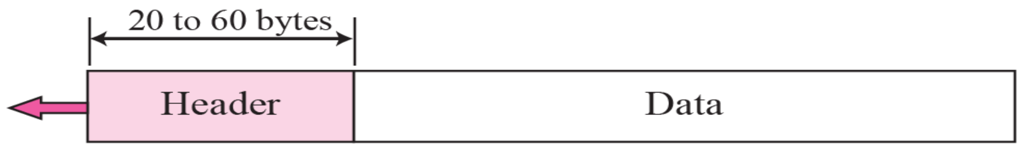
Transport Layer

Anand Baswade

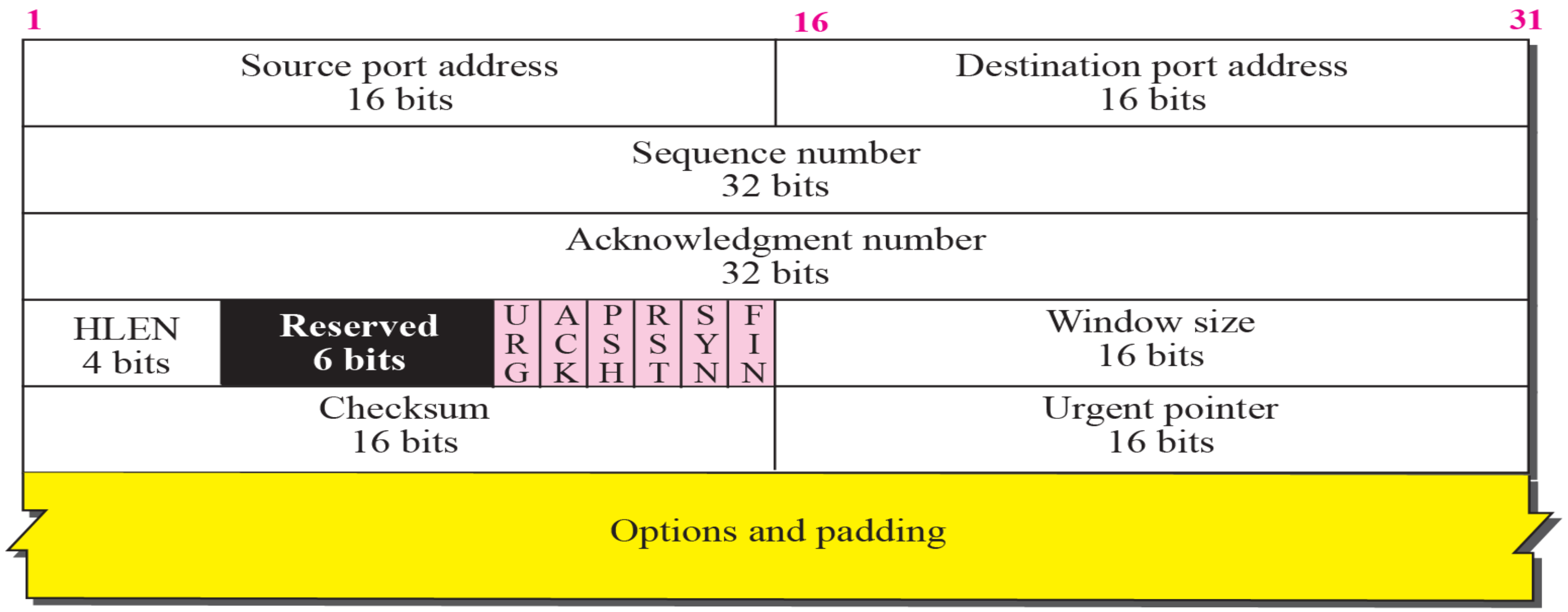
anand@iitbhilai.ac.in

TCP segment format

- Before discussing TCP in more detail, let us discuss the TCP packets themselves. A packet in TCP is called a segment.



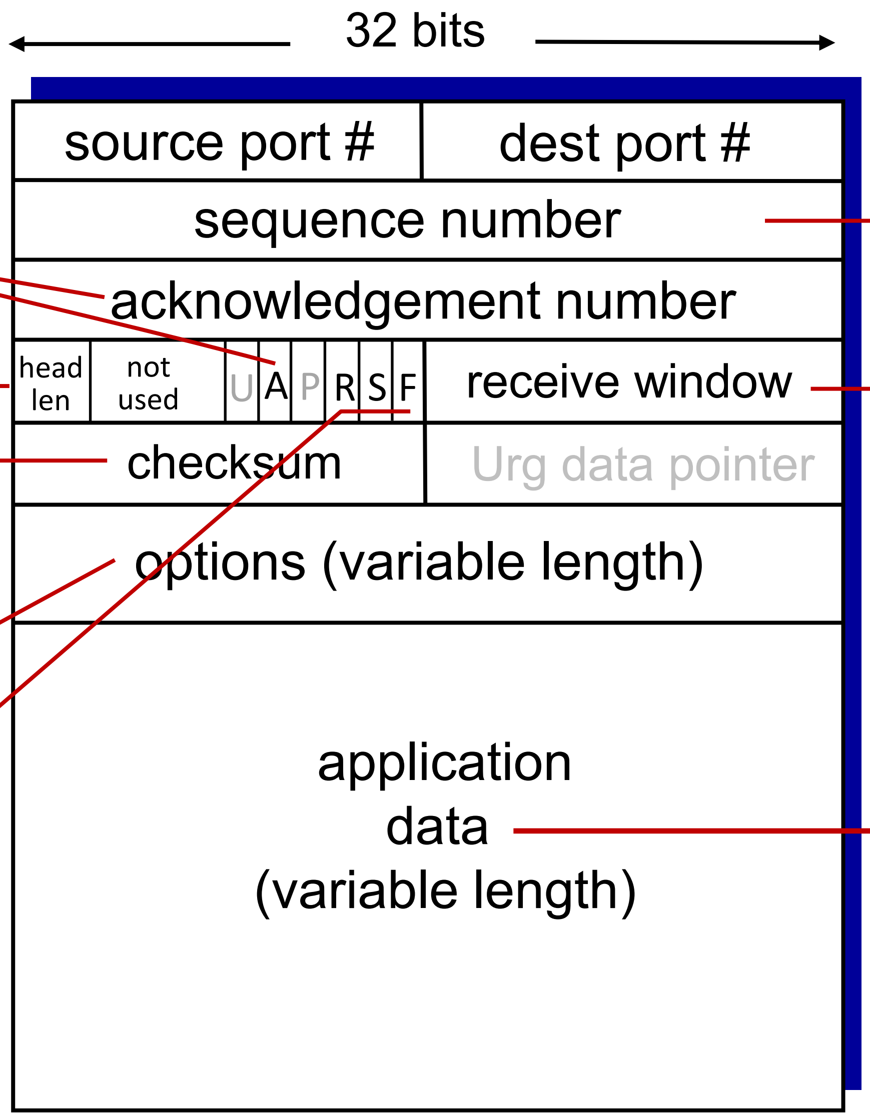
a. Segment



b. Header

TCP/IP Protocol Suite

TCP segment structure



ACK: seq # of next expected byte; A bit: this is an ACK

length (of TCP header)

Internet checksum

TCP options

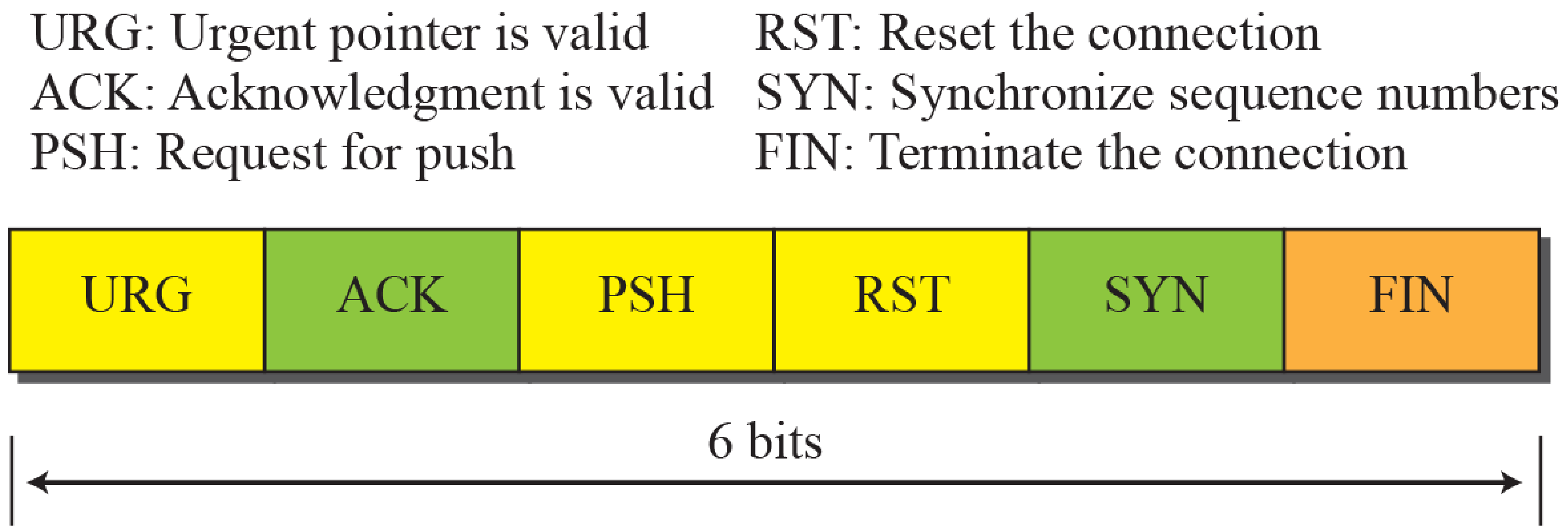
RST, SYN, FIN: connection management

segment seq #: counting bytes of data into bytestream (not segments!)

flow control: # bytes receiver willing to accept

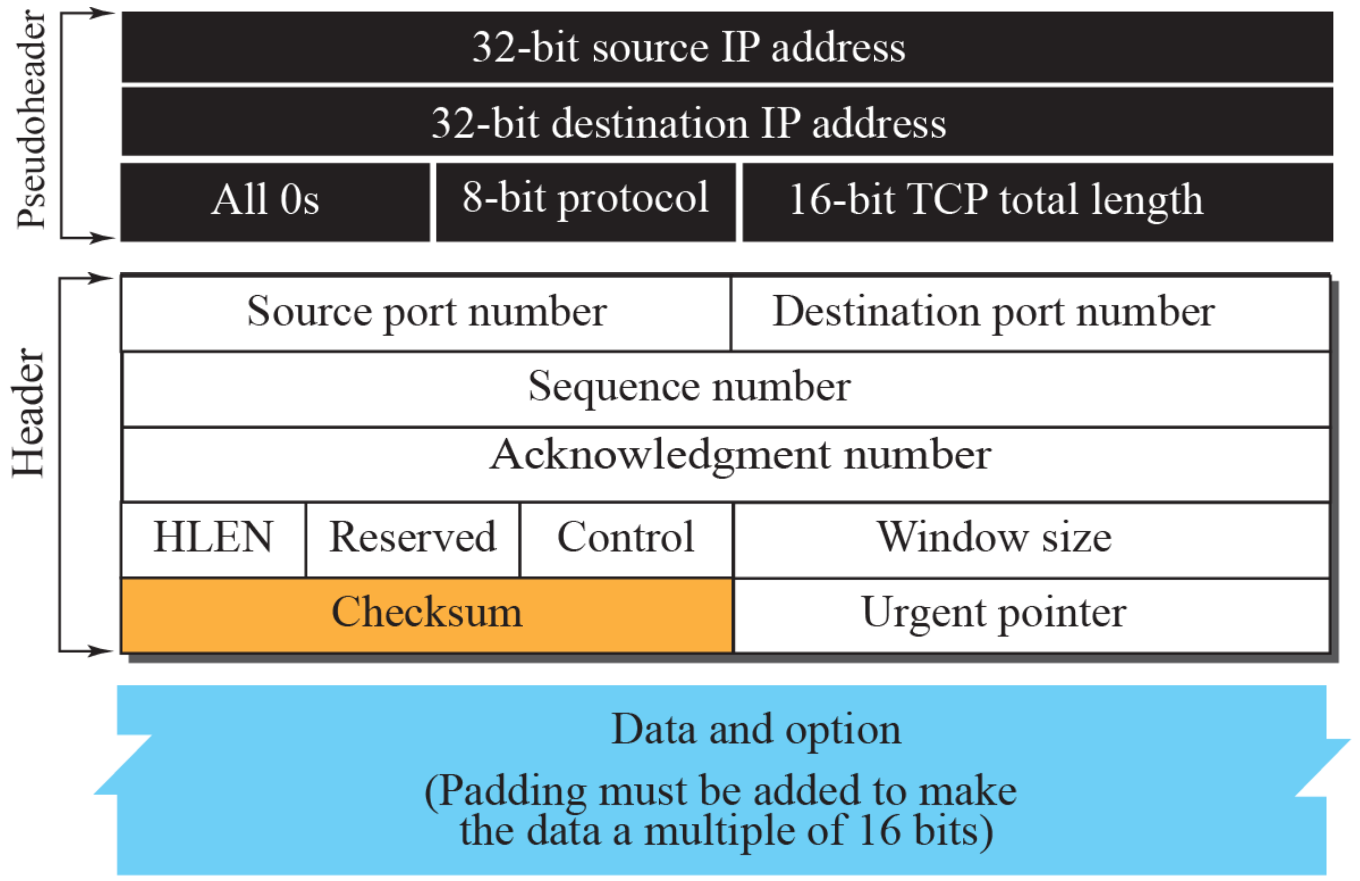
data sent by application into TCP socket

TCP Flag Bits


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
6 bits

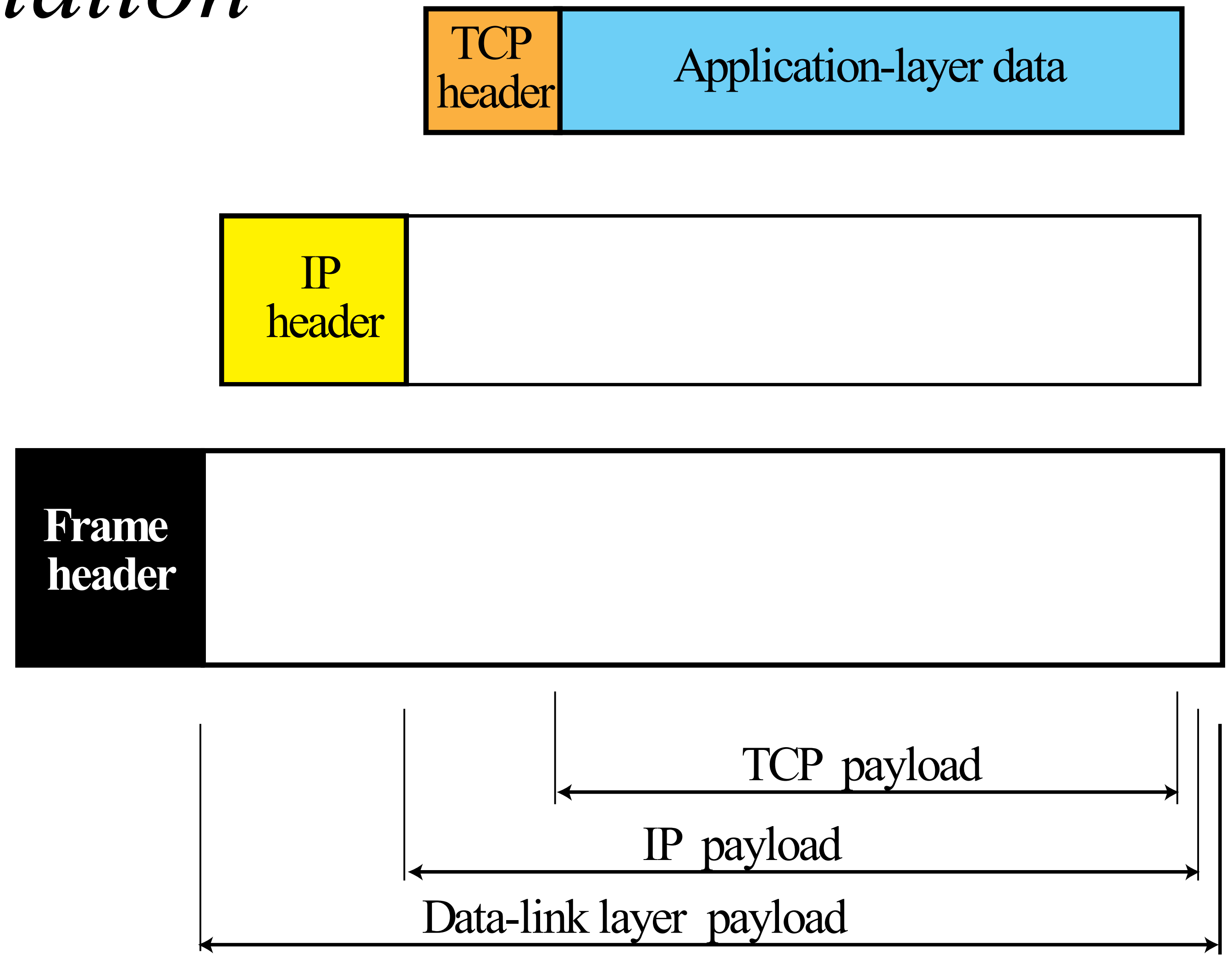
In practice URG and the urgent pointer are not used.

Pseudoheader added to the TCP segment



The use of the checksum in TCP is mandatory.

Encapsulation

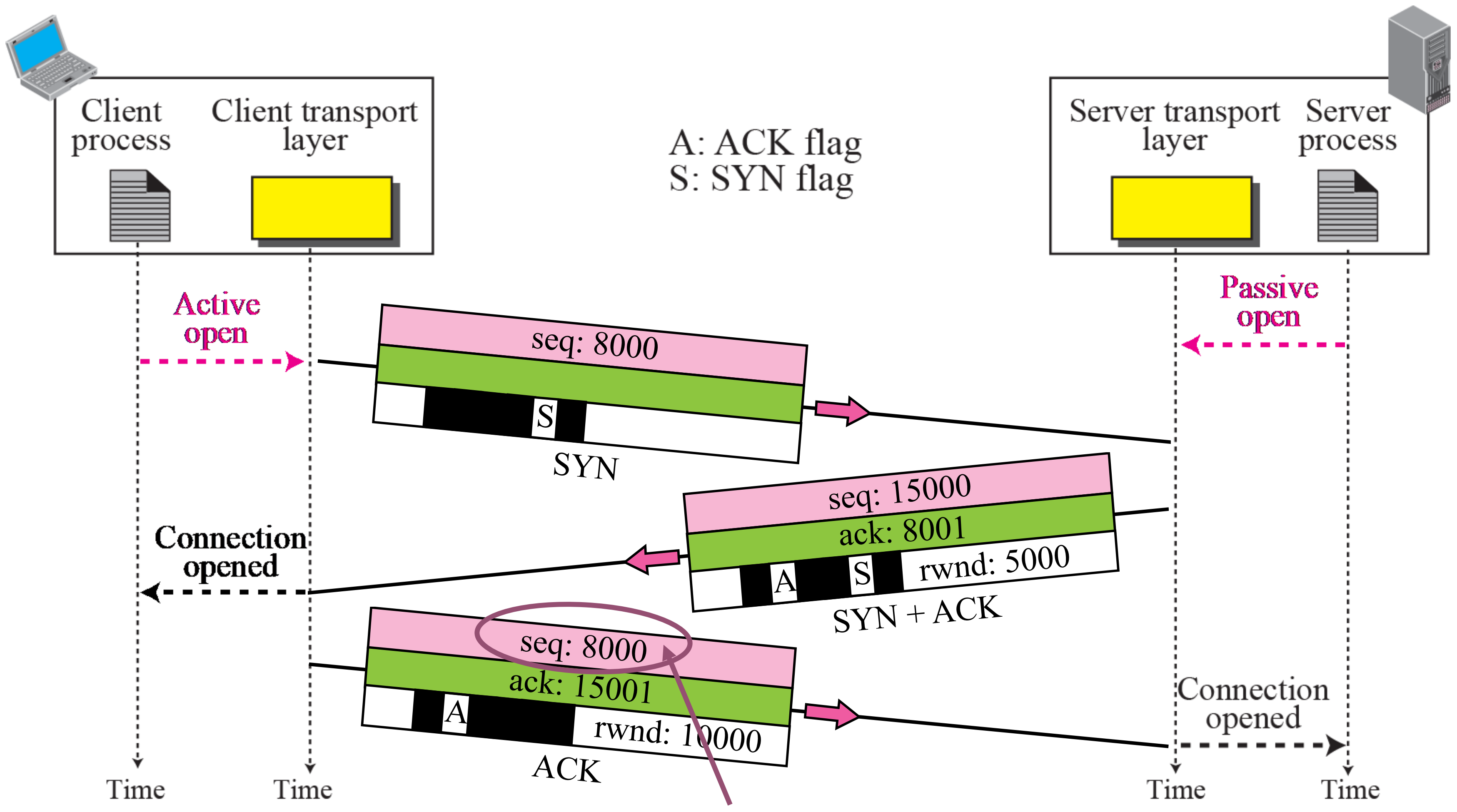


TCP/IP Protocol Suite

TCP Connection

- TCP is connection-oriented. It establishes a virtual path between the source and destination. All of the segments belonging to a message are then sent over this virtual path.
- You may wonder how TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented. The point is that a TCP connection is virtual, not physical.
- TCP operates at a higher level. TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself. If a segment is lost or corrupted, it is retransmitted.

Connection establishment using three-way handshake

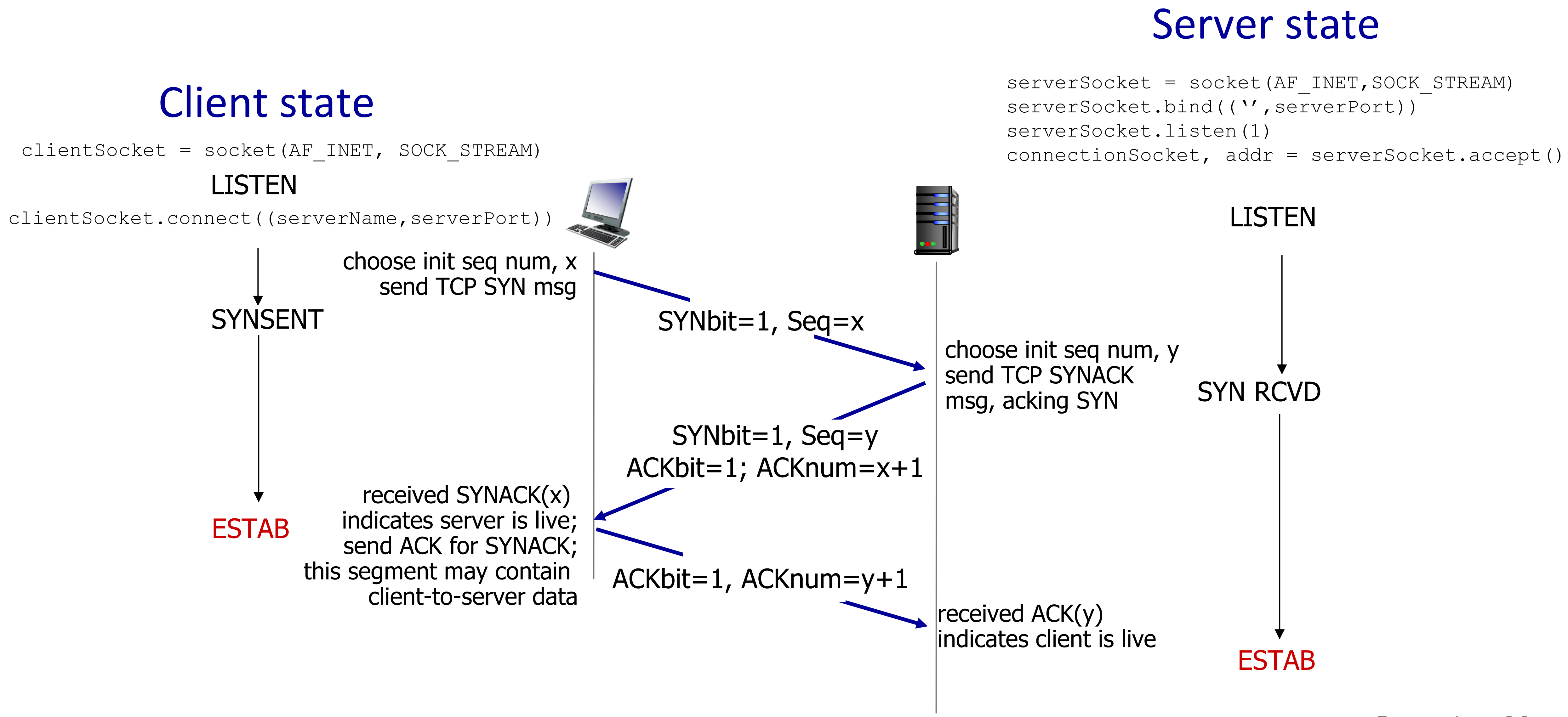


Means "no data" !

seq: 8001 if piggybacking

TCP/IP Protocol Suite

TCP 3-way handshake



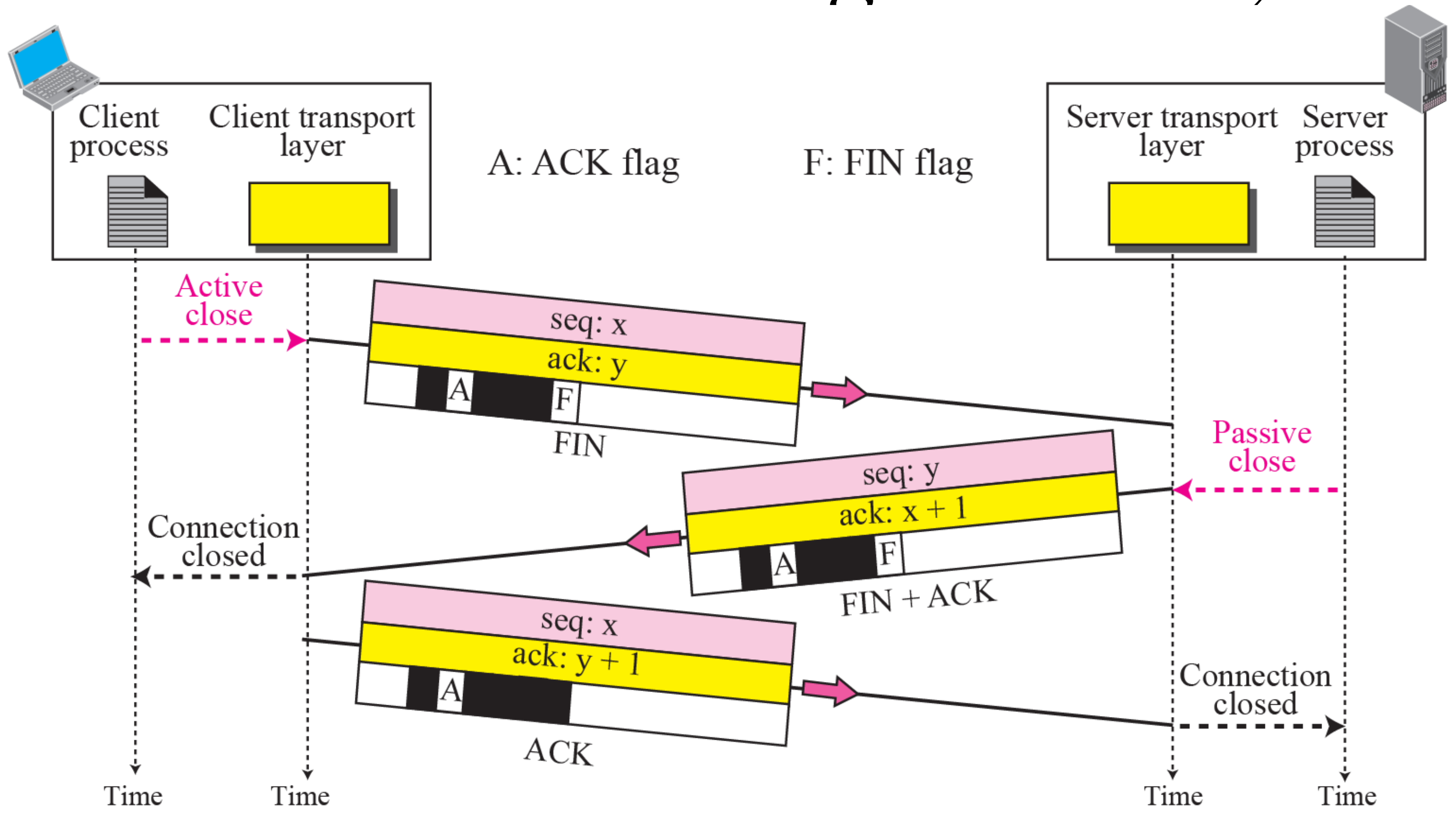
```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
```

```
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
connectionSocket, addr = serverSocket.accept()
```

Cont..

- 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 termination using three-way handshake

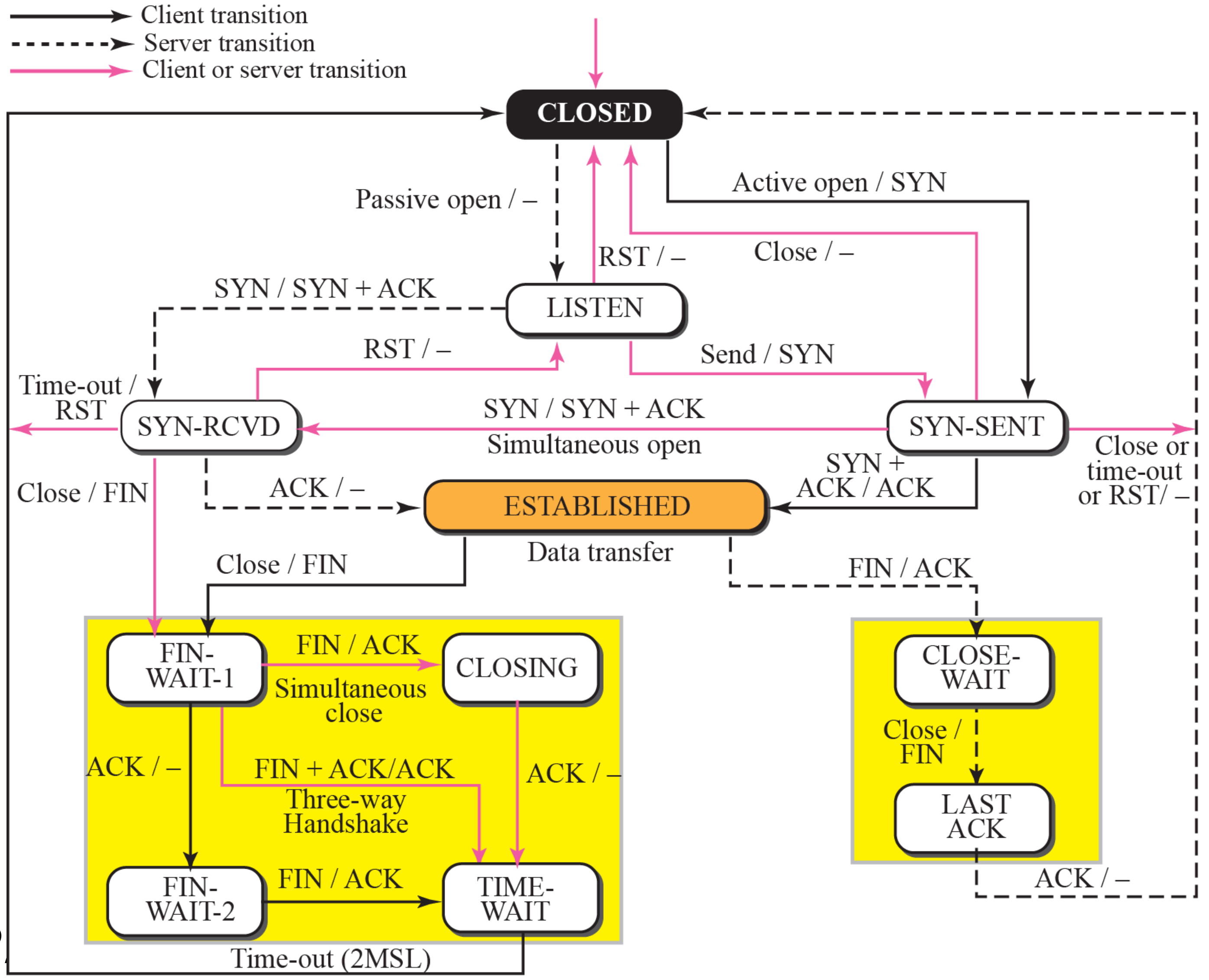


- The FIN segment consumes one sequence number if it does not carry data.
- The FIN + ACK segment consumes one sequence number if it does not carry data.

Closing a TCP 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

State transition diagram



TCP

States of TCP

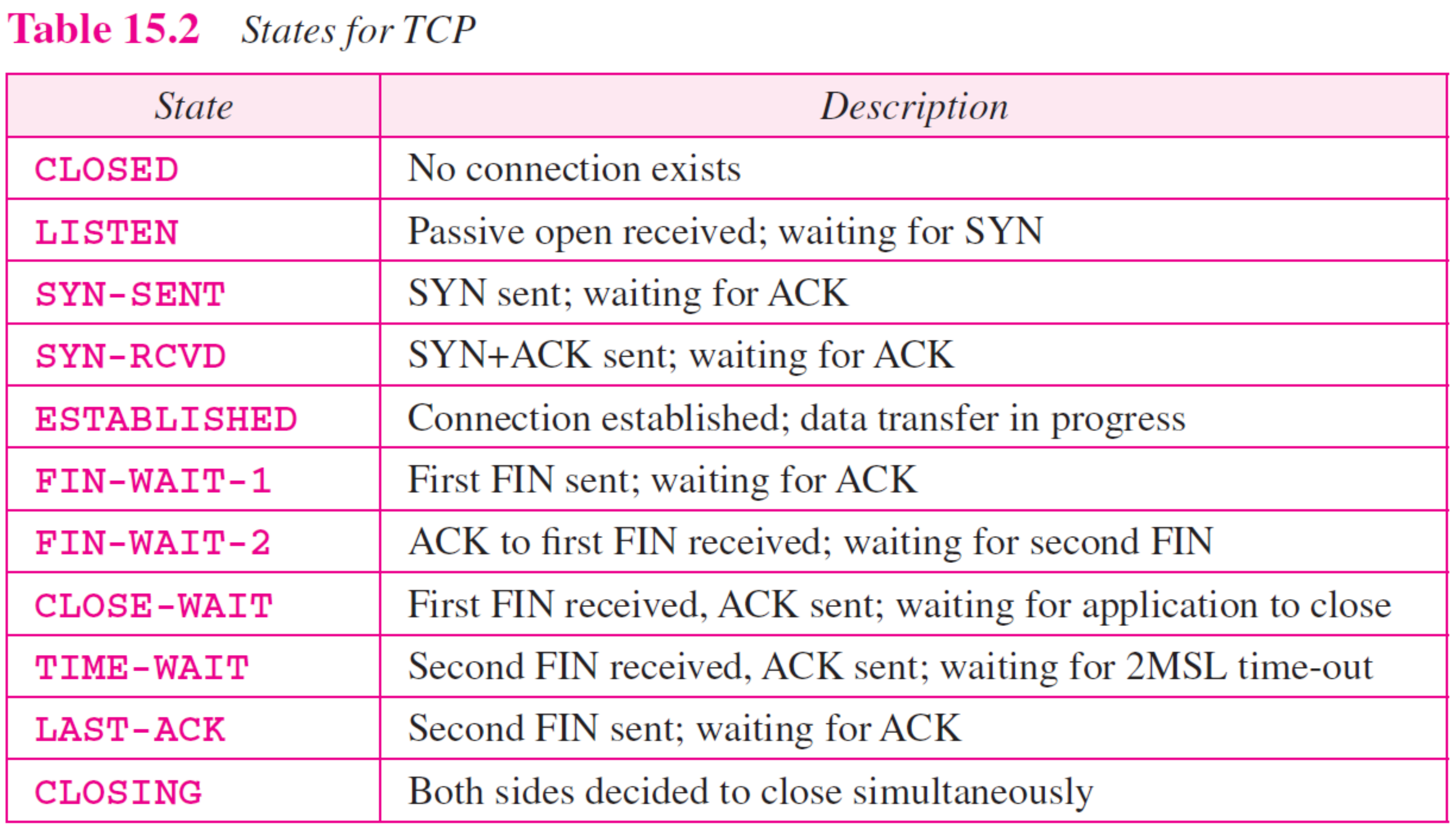
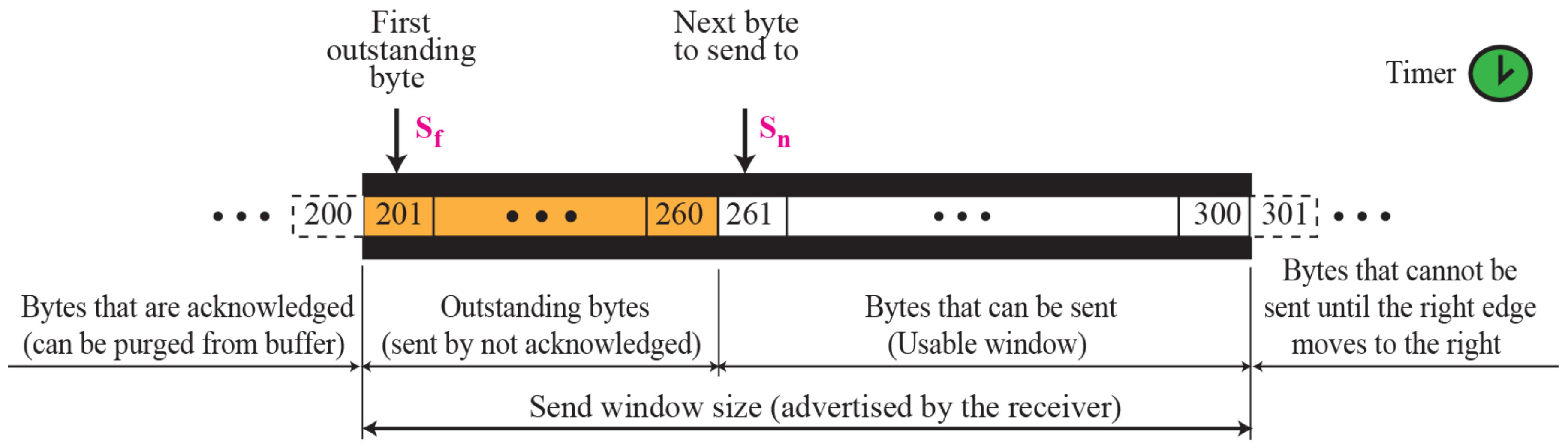


Table 15.2 *States for TCP*

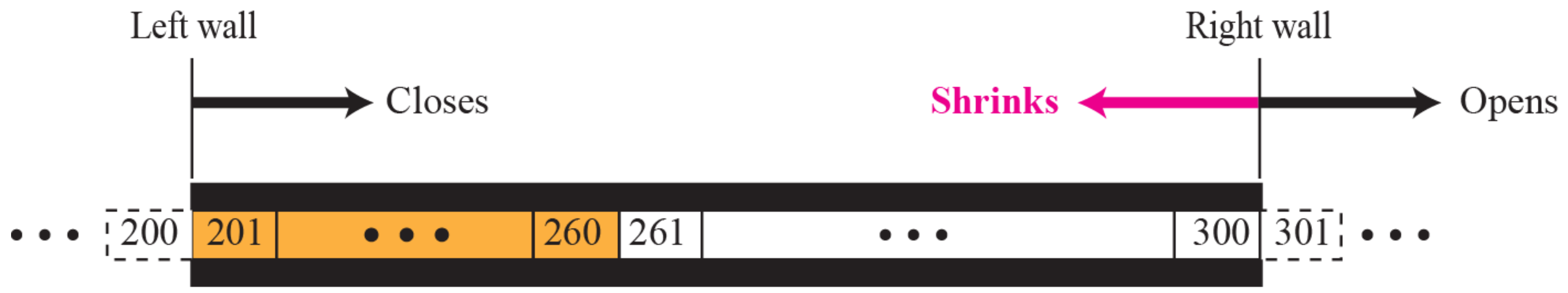
State	*Description*
`CLOSED`	No connection exists
`LISTEN`	Passive open received; waiting for SYN
`SYN-SENT`	SYN sent; waiting for ACK
`SYN-RCVD`	SYN+ACK sent; waiting for ACK
`ESTABLISHED`	Connection established; data transfer in progress
`FIN-WAIT-1`	First FIN sent; waiting for ACK
`FIN-WAIT-2`	ACK to first FIN received; waiting for second FIN
`CLOSE-WAIT`	First FIN received, ACK sent; waiting for application to close
`TIME-WAIT`	Second FIN received, ACK sent; waiting for 2MSL time-out
`LAST-ACK`	Second FIN sent; waiting for ACK
`CLOSING`	Both sides decided to close simultaneously

TCP/IP Protocol Suite

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)

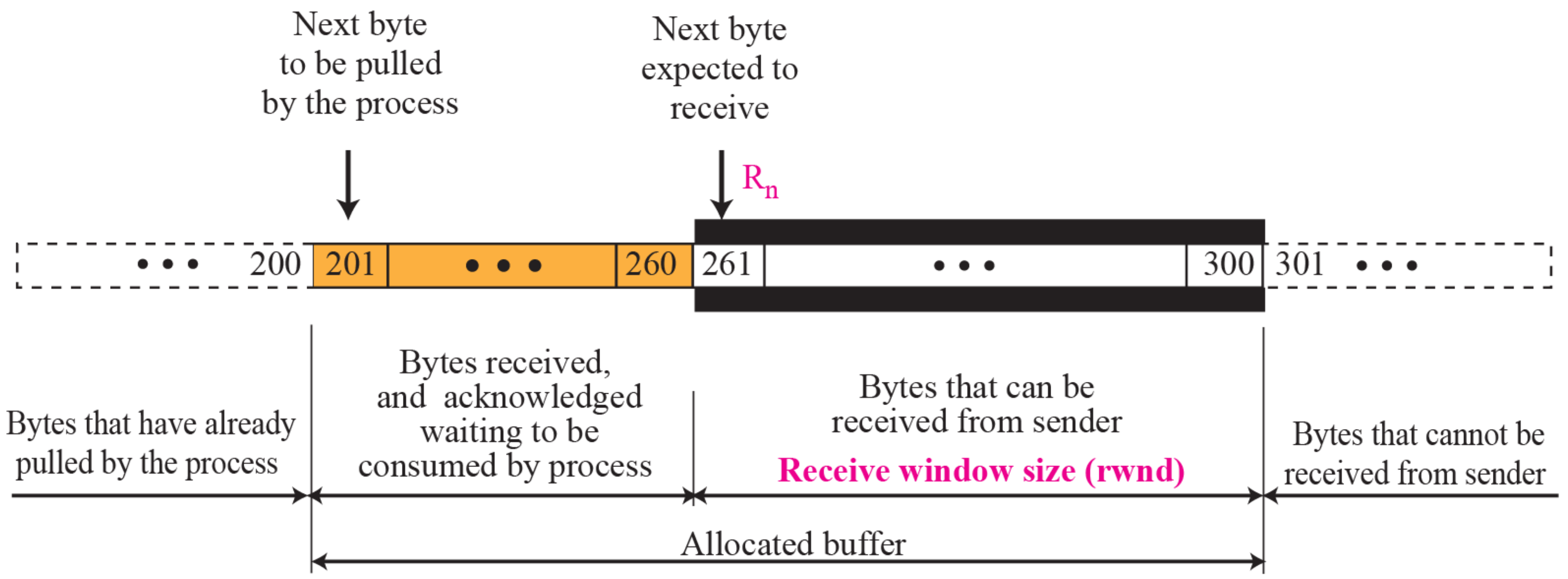
a. Send window



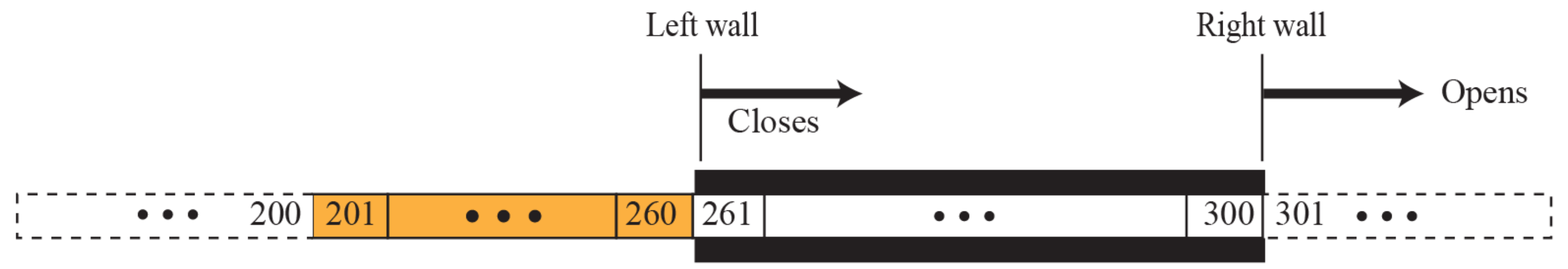
b. Opening, closing, and shrinking send window

TCP/IP Protocol Suite

Receive window in TCP


Next byte to be pulled by the process
Next byte expected to receive
R_n
• • • 200
201
• • •
260
261
• • •
300
301 • • •
Bytes that have already pulled by the process
Bytes received, and acknowledged waiting to be consumed by process
Bytes that can be received from sender
Receive window size (rwnd)
Bytes that cannot be received from sender
Allocated buffer

a. Receive window and allocated buffer



b. Opening and closing of receive window

TCP/IP Protocol Suite