Reliable Stream Transport (TCP)

Sections 5.2.1-5.2.4 in Peterson & Davie
(ComputerNetworks: A Systems Approach)

Sections 12.1-12.11, 12.15. 12.119 and 12.24-12.28 in Comer (Internetworking with TCP/IP, vol 1)

Introduction

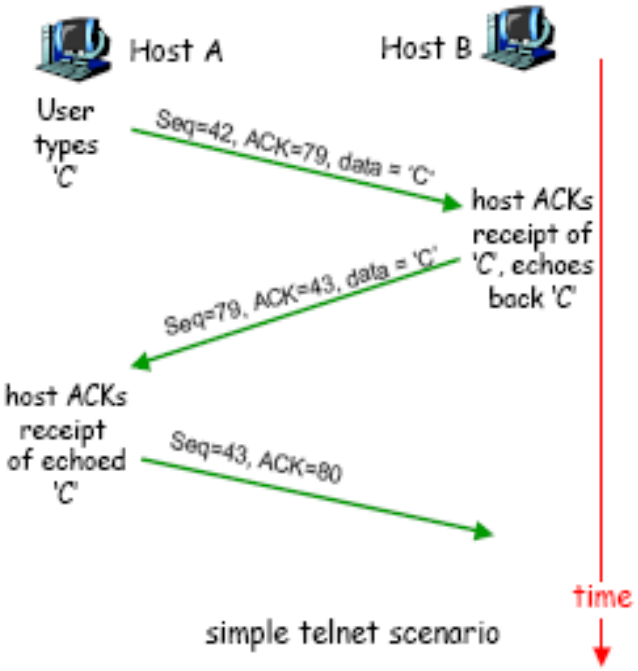
- Applications programs often need to send a large volume of data from one computer to another.
- Need for stream delivery:
  - Reliable.
  - Connection-oriented.
  - Byte-stream service.
- Run above IP
  - No assumptions about physical network.

Properties

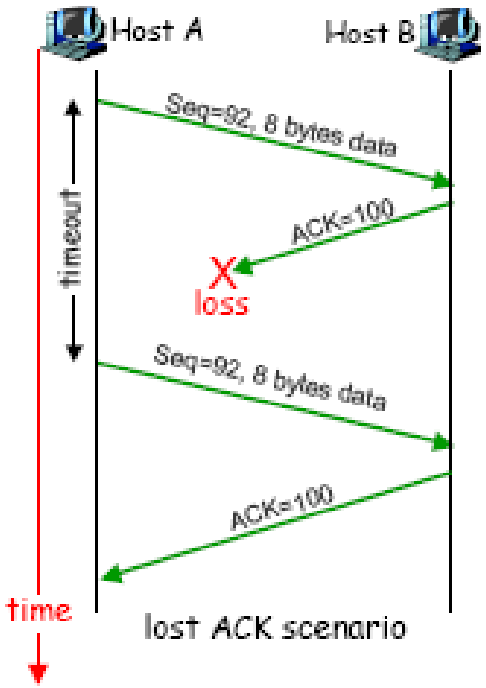
- Guarantees the reliable, in-order delivery of a stream of bytes.
- Full-duplex protocol: Each TCP connection supports a pair of byte streams, one flowing in each direction.
- Flow-control mechanism: Allows the receiver to limit how much data the sender can transmit at a given time.
- TCP implements a congestion-control mechanism.
- TCP (like UDP) allows multiple application programs on any given host to simultaneously carry on a conversation with their peers.

Reliable Delivery

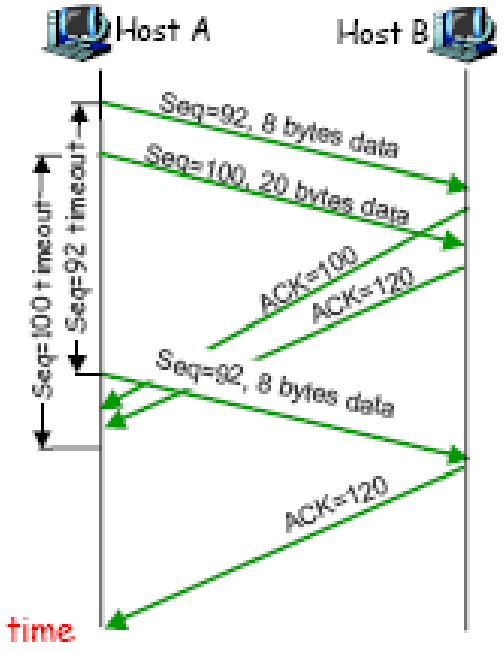
(Positive Acknowledgement, Retransmission, Sliding Window)



Positive Acknowledgement



Retransmission



Sliding Window

End-to-End issues

(1) Logical connections

- Logical connections between processes
  - TCP needs an explicit connection establishment phase during which the two sides of the connection agree to exchange data with each other. The two parties establish some shared state to enable the sliding window algorithm to begin.
  - Connection teardown is needed so each host knows it is OK to close the connection and free any resource allocated.

End-to-End issues
(2) Different round-trip times

- TCP connections are likely to have widely different round-trip times.
- The same TCP protocol must be able to support both of these connections.
- The timeout mechanism that triggers retransmissions must be adaptive.

End-to-End issues

(3) Packets reordering – duplicating - looping

- Packets may be reordered as they cross the Internet
  - Packets slightly out of order do not cause a problem (sliding window algorithm can reorder packets correctly)
  - TCP has to be prepared for very old packets to suddenly show up at the receiver
- Packets may be duplicated
  - Lost Acknowledgment causing a timeout and retransmission.
  - Unexpectedly delayed packets
- Packets may “lost” the route
  - Such message must be eventually discarded from the network.
  - Use of Time To Live (TTL) field

End-to-End issues
(4) Flow-control issue

- Almost any kind of computer can be connected to the Internet.
  - Resources dedicated to any one TCP connection are highly variable.
  - TCP must include a mechanism to "learn" what resources (e.g., how much buffer space) the other side is able to apply to the connection.

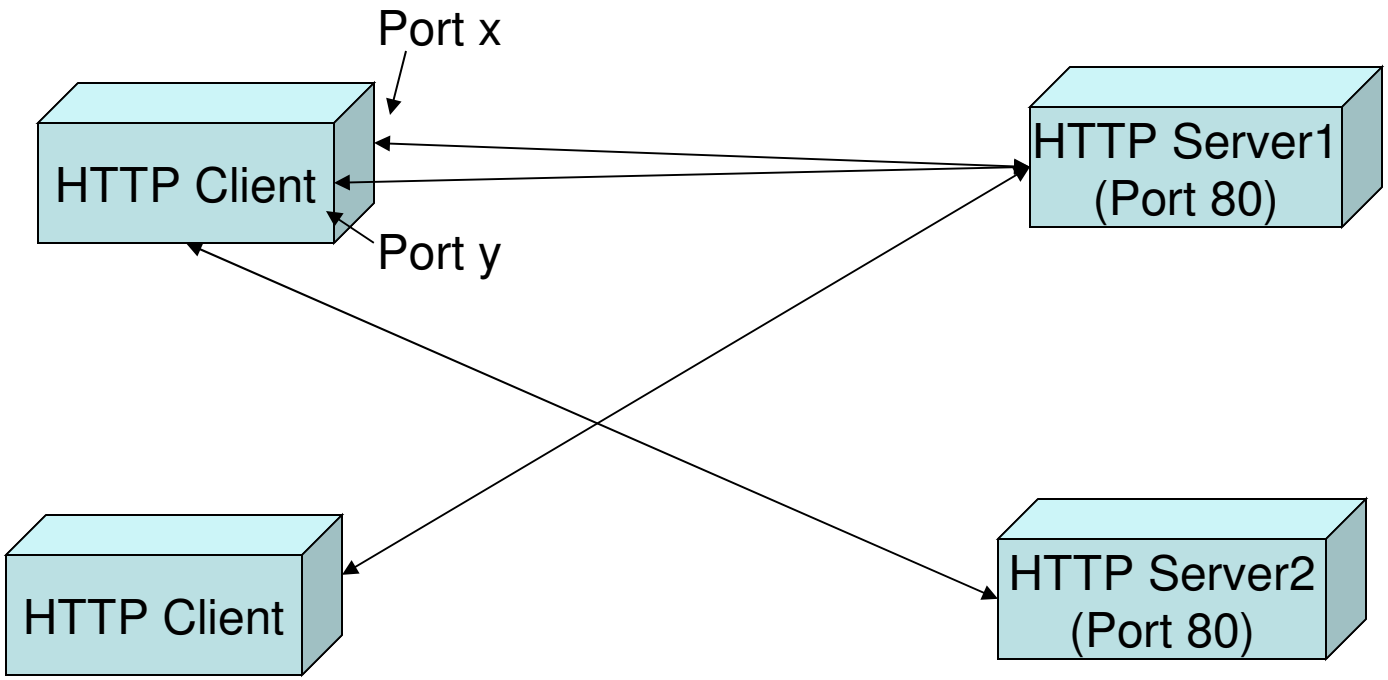
End-to-End issues
(5) Congestion Control

- Bottleneck links exists in the Internet.
- Available bandwidth of a link changes over time.
- TCP should be able to adjust the data injection rate, using feedback received from the network.

Ports, Endpoints, Connections

- Port
  - queue into which arriving datagram are placed.
  - shared among objects
- Endpoint: Pair of integers (host, port)
  - Host is the IP address for a host
  - Port is a TCP port on that host
  - Example: (128.10.2.3, 25)
- TCP creates Virtual Connections
  - Are identify by a pair of endpoints (sending and receiving endpoint)
  - Example: (18.26.0.1069) and (128.10.2.3, 25)
  - **Because TCP identifies a connection by a pair of endpoints, a given TCP port number can be shared by multiple connections on the same machine.**

Ports, Endpoints, Connections



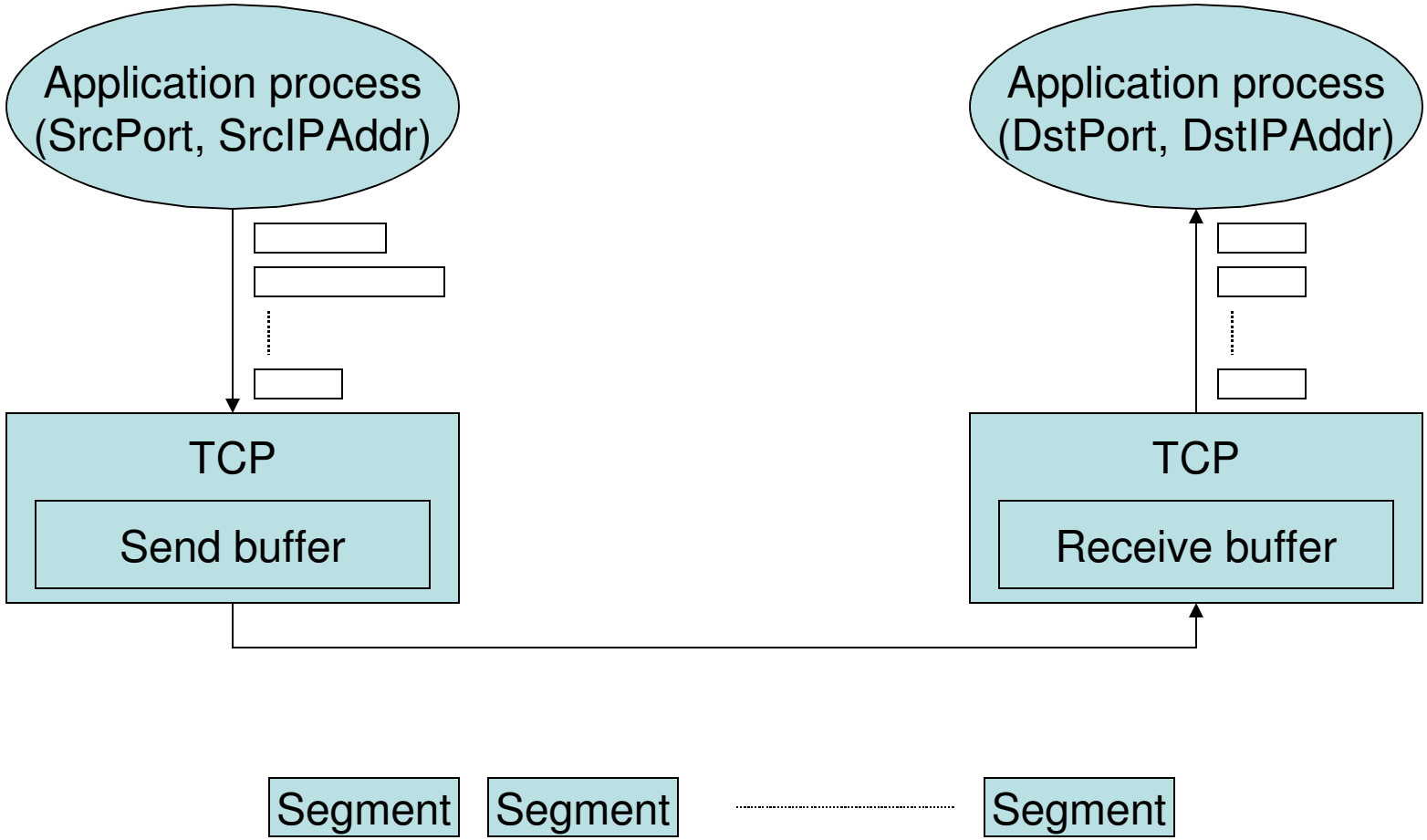
Passive and Active Opens

- Passive Open: An application program requests from the operating system and allocates a particular port, where it will accept incoming connections (local port number specified).
- Active Open: An application program contacts the operating system requesting to establish a connection (remote port number specified).

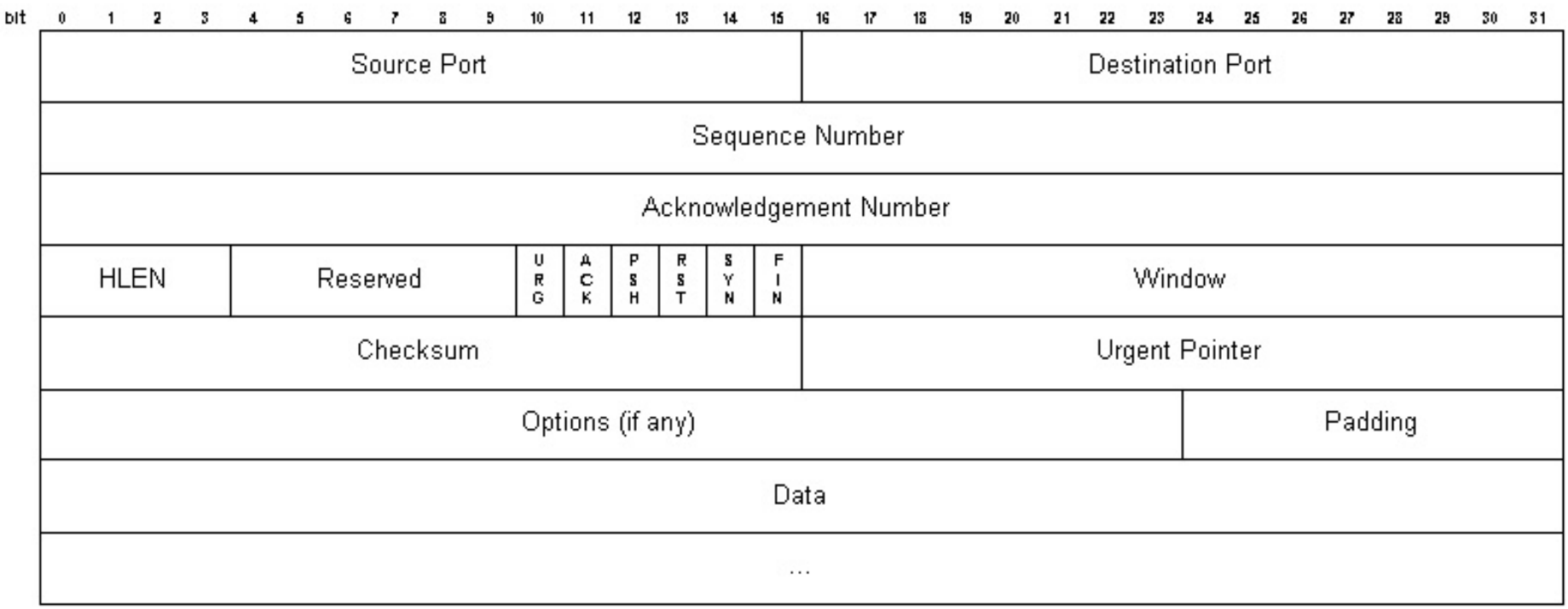
Segments

- TCP offers “Byte Stream” service to the application layer
  - The sender writes bytes into a TCP connection and the receiver reads bytes out of the TCP connection.
- TCP on the source host buffers enough bytes from the sending process to fill a reasonably sized packet
- Sends this packet (called “segments”) to its peer on the destination host.
- TCP on the destination host then places the contents of the packet into a receive buffer
- The receiving process reads from this buffer.

Segments


Application process
(SrcPort, SrcIPAddr)
Application process
(DstPort, DstIPAddr)
TCP
Send buffer
TCP
Receive buffer
Segment
Segment
Segment

TCP header


bit 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port
Destination Port
Sequence Number
Acknowledgement Number
HLEN
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window
Checksum
Urgent Pointer
Options (if any)
Padding
Data
...

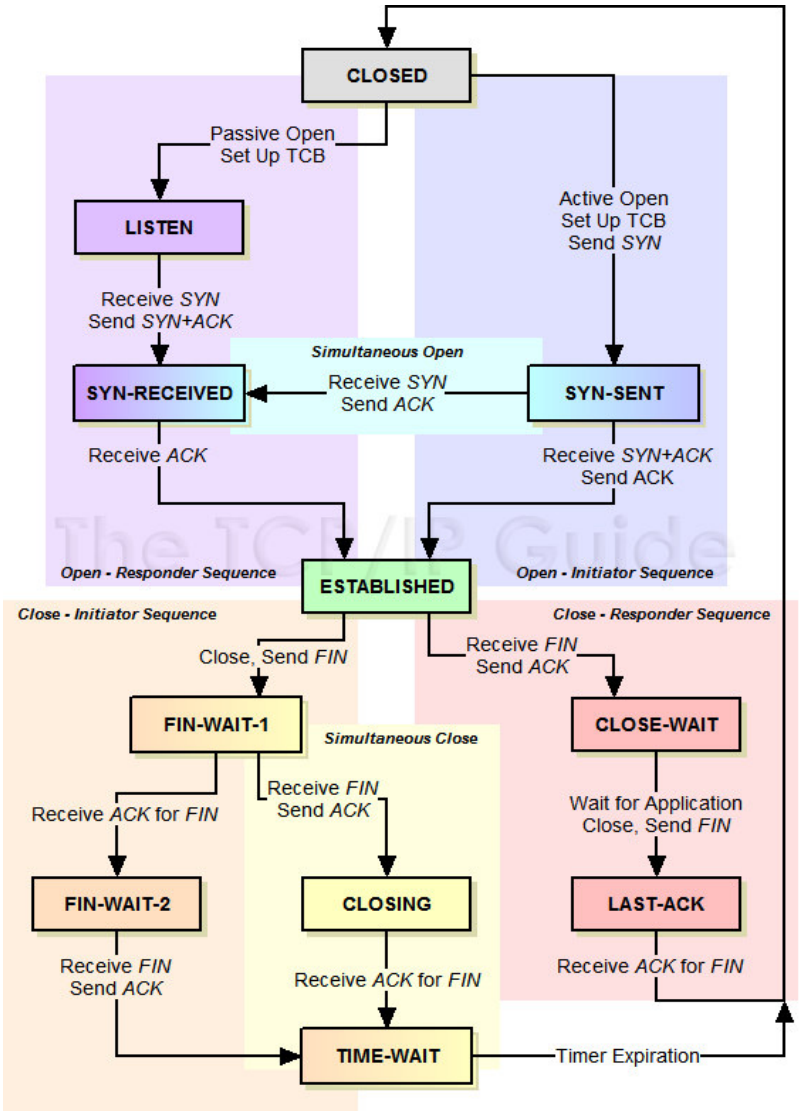
TCP header

- Sequence Number: the sequence number for the first byte of data carried in that segment.
- Acknowledgment: Number of the octet that the source expects to receive next.
- Window: The data that the source is willing to accept.
- HLEN: Length of the segment header measured in 32-bits.
- Urgent pointer: Position in the segment where urgent data ends.
- Flags: Used to relay control information between TCP peers.
  - SYN and FIN flags are used when establishing and terminating a connection, respectively.
  - ACK flag is set any time the Acknowledgment field is valid.
  - URG flag signifies that this segment contains urgent data.
  - PUSH flag signifies that the sender invoked the push operation.
  - RESET flag signifies that the receiver wants to abort the connection.

TCP State Machine

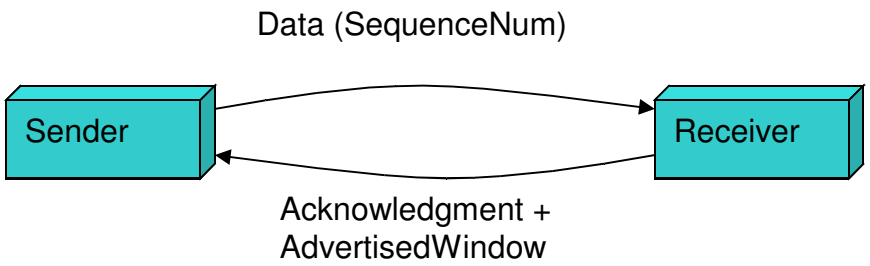
- The behavior of TCP when opening and closing a connection is described by a SM
- The SM shows:
  - States involved in opening a connection (everything above ESTABLISHED)
  - States involving in closing a connection (everything below ESTABLISHED).
  - Everything that goes on while a connection is open is hidden in the ESTABLISHED state.

TCP State Machine



Established

- Two sides begin sending data

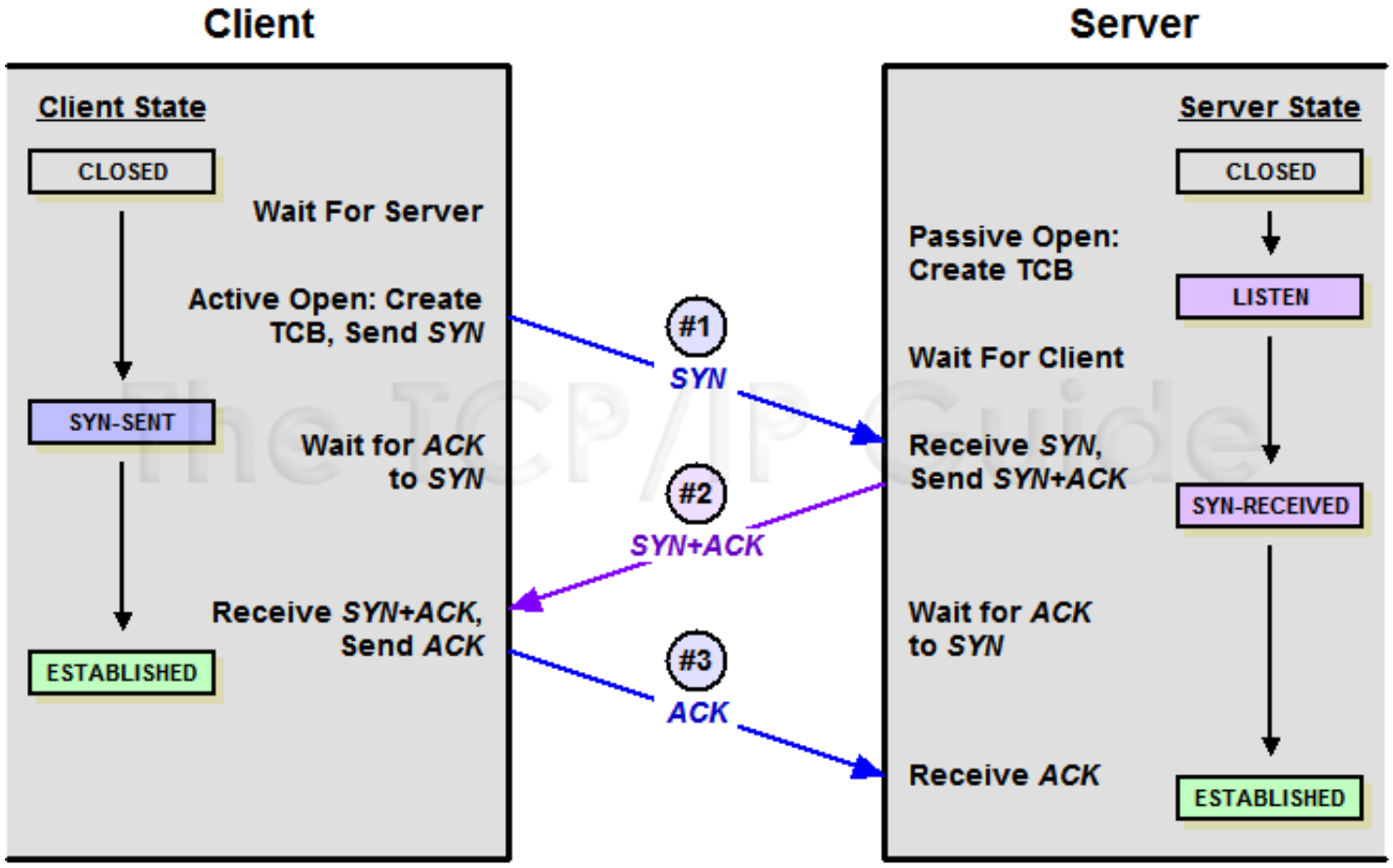


Connection Establishment
(Three-Way handshake)

- The two parties want to agree on a set of parameters:
  1. The client (the active participant) sends a segment to the server (the passive participant) stating the initial sequence number it plans to use (Flags = SYN, Sequence Number = x).
  2. The server then responds with a single segment that both acknowledges the client's sequence number (Flags = ACK, Ack = x + 1) and states its own beginning sequence number (Flags = SYN, Sequence Number = y).
  3. The client responds with a third segment that acknowledges the server's sequence number (Flags = ACK, Ack = y + 1).
- Acknowledgment field identifies the "next sequence number expected“.
- TCP specification requires that each side of a connection select an initial starting sequence number at random.

Connection Establishment

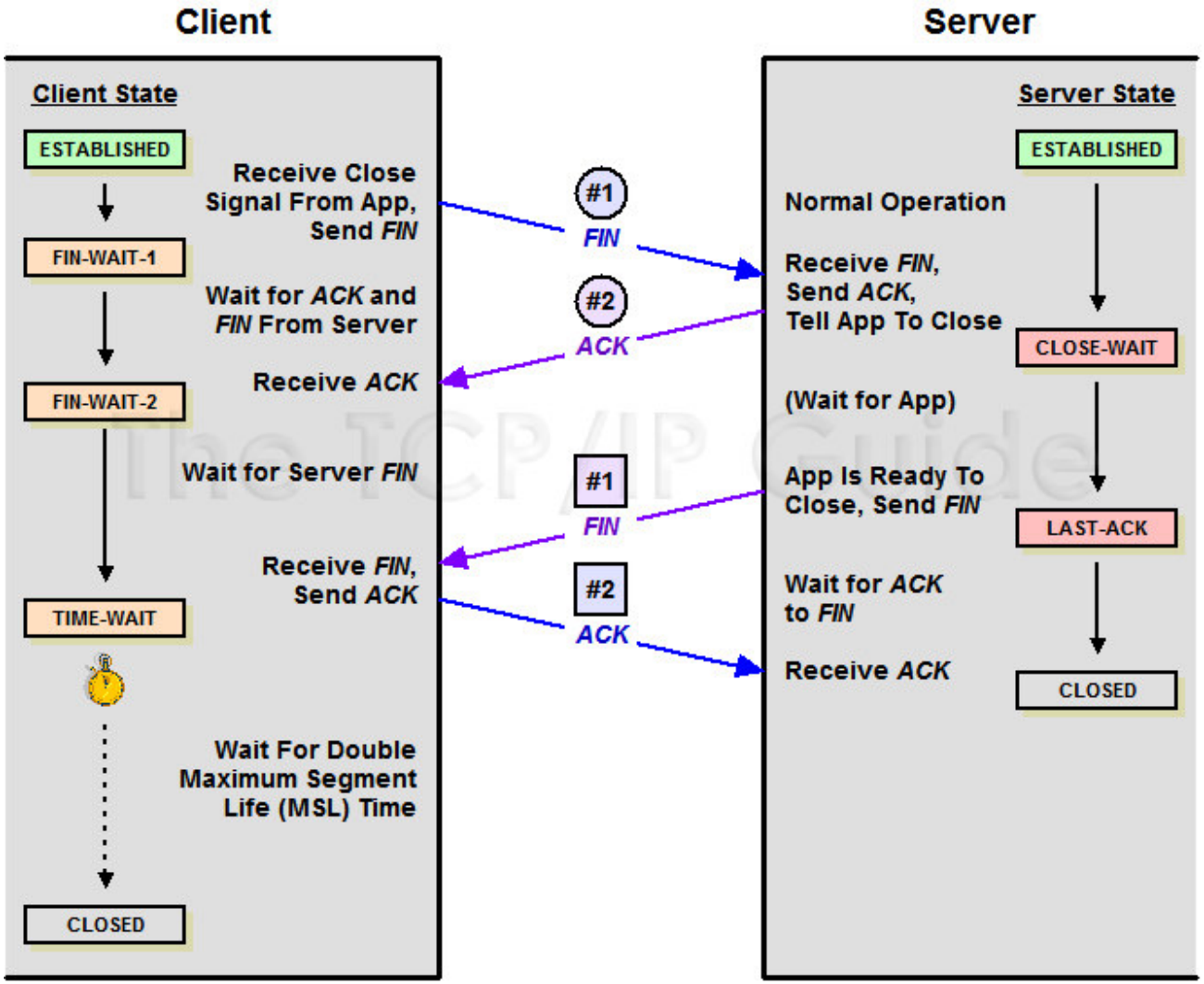
(Three-Way handshake)



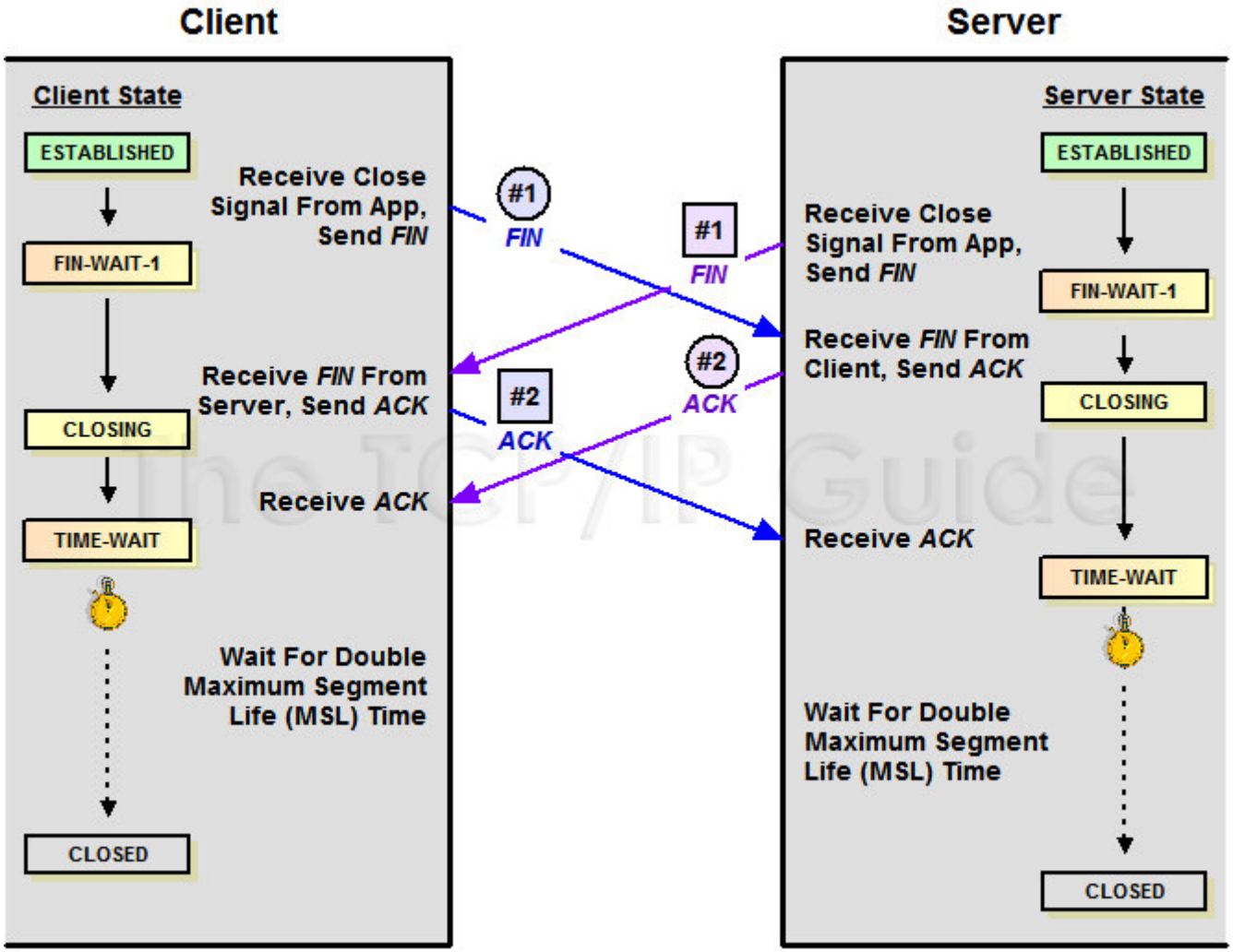
Closing TCP connection

- There are three combinations of transitions that get a connection from the ESTABLISHED state to the CLOSED state:
  - This side closes first: ESTABLISHED → FIN_WAIT_1 → FIN_WAIT_2 → TIME_WAIT → CLOSED.
  - The other side closes first: ESTABLISHED → CLOSE_WAIT → LAST_ACK → CLOSED.
  - Both sides close at the same time: ESTABLISHED → FIN_WAIT_1 → CLOSING → TIME_WAIT → CLOSED.

Closing TCP connection (example 1)



Closing TCP connection (example 2)



TCP connection Reset

- Used in abnormal conditions forcing an application to break a connection
- One side initiates termination by sending a segment with the RST bit set.
- The receiver of such segment immediately aborts the connection.

Flow Control

- Used for avoiding buffers overflow at the receiver's site.
- Utilizes window field of TCP header: Specifies how many additional data the receiver is prepared to accept.
- Based on explicit feedback (receiver_advertisement).
- Allowed_window = min(receiver_advertisement, congestion_window)

Congestion Control

- Based on heuristics.
- Uses implicit feedback:
  - RTT=(α*Old_RTT)+((1- α)*New_Round_Trip_Sample)
  - Timeout = β*RTT
  - Congestion Window Size
    - Start with w=1
    - w ← w - aw (when a loss is detected)
    - w ← w + b/w (when an ACK is received)
    - Many different approaches for calculating congestion window have been proposed.