



21CSC302J- Computer Networks



Unit V



Syllabus - Unit V

- Port Numbers
- User Datagram Protocol
- Transmission Control Protocol
- WWW
- HTTP
- FTP
- Email
- Telnet
- DNS

PORT NUMBERS





PORT NUMBERS

- A port number refers to the type of addressing information that identifies the receiver as well as the sender of a message in computer networking.
- The primary use of different port numbers is to identify the protocol at which a network must direct the incoming traffic.
- Port numbers determine the specific processes to which a network must forward the internet or any specific message (once it arrives at the intended server).
- There are ports for every protocol and these act as endpoints of communications



PORT NUMBERS

- A port gets represented by various 16-bit numbers. Here, the numbers from 0 to 1023 are restricted, and only well-known protocol services use them.
- The port numbers from 1024 to 49151 are the registered ones.
- It means that you can register these numbers to any particular protocol using the software corporations.
- At last, the numbers from 49152 to 65536 act as private ports. Anybody can use these.



PORT NUMBERS

- For example, a user request for a file transfer from a client, or local host, to a remote server on the internet uses File Transfer Protocol (FTP) for the transaction.
- Both devices must be configured to transfer files via FTP.
- To transfer the file, the Transmission Control Protocol (TCP) software layer in local host identifies the port number of 21, which, by convention, associates with an FTP request -- in the 16-bit port number integer that is appended to the request.
- At the server, the TCP layer will read port number 21 and forward the request to the FTP program at the server.



PORT NUMBERS

- There are 65,535 port numbers, but not all are used every day.
- Restricted port numbers are reserved by prominent companies and range from 0 to 1023 like Apple QuickTime, Structured Query Language.
- Those who want to register a specific port number can choose from 1024 to 49151.
- Software companies typically register these port numbers.
- Dynamic or private ports ranging from 49152 to 65536 are available for anyone's use
- A port number is assigned temporarily -- for the duration of the request and its completion -- from a range of assigned port numbers. This is called a ***temporary port number***.



PORT NUMBERS

Some Commonly used Port Numbers

Port No	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active user
13	Daytime	Returns the data 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
22	ssh	Secure shell
123	Ntp	Network time protocol
110	Pop3	Post office protocol



PORT NUMBERS VS. IP ADDRESS

Parameters	IP Address	Port Number
Meaning and Purpose	The IP address refers to the Internet Protocol Address. These basically identify a host present in a network.	We use Port numbers for identifying any process/ service present on your system.
Bits Used	The size of IPv4 is 4 bytes (32 bits), and that of IPv6 is 16 bytes (128 bits).	A typical port number is 16 bits in size.
Layer of Protocol	This address refers to that of the IP protocol of Layer-3.	This address refers to that of the protocol of Layer-4.
Address Provider	A network administrator or system admin provides a user with their IP address.	The kernel of the OS (operating system) provides an application with its port number.
Command Used	One can use the ipconfig command for finding an IP address.	One can use the netstat command for finding the available TCP ports along with the network statistics.
Uses	This type of address identifies a computer/ host device on the computer network.	These numbers act as the logical interfaces utilized by the communication protocols.
Examples	A few examples of IP addresses are 172.16.0.2, 192.168.0.2, etc.	A few examples of port numbers are 67 and 68 for DHCP traffic, 80 for HTTP, 22 for SSH, 123 for NTP, etc.



OSI LAYERS AND PROTOCOLS

OSI LAYERS

EXAMPLE PROTOCOLS

APPLICATION LAYER	HTTP, FTP, IRC, SSH, DNS
PRESENTATION LAYER	SSL, FTP, IMAP, SSH
SESSION LAYER	VARIOUS API'S, SOCKETS
TRANSPORT LAYER	TCP, UDP, ECN, SCTP, DCCP
NETWORK LAYER	IP, IPSec, ICMP, IGMP
DATA-LINK LAYER	Ethernet, SLIP, PPP, FDDI
PHYSICAL LAYER	Coax, Fiber, Wireless

A red speech bubble with a white outline and a slight shadow, containing the text 'UDP' and 'User Datagram Protocol'.

UDP

User Datagram
Protocol



User Datagram Protocol (UDP)

UDP – An Introduction

- Connectionless service
- Unreliable transport protocol.
- No flow control / No Acknowledgement
- Process to Process communication
- Powerless
- It uses minimum of over heads
- No reliability is obtained using UDP
- Less interaction between sender and receiver



User Datagram Protocol (UDP)

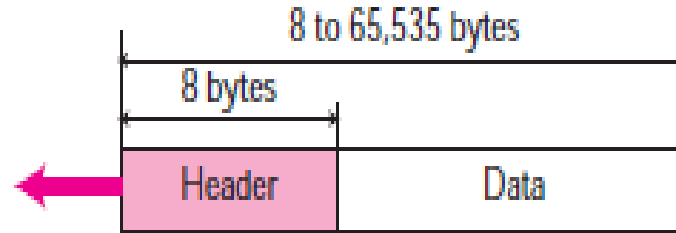
UDP – An Introduction Contd...

- UDP is used when acknowledgement of data does not hold any significance
- UDP is good protocol for data flowing in one direction
- UDP is simple and suitable for query based communications
- UDP does not provide congestion control mechanism
- UDP does not guarantee ordered delivery of data
- UDP is stateless
- UDP is suitable protocol for streaming applications such as VoIP, multimedia streaming

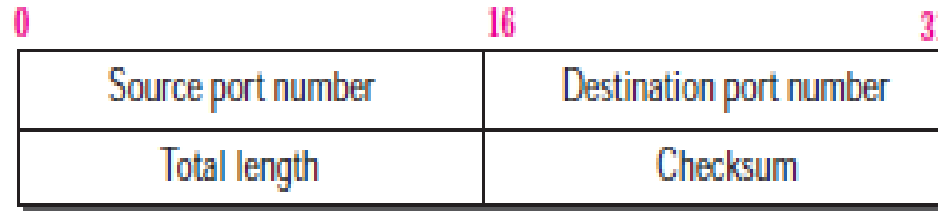


User Datagram Protocol (UDP)

USER DATAGRAM



a. UDP user datagram



b. Header format

User datagram header format

- Above picture depicts user datagram format
- UDP packets are called user datagrams(messages)
- It has fixed size header of 8 bytes



User Datagram Protocol (UDP)

UDP Datagram various fields

➤ Source Port Number:

- 16 bits long – port ranges from 0-65535.
- This port number will be used by the source host for identification.

➤ Destination Port Number:

- 16 bits long.
- Used by the process running on the Destination machine.
- Application level service on end machine



User Datagram Protocol (UDP)

UDP Datagram various fields

➤ Length:

UDP length=IP length – IP headers Length

- Length field specifies the entire length of UDP packet (including header).
- It is 16-bits field and minimum value is 8-byte, i.e. the size of UDP header itself.
- A user datagram is encapsulated in an IP datagram.

➤ Checksum:

- This field is used to detect errors over the entire user datagram (header plus data)



User Datagram Protocol (UDP)

Applications that use UDP

- Best Effort Delivery applications
- Lightweight applications
- Applications with their own mechanisms for reliable transmission
- Multicast Applications
- Real-Time applications

TCP Header





Transmission Control Protocol (TCP)

TCP Segment

- A packet in TCP is called a *Segment*
- *Segment – Format*
 - Header - 20 to 60 Bytes
 - In case of no options Header is of size 20 bytes
 - When options are used Header size extends up to 60 Bytes
 - Data from application program



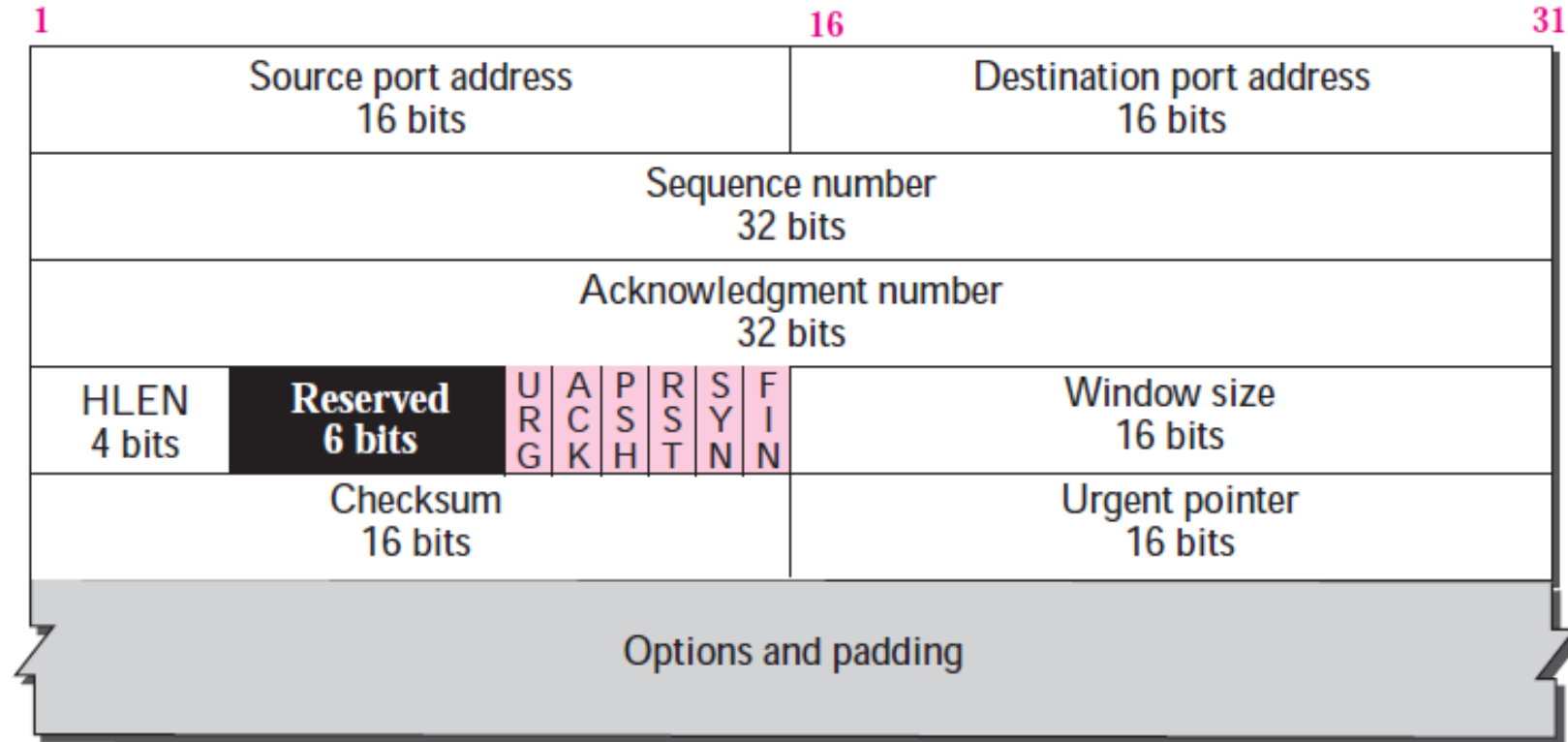
Format of Segment



Transmission Control Protocol (TCP)

TCP Header

- Size of TCP Header varies from 20 to 60 bytes



Format of TCP Header



Transmission Control Protocol (TCP)

TCP Header Contd...

➤ The various fields of TCP header are as follows:

- i. Source Port Address
- ii. Destination Port Address
- iii. Sequence Number
- iv. Acknowledgement Number
- v. Header Length
- vi. Reserved

- vii. Control
- viii. Window Size
- ix. Urgent Pointer
- x. Options
- xi. Checksum



Transmission Control Protocol (TCP)

TCP Header Contd...

i. Source Port Address

- 16 bit Field
- Defines port number of application program in host sending the segment

ii. Destination Port Address

- 16 bit Field
- Defines port number of application program in host receiving the segment

iii. Sequence Number

- 32 bit field
- Defines number of first byte of data contained in Segment
- Each byte is numbered for connectivity reasons
- Sequence number provides information regarding the first byte of segment to destination host



Transmission Control Protocol (TCP)

TCP Header Contd...

iv. Acknowledgement Number

- 32 bit Field
- Defines the byte number the receiver expects to receive from senders
- If 'n' bytes are received from sender 'n+1' is sent as acknowledgement number
- Acknowledgment and Data go hand in hand

v. Header Length

- 4 bit Field
- Indicates number of 4 byte words in Header
- Value of header length varies between 5 ($5 \times 4 = 20$) and 20 ($20 \times 4 = 60$)

vi. Reserved

- 6 bit field
- Reserved for future use

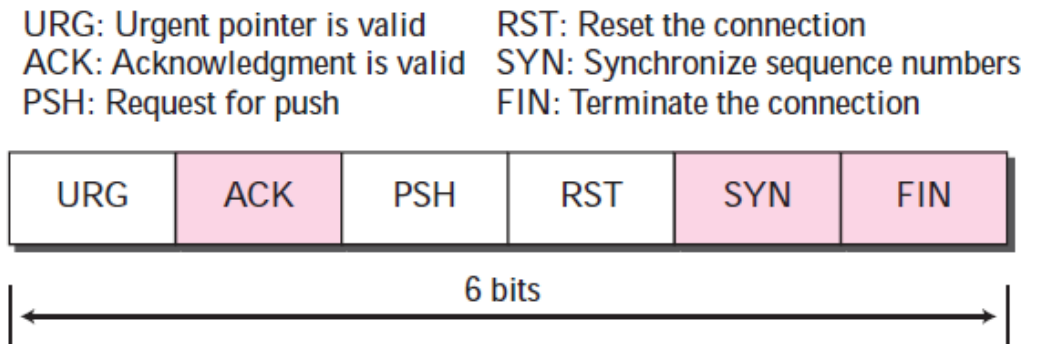


Transmission Control Protocol (TCP)

TCP Header Contd...

vii. Control

- 6 bit Field
- Defines 6 unique control bits / flags
- Can be set one / more at a time
- Enables
 - Flow Control
 - Connection Establishment, Termination and Abortion
 - Mode of TCP Data transfer



Control Field of TCP Header



Transmission Control Protocol (TCP)

TCP Header Contd...

viii. Window Size

- 16 bit Field
- Maximum Window Size: 65, 535 bytes
- Defines Window size of sending TCP
- Determined by Receiver
- Referred as (*rwnd*)
- Sender must adhere to the window size fixed by the receiver

ix. Urgent Pointer

- 16 bit field
- Used only if urgent data is part of segment
- Valid only when urgent flag is set

x. Options

- Can accommodate up to 40 bytes of optional information

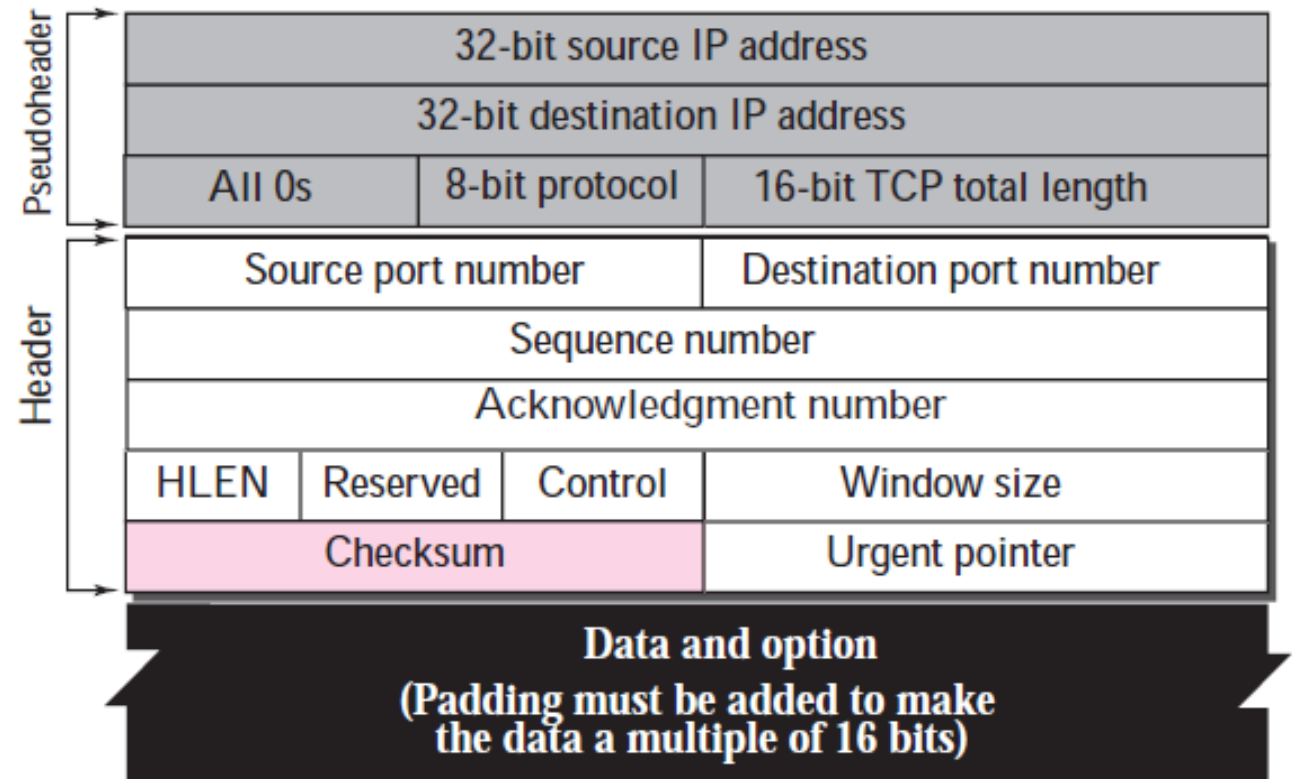


Transmission Control Protocol (TCP)

TCP Header Contd...

xi. Checksum

- 16 bit field
- Mandatory in TCP
- Detects error over entire TCP
- Pseudoheader added to segment



Pseudoheader Added to the TCP Datagram

A photograph of a server room with blue lighting and bokeh light effects. The text "A TCP Connection" is overlaid in white, italicized font. Below the text is a solid orange horizontal bar.

A TCP Connection



Transmission Control Protocol (TCP)

A TCP Connection

- Connection Oriented Protocol
- Virtual path established between source and destination
- TCP Segments sent and received over Virtual path
 - Enables retransmission of Lost / Damaged segments
 - Waits for segments arriving out of order
- Phases of TCP Connection
 - i. Connection Establishment
 - ii. Data Transfer
 - iii. Connection Termination



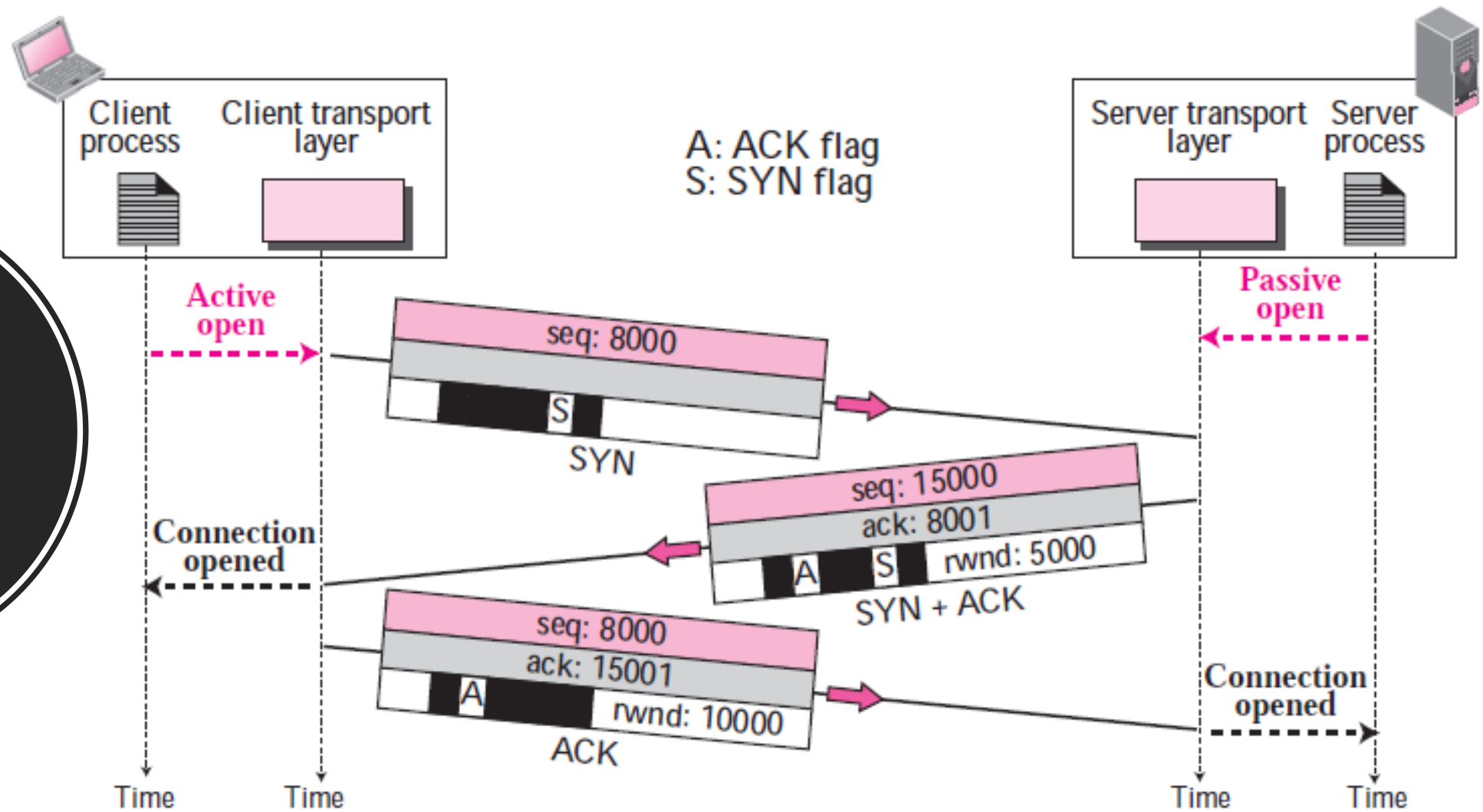
Transmission Control Protocol (TCP)

A TCP Connection Contd...

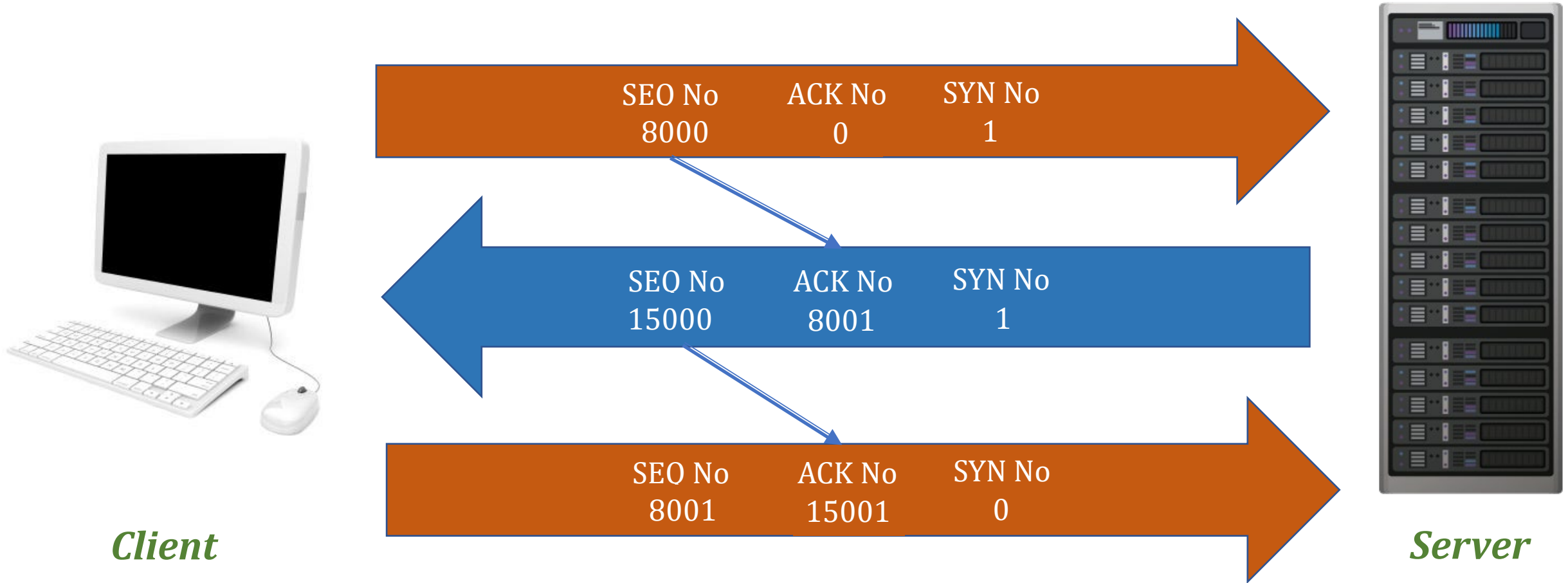
i. Connection Establishment

- Transmission mode in TCP: *Full-Duplex*
- Simultaneous transfer of data between Sender and Receiver
- Mutual approval of Communication between sender and receiver mandatory before data transfer
- Connection establishment performed through *3-Way Handshaking*
 - *Passive Open* – Server program informs TCP that it is ready to accept connection
 - *Active Open* – Client program intending to connect to a Server informs TCP

Connection Establishment using 3-way Handshaking



3-Way Handshaking for TCP Connection





Transmission Control Protocol (TCP)

A TCP Connection Contd...

i. Connection Establishment Contd...

➤ *Step 1*

- Client is in active open mode
- Client sends first SYN segment and sets flag to SYN
- ***Initial Sequence Number (ISN)*** : 8000 (Random Number) sent to Server
- SYN segment (Control Segment) carries no data
- SYN segment consumes one Sequence number
- ***Note:*** No ACK / Window Size sent along with ISN



Transmission Control Protocol (TCP)

A TCP Connection Contd...

i. Connection Establishment Contd...

➤ Step 2

- Server sends second segment (SYN + ACK) and sets flag to SYN & ACK
 - **ACK** - Server acknowledges receipt of first segment from Client
 - **SYN** – Segment for communication from Server to Client
 - New Sequence Number : **15000** (Random Number)
 - Acknowledgement Number for Segment from client: **8001** (Inc. by 1)
 - Window Size (**rwnd**): Set to **5000** by Server (to be used by Client)
 - (SYN+ACK) segment consumes one Sequence number





Transmission Control Protocol (TCP)

A TCP Connection Contd...

i. Connection Establishment Contd...

➤ *Step 3*

- Client sends third segment (ACK) and sets ACK flag
 - **ACK** – Client acknowledges receipt of second segment from Server
 - Sequence Number : **8000** (Used in first Segment)
 - Acknowledgement Number for Segment from Server: **15001** (Inc. by 1)
 - Window Size (**rwnd**) : Set to **10000** by Client (to be used by Server)
 - (ACK) segment consumes no Sequence number





Transmission Control Protocol (TCP)

A TCP Connection Contd...

i. Connection Establishment Contd...

➤ *Synchronous Flooding Attack*

- Type of Denial of Service Attack (DoS)
- Malicious attackers send large count of SYN segments to a Server
- SYN segments pretend to come from various Clients using different IP Address
- Server assumes that Clients are in active open mode and allocates resources
- Server Sends SYN + ACK segments to fake clients and waits for response
- Server runs out of resources and is unable to accept connection requests from legitimate clients



Transmission Control Protocol (TCP)

A TCP Connection Contd...

i. Connection Establishment Contd...

➤ *Simultaneous Open*

- Client and Server issues active open mode (Rare Phenomenon)
- SYN + ACK segment sent from Client to Server and vice versa



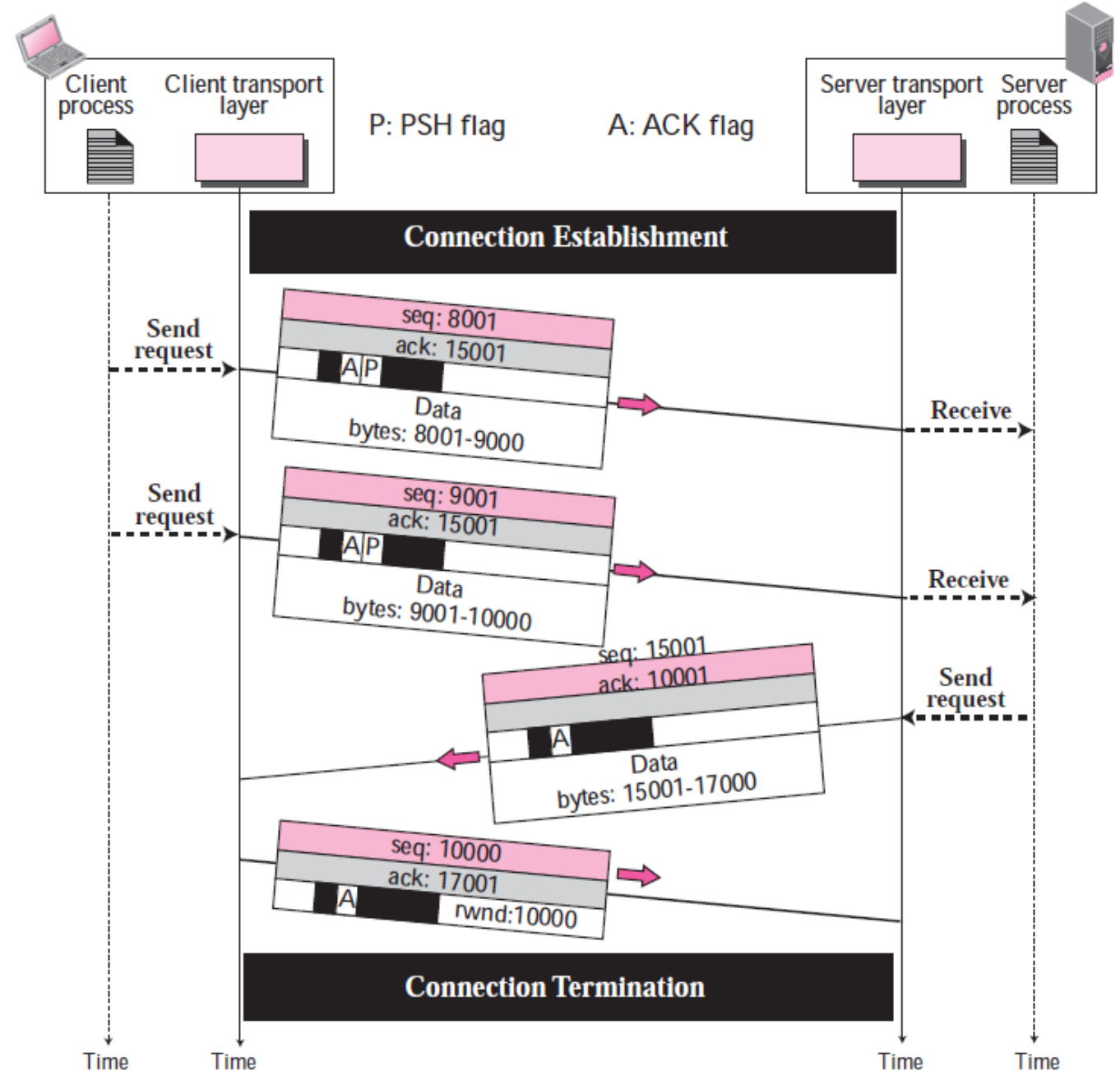
Transmission Control Protocol (TCP)

A TCP Connection Contd...

ii. TCP Data Transfer

- Data transfer is bidirectional after connection is established
- Data and acknowledgements can be sent between client and server bidirectionally
- *Example (After Connection Establishment)*
 - Client transmits 2000 bytes of data in 1st and 2nd segment to the Server
 - Push flag (PSH) & ACK flags set for Data Segments sent by client
 - *Segment 1:* seq(8001), ack(15001), A & P flags set, Data bytes (8001-9000)
 - *Segment 2:* seq (9001), ack (15001), A & P flags set, Data bytes (9001 – 10000)

- Server sends 2000 bytes of data in the 3rd segment
- **Segment 3:** seq (15001), ack (10001), A flag set, Data bytes (15001 – 17000)
- ACK flag set & SYN flag not set for data segments sent by server
- **Segment 4:** seq (10001), ack (17001), ACK flag set, rwnd (10000)



Data Transfer in TCP



Transmission Control Protocol (TCP)

A TCP Connection Contd...

ii. TCP Data Transfer Contd...

➤ *Flexibility of TCP*

- Sending TCP uses buffer to store incoming data stream from applications
- Receiving TCP uses buffer to store arriving data and send to applications
- Disadvantage: Delayed delivery of data

➤ *Pushing Data*

- Sending TCP creates a segment and transfers immediately by setting PSH bit
- PSH bit indicates that receiving TCP must deliver the received data segment to application program immediately



Transmission Control Protocol (TCP)

A TCP Connection Contd...

ii. TCP Data Transfer Contd...

- Data presented from application program to TCP as stream of bytes
 - Consecutive positions assigned to each byte of data
 - **Exception:** Application program needs to send Urgent bytes that needs to be specially treated (With top priority)
- ***Solution: Urgent Data***
 - Send a data segment by setting the URG bit
 - Urgent data inserted at segment beginning by the sending TCP
 - End of the urgent data in segment indicated by urgent pointer field in header



Transmission Control Protocol (TCP)

A TCP Connection Contd...

ii. TCP Data Transfer Contd...

- Segment with URG bit is received by the receiving TCP
- Receiving TCP informs receiving application of the segment with URG bit



Transmission Control Protocol (TCP)

A TCP Connection Contd...

iii. TCP Connection Termination

- Connection termination initiated by the Client by default
- However, Server can also choose to close the TCP connection with client
- Options for connection termination
 - a) 3-way Handshaking
 - b) 4-way Handshaking with Half-Close Option



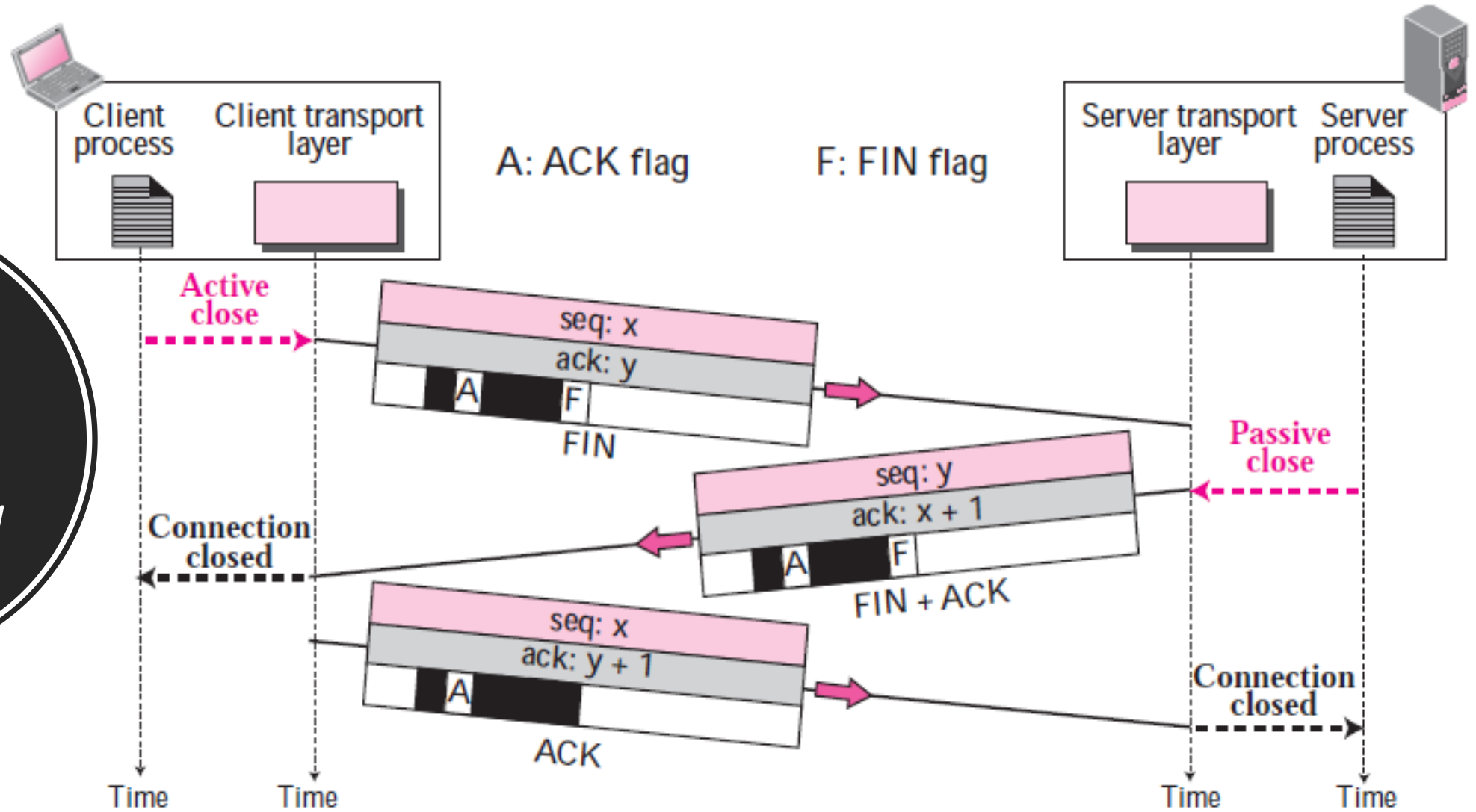
Transmission Control Protocol (TCP)

A TCP Connection Contd...

a) Connection Termination using 3-way Handshaking

- *Passive Close* – Server program informs TCP that it is ready to close connection
- *Active Close* – Client program intending to close connection to a Server informs TCP
- *Step 1*
 - Client process sends close command to Client TCP
 - Client TCP sends 1st Segment (FIN) with FIN flag and ACK flag set
 - FIN Segment consumes one sequence number (if it carries no data)
 - Note: FIN segment may carry last chunk of data also

Connection Termination using 3-way Handshaking





Transmission Control Protocol (TCP)

d) A TCP Connection Contd...

a) Connection Termination using 3-way Handshaking

➤ *Step 2*

- Server TCP sends 2nd segment (FIN + ACK) to confirm receipt of FIN segment
- Server announces connection closing from its side
- 2nd Segment may contain last chunk of data

➤ *Step 3*

- Client TCP sends 3rd segment (ACK) that confirms FIN receipt from Server
- Acknowledgment number – Sequence number + 1
- 3rd Segment cannot carry data



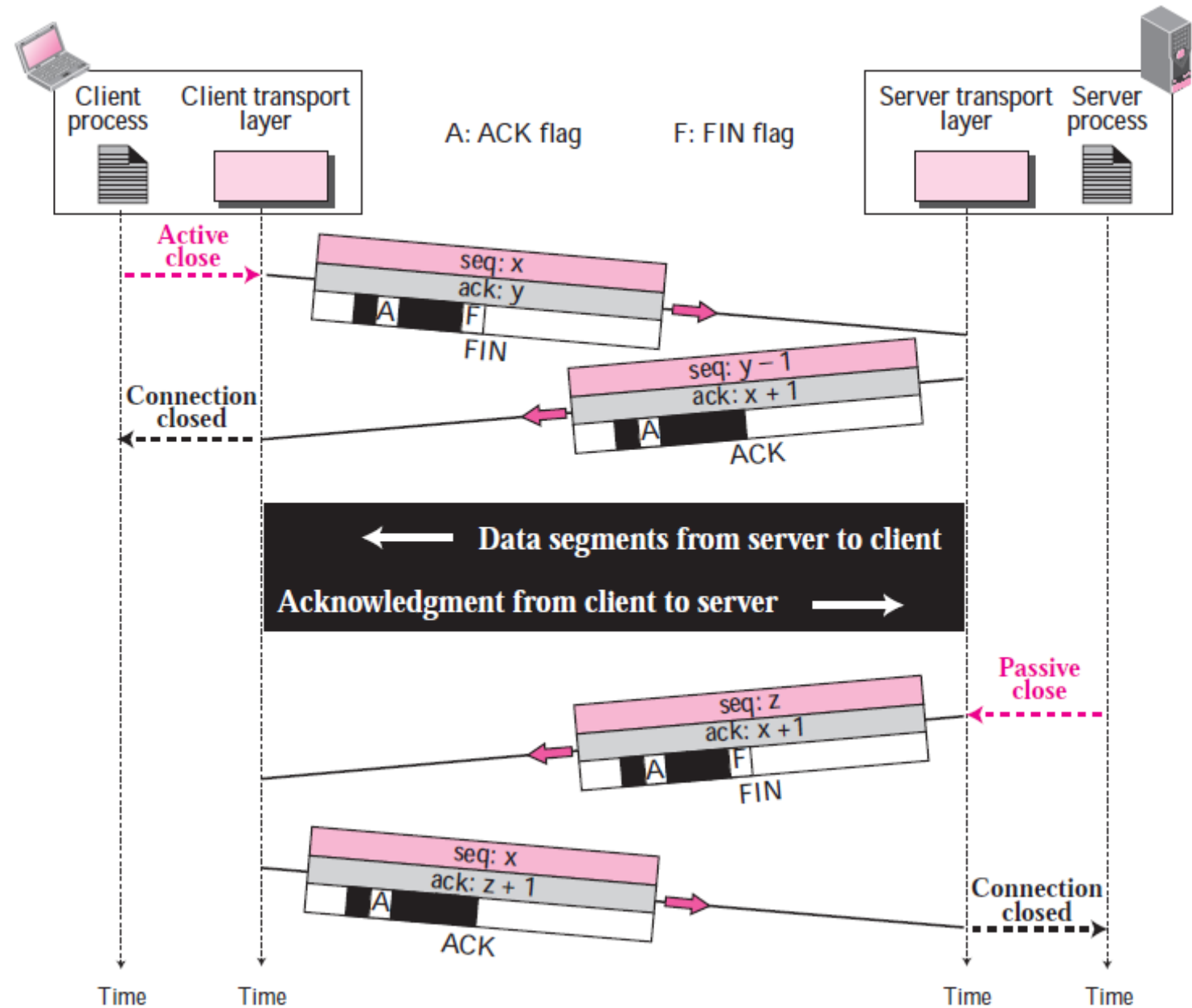
Transmission Control Protocol (TCP)

A TCP Connection Contd...

b) Half Close Connection Termination

- *Half Close* – In client / Server communication if one can stop sending data while the other can send data it is called an Half-Close
 - The Client or Server can issue a Half-Close request
- Example – Sorting
 - Client sends data for sorting to Server
 - Client closes connection in Client-Server direction
 - Server receives data from client & keeps connection open
 - Till Sorting is completed & result sent back to Client

Half-Close





Transmission Control Protocol (TCP)

A TCP Connection Contd...

iv. TCP Connection Reset

- Reset Flag (RST) - Denies connection / Aborts connection / Terminates existing idle connection
 - *Denying Connection*
 - Client requests to a non-existent server port
 - Server sends segment with RST flag set and denies request
 - *Aborting Connection*
 - Client / Server TCP aborts existing connection by sending RST segment to abort connection



Transmission Control Protocol (TCP)

A TCP Connection Contd...

iv. TCP Connection Reset Contd...

➤ *Termination Idle Connection*

- Client finds server idle or vice versa
- Client / Server sends RST segment to terminate connection
- Similar to abort



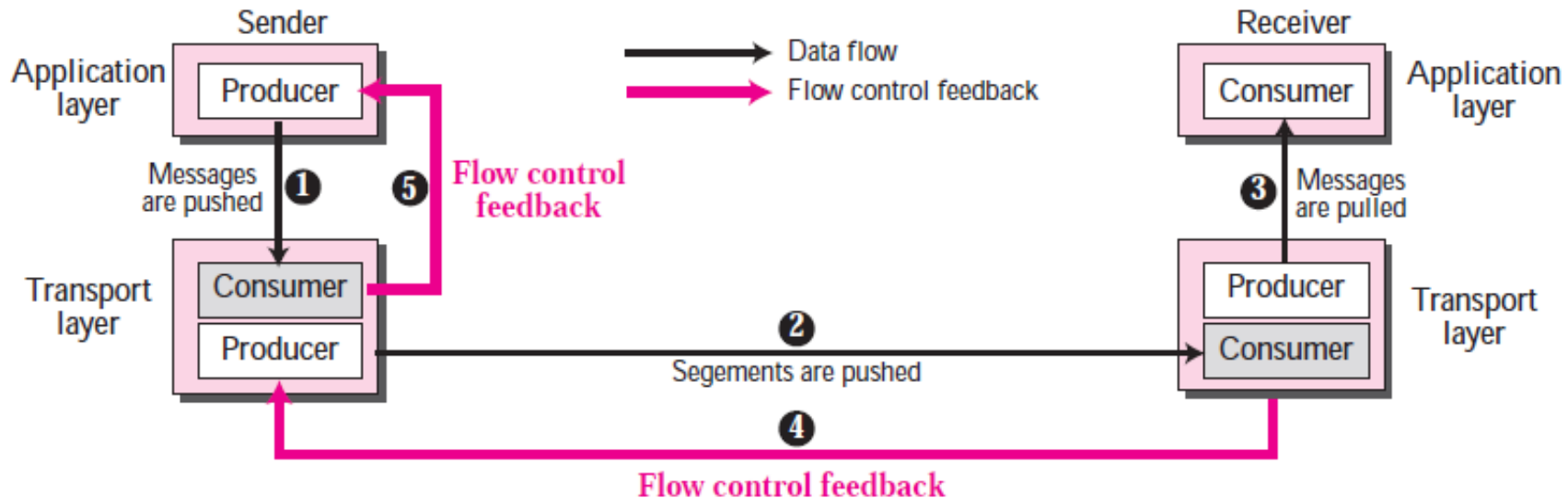
TCP Flow Control



Transmission Control Protocol (TCP)

TCP Flow Control

- Creates a balance between rate of data production and the rate of data consumption
- **Assumption:** Channel between sender & receiver is error-free

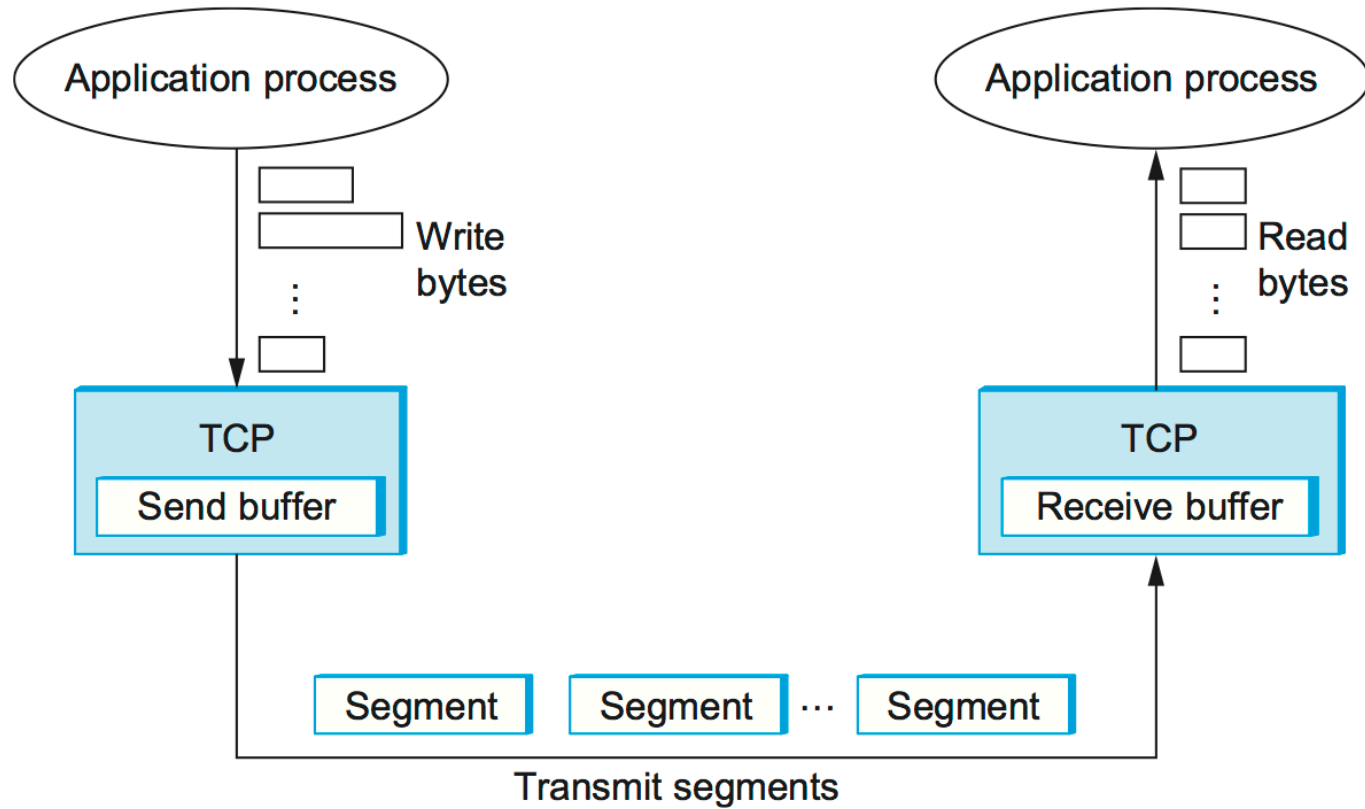


Data Flow and Flow Control Feedbacks in TCP



Transmission Control Protocol (TCP)

TCP Flow Control Contd...



How TCP manages byte Stream?



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

- 1) Messages are pushed from the Sending application to TCP Client
- 2) Message segment from TCP Client is pushed to TCP Server
- 3) Messages are pulled by receiving application from TCP Server
- 4) Flow control feedback is sent from TCP server to TCP client
- 5) TCP client forwards the flow control feedback to sending application

➤ *Opening and Closing Windows*

- Buffer size of sender & receiver is fixed during connection establishment
- Window sizes of Sender / Receiver is controlled and adjusted by TCP Server
- Opening / Closing / Shrinking of client window is controlled by receiver



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

Scenario – TCP Flow Control

➤ *Segment 1*

- Connection request – (SYN) segment from client to server
- Initial Sequence Number (ISN): 100 (Next byte to Arrive : 101)
- Server allots buffer size = 800 (Assumption)
- Server allots Window size (rwnd) = 800

➤ *Segment 2*

- (ACK + SYN) segment from Server to Client
- Set ACK no = 101 & Buffer Size = 800 bytes



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

Scenario – TCP Flow Control

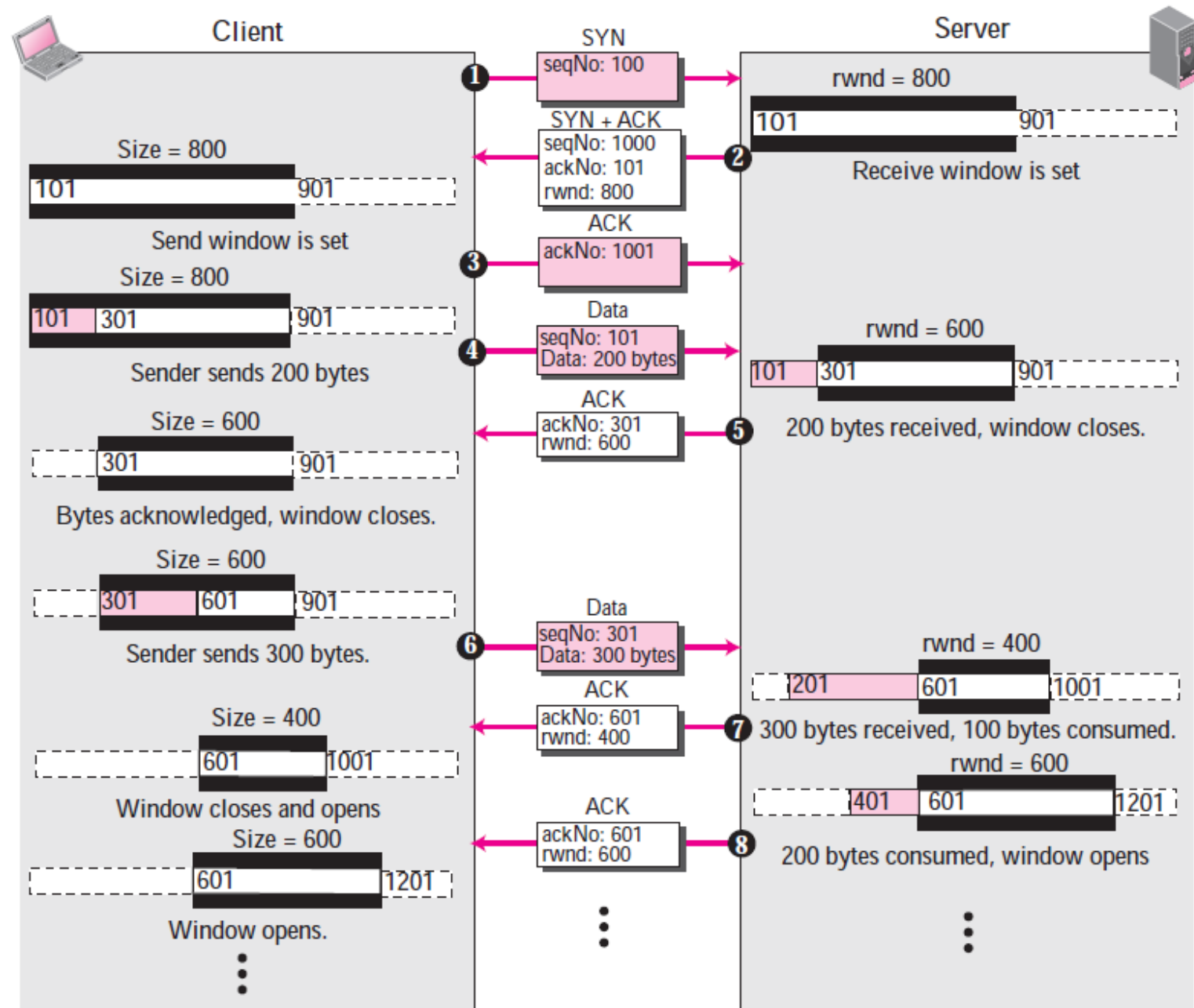
➤ *Segment 3*

- ACK segment sent from client to server

➤ *Segment 4*

- Client sets window size to 800 (since rwnd from server is 800)
- Client process pushes 200 bytes of data to TCP client
- TCP client creates data segment with bytes (101-300) and sends to Server
- Client window adjusted
 - Shows 200 bytes as sent but no ACK received from Server

Flow Control – A Scenario





Transmission Control Protocol (TCP)

TCP Flow Control Contd...

Scenario – TCP Flow Control

- Shows 200 bytes as sent but no ACK received from Server
- Server stores 200 bytes in buffer & closes receive window
- Server indicates the next expected byte as 301
- *Segment 5*
 - Server acknowledges receipt of 200 bytes from client & reduces rwnd to 600
 - Client receives acknowledgement and resizes window size to 600
 - Client closes the window (101-300) from left to right
 - Client indicates the next byte to send as 301



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

Scenario – TCP Flow Control

➤ *Segment 6*

- Client pushes 300 more bytes to the server (Seq. No = 301 & Data = 300 bytes)
- Server stores 300 bytes in buffer
- 100 bytes of data are pulled by the Client process
- So, Window size is reduced by 100 bytes to the left & opened by 100 bytes to the right
- Overall, TCP client window size is reduced by 200 bytes
- Now, Receiver window size (rwnd) = 400



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

Scenario – TCP Flow Control

➤ *Segment 7*

- TCP Server acknowledges receipt of 300 bytes and sets window size (rwnd) = 400 & TCP Client reduces window size to 400
- Sender windows closes by 300 bytes from the left and opens by 100 bytes to the right
- This process continues until all the data segments are sent from server to client and connection gets closed



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

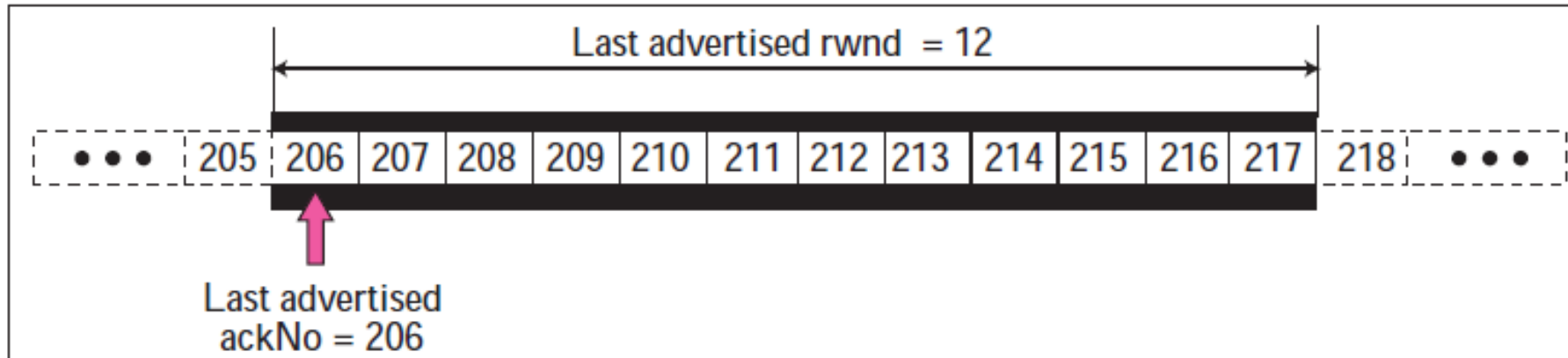
➤ *Shrinking of Windows*

- Shrinking : Decreasing window size i.e., right wall moves towards the left
- Sender window may shrink based on rwnd value defined by receiver
- Receiver window cannot shrink

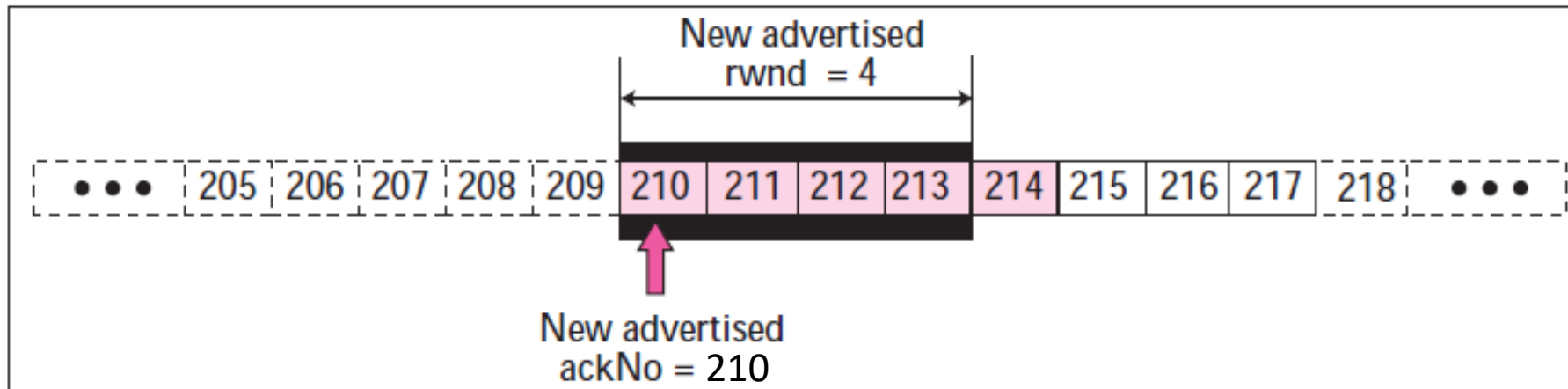
➤ *Illustration*

- Upto 205 bytes of data received and acknowledged by sender
- Last advertised rwnd = 12 (Window size) & last advertised ACK No = 206
 - Data can be sent from byte 206 to byte 217 (Since rwnd = 12)
- New advertised rwnd = 12 (Window size) & new advertised ACK No = 210

The window after the new advertisement; Window has shrunk



a. The window after the last advertisement



New advertised
ackNo = 210



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

➤ *Shrinking of Windows Contd...*

- Shrinking of window has occurred from byte 217 to byte 213 (Window had moved from right to left)
- Shrinking can be prevented by the relation given below:

$$\text{New ACK number} + \text{new rwnd} \geq \text{Last ACK Number} + \text{Last rwnd}$$

- To prevent shrinking,
 - Wait until enough buffer locations are available in its window



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

➤ *Silly Window Syndrome*

- Occurs when either the sending process creates data slowly (or) the receiving process consumes data slowly (or) both
- Silly window syndrome sends / receives data in small segments thus resulting in poor efficiency

➤ *Example*

- A 42 byte TCP datagram is needed to send Segment with 2 bytes of data
- Overhead : $42 / 2 \Rightarrow$ Network capacity used inefficiently



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

➤ *Silly Window Syndrome Created by Sender*

- Sending TCP creates syndrome since sending application program creates data slowly i.e., 1 byte at an instance

➤ *Suggestions*

- Prevent sending TCP from transmitting data byte by byte
- Sending TCP made to wait & collect data to send data in larger block
- ***Disadvantage:*** Waiting too long delays the process

➤ *Solution*

- Nagle's Algorithm



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

- *Nagle's Algorithm for Silly Window Syndrome Created by Sender*
 - i. First segment of data sent by sending TCP irrespective of the size of segment
 - ii. Data is accumulated in buffer by sending TCP until acknowledgment is received from receiving TCP (or) enough data accumulated to send a segment
 - iii. Repeat step 2 until transmission completes
- Nagle's algorithm works based on speed of application program and the network speed
 - Faster the application program larger the segment size
- **Advantage:** Simple to implement



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

➤ *Silly Window Syndrome Created by Receiver*

➤ Syndrome created when serving application consumes slowly

➤ *Example*

➤ 1 KB data blocks created by sending application

➤ 1 byte of consumed at a time by receiving application

➤ Once sending window buffer is full, window size (rwnd) becomes 0

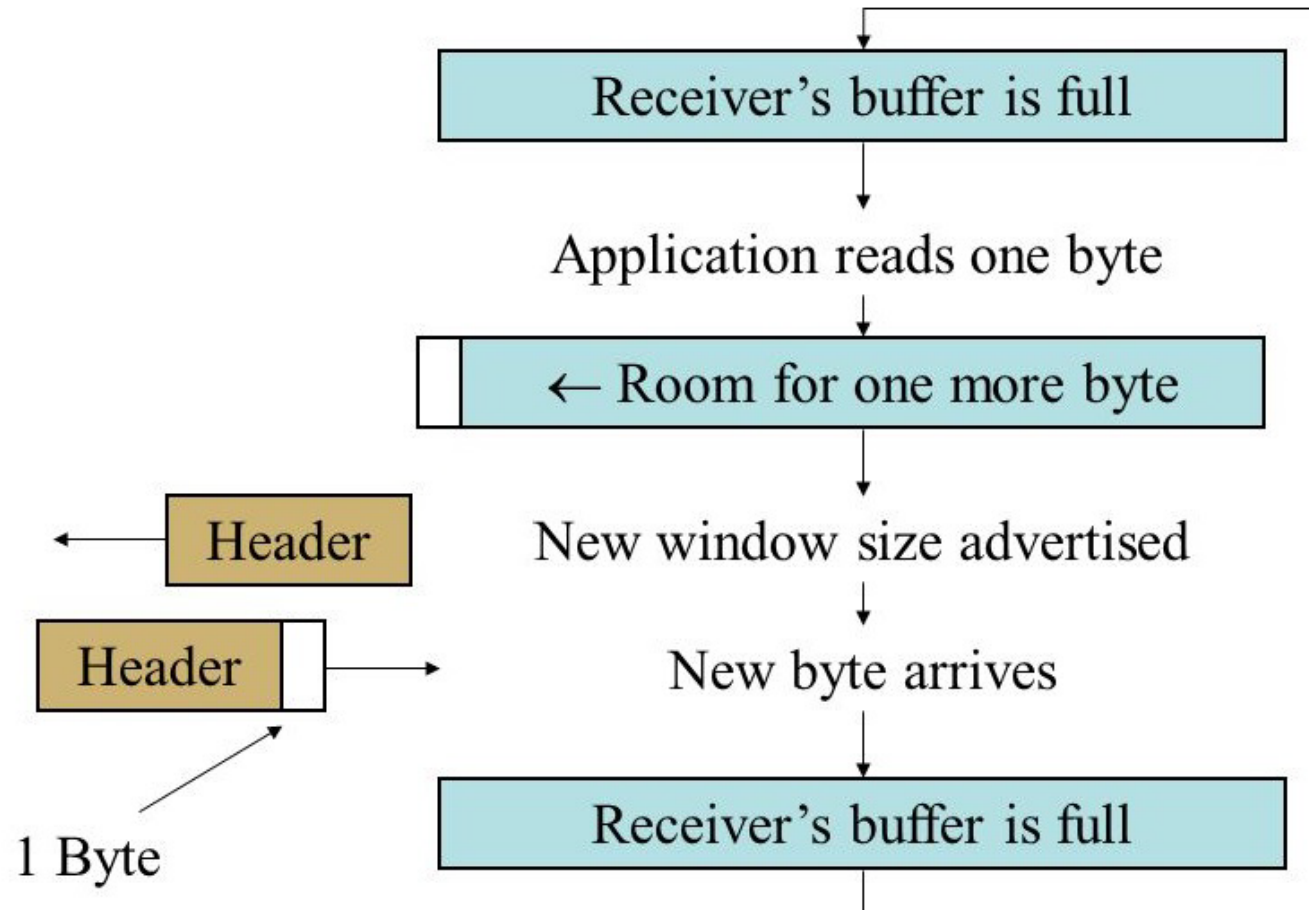
➤ *Solution 1:* Clarks's Solution

➤ Send ACK as data segment arrives

➤ Set rwnd = 0 iff receive buffer is half empty (or) there is enough space



Transmission Control Protocol (TCP)



Silly Window Syndrome at Receiver's Side



Transmission Control Protocol (TCP)

TCP Flow Control Contd...

➤ *Silly Window Syndrome Created by Receiver Contd...*

➤ **Solution 2:** Delayed Acknowledgment

- Acknowledgement is withheld when segment arrives
- Receiver waits for space in incoming buffer before acknowledging
- Delayed ACK prevents sending TCP from sliding
- Delayed ACK reduces network traffic
- Note: ACK not to be delayed by more than 500 ms



TCP Error Control



Transmission Control Protocol (TCP)

TCP Error Control

- TCP – Reliable Transport layer Protocol
- Entire data stream to be delivered without error / loss / duplication
- Reliability is provided by TCP using Error Control
- Error Control includes:
 - Finding and resending corrupted segments
 - Resending lost segments
 - Storing out-of-order segments till missed segments arrive
 - Discarding duplicate segments
- Error control achieved by: Checksum, Acknowledgement and Time-out



Transmission Control Protocol (TCP)

TCP Error Control Contd...

a) Checksum

- TCP uses 16-bit mandatory checksum field
- Checksum field associated with each segment
 - Checks for corrupted segment
- Invalid Checksum: Segment discarded by receiving TCP & considered lost



Transmission Control Protocol (TCP)

TCP Error Control Contd...

b) Acknowledgement

- Acknowledgement segments (ACK) carry no data & confirms data segment receipt
- Types : Cumulative and Selective Acknowledgment
- *Cumulative Acknowledgment (ACK)*
 - 32-bit ACK field used
 - Acknowledges segments cumulatively (sets ACK flag to 1)
 - No feedback provided for discarded, lost or duplicate segments
- *Selective Acknowledgement (SACK)*
 - Reports out of order & duplicate segments



Transmission Control Protocol (TCP)

TCP Error Control Contd...

b) Acknowledgement Contd...

- No provision for SACK in TCP header
- SACK included as part of options field in the TCP header
- *Rules for Generating Acknowledgments*
 - 1) When data is sent from sender to receiver, ACK provides the next Seq. No expected to be received
 - This results in less traffic and less segments between sender and receiver
 - 2) In case of one in-order segment remaining, receiver needs to delay sending ACK segment. Network traffic is thus reduced.



Transmission Control Protocol (TCP)

TCP Error Control Contd...

b) Acknowledgement Contd...

- 3) At no point of time there should be more than two in-order segments unacknowledged. (Thwarts unnecessary retransmission)
- 4) Receiver acknowledges (ACK) an higher out-of-order sequence number immediately leading to fast retransmission of next segment
- 5) Receiver sends ACK when a missing segment arrives. Segments reported as missing are thus informed to the receiver
- 6) Receiver discards duplicate segment & sends ACK indicating the next in-order segment. Lost ACK segment problems are thus solved.



Transmission Control Protocol (TCP)

TCP Error Control Contd...

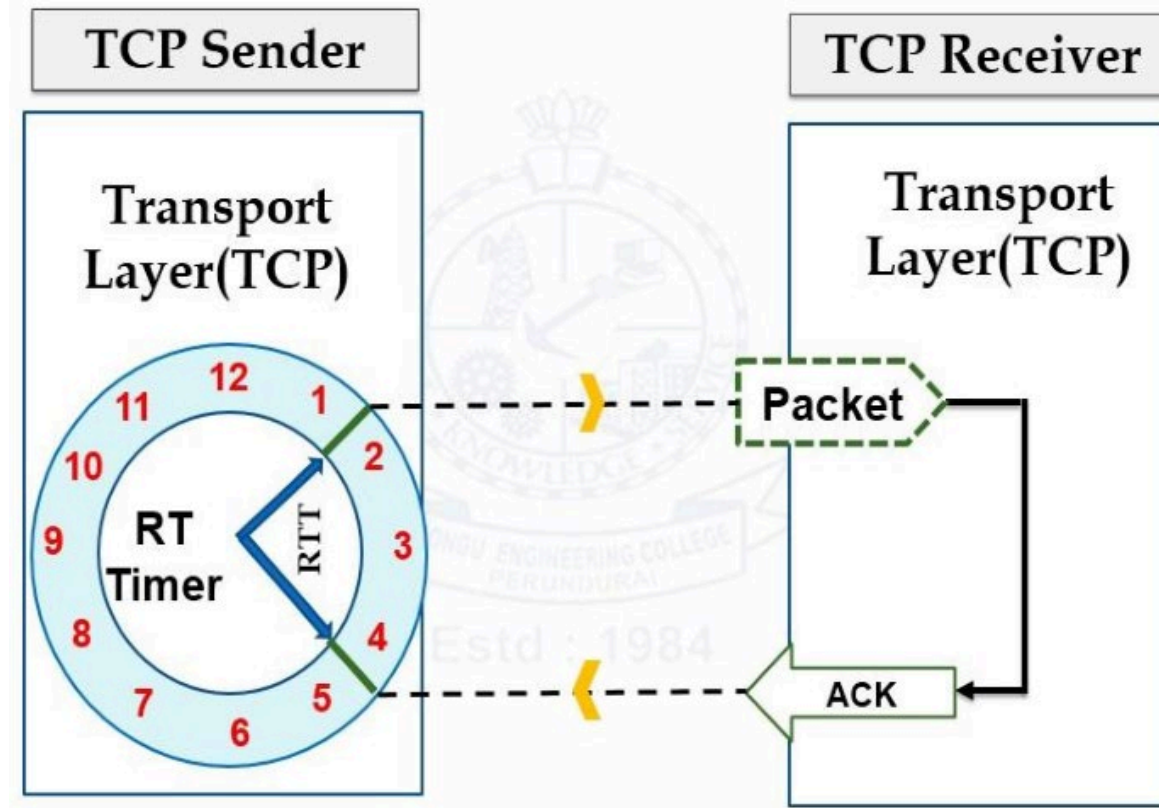
c) Retransmission of Segments

- When retransmission occurs?
 - Expiry of retransmission timer (or)
 - Sender receives more than 2 duplicate ACK's for 1st segment
- *Retransmission after RTO*
 - One retransmission time-out (RTO) maintained by sending TCP for each connection
 - In case of time-out, timer is restarted by TCP & first segment of Queue is sent
 - This version of TCP is called *Tahoe*



Transmission Control Protocol (TCP)

TCP Error Control Contd...



Retransmission after Retransmission Timeout (RTO)



Transmission Control Protocol (TCP)

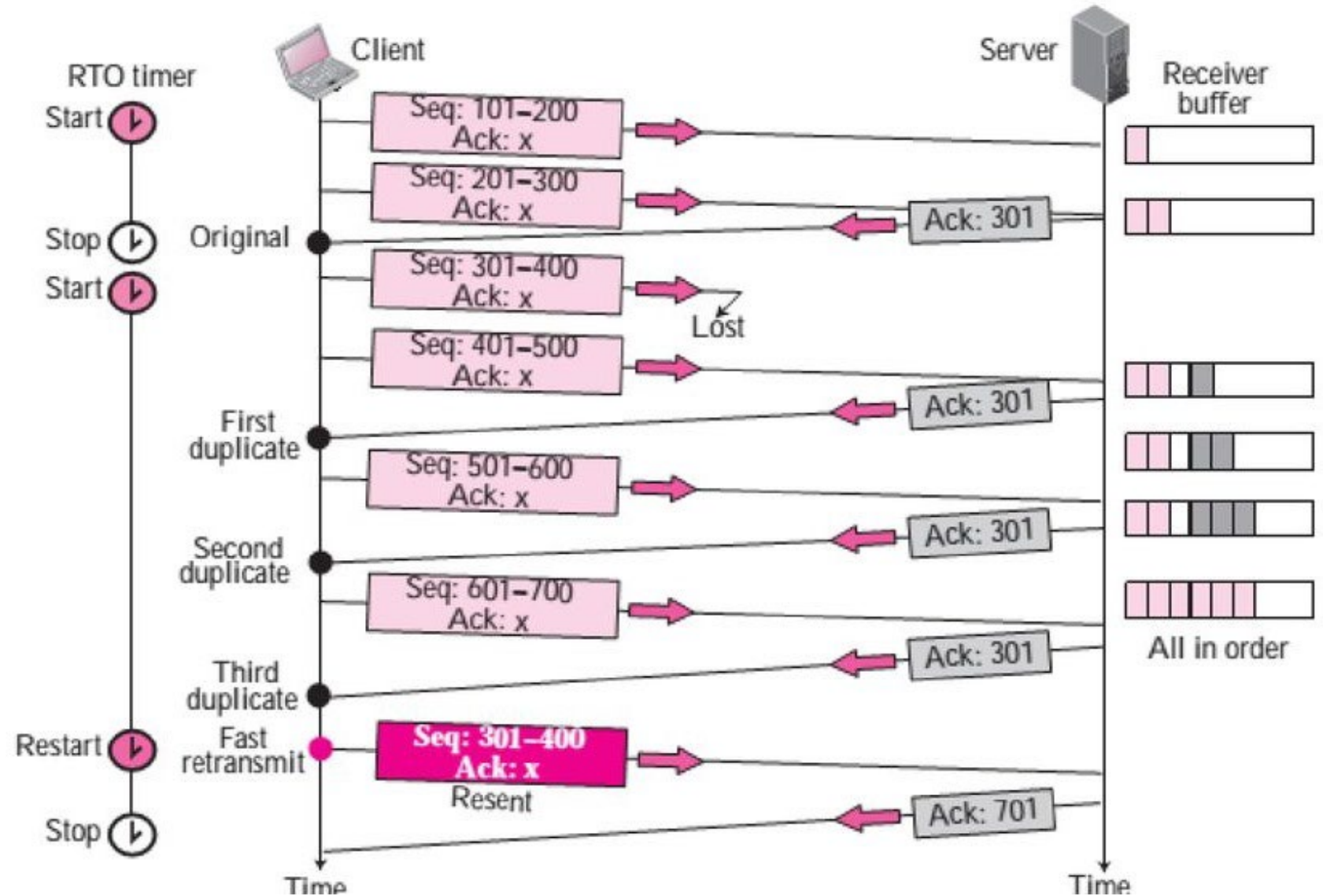
TCP Error Control Contd...

c) Retransmission of Segments Contd...

➤ *Retransmission after 3 Duplicate Segments*

- Also called as Fast Retransmission & followed by most implementations
- TCP version called as *Reno*
- If 3 identical duplicate ACK's along with the original ACK are received for a segment, the next segment is retransmitted
- Note: Retransmission does not wait for time-out in this case

Retransmission after 3 Duplicate Segments





Transmission Control Protocol (TCP)

TCP Error Control Contd...

d) Out-of-Order Segments

- Out-of-Order segments are not discarded by TCP
- TCP flags such segments as out-of-order and store them temporarily until missing segments arrive
- TCP makes sure that data segments are delivered in sequence to the process



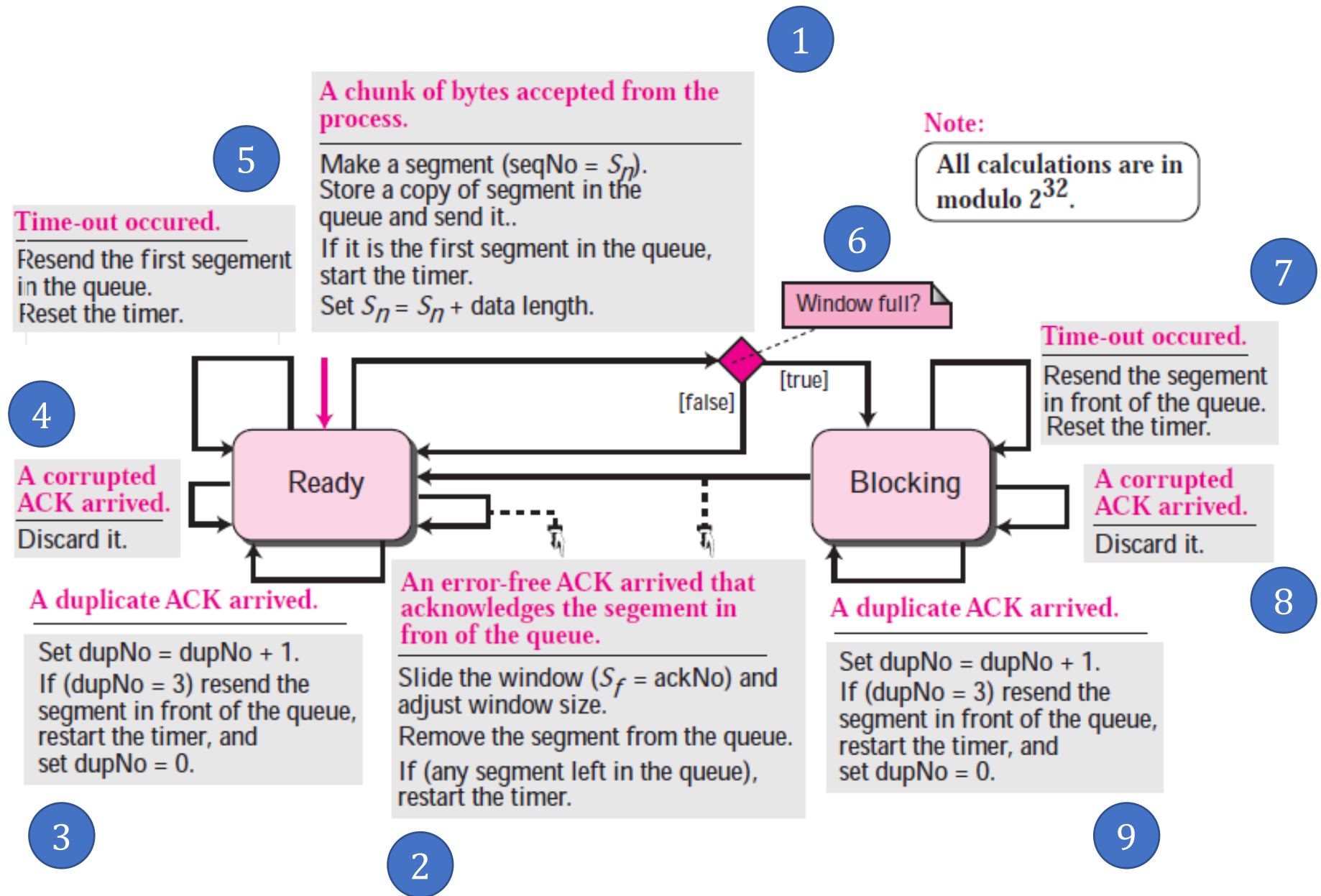
Transmission Control Protocol (TCP)

TCP Error Control Contd...

e) FSM for Data Transfer in TCP

- FSM – Finite State Machine
- Similar to Selective repeat and Go Back-N protocol
- *Sender-side & Receiver Side FSM*
 - *Assumption :* Unidirectional communication
 - *Ignored Parameters:* Selective ACK and Congestion Control
 - Nagle's algorithm / Windows shutdown not included in FSM
 - *Advantage:* Fast transmission policy using 3 duplicate ACK segments
 - *Bi-directional FSM :* Complex and more practical

Simplified FSM for TCP Sender Side



Simplified FSM for TCP Receiver Side

Note:

All calculations are in
modulo 2^{32} .

An expected error-free segment arrived.

Buffer the message.

$R_n = R_n + \text{data length}$.

If the ACK-delaying timer is running, stop
the timer and send a cumulative ACK. Else,
start the ACK-delaying timer.

1

5

ACK-delaying timer expired.

Send the delayed ACK.

**An error-free, but out-of
order segment arrived**

Store the segment if not duplicate.
Send an ACK with ackNo equal
to the sequence number of expected
segment (duplicate ACK).

4

A corrupted segment arrived

Discard the segment.

2

3

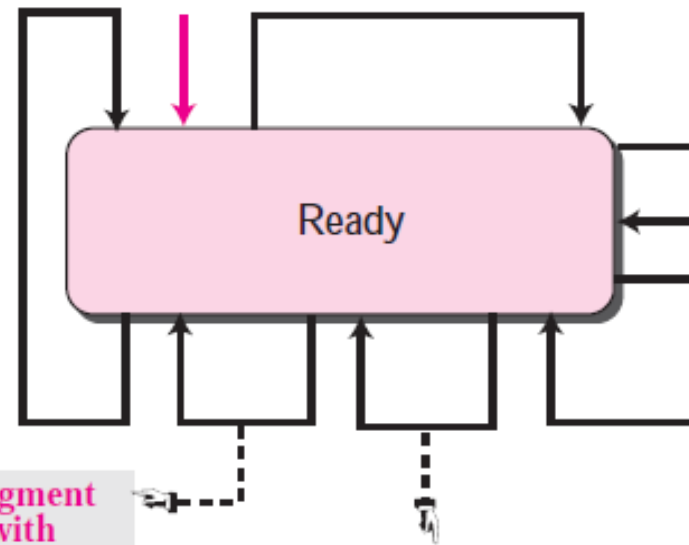
**An error-free duplicate segment
or an error-free segment with
sequence number outside
window arrived**

Discard the segment.
Send an ACK with ackNo equal
to the sequence number of expected
segment (duplicate ACK).

**A request for delivery of
k bytes of data from
process came**

Deliver the data.
Slide the window and adjust
window size.

6





TCP Congestion Control



Transmission Control Protocol (TCP)

TCP Congestion Control

- Congestion window and congestion policy handles TCP congestion

a) Congestion Window

- Client window size (rwnd) decided by the available buffer space of Server
- Ignored entity in deciding window size : Network Congestion
- Sender window size determined by,
 - rwnd (receiver advertised window size) &
 - cwnd (Congestion window size)

Actual window size = $\min(\text{rwnd}, \text{cwnd})$



Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

TERMINOLOGIES & ABBREVIATIONS USED

- *rwnd* – Sender Window Size
- *cwnd* – Congestion Window Size
- *ssthresh* – Slow Start Threshold
- *MSS* – Maximum Segment Size
- *ACK* – Acknowledgement
- *RTT* – Round Trip Time
- *RTO* – Retransmission Time-out



Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

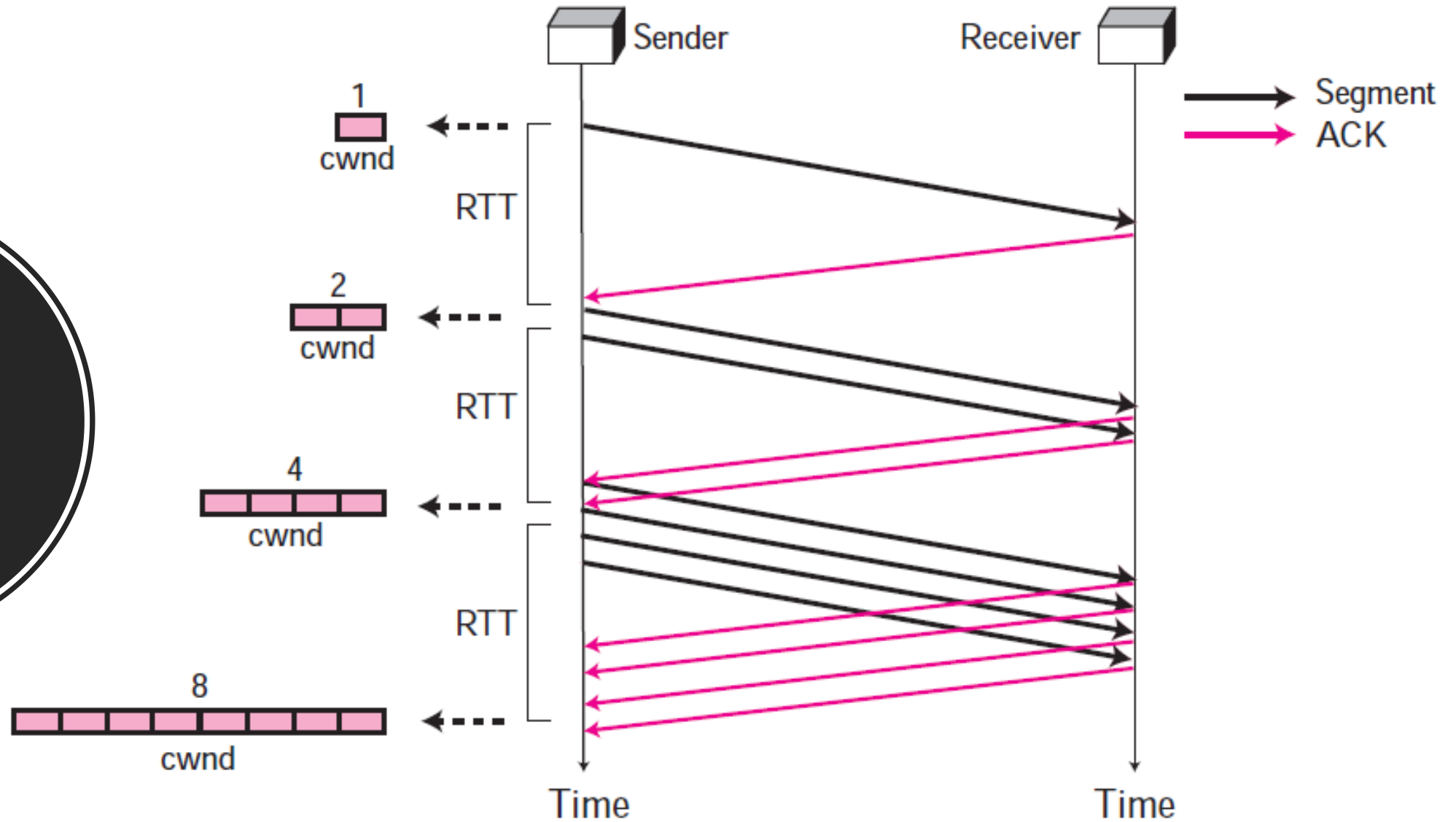
b) Congestion Policy

- Three phases : Slow Start, Congestion avoidance & Congestion detection

i. Slow Start (Exponential Increase)

- **Assumption:** $rwnd > cwnd$
- $cwnd$ initialized to one Maximum window size (MSS)
- On arrival of each ACK, $cwnd$ increases by 1
- Algorithm starts slowly & grows exponentially
- Delayed ACK policy is ignored
- **Note:** Consider each segment is individually acknowledged

Slow Start, Exponential Increase





Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

b) Congestion Policy

i. Slow Start (Exponential Increase) contd...

- Initial value of cwnd = 1 MSS

No of MSS sent	No of Segments Acknowledged	RTT	cwnd in MSS
Nil	Nil	-	1
1	1	1	$1 \times 2 = 2 \Rightarrow 2^1$
2	2	2	$2 \times 2 = 4 \Rightarrow 2^2$
4	4	3	$4 \times 2 = 8 \Rightarrow 2^3$
8	8	4	$8 \times 2 = 16 \Rightarrow 2^4$



Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

b) Congestion Policy

i. Slow Start (Exponential Increase) contd...

- For Delayed Acknowledgements
 - If multiple segments are acknowledged accumulatively, cwnd increases by 1
 - **Example:** if ACK = 4 then cwnd = 1
 - Growth is exponential but not to the power of 2
 - Slow start stops with a threshold value ***ssthresh***
 - It stops when ***window size = ssthresh***



Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

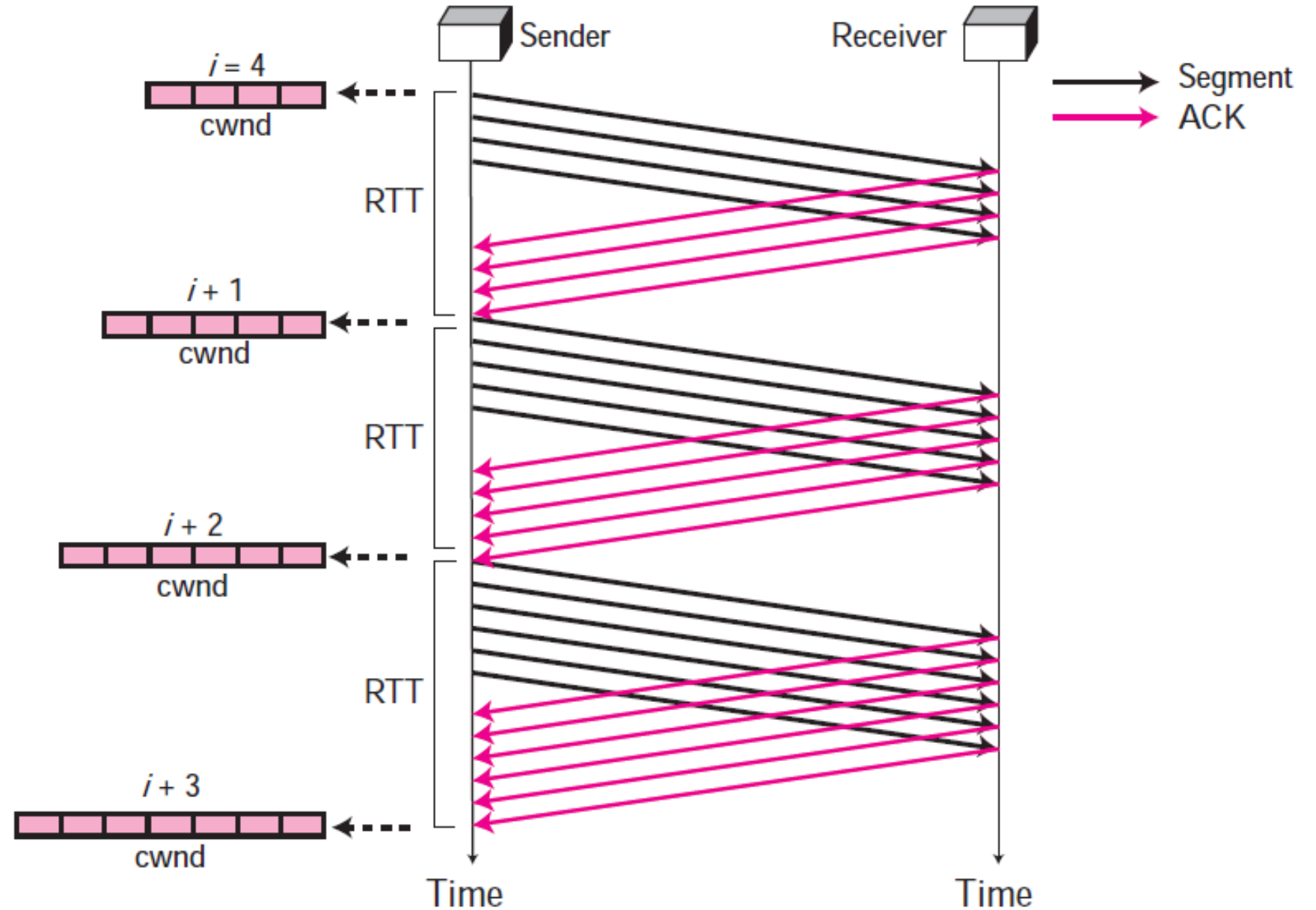
b) Congestion Policy

ii. Congestion Avoidance : Additive Increase

- Slow start increases congestion window size (cwnd) exponentially
- Congestion avoidance increases cwnd additively
- Additive phase begins when slow start reaches ssthresh i.e. $cwnd = I$
- Increase in cwnd is based on RTT & not on number of ACK's

RTT	cwnd in MSS
1	i
2	i + 1

Congestive avoidance, Additive Increase





Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

b) Congestion Policy

iii. Congestion Detection : Multiplicative Decrease

- Size of Cwnd must be decreased in case of congestion
- Retransmission occurs during missing segments / lost segments
- Retransmission helps identify whether congestion has occurred or not
- Retransmission occurs when
 - There is RTO Time-out
 - On receipt of three duplicate ACK's
- Note: In both cases, ssthresh is reduced by half (Multiplicative decrease)



Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

b) Congestion Policy

iii. Congestion Detection : Multiplicative Decrease Contd...

a) Time-out increases possibility of congestion. TCP reacts as follows:

- Ssthresh set to half the value of rwnd
- Cwnd initialized to 1
- Slow start phase is initiated again

b) Three duplicate ACK's indicates a weaker possibility of Congestion. Also called as fast transmission & fast recovery. TCP reacts as follows:

- Ssthresh set to half the value of rwnd



Transmission Control Protocol (TCP)

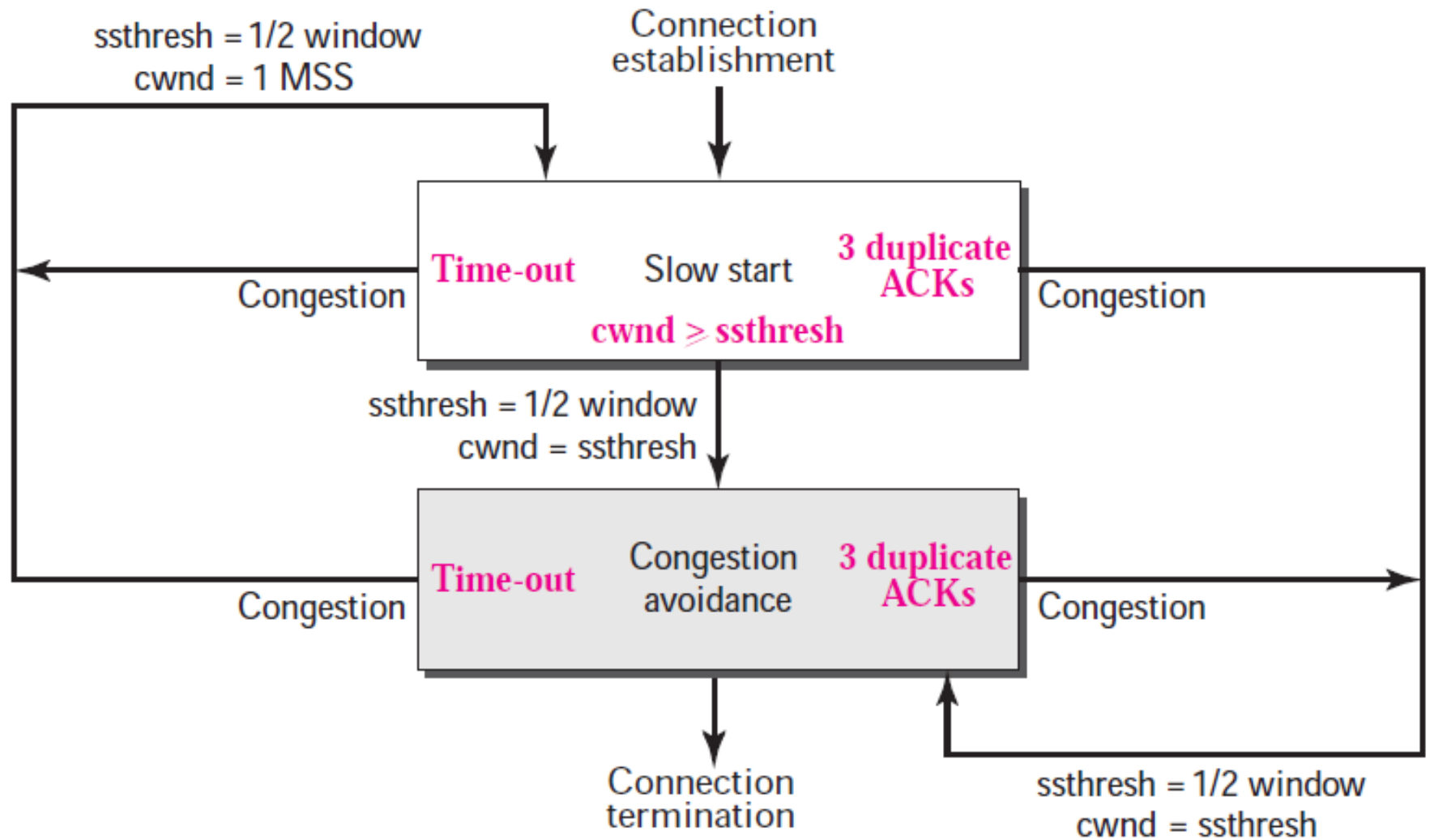
TCP Congestion Control Contd...

b) Congestion Policy

iii. Congestion Detection : Multiplicative Decrease Contd...

- $Cwnd = ssthresh$
- Congestion avoidance phase is initiated again

TCP Congestion Policy : A Summary





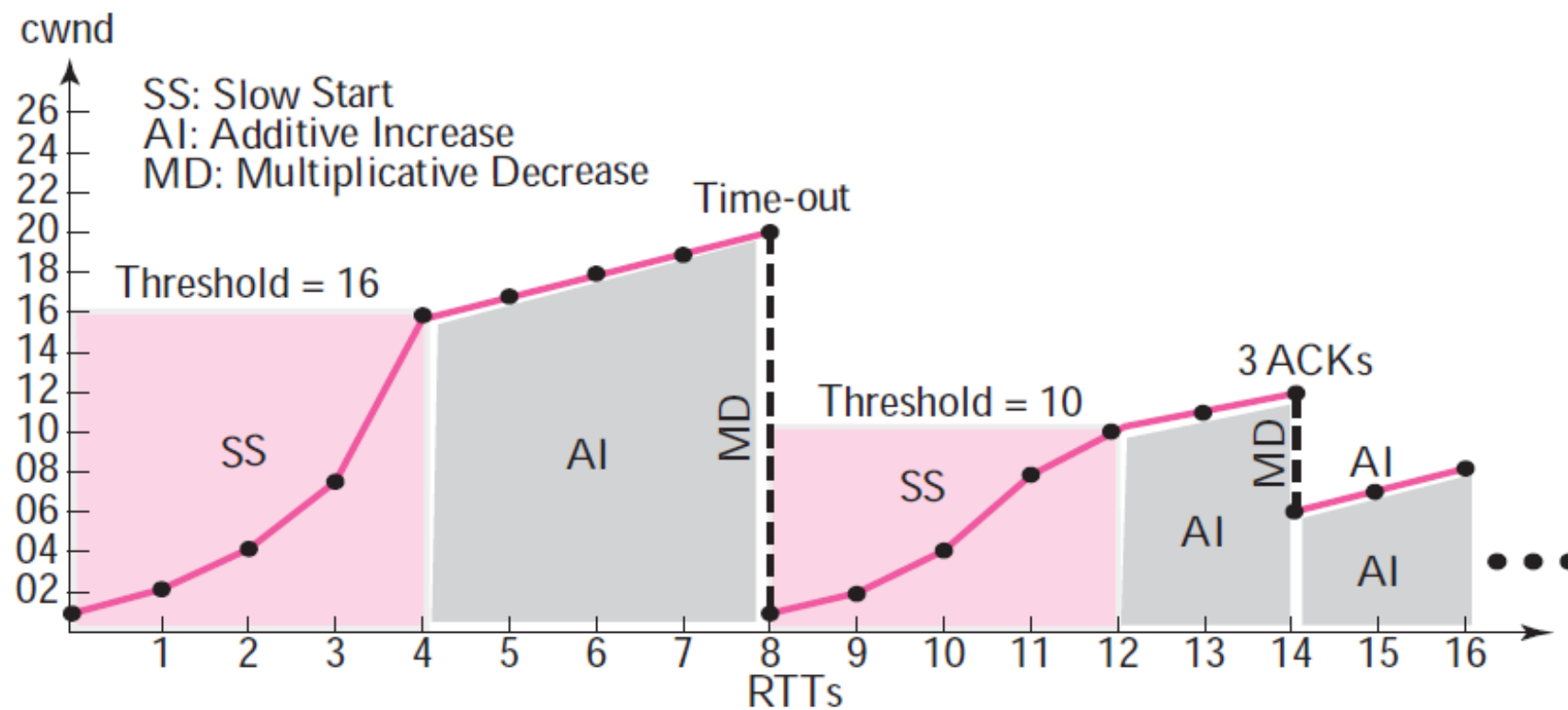
Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

Summarization with Example

- Assumptions
 - Maximum window Size (MSS) = 32
 - Threshold (sssthresh) = 16
- TCP moves to slow start
 - rwnd starts from 1 and grows exponentially till it reaches sssthresh (16)
- Additive increase increases rwnd from 16 to 20 (one by one)
 - When rwnd = 20, time-out occurs
- Multiplicative Decrease: sssthresh reduced to 10 (half the window size)

Congestion Example





Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

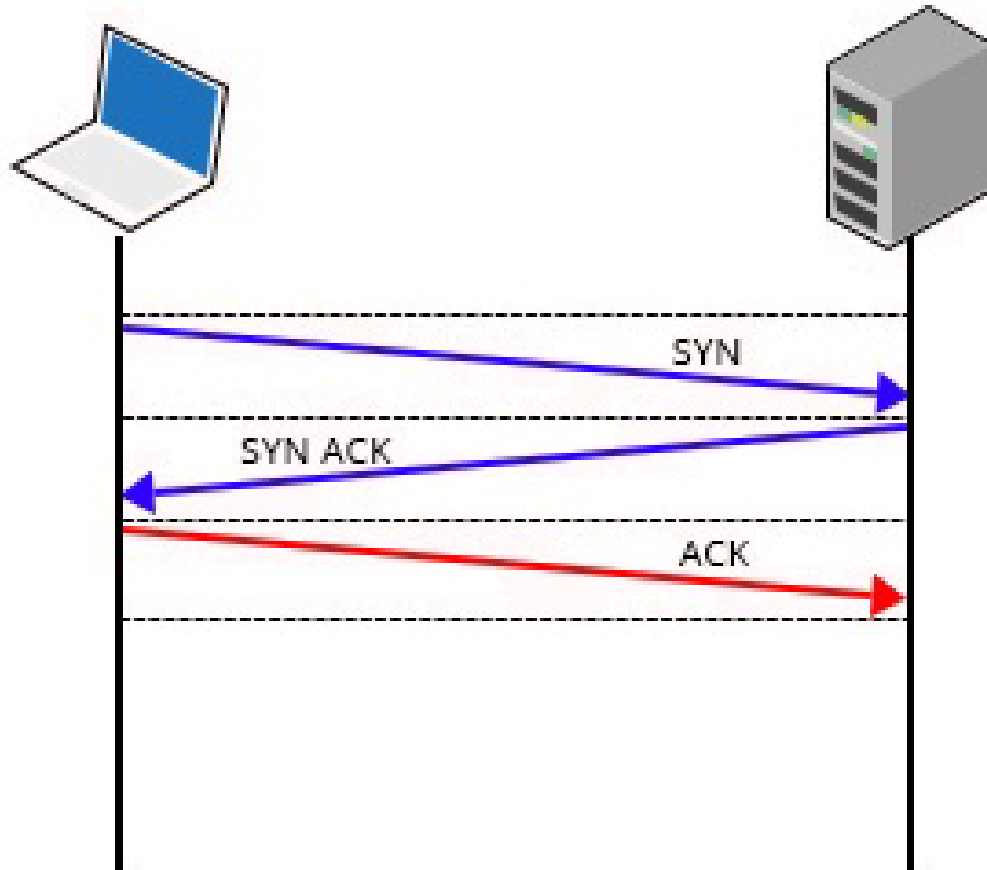
Summarization with Example Contd...

- New ssthresh = 10
- TCP moves to Slow start again
 - rwnd starts from 1 and grows exponentially till it reaches new ssthresh (10)
- Additive increase increases rwnd from 10 to 12 (one by one)
- 2 duplicate ACK's are received by the sender
- Multiplicative Decrease: ssthresh reduced to 6 (half the window size)

TCP

Sender

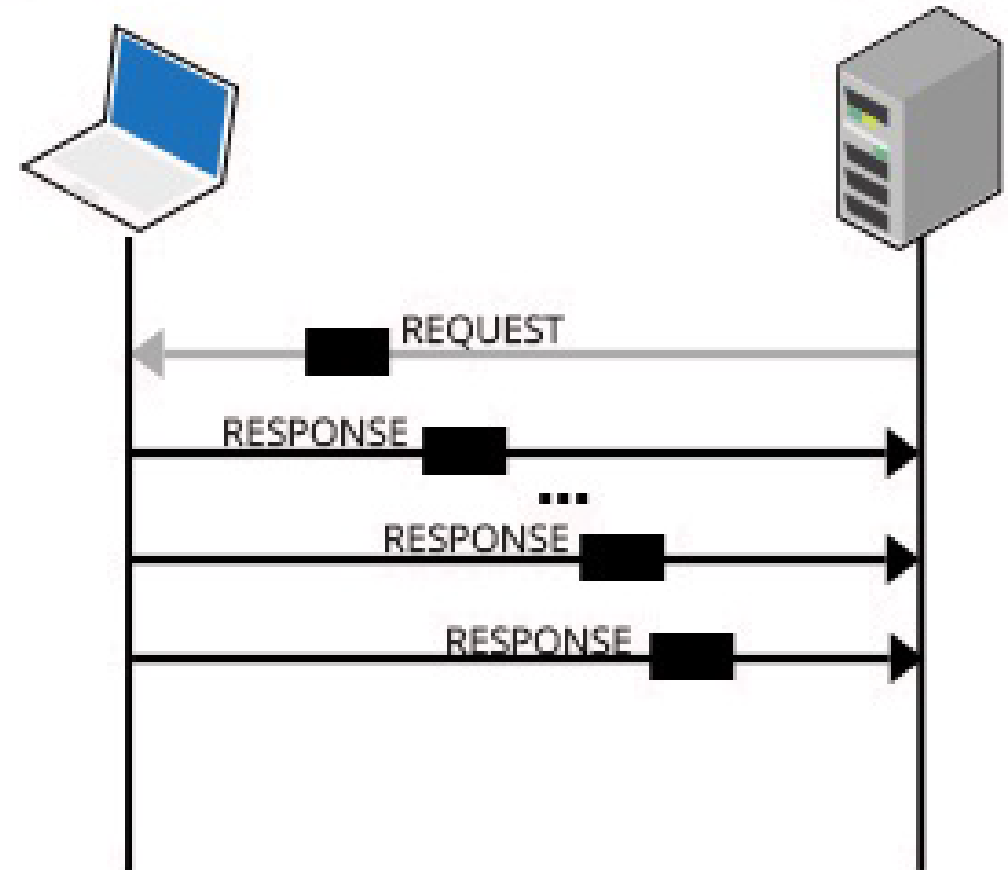
Receiver



UDP

Sender

Receiver



Difference between TCP & UDP



Transmission Control Protocol (TCP)

Difference between TCP & UDP

	TCP	UDP
Connection	Connection-oriented protocol	Connection-less protocol
Speed	The speed for TCP is slower	The speed for UDP is higher
Error Detection	Yes	No
Data Packets	Data packets are arranged in the correct order	Data packets are independent of each other, and therefore does not follow a sequence.
Acknowledgement	Acknowledgement Segments	No Acknowledgement Segments
Handshake Protocol	Uses protocols such as: - SYN, ACK, SYN-ACK	No Handshake, and therefore connectionless protocol.
Reliability	Successful deliverance of data to destination router, and therefore reliable.	Deliverance of data to the destination router is not reliable.



References

Learning Resources Used for Unit I

- 1) Behrouz A. Forouzan, “TCP IP Protocol Suite ” 4th edition, 2010, McGraw-Hill ISBN: 0073376043
- 2) <https://indigoo.com/petersblog/?p=185>
- 3) <https://www.ionos.com/digitalguide/server/know-how/sctp/>

Thank You