TCP

CSC 512 – Networks: Architectures and Protocols

Instructor: Dr. Frye

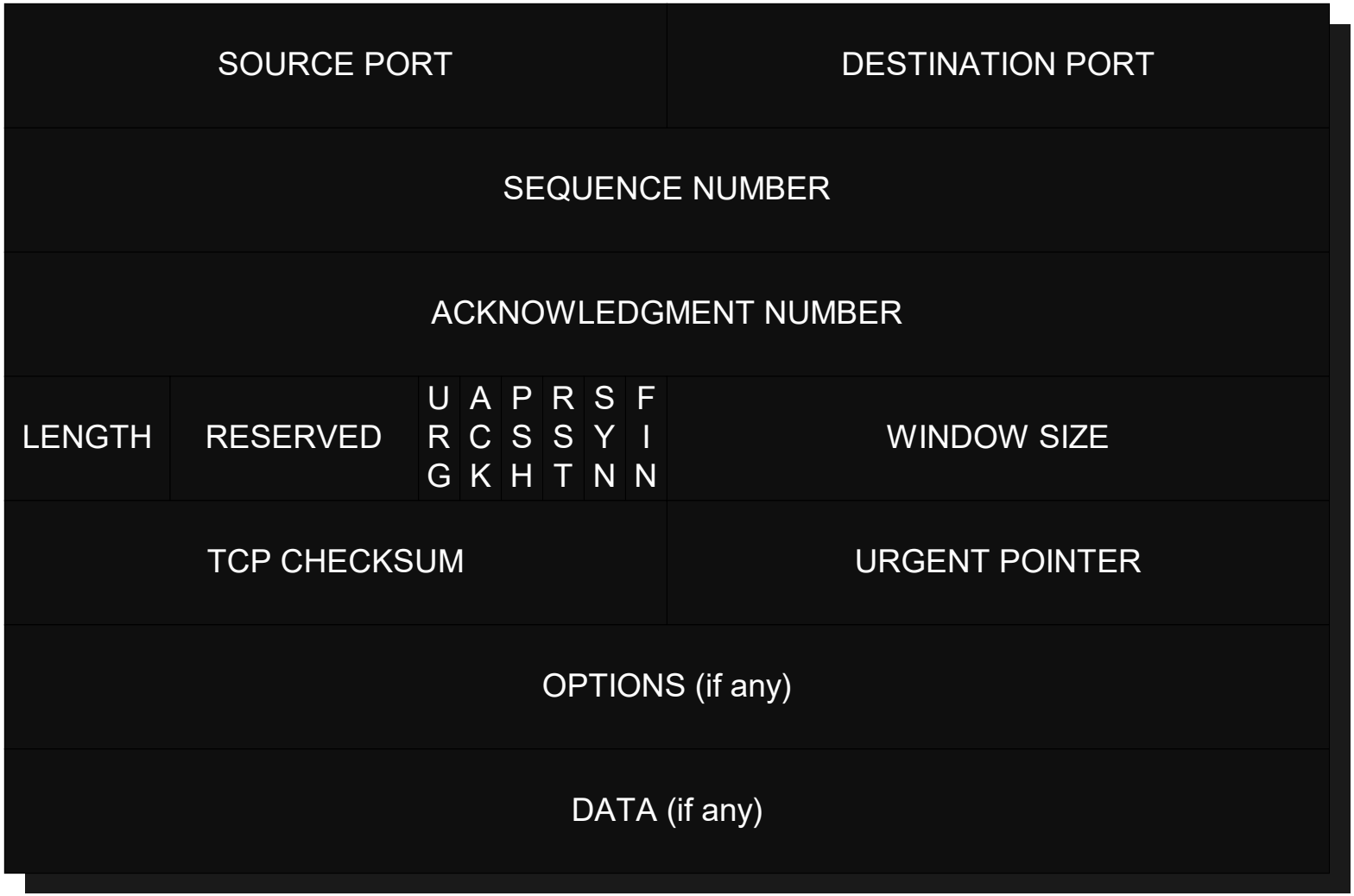
frye@kutztown.edu

Computer Science & Information Technology Department, Kutztown University

TRANSMISSION CONTROL PROTOCOL OVERVIEW

- Provides a connection-oriented reliable packet delivery by
  - Sending acks
  - Maintaining a retransmission timer
  - Checksum on header and data
  - Segment and resequence data
  - Checks for and discards duplicates
  - Provides flow control

TCP Segment Structure



(TCP) Segment Structure

TCP FIELD	DESCRIPTION
Source Port Number	Identifies the sending application
Destination Port Number	Identifies the receiving application
Sequence Number	Identifies the byte in the stream of data
Acknowledgement Number	Identifies the next sequence number that the sender expects the to receive.
Length	4-bit Header Length
URG	Urgent Pointer
ACK	Acknowledgment Number is valid
PSH	Receiver should pass this data to the application as soon as possible
RST	Reset the connection
SYN	Synchronize sequence numbers to initiate a connection
FIN	The sender is finished sending data
Window Size	The number of outstanding segments allowed at any one time without being acknowledged
Checksum	Covers the header and data
Urgent Pointer	Positive offset that must be added to the sequence number to yield the number of the last byte of data
Options	usually Maximum Segment Size (MSS)

Some Header OPTIONS

- Maximum Segment Size
  - Default = 536
  - Optimum value
- TCP Checksum

Sequence Numbers

- Sequence Numbers
  - First byte numbered 0
  - File size 500,000 bytes
  - MSS 1,000 (500 segments)
  - Sequence #1=0, Sequence #2=1000, Sequence #3=2000, etc.

Acknowledgement Numbers

- Sequence number of next segment expected
  - Received bytes 0 through 535
  - Waiting for byte 536
  - Puts 536 in acknowledgement number field of segment

TCP CONNECTION ESTABLISHMENT

- Three-way Handshake
- Requesting end sends a SYN segment
  - port number of server
  - initial sequence number
- Server responds with its own SYN
  - contains server's ISN
  - ACKs the client's SYN with client's ISN + 1
- Client acknowledges with SYN ACK

TCP Connection

- Connection now established
  - Send and receive buffers
- Sender
- Receiver

Reliable-Data Transfer Service

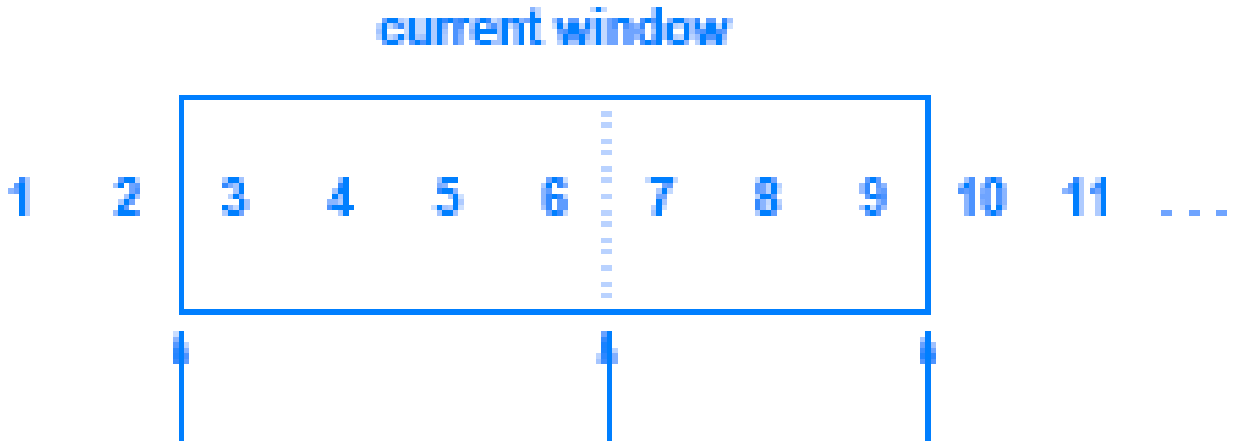
- How does TCP provide a reliable-data transfer service?

TCP CONNECTION TERMINATION

- Modified three-way handshake
  - Passive close
    - The client sends a FIN to the server application
    - The server ACKs the FIN of the received sequence number + 1
  - Active close
    - The server sends a FIN to the client
    - The client ACKs the FIN of the server

Sliding Window

- What is pipelining?
- Octet vs. Byte



Sliding Window cont.

- http://www2.rad.com/networks/2004/sliding_window/

- Default window size = 8760 bytes
- Window advertisement

Timeout and Retransmission

- Varying transmission time
- Adaptive retransmission algorithm
- Sample round trip time

Round-Trip Time

- SampleRTT – time from segment sent until ACK received
- Why will the SampleRTT change from segment to segment on same TCP connection?

Average RTT

- averageRTT = (α * Old_ averageRTT) + ((1 – α) * SampleRTT)
- Value for α
- Why do you think the averageRTT is weighted the way it is?

Timeout value

- What is a good general (larger than, smaller than, etc) value for the timeout value?
- Timeout = β * averageRTT

Calculation of RTT values

- Why does the way TCP really works make the calculation of a sample round trip time non-trivial?

- Acknowledgement ambiguity
- Solution?

Karn's Algorithm

- Only deal with unambiguous ACKs
- Timer backoff strategy
  - Timeout event – increase timeout value
  - new_timeout = γ * timeout
- Variation in delay
  - Average RTT and variance

TCP Flow Control

- Receive Window
- Sliding Window

Receiver Variables

- RcvBuffer
- LastByteRead
- LastByteRcvd
- RcvWindow
- LastByteRcvd – LastByteRead <= RcvBuffer
- RcvWindow = RcvBuffer – (LastByteRcvd – LastByteRead)

Sender Variables

- LastByteSent
- LastByteAcked

- LastByteSent – LastByteAcked = amount of unacknowledged data sender sent to the receiver.

Silly Window Syndrome

- What if RcvWindow = 0?
  - Receiver reads one byte from buffer
    - Receiver sends ACK
    - Sender sends 1-byte segments
  - What is wrong with this?
  - No data from receiver to sender
    - Sender continues to send 1-byte segments

Silly Window Syndrome Avoidance

- Sender – shortly
- Receiver
  - Keep track of current window size
  - Only increase to sender if “significant”
- What is wrong with this?

Solutions

- Send ACK, don't increase window size
- Delay the ACK
  - Advantage – decrease traffic
    - How is the traffic decreased?
  - Disadvantages
    - Delayed too long – what happens?
    - Confuse averageRTT

Sender Silly Window Syndrome Avoidance – Nagle's Algorithm

- Clumping
  - TCP waits to create segment
- How long should the wait be?
- Self-clocking
  - No computation
  - Arrival of ACK triggers transmission

TCP Congestion Control

- End-end congestion control
- Network-assisted congestion control

- Congestion collapse

Congestion Control Variables

- CongWin
- Threshold

- Allowed window = min{CongWin, RcvWin}

TCP Congestion Control Algorithm

- Three main components
  - Additive-increase, multiplicative-decrease
  - Slow start
  - Reaction to timeout events

Slow start phase

- Start CongWin = 1 MSS
- Receive ACK, CongWin = CongWin + 1
- Increases exponentially
- Congestion avoidance
  - Only increase CongWin if all segments ACK'd

Multiplicative Decrease Congestion Avoidance

- Lost segment
  - CongWin reduced by half
  - Minimum value is 1 MSS
- Additive Increase, Multiplicative Decrease (AIMD)

Reaction to Timeout Events

- Enter slow start phase
- Grow exponentially until ½ value before timeout
- Threshold value
  - Initially very large (65KB)
  - Lost segment: Threshold = ½ * CongWin

- Why have different ways to handle congestion control?

Versions of TCP

- Tahoe
- Reno
  - Fast recovery or fast retransmit
- NewReno
- SACK
- Explicit Congestion Notification (ECN)

Random Early Discard (RED)

- Tail-drop policy of routers
- Causes TCP connections to enter slow-start
- Global synchronization

- Random Early Drop
- Random Early Detection

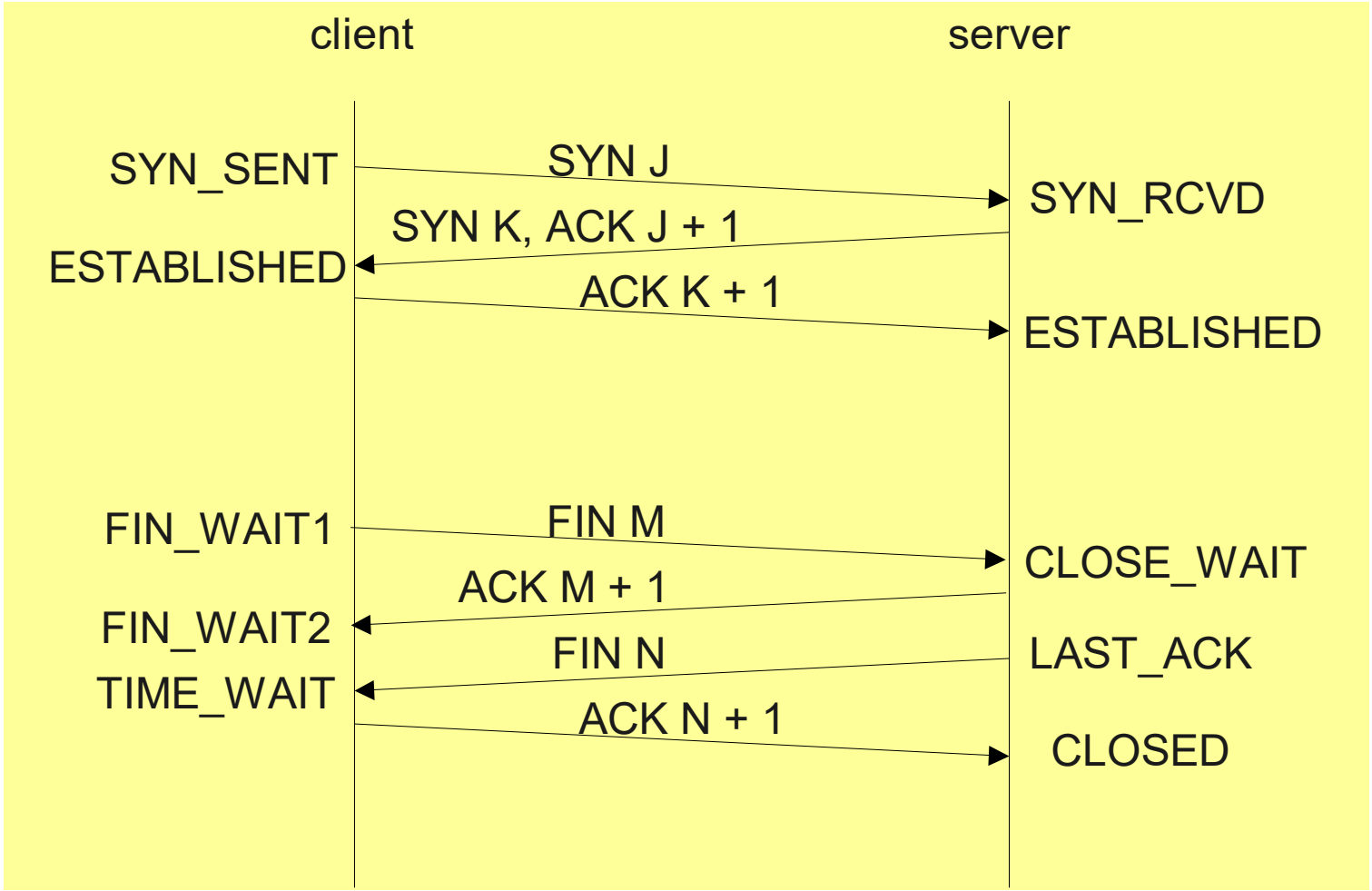
RED Variables

- Two threshold variables
  - T_{min}
  - T_{max}
- Probability, p

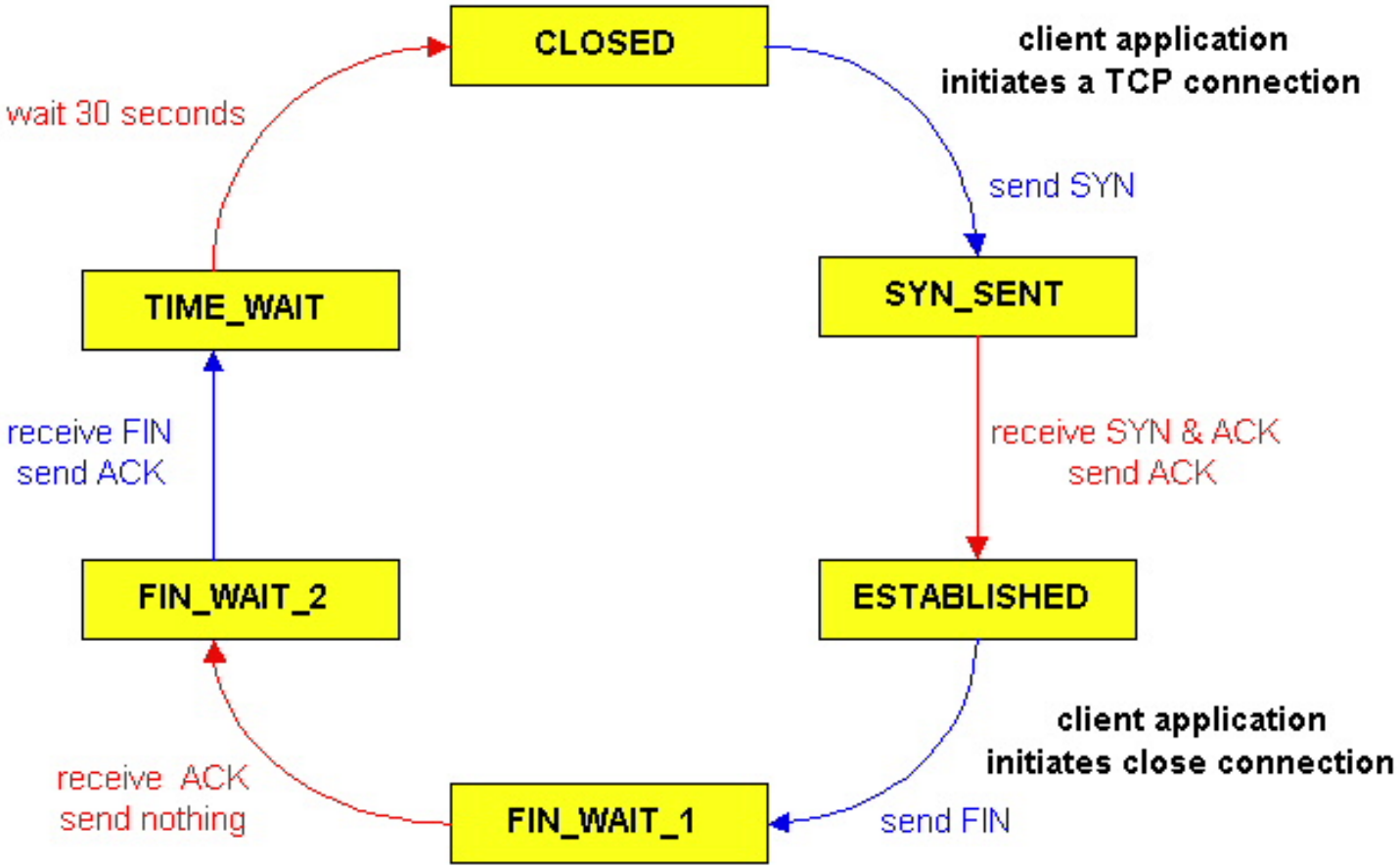
RED Implementation

- If the queue contains fewer than T_{min} datagrams, add the new datagram to the queue.
- If the queue contains more than T_{max} datagrams, discard the new datagram.
- If the queue contains between T_{min} and T_{max} datagrams, randomly discard the datagram according to a probability, p.

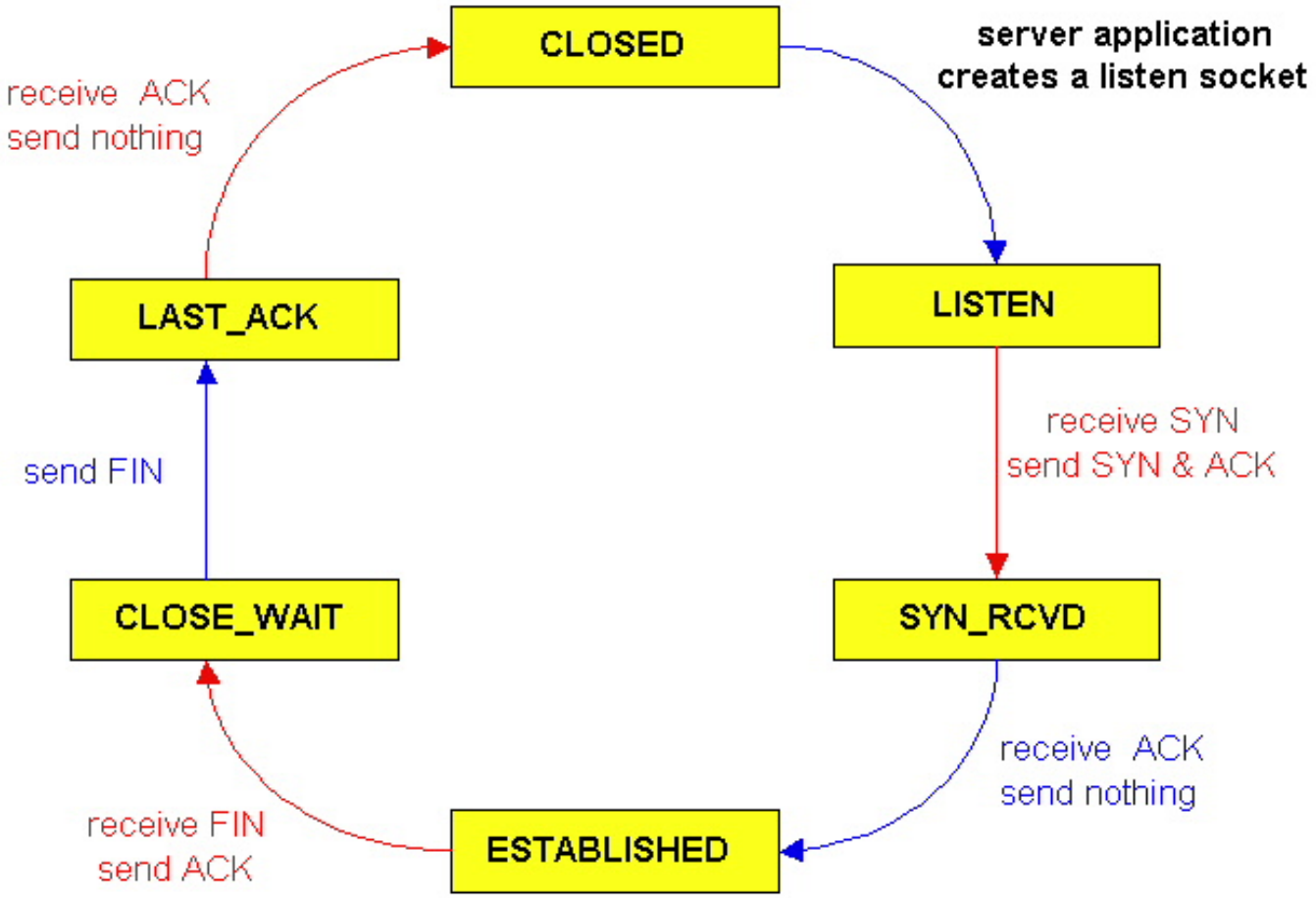
TCP STATE TRANSITION DIAGRAM

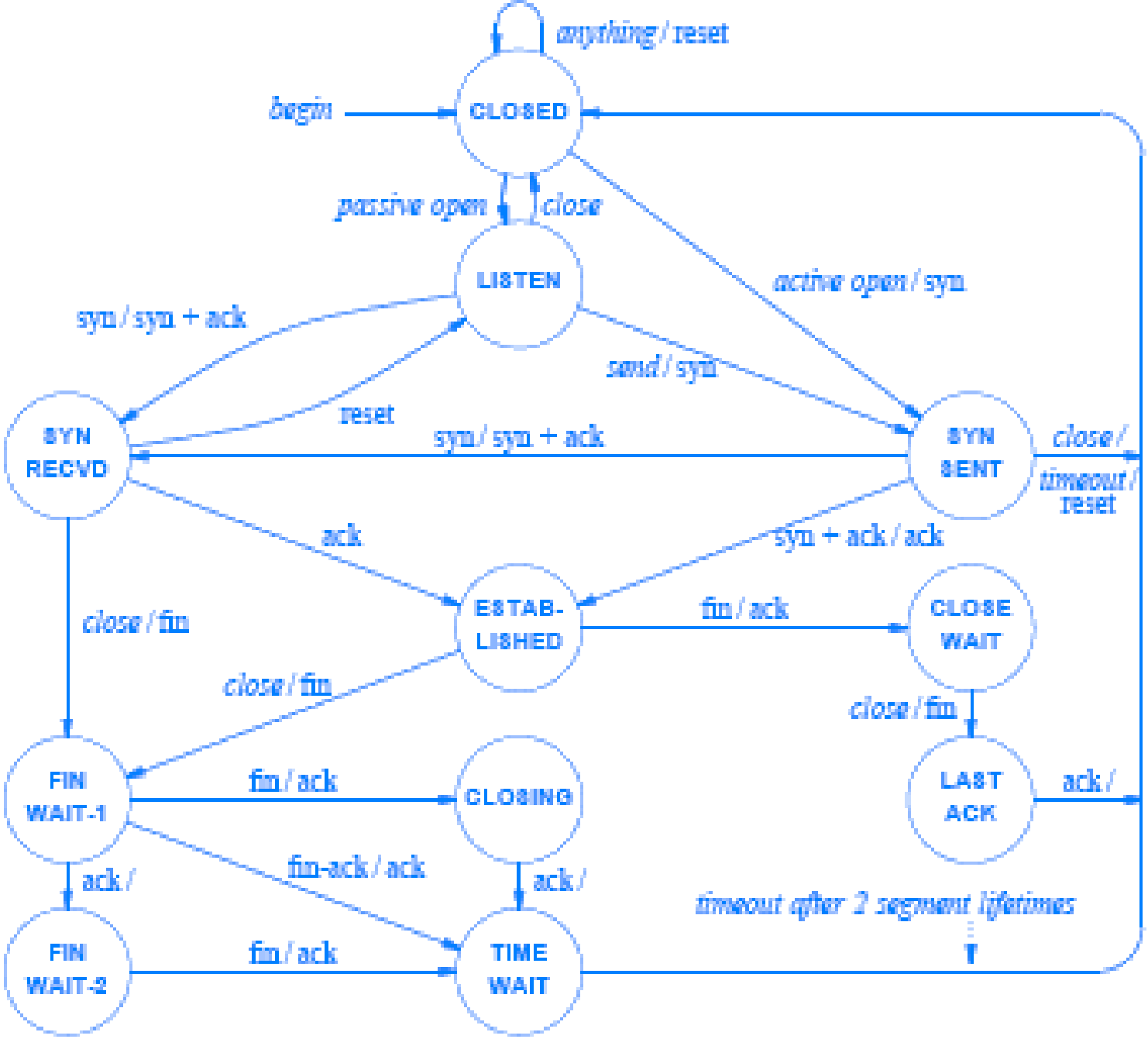


Client Lifecycle



Server Lifecycle




anything / reset
begin
CLOSED
passive open
close
LISTEN
active open / syn
syn / syn + ack
send / syn
reset
SYN RECVD
syn / syn + ack
SYN SENT
close /
timeout / reset
ack
syn + ack / ack
close / fin
ESTAB-LISHED
fin / ack
CLOSE WAIT
close / fin
close / fin
FIN WAIT-1
fin / ack
CLOSING
LAST ACK
ack /
fin-ack / ack
ack /
ack /
timeout after 2 segment lifetimes
FIN WAIT-2
fin / ack
TIME WAIT