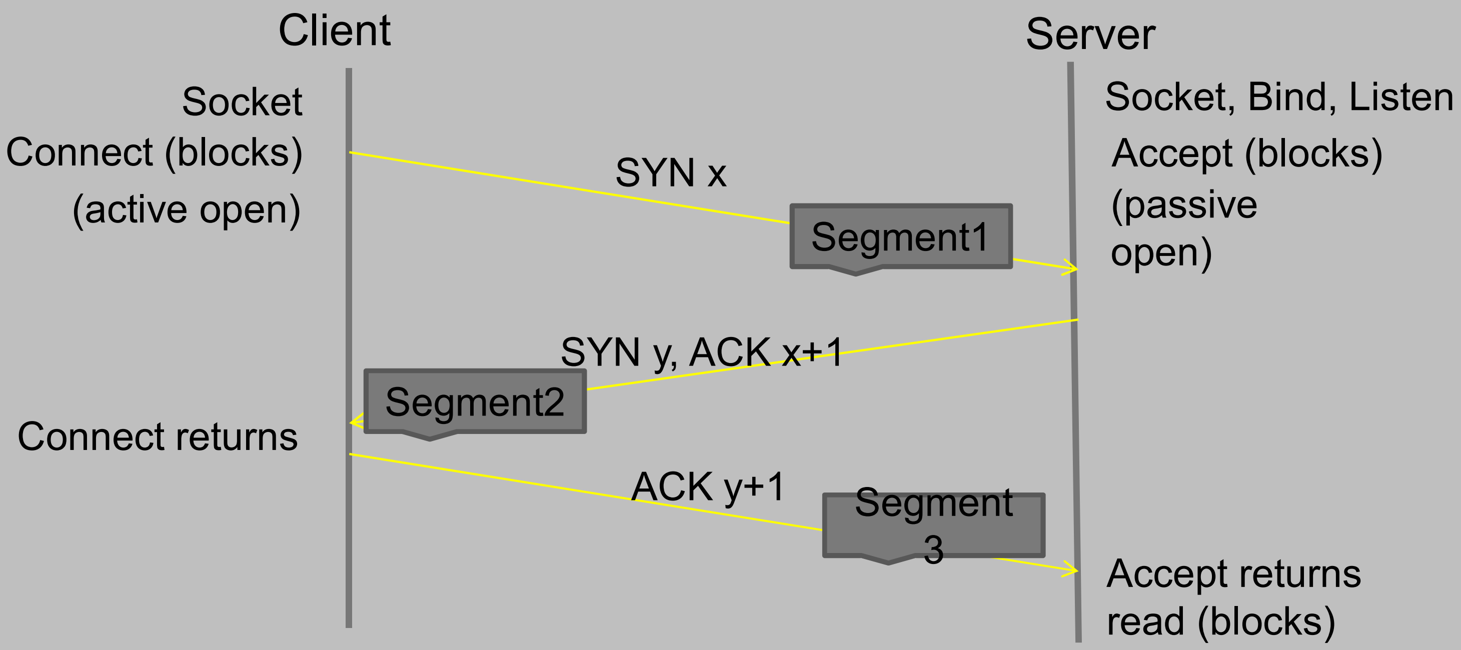
The client issues an active open by calling connect.

This initiates the Three way Handshake

1. Initially, the client TCP sends a "synchronize" (SYN) segment, which contains the client's initial sequence number for the data to be sent on this connection
2. The server must acknowledge (ACK) the client's SYN and the server also sends its own SYN in a single segment.
3. The client must acknowledge the server's SYN by sending ACK.

TCP CONNECTION ESTABLISHMENT

Three way handshake



Client

Server

Socket

Connect (blocks)

(active open)

Socket, Bind, Listen

Accept (blocks)

(passive open)

SYN x

Segment1

SYN y, ACK x+1

Segment2

Connect returns

ACK y+1

Segment 3

Accept returns

read (blocks)

TCP CONNECTION TERMINATION

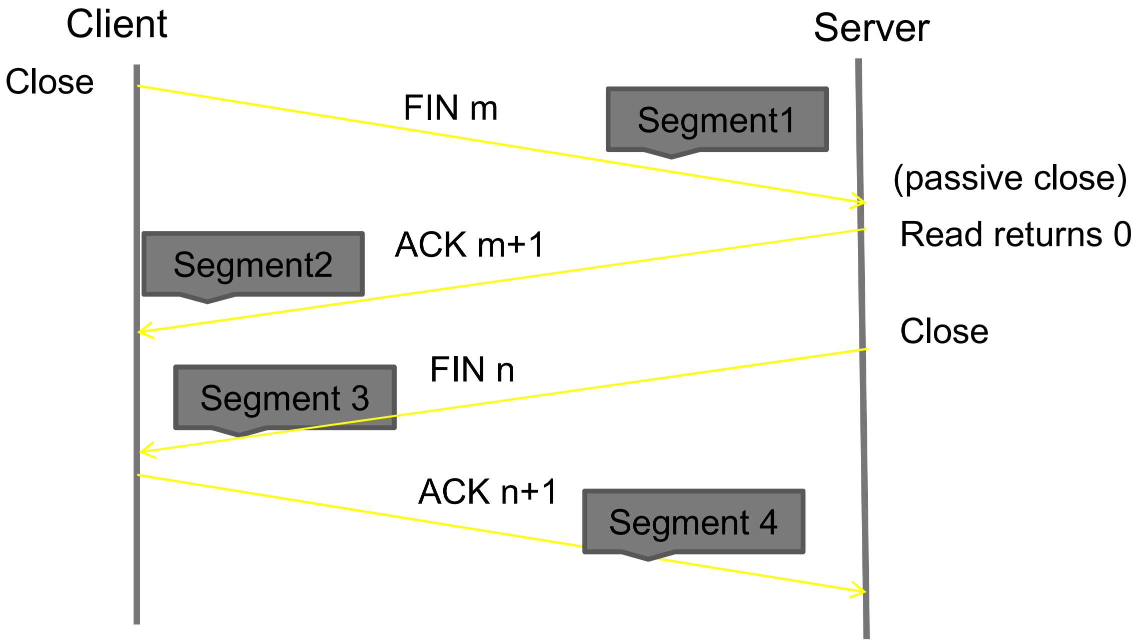
The server closes its connection with the client by calling close

Steps involved in terminating TCP connection are

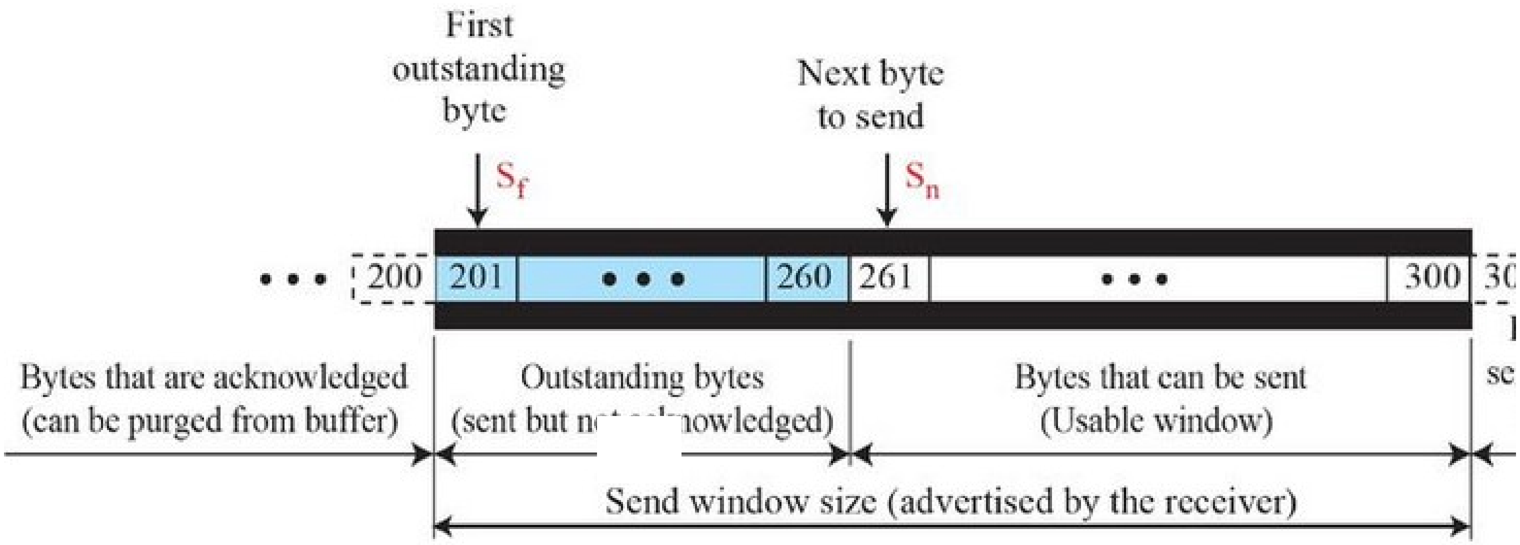
1. The client TCP sends a “finish" (FIN) segment
2. The server TCP acknowledges by sending ACK to the client
3. Then the server TCP sends a “finish" (FIN) segment for mutual termination
4. Finally, the client TCP acknowledges by sending ACK to the server

TCP CONNECTION TERMINATION

Four way teardown

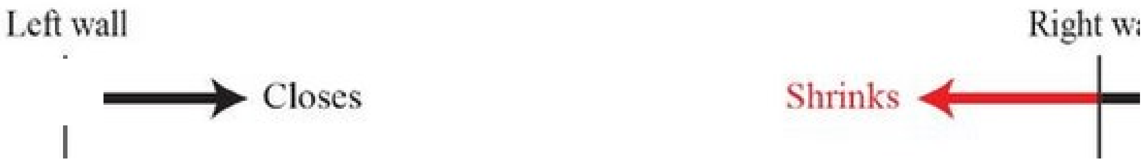


SEND WINDOW

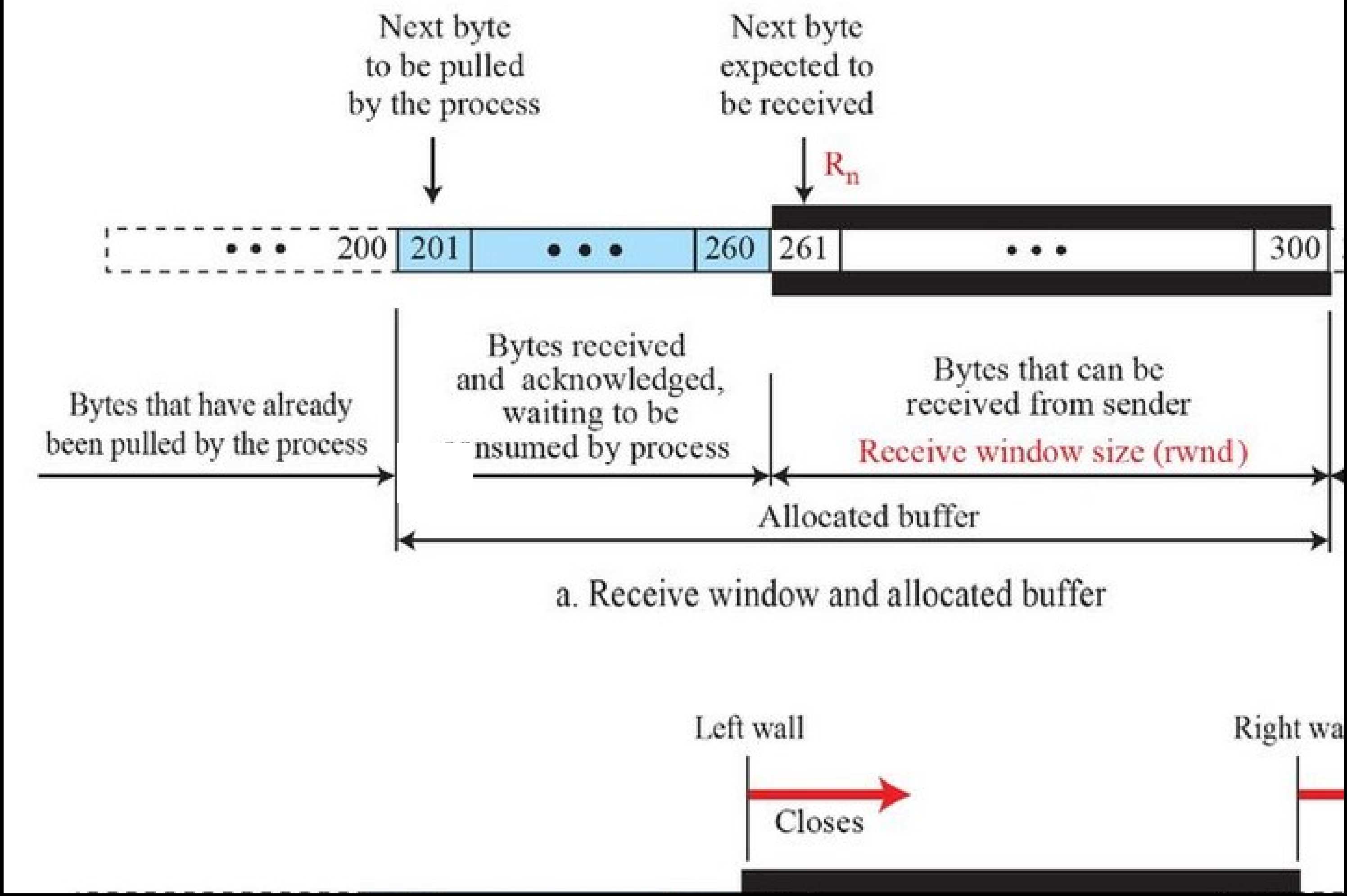


First outstanding byte
S_f
Next byte to send
S_n
• • •
200
201
• • •
260
261
• • •
300
30
Bytes that are acknowledged (can be purged from buffer)
Outstanding bytes (sent but n nowledged)
Bytes that can be sent (Usable window)
se
Send window size (advertised by the receiver)

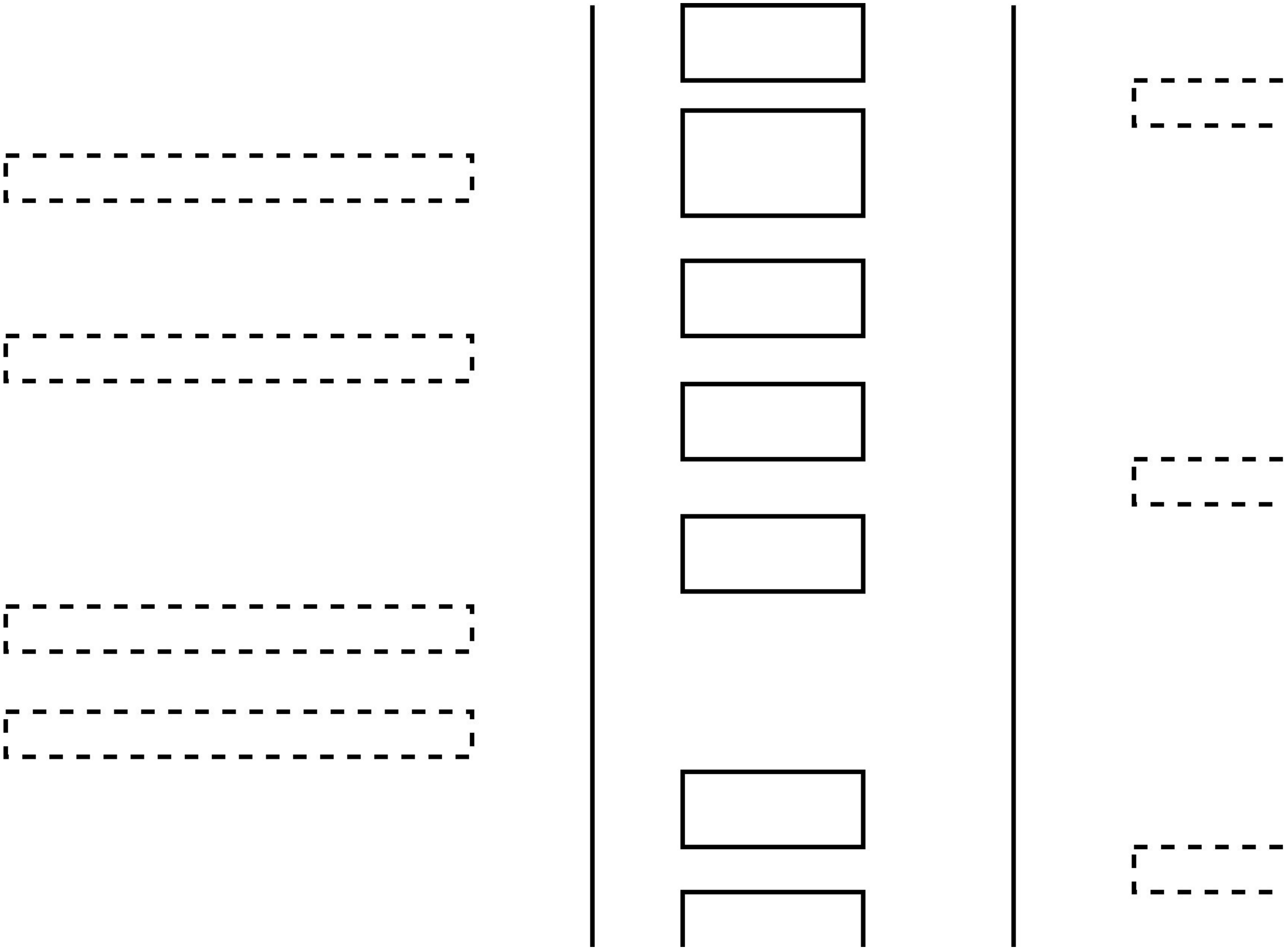
a. Send window

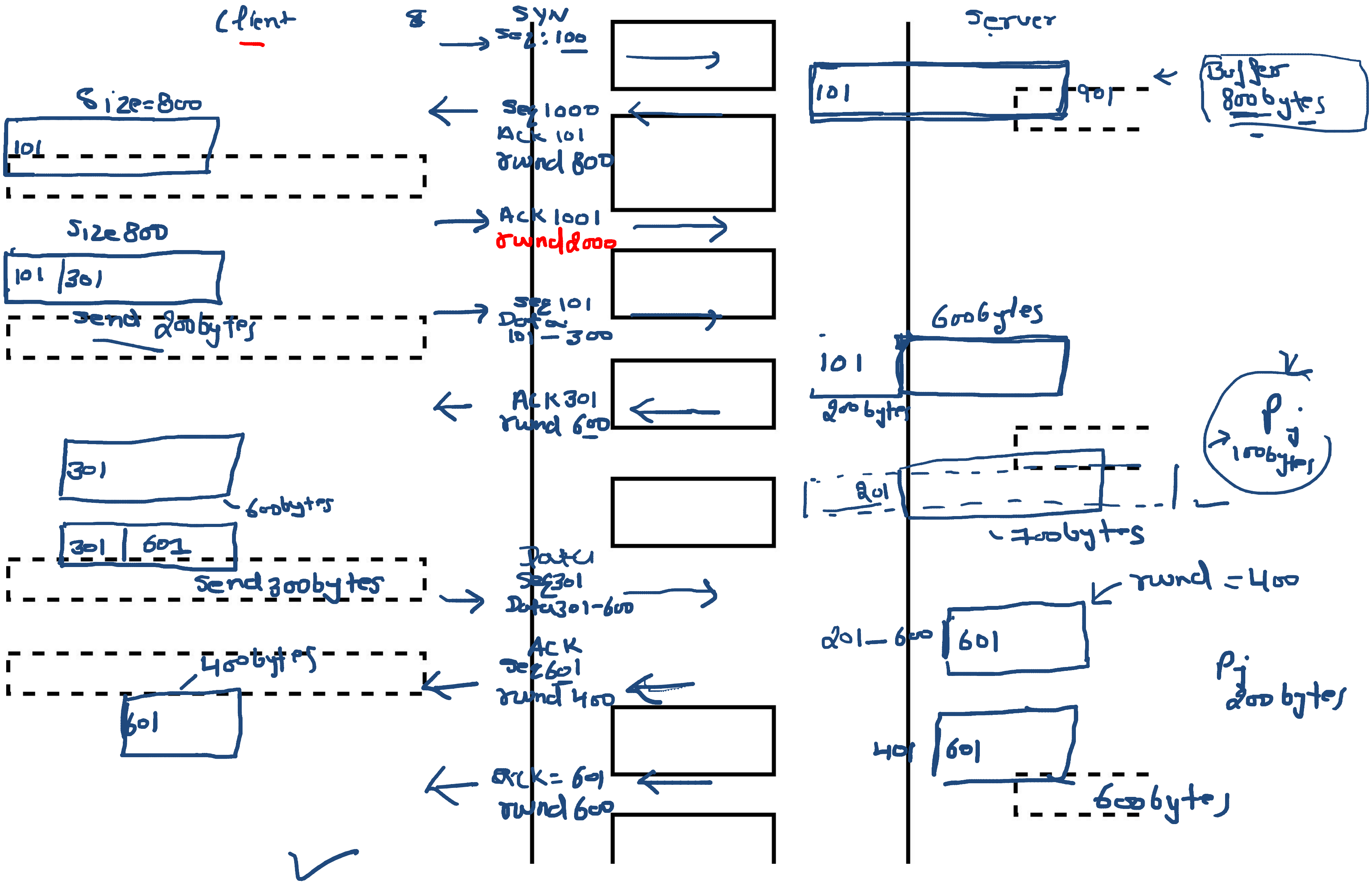


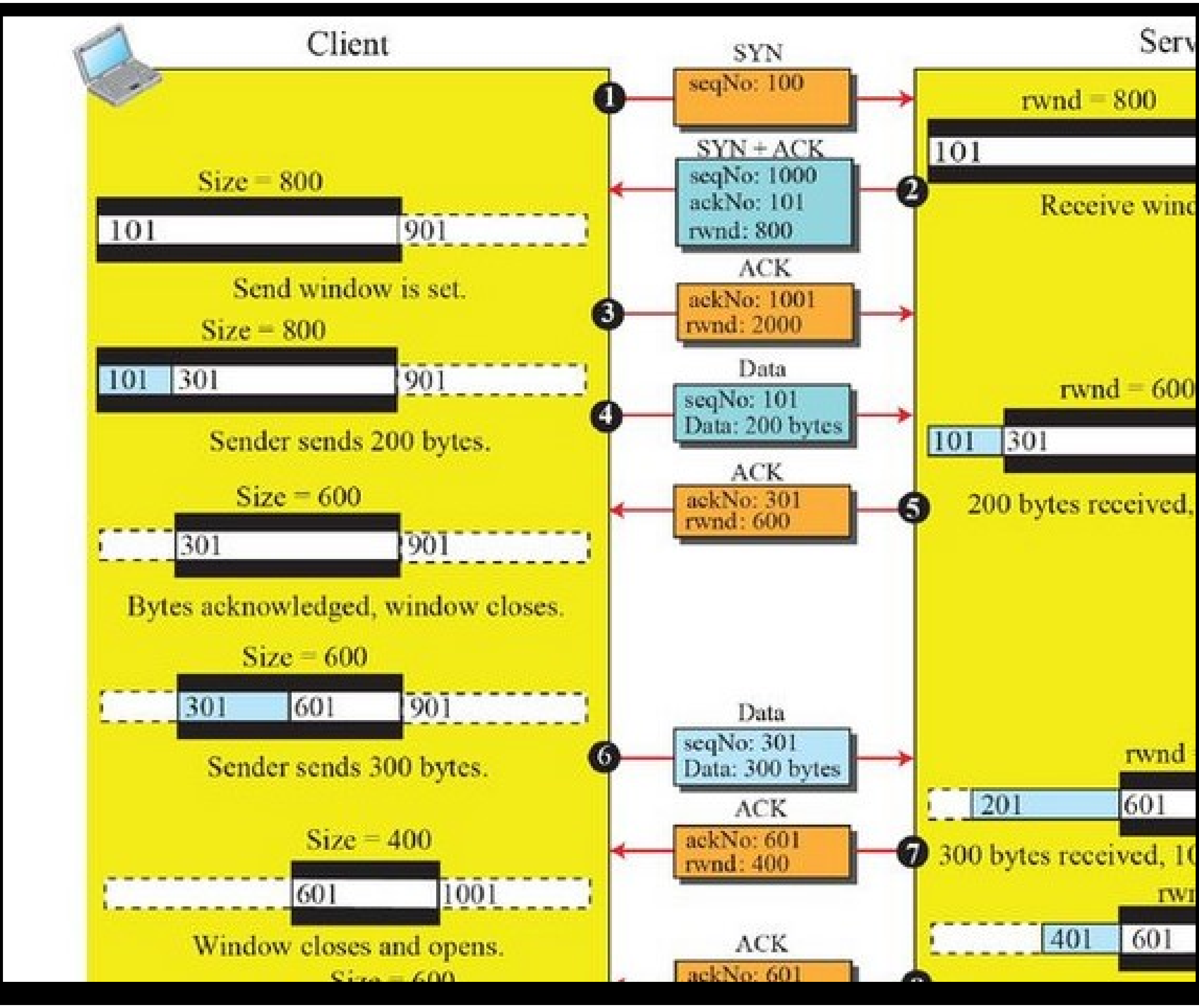
Receive Window


Next byte to be pulled by the process
Next byte expected to be received
R_n
200
201
260
261
300
Bytes that have already been pulled by the process
Bytes received and acknowledged, waiting to be nsumed by process
Bytes that can be received from sender
Receive window size (rwnd)
Allocated buffer
Left wall
Right wa
Closes

a. Receive window and allocated buffer




Client
Server
SYN
Seq: 100
Buffer
800 bytes
101
901
Size=800
101
Seq 1000
Ack 101
rwnd 800
Ack 1001
rwnd 2000
Size 800
101
301
Seq 101
Data
101 — 300
Send 200 bytes
600 bytes
101
200 bytes
ACK 301
rwnd 600
301
600 bytes
Pj
100 bytes
201
700 bytes
301
601
Send 300 bytes
Data
Seq 301
Data 301-600
rwnd = 400
201 — 600
601
400 bytes
ACK
Seq 601
rwnd 400
601
Pj
200 bytes
401
601
ACK = 601
rwnd 600
600 bytes


Client
Serv
SYN
seqNo: 100
1
rwnd = 800
101
SYN + ACK
seqNo: 1000
ackNo: 101
rwnd: 800
2
Receive wind
Size = 800
101
901
Send window is set.
ACK
ackNo: 1001
rwnd: 2000
3
Size = 800
101
301
901
Data
seqNo: 101
Data: 200 bytes
4
rwnd = 600
101
301
Sender sends 200 bytes.
ACK
ackNo: 301
rwnd: 600
5
200 bytes received,
Size = 600
301
901
Bytes acknowledged, window closes.
Size = 600
301
601
901
Data
seqNo: 301
Data: 300 bytes
6
rwnd
201
601
Sender sends 300 bytes.
ACK
ackNo: 601
rwnd: 400
7
300 bytes received, 1(
Size = 400
601
1001
rwi
401
601
Window closes and opens.
ACK
ackNo: 601
Size = 600