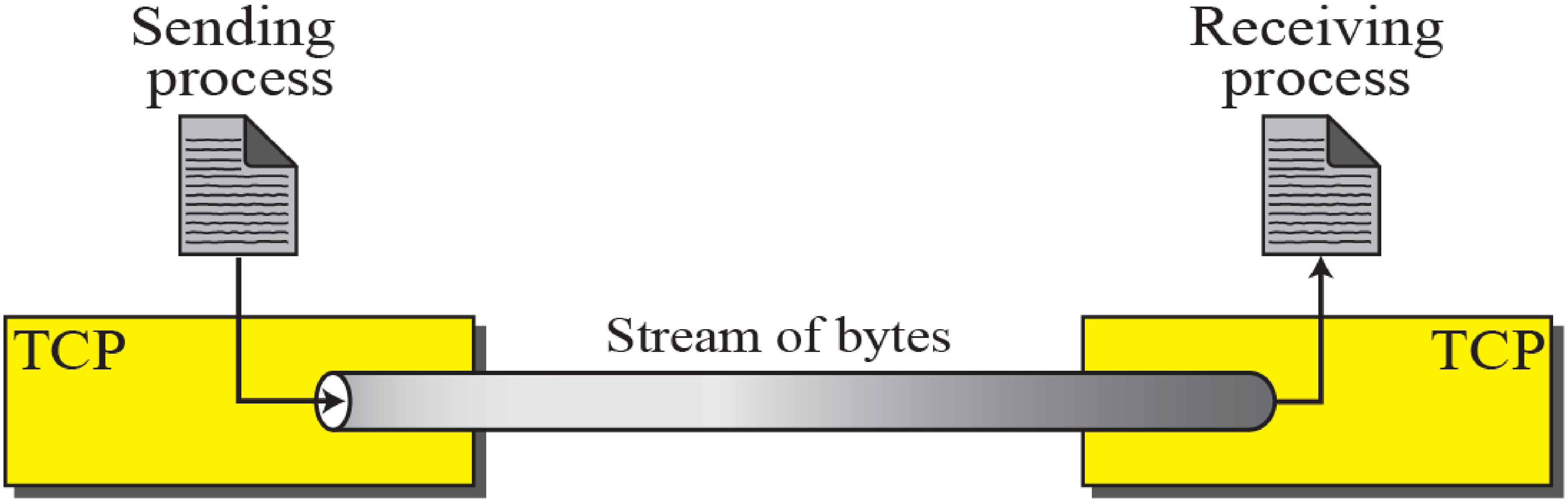
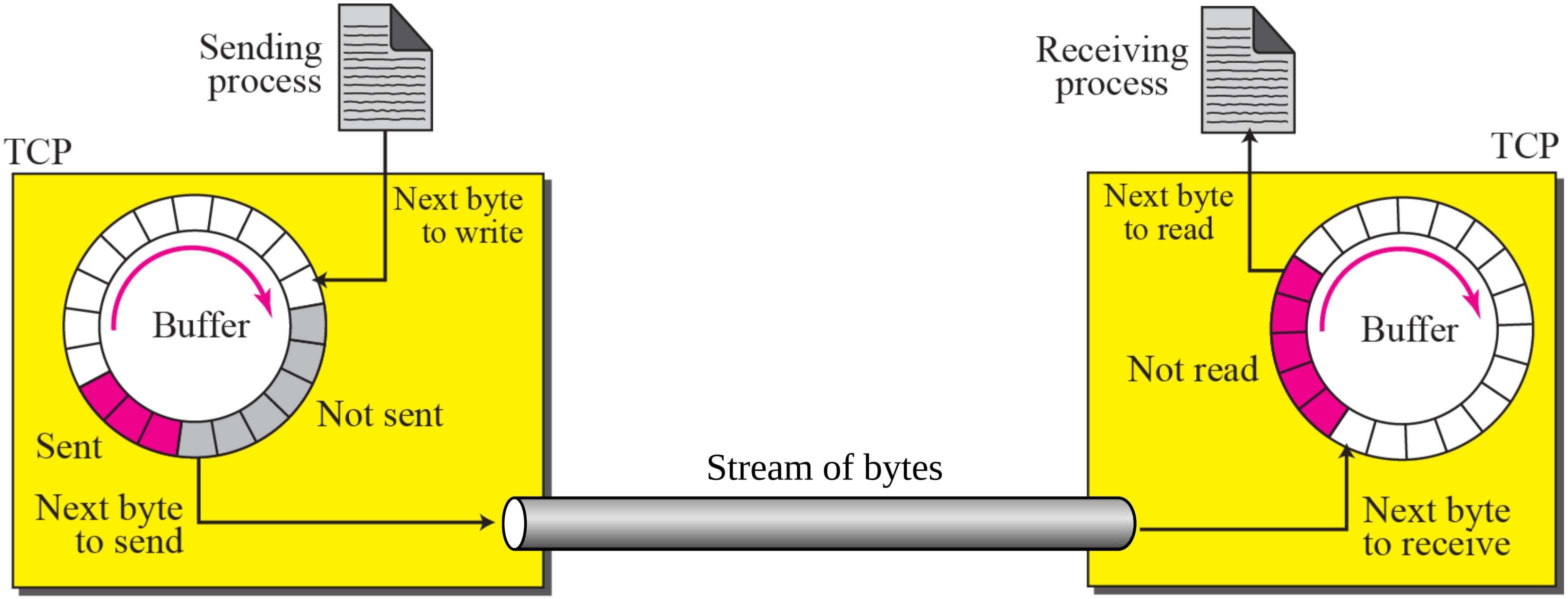
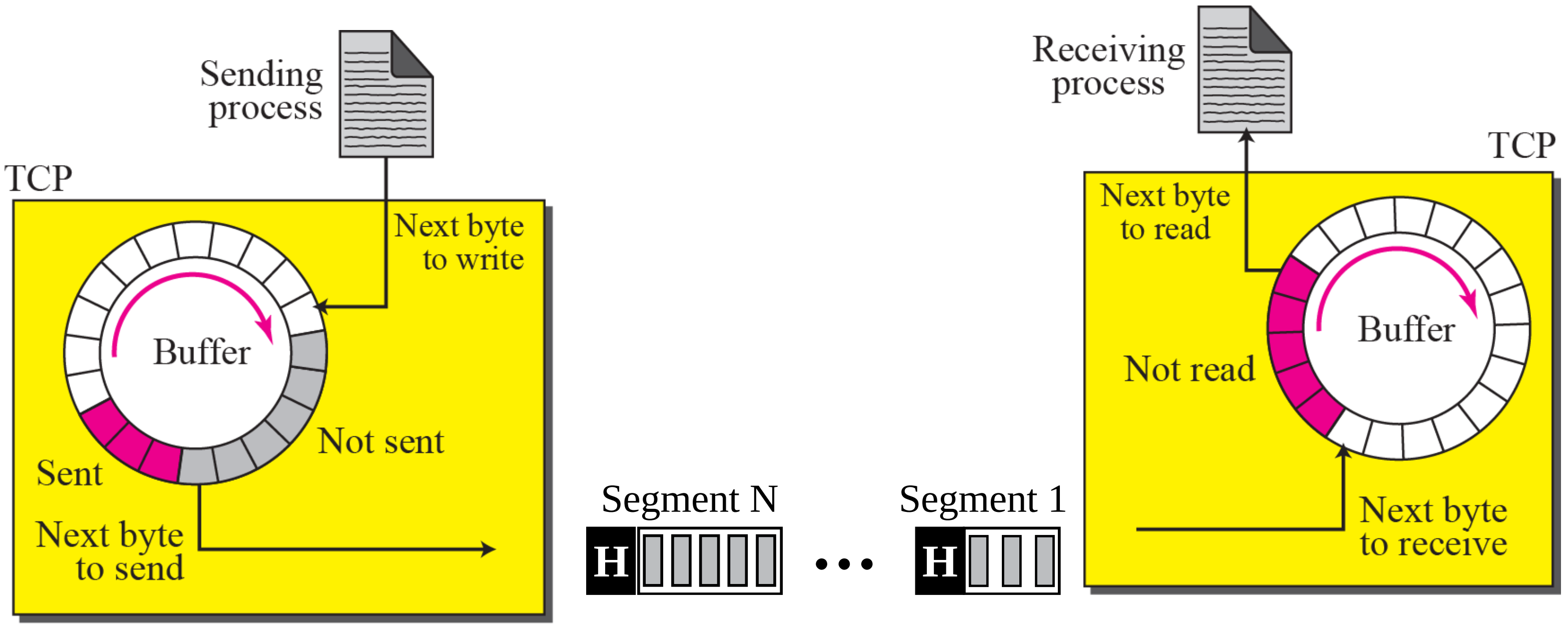
Transmission Control Protocol (TCP)


Sending process
Receiving process
TCP
Stream of bytes
TCP

Sending and receiving buffers



TCP segments



Numbering:

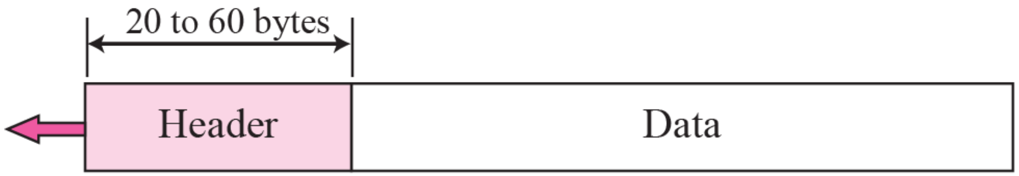
Byte Number

Sequence Number

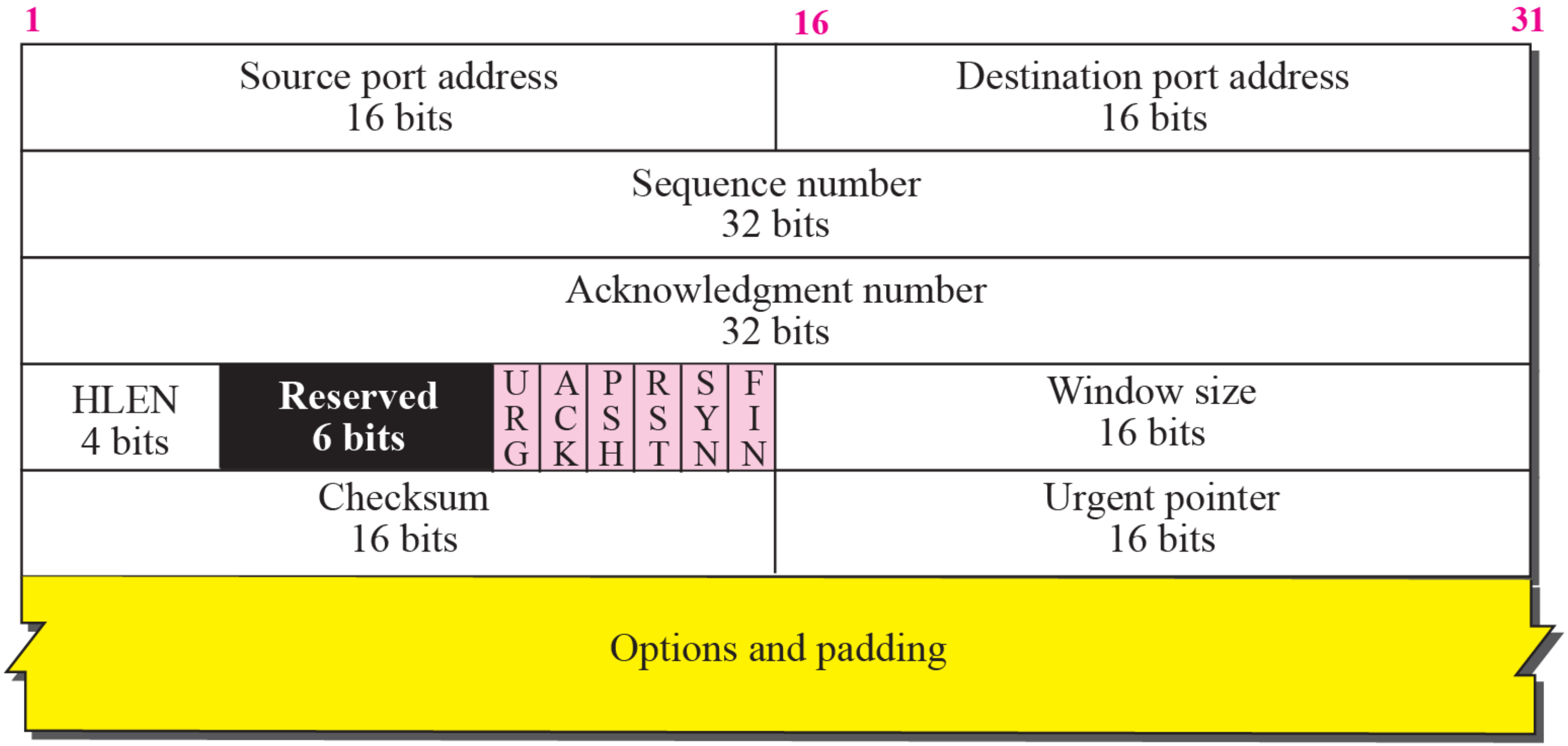
Data + Ctrl, Data, Ctrl (response) - Valid Sequence number

Acknowledge Number

TCP segment format



a. Segment



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

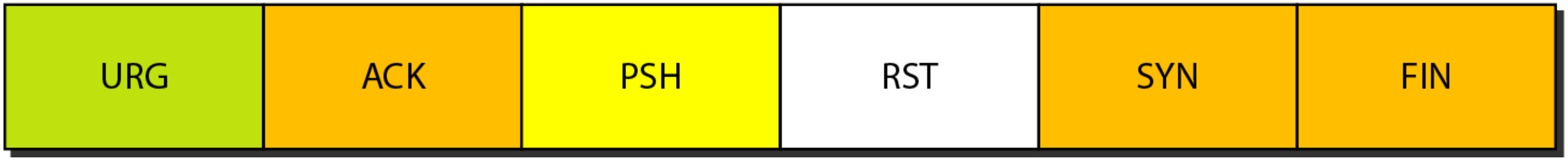
Options and padding

b. Header

Control field

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

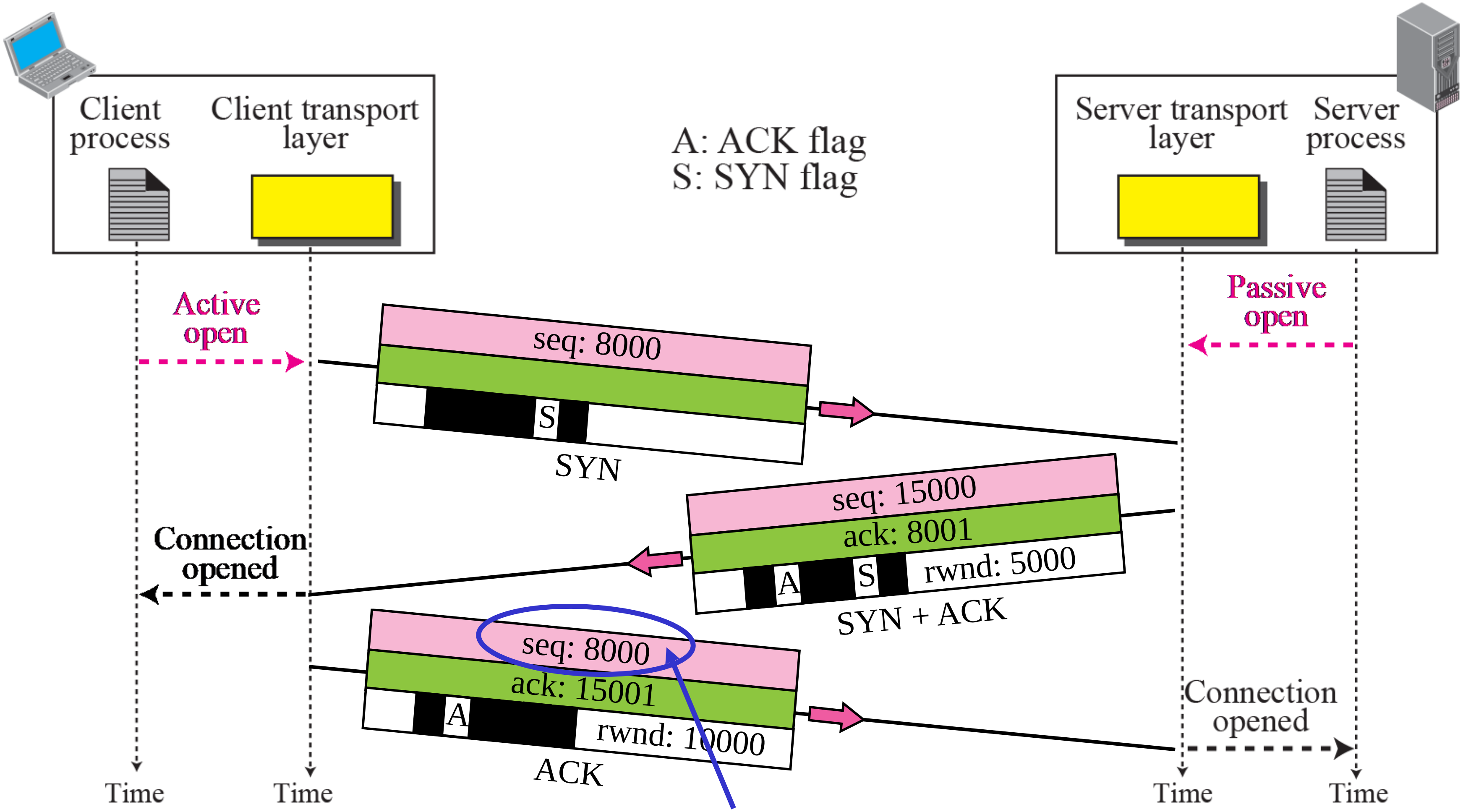


URG	ACK	PSH	RST	SYN	FIN

TCP Connections

- ✓ Connection Establishment
- ✓ Data Transfer
- ✓ Connection Termination
- ✓ Connection Reset

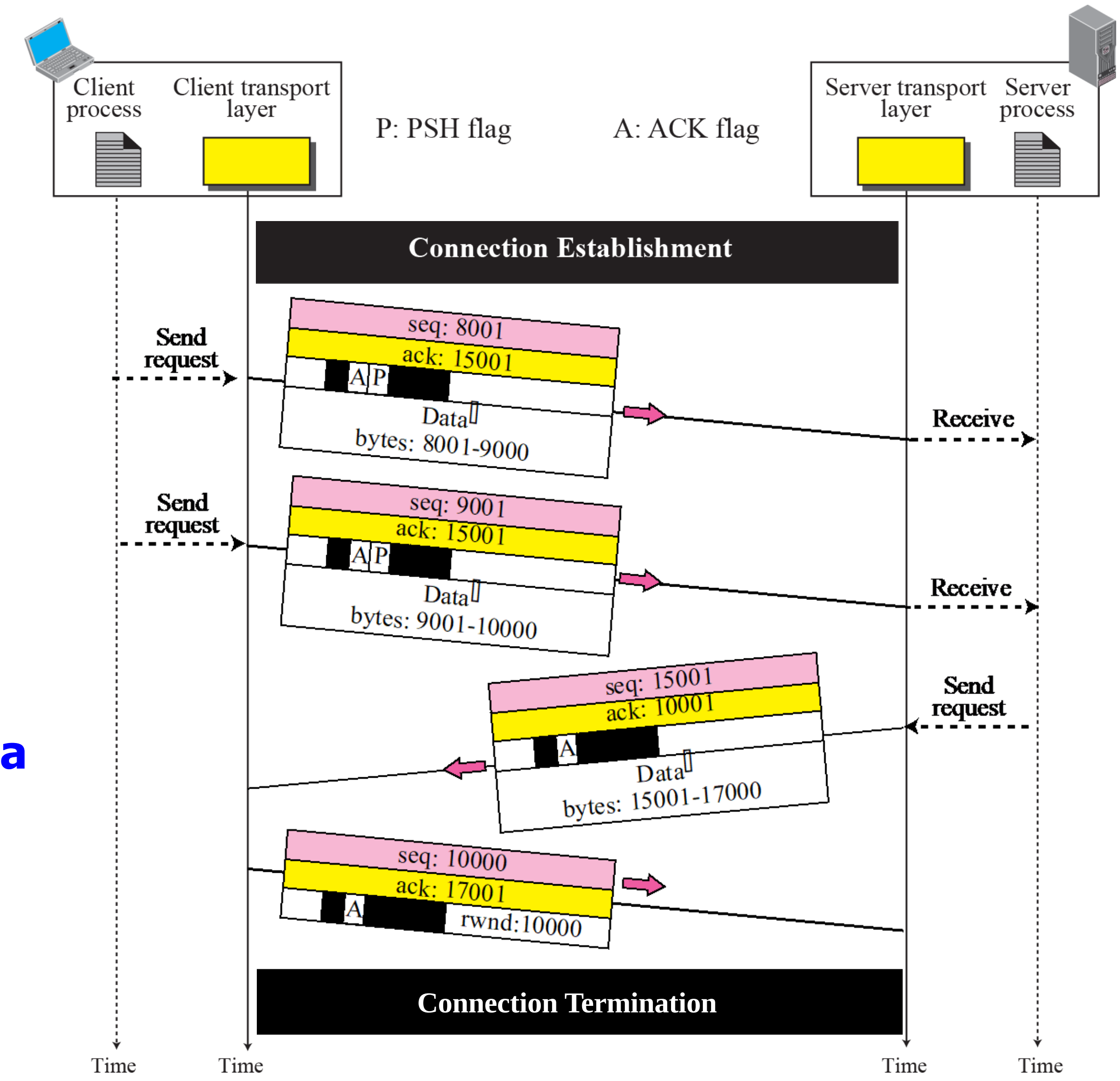
Connection establishment using three-way handshake



Means "no data" !

seq: 8001 if piggybacking

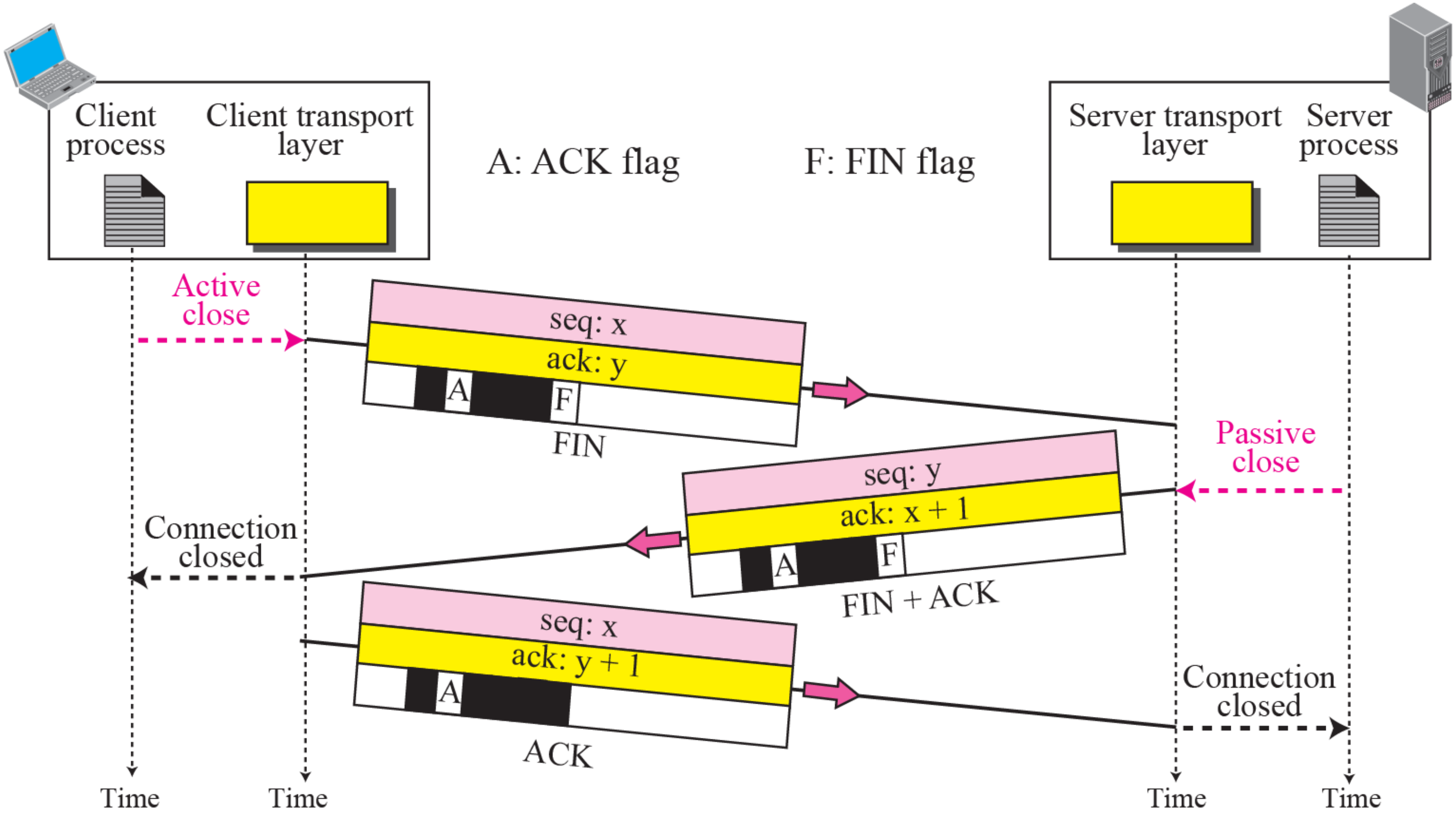
Data Transfer



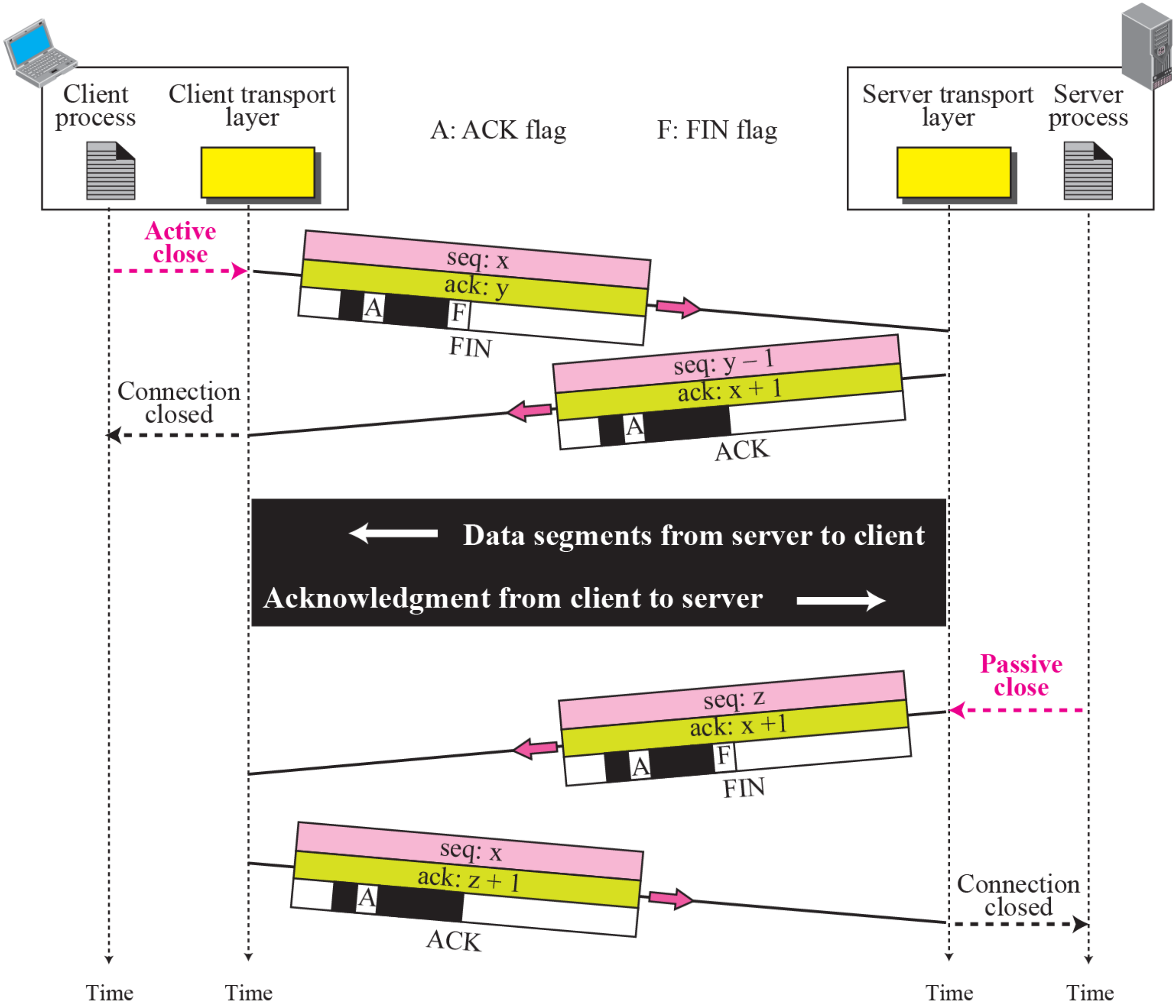
Pushing data

Urgent data

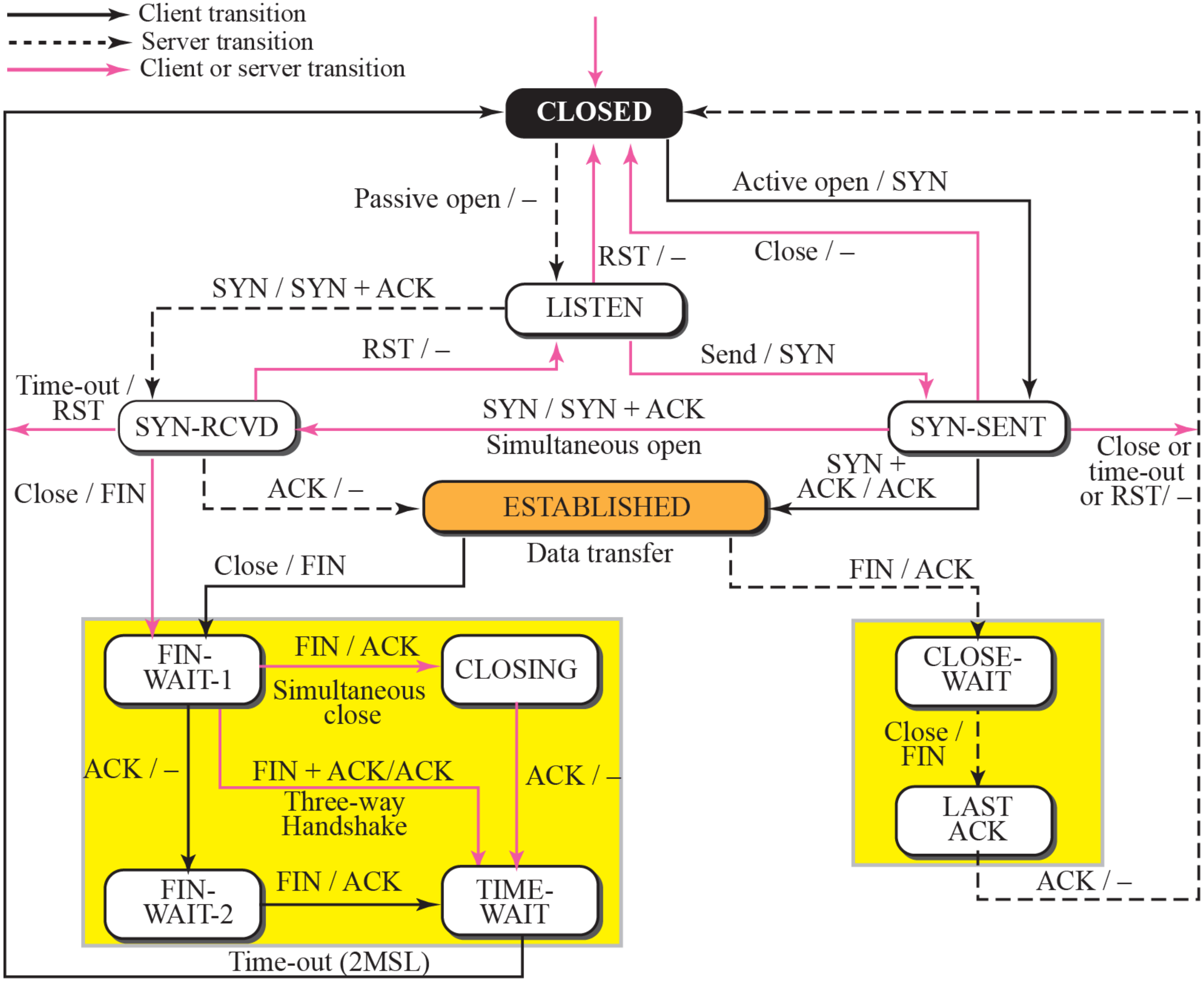
Connection termination using three-way handshake



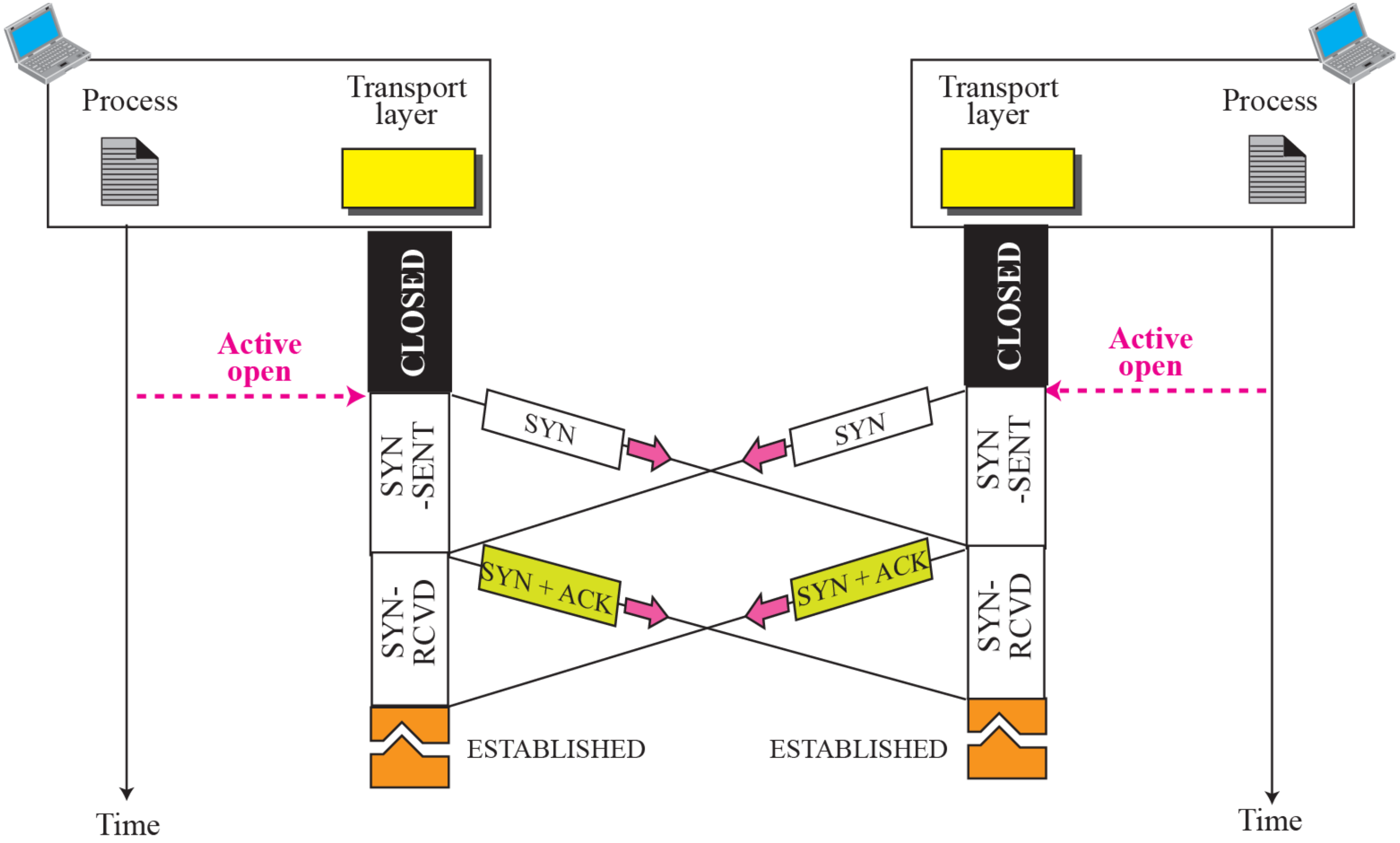
Half-Close


Client process
Client transport layer
A: ACK flag
F: FIN flag
Server transport layer
Server process
Active close
seq: x
ack: y
A
F
FIN
seq: y – 1
ack: x + 1
A
ACK
Connection closed
Data segments from server to client
Acknowledgment from client to server
Passive close
seq: z
ack: x +1
A
F
FIN
seq: x
ack: z + 1
A
ACK
Connection closed
Time
Time
Time
Time

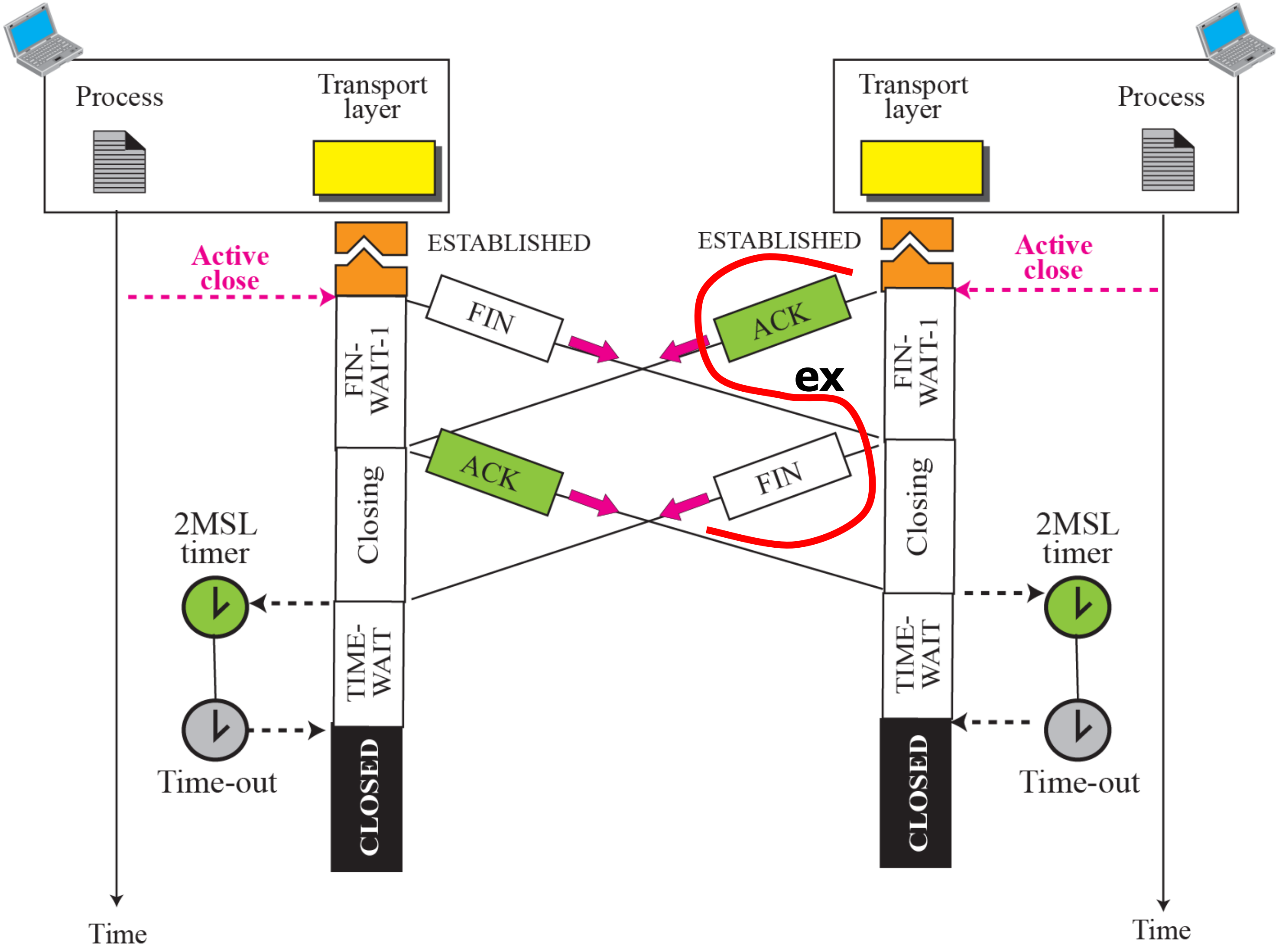
STATE TRANSITION DIAGRAM



Simultaneous open

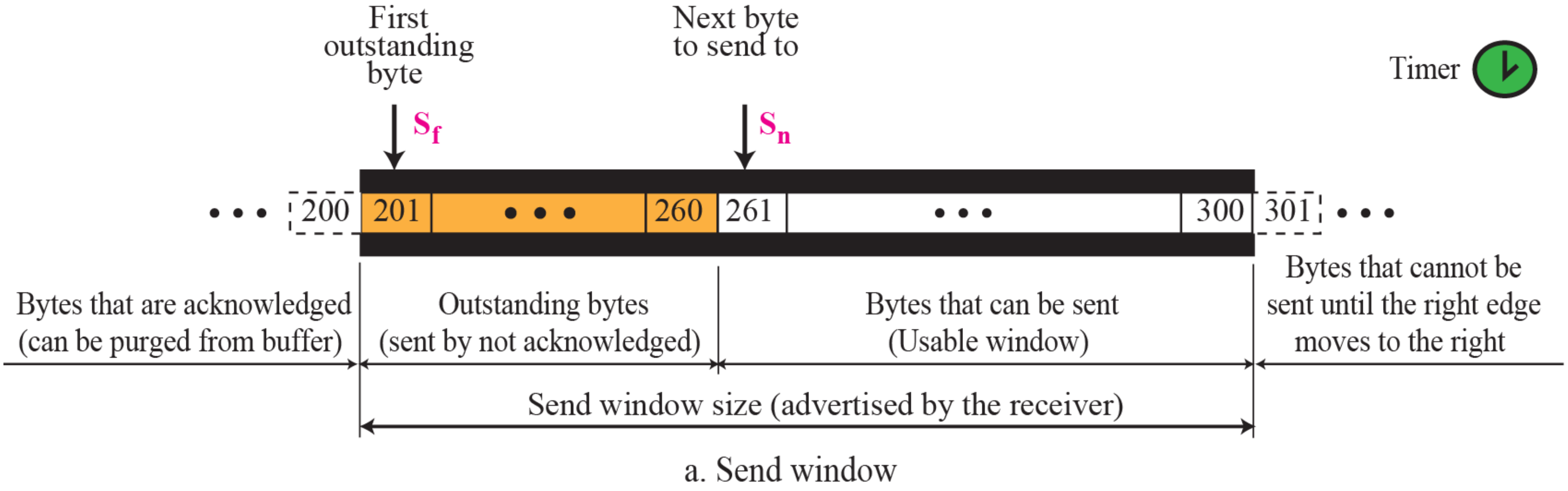


Simultaneous close

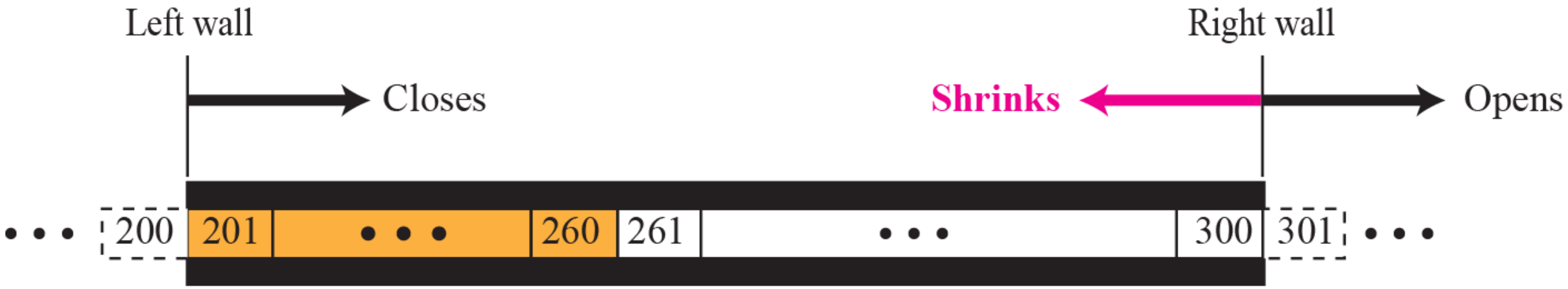


WINDOWS IN TCP

Send window in TCP

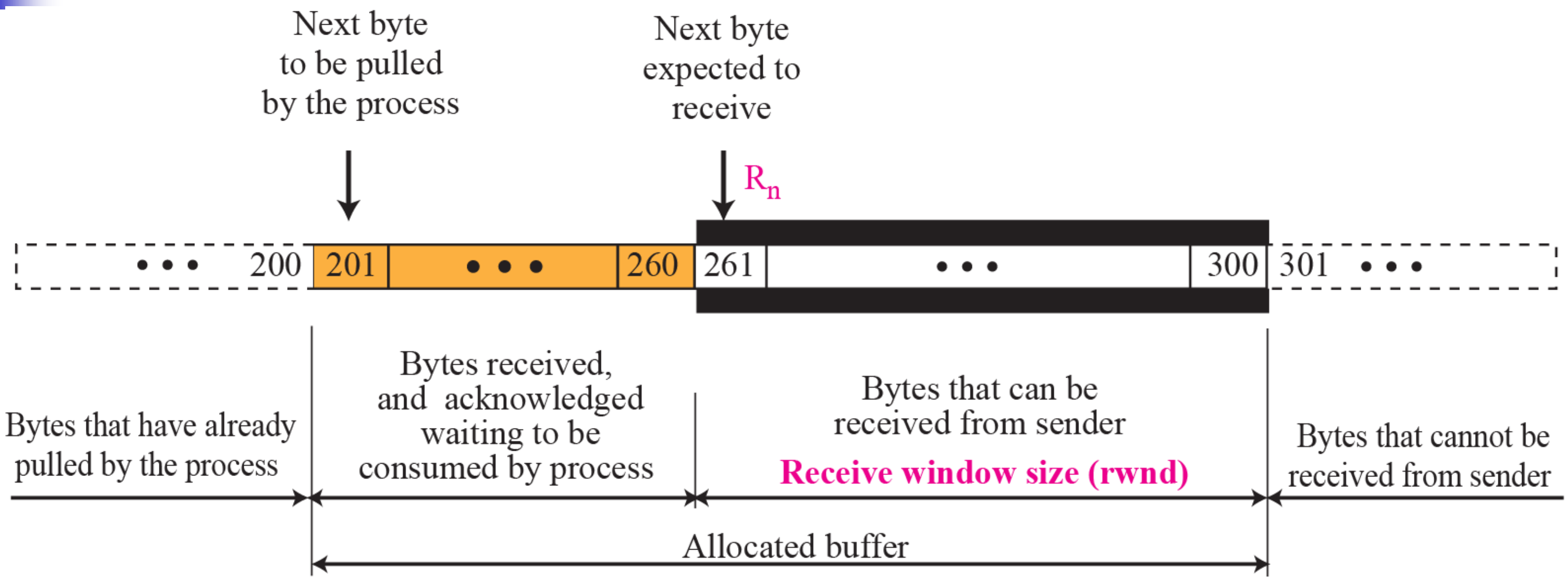

First outstanding byte
S_f
Next byte to send to
S_n
Timer
. . .
200
201
. . .
260
261
. . .
300
301
. . .
Bytes that are acknowledged (can be purged from buffer)
Outstanding bytes (sent by not acknowledged)
Bytes that can be sent (Usable window)
Bytes that cannot be sent until the right edge moves to the right
Send window size (advertised by the receiver)

a. Send window

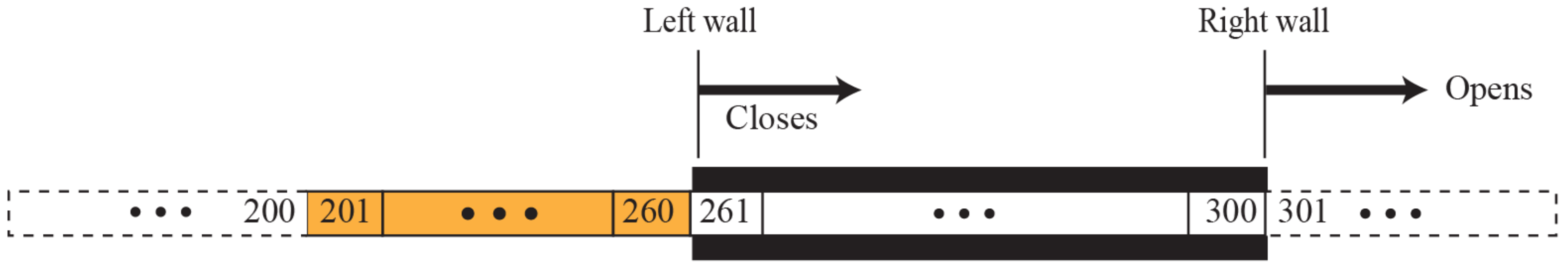


b. Opening, closing, and shrinking send window

Receive window in TCP



a. Receive window and allocated buffer

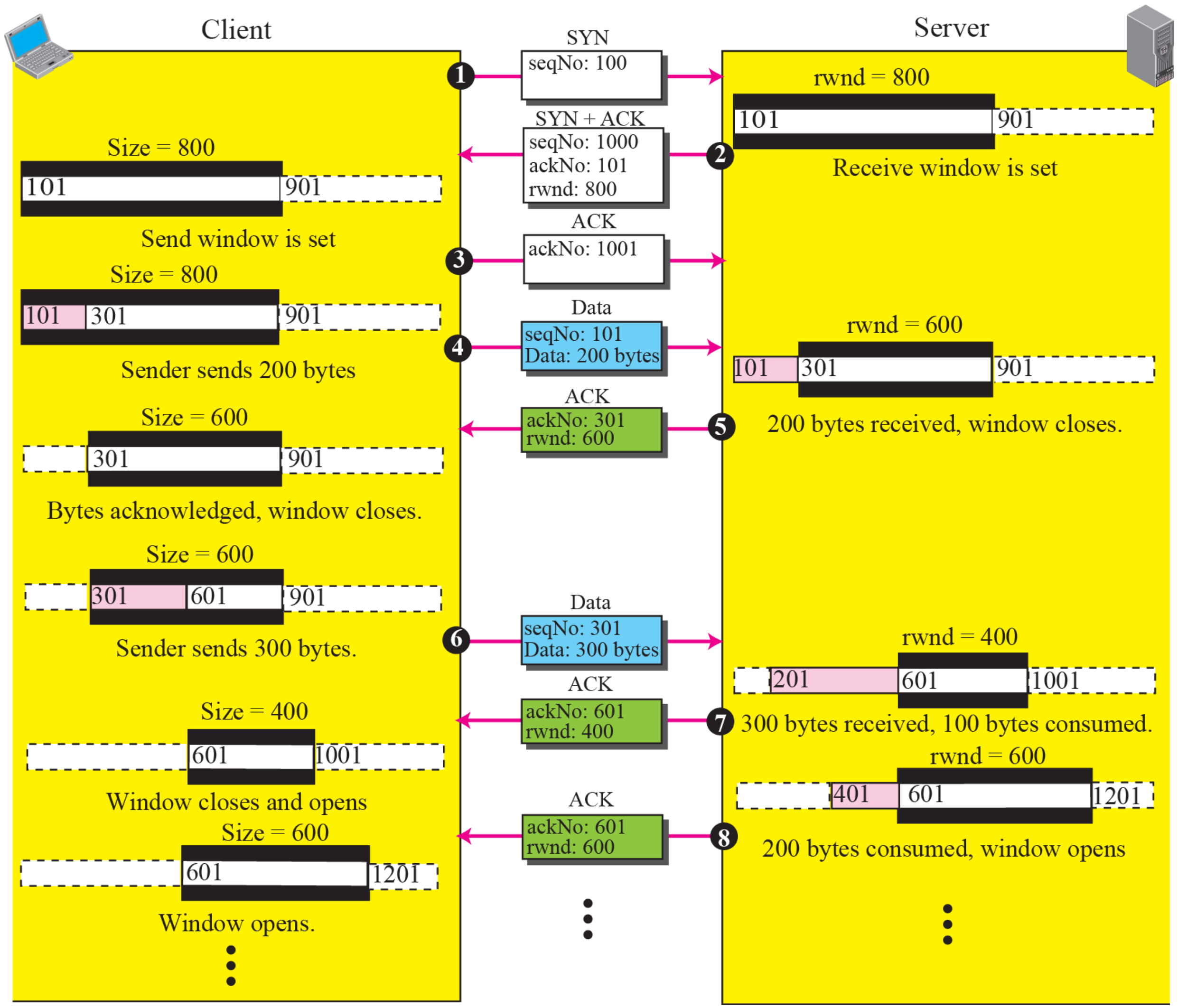


b. Opening and closing of receive window

FLOW CONTROL

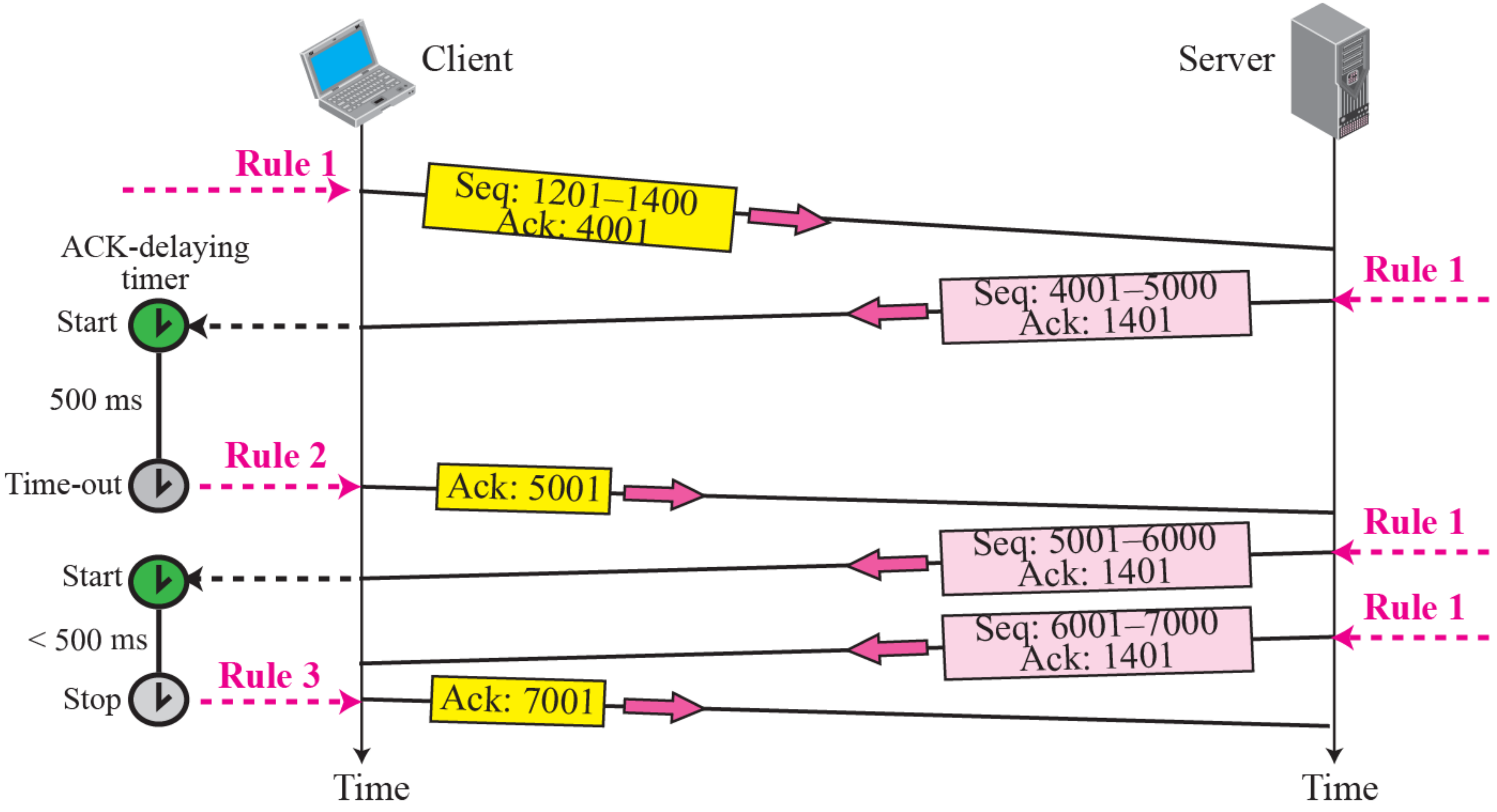
An example of flow control

Note: We assume only unidirectional communication from client to server. Therefore, only one window at each side is shown.

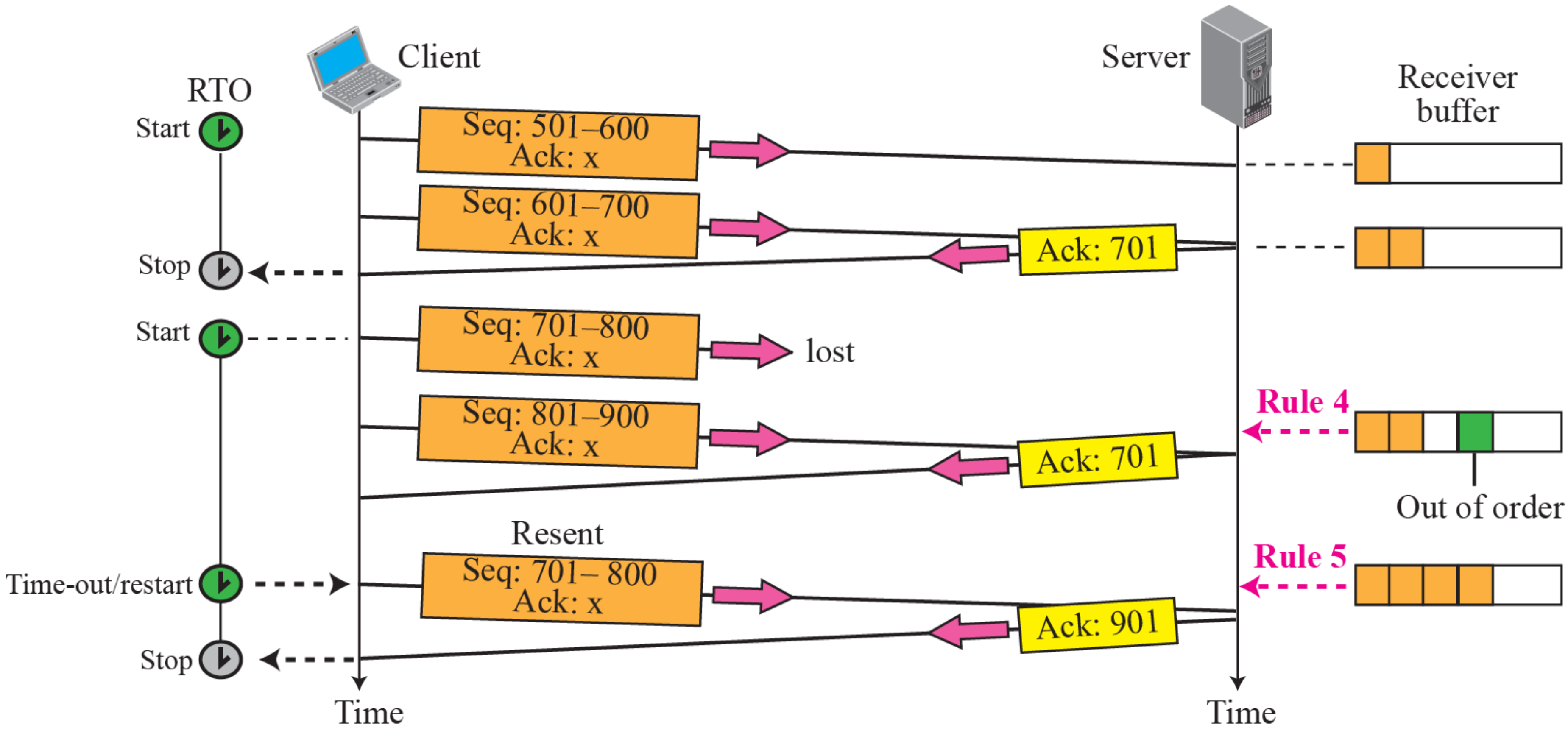


ERROR CONTROL

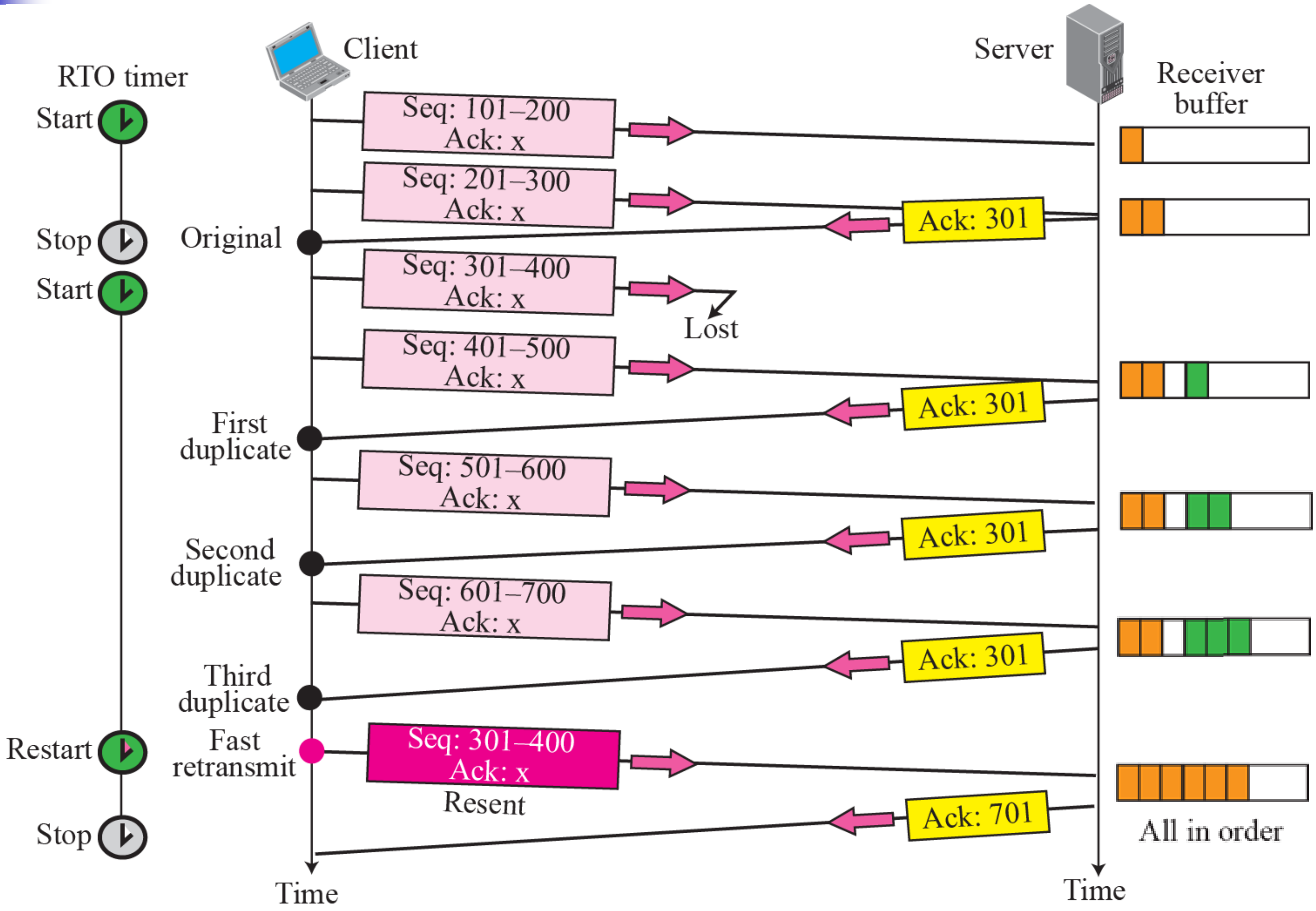
Normal operation



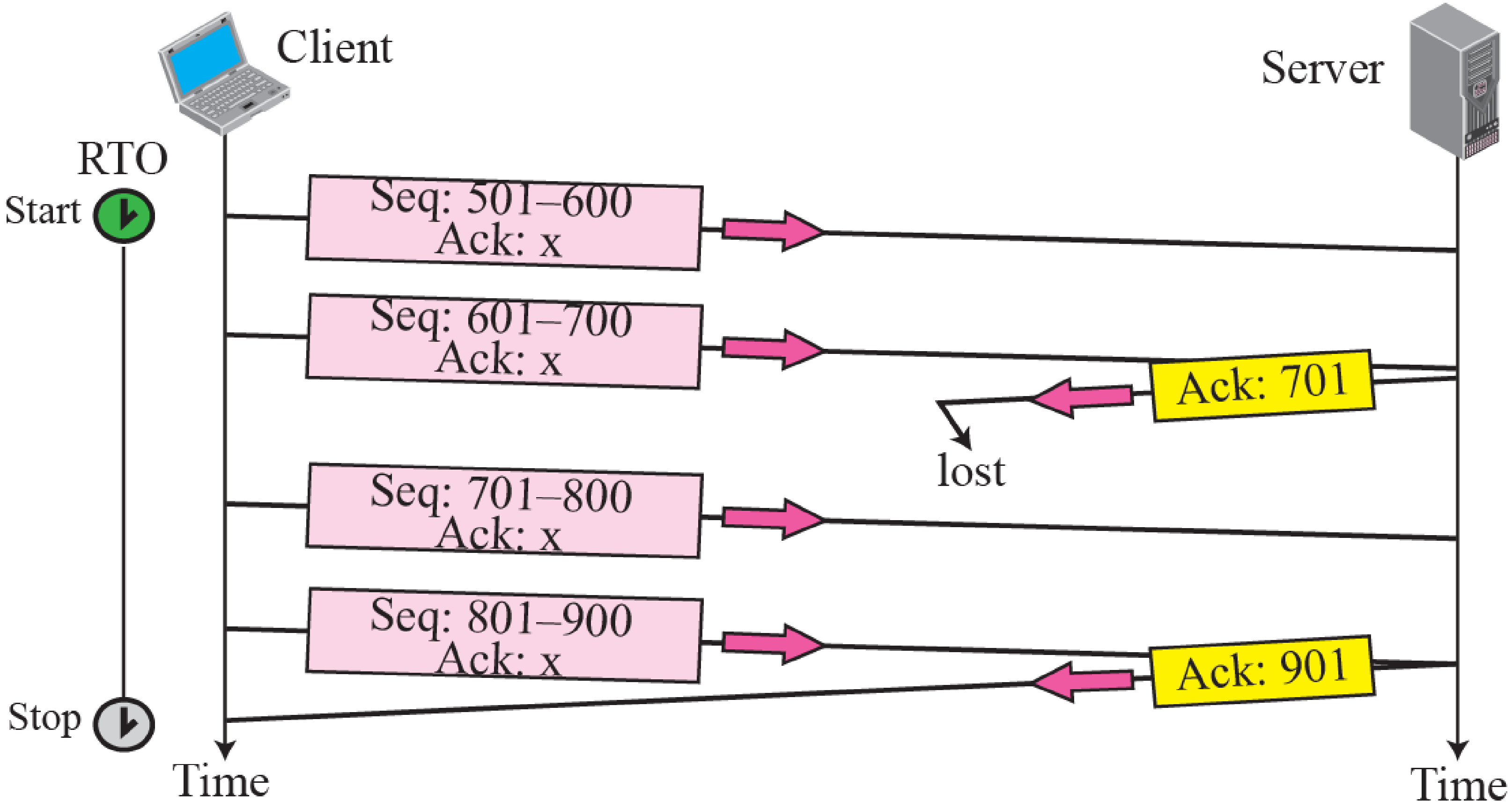
Lost segment



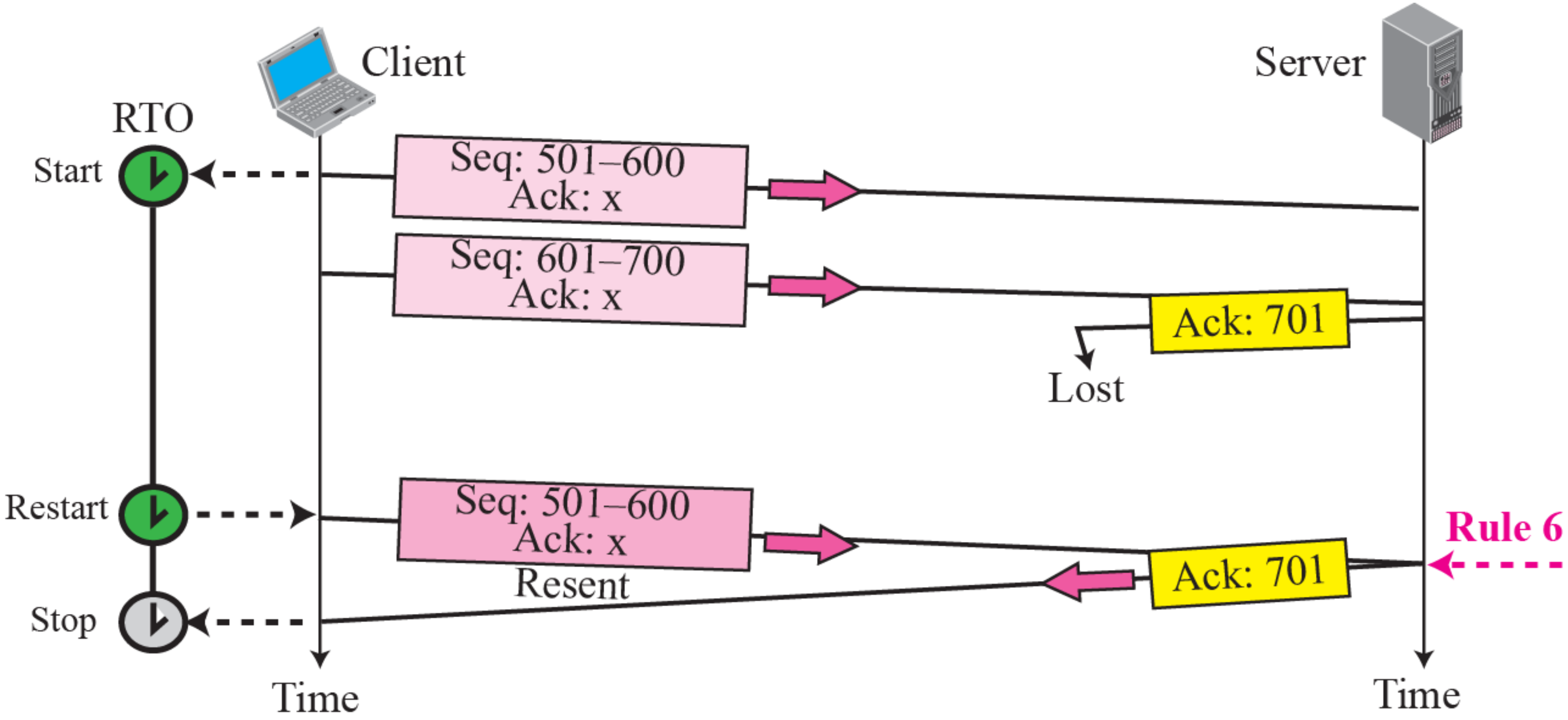
Fast retransmission



Lost acknowledgment

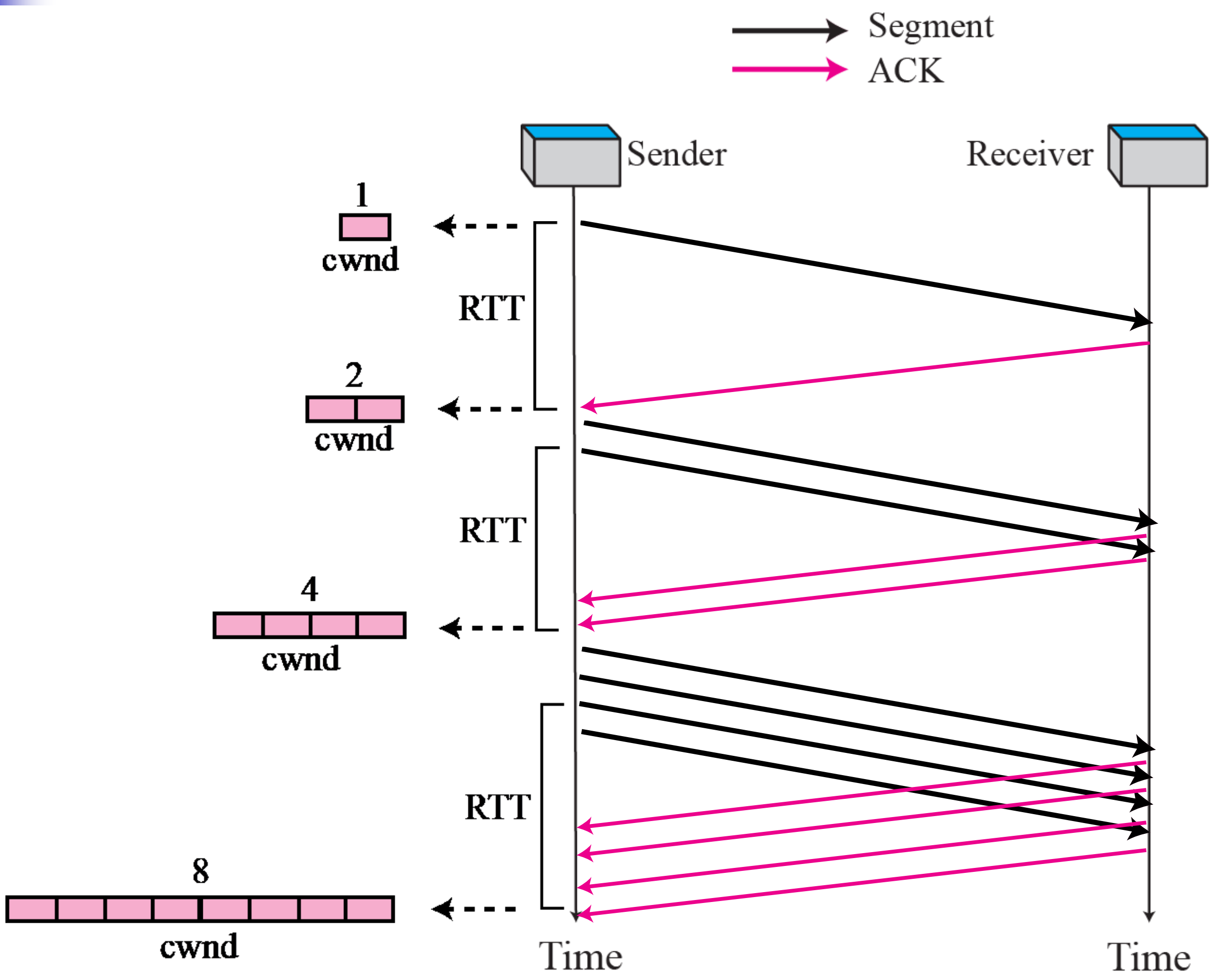


Lost acknowledgment corrected by resending a segment

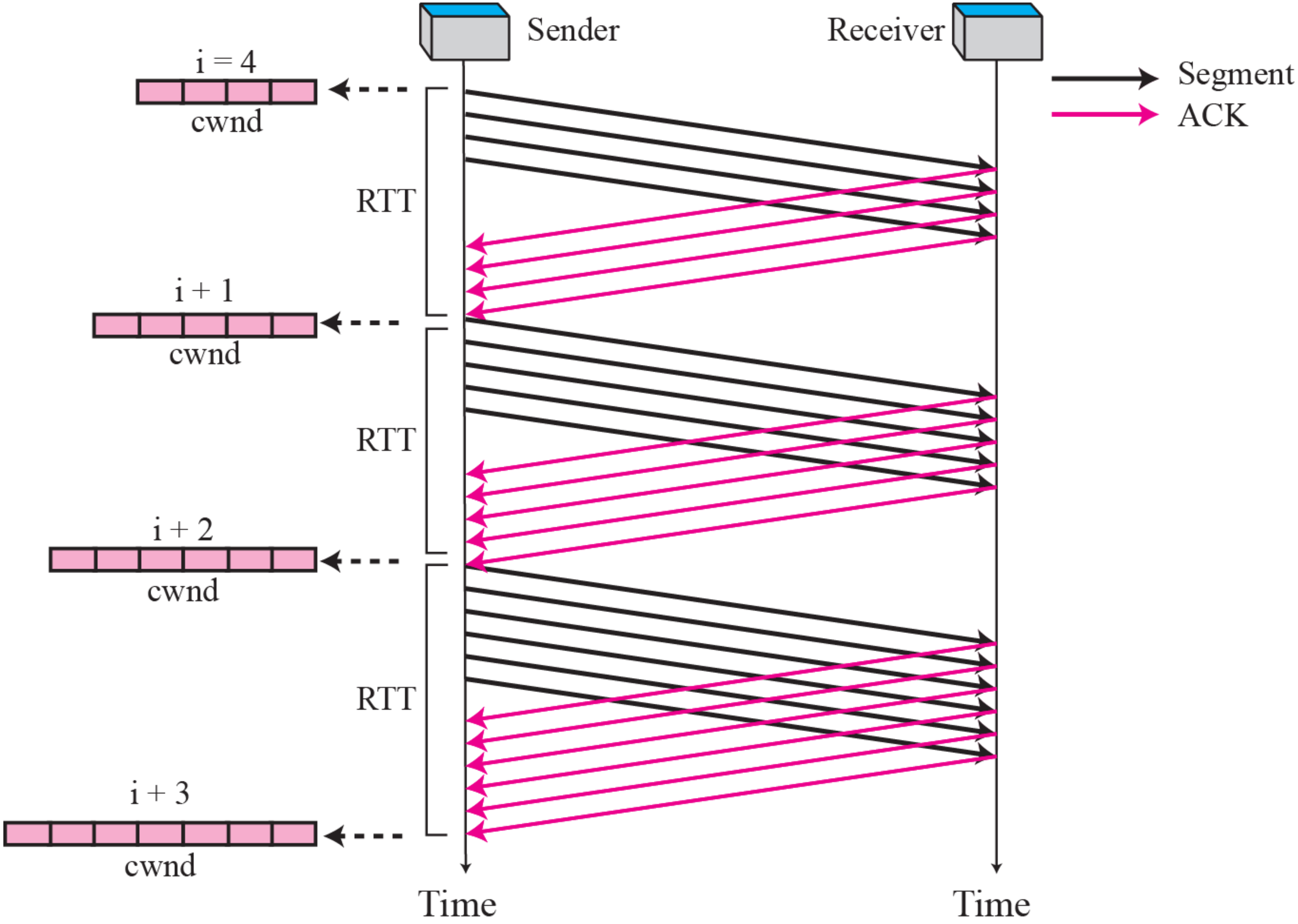


CONGESTION CONTROL

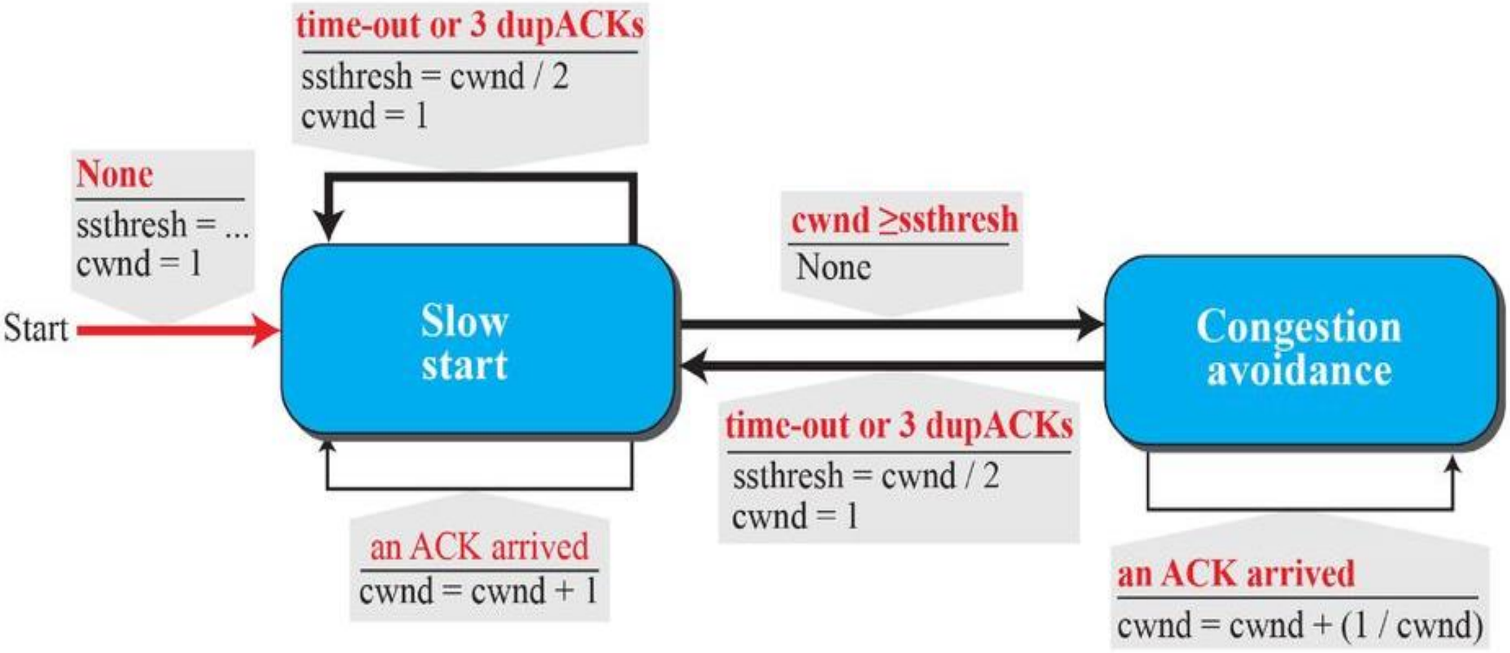
Slow start, exponential increase



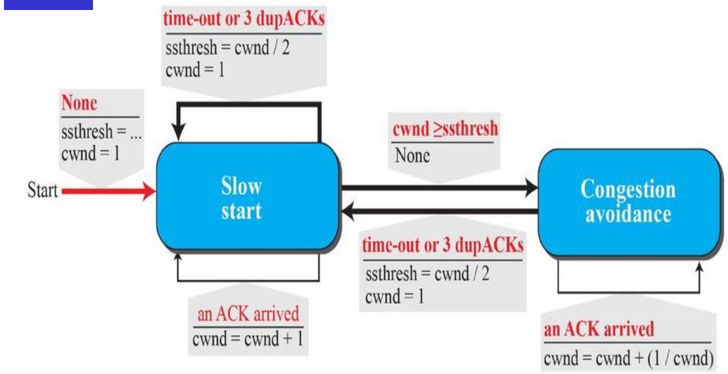
Congestion avoidance, additive increase

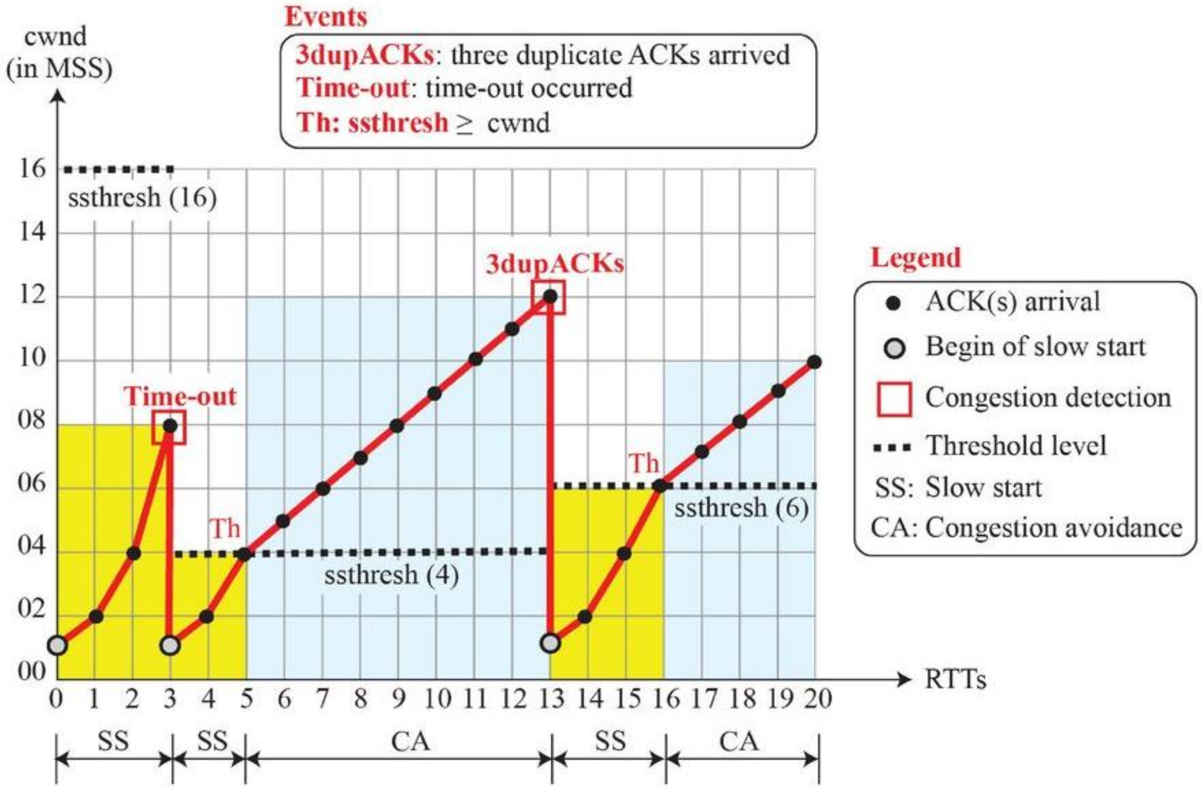


TCP Taho

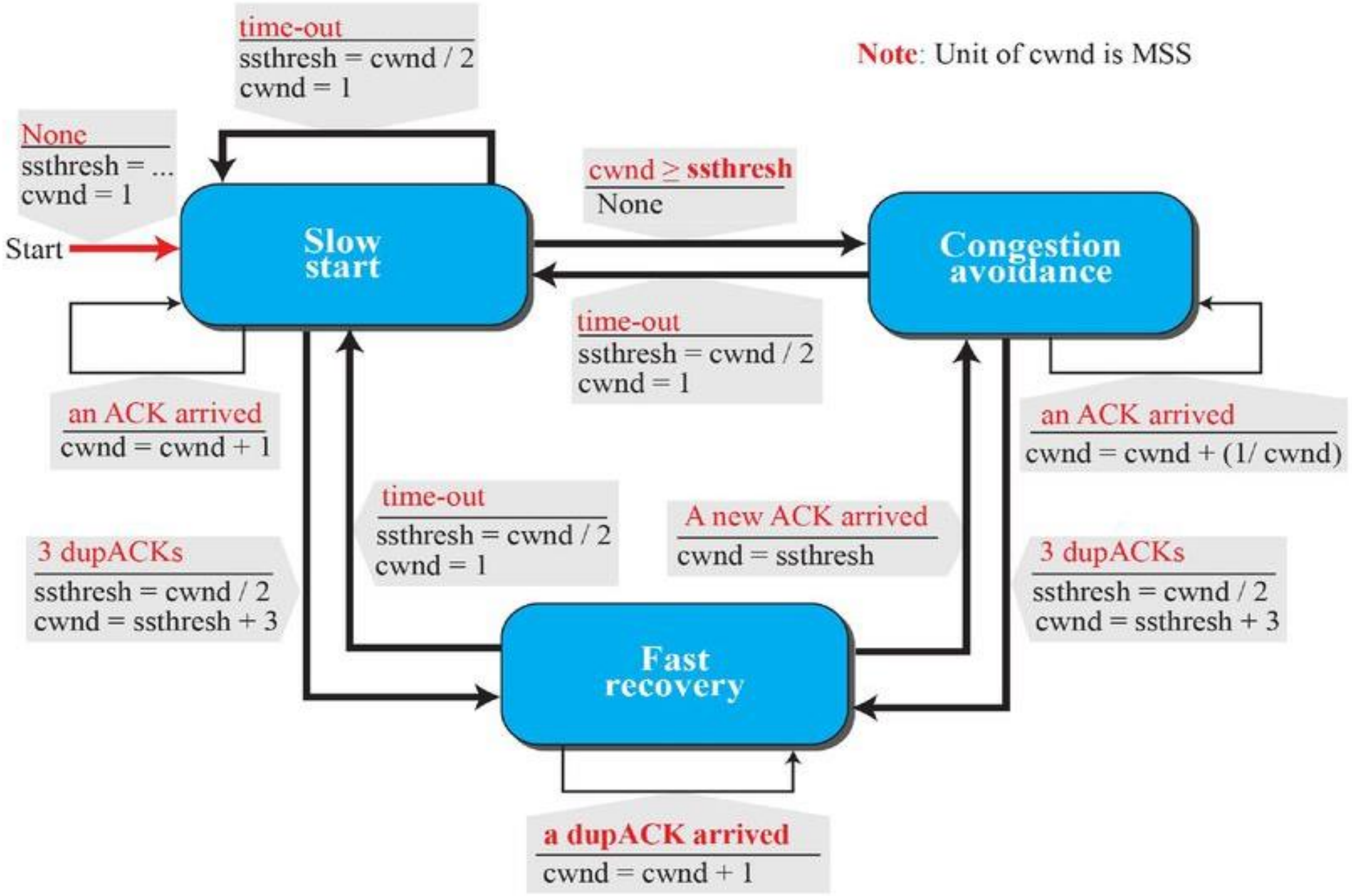

time-out or 3 dupACKs
ssthresh = cwnd / 2
cwnd = 1
None
ssthresh = ...
cwnd = 1
Start
Slow start
cwnd ≥ssthresh
None
Congestion avoidance
time-out or 3 dupACKs
ssthresh = cwnd / 2
cwnd = 1
an ACK arrived
cwnd = cwnd + 1
an ACK arrived
cwnd = cwnd + (1 / cwnd)

TCP Taho - Example

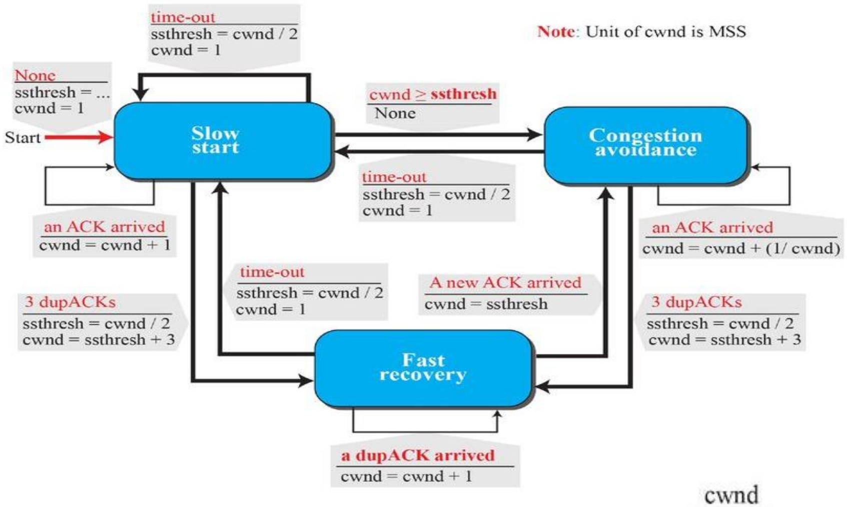


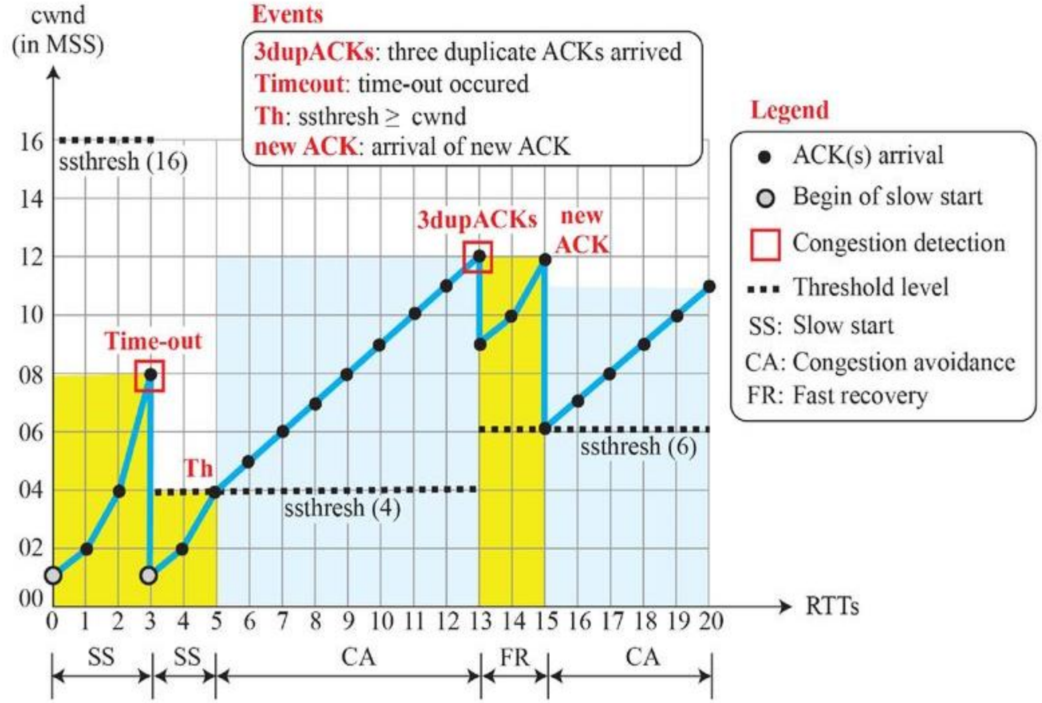


TCP Reno


Note: Unit of cwnd is MSS
time-out
ssthresh = cwnd / 2
cwnd = 1
None
ssthresh = ...
cwnd = 1
Start
Slow start
cwnd ≥ ssthresh
None
Congestion avoidance
time-out
ssthresh = cwnd / 2
cwnd = 1
an ACK arrived
cwnd = cwnd + 1
an ACK arrived
cwnd = cwnd + (1/ cwnd)
time-out
ssthresh = cwnd / 2
cwnd = 1
A new ACK arrived
cwnd = ssthresh
3 dupACKs
ssthresh = cwnd / 2
cwnd = ssthresh + 3
3 dupACKs
ssthresh = cwnd / 2
cwnd = ssthresh + 3
Fast recovery
a dupACK arrived
cwnd = cwnd + 1

TCP Reno - Example


Note: Unit of cwnd is MSS
time-out
ssthresh = cwnd / 2
cwnd = 1
None
ssthresh = ...
cwnd = 1
Start
Slow start
cwnd ≥ ssthresh
None
Congestion avoidance
time-out
ssthresh = cwnd / 2
cwnd = 1
an ACK arrived
cwnd = cwnd + 1
an ACK arrived
cwnd = cwnd + (1/ cwnd)
time-out
ssthresh = cwnd / 2
cwnd = 1
A new ACK arrived
cwnd = ssthresh
3 dupACKs
ssthresh = cwnd / 2
cwnd = ssthresh + 3
3 dupACKs
ssthresh = cwnd / 2
cwnd = ssthresh + 3
Fast recovery
a dupACK arrived
cwnd = cwnd + 1


cwnd
(in MSS)
Events
3dupACKs: three duplicate ACKs arrived
Timeout: time-out occured
Th: ssthresh ≥ cwnd
new ACK: arrival of new ACK
Legend
● ACK(s) arrival
○ Begin of slow start
□ Congestion detection
▪▪▪ Threshold level
SS: Slow start
CA: Congestion avoidance
FR: Fast recovery
ssthresh (16)
Time-out
Th
ssthresh (4)
3dupACKs
new ACK
ssthresh (6)
RTTs
SS
SS
CA
FR
CA