Network : Transport Protocols (2)

Jae Hyeon Kim

Reference

William Stalling, Data and Computer Communications 10/E, Prentice Hall

Ordered Delivery

- With an unreliable network service it is possible that segments may arrive out of order
- Solution is to number segments sequentially
  - TCP makes use of a scheme where each data octet is implicitly numbered

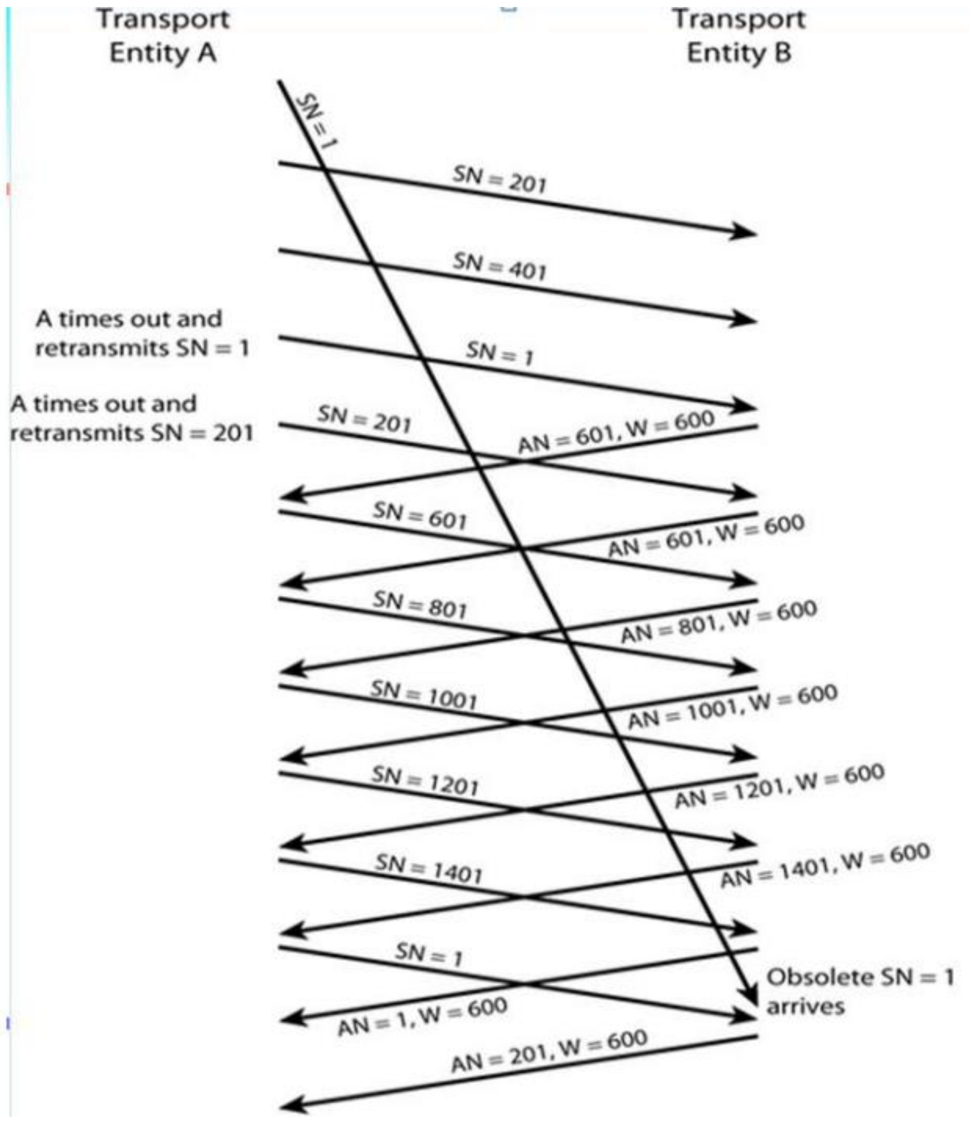
Retransmission Strategy

- Events necessitating retransmission:
    - Segment may be damaged in transit but still arrives at its dest.
    - Segment fails to arrive
- Sender does not know transmission was unsuccessful
- Receiver acknowledges successful receipt by returning a segment containing an acknowledgement number
- Retransmission strategy
    - No ACK will be issued if a segment does not arrive successfully
    - A timer needs to be associated with segment at it is sent
    - If timer expires before acknowledgement is received, sender must retransmit

Duplication Detection

- Receiver received a segment, but lost its Ack.
    - Eventually the sender try to retransmit the segment
    - Receiver must be able to recognize duplicates
    - Segment sequence numbers help
- Complications arise if:
    - A duplicate is received prior to the close of the connection
    - Sender must not get confused if it receives multiple acknowledgements to the same segment
    - Sequence number space must be long enough
    - A duplicate is received after the close of the connection

– Incorrect Duplicate Detection



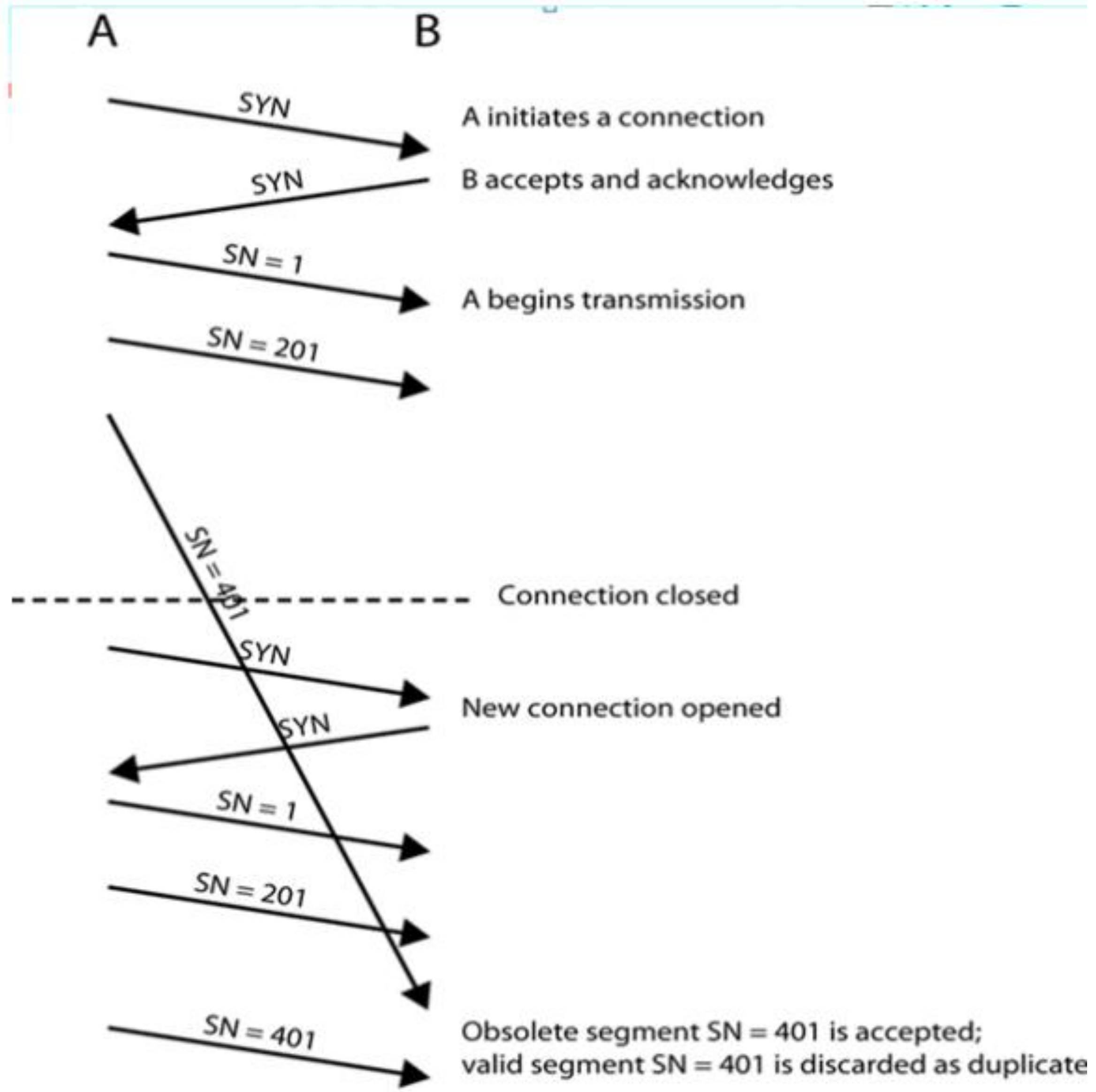
Flow Control

- Credit allocation quite robust with unreliable network
  - Can ack data & grant credit
  - Lost ACK recovers on next received
- Have problem if AN=I, W=0 closing window
  - Then send AN=I, W=j to reopen, but this is lost
  - Sender thinks window closed, receiver thinks it open
- Solution is to use persist timer
- If timer expires, send something
  - Could be re-transmission of previous segment

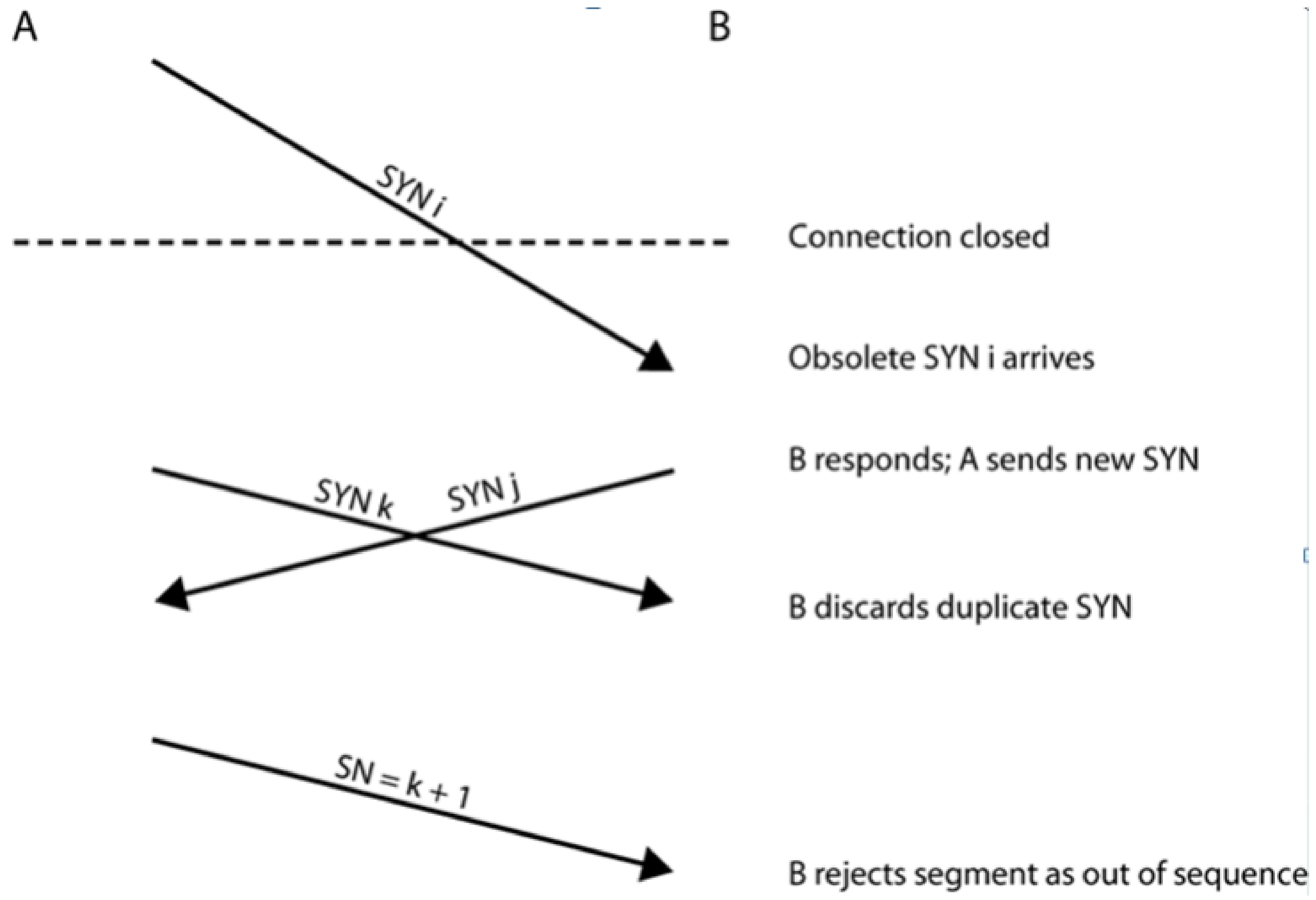
Connection Establishment

- Two way handshake
  - A send SYN, B replies with SYN
  - Lost SYN handled by re-transmission
  - Ignore duplicate SYNs once connected
- Lost or delayed data segments can cause connection problems
  - With data segment from old connections
  - Make use SYN i, where i is the sequence # of the first data
  - Start segment # far removed from previous connection
  - There is still a problem with SYN segment from old connections
  - Need ACK to include i
  - So, three way handshake

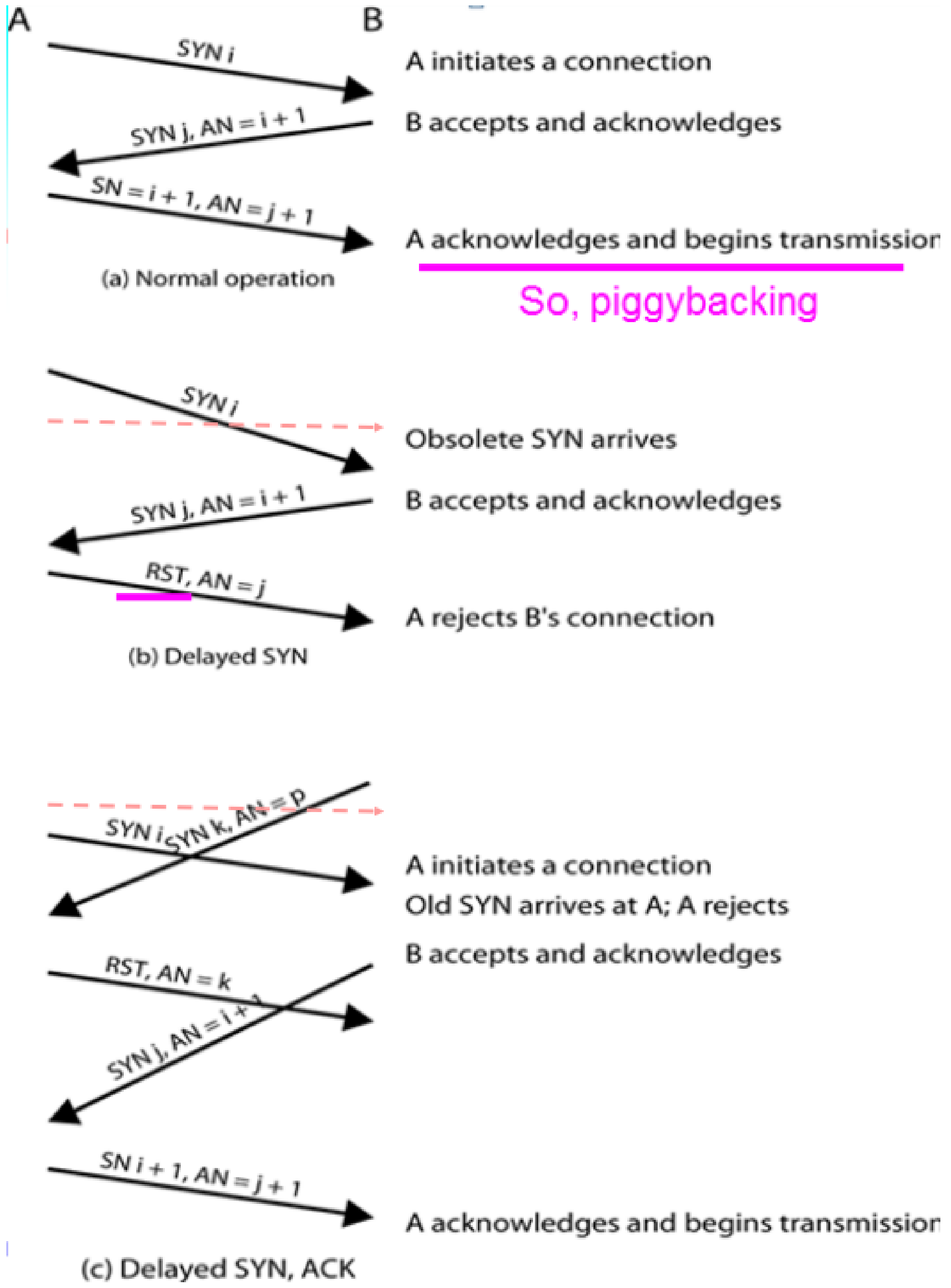
Two Way Handshake problem with Obsolete Data Segment



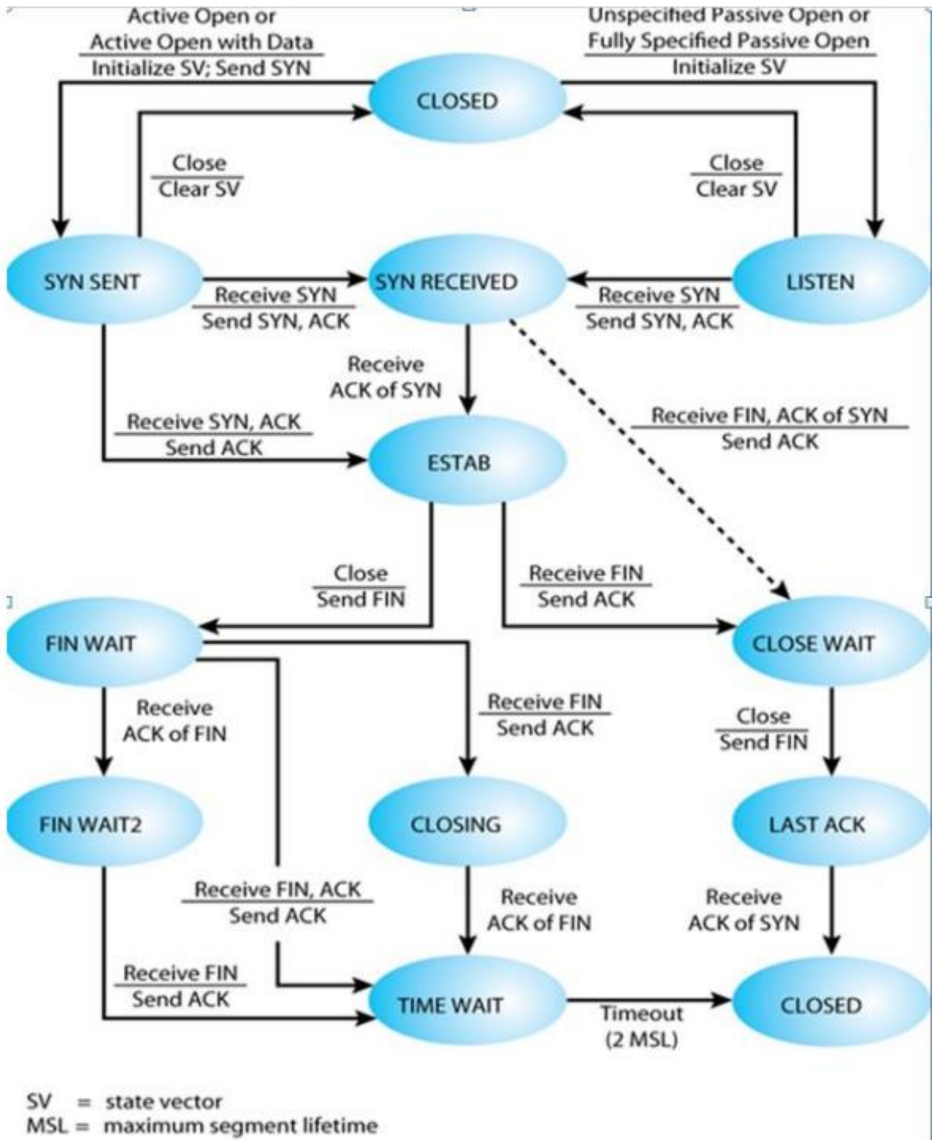
– Obsolete SYN Segment



Three Way Handshake : Examples



– TCP Entity State Diagram



Connection Termination

- 2-way handshake is inadequate on an unreliable network
  - Like connection establishment, it need 3-way handshake
- Out of order segments could cause:
  - The FIN segment to arrive before the last data segment
- To avoid this problem the next sequence number after the last octet of data can be assigned to FIN
  - Each side must explicitly acknowledge the FIN of the other using an ACK with the sequence number of the FIN to be acknowledged

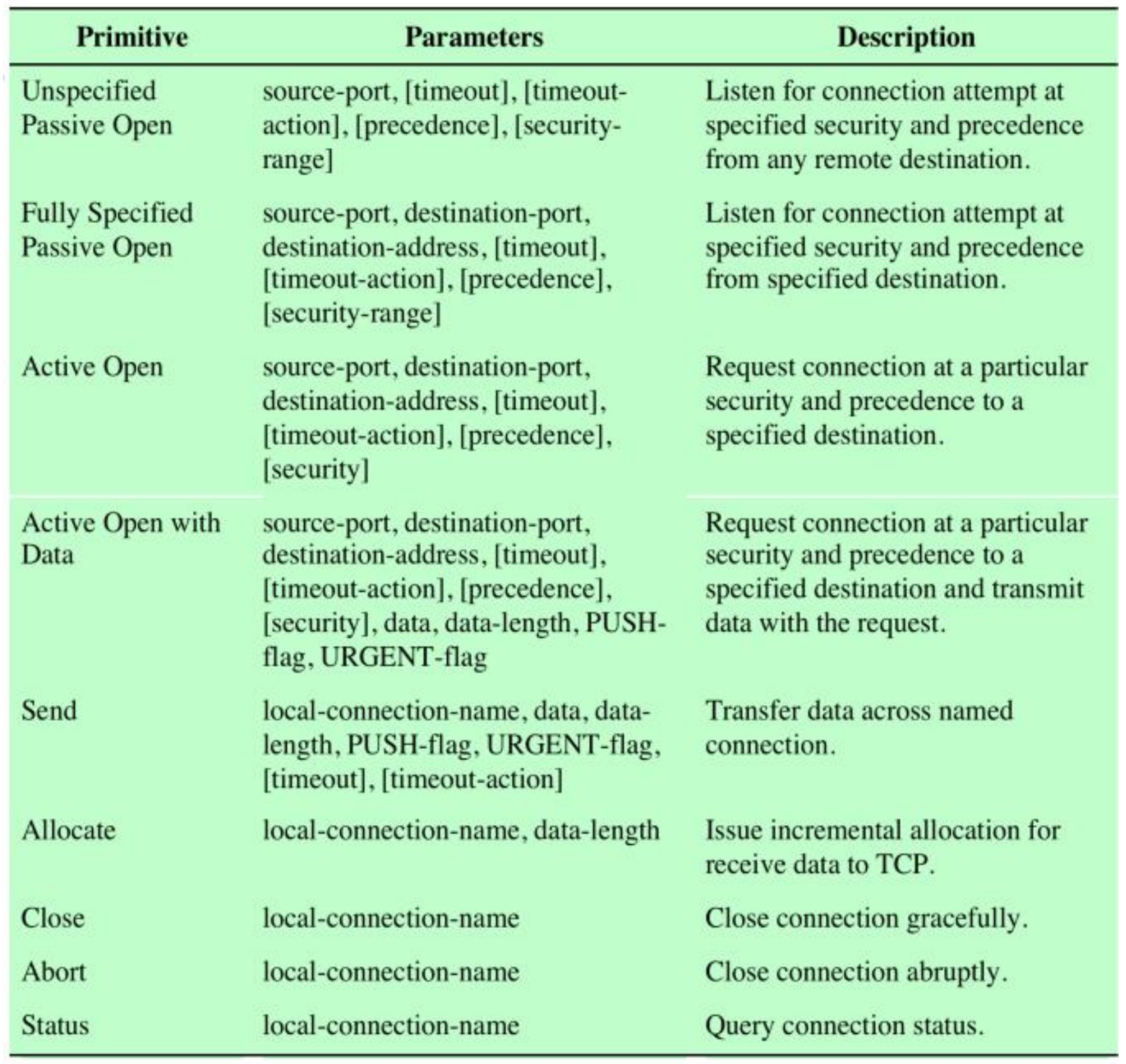
Failure Recovery

- After restart a system, the state information of all active connections is lost
  - May have half open connection because side that did not crash still thinks it is connected
- Still active side of a half-open connection can close the connection using a keepalive timer
  - Wait for ACK for (time out) * (number of retries)
  - When expired, close connection and inform user
- Or, failed side returns an RST i to every segment i that it receives
  - RST i must be checked for validity on the other side
  - If valid, an abnormal termination occurs

TCP Services

- TCP (Transmission Control Protocol) : RFC 793
    - Connection oriented, reliable communication over reliable and unreliable (inter)networks
- Two ways of labeling data:
    - Data stream push
    - User requires transmission of all data up to push flag
    - Receiver will deliver in same manner
    - Avoids waiting for full buffers
    - Urgent data signal
    - Indicates urgent data is upcoming in stream
    - User decides how to handle it

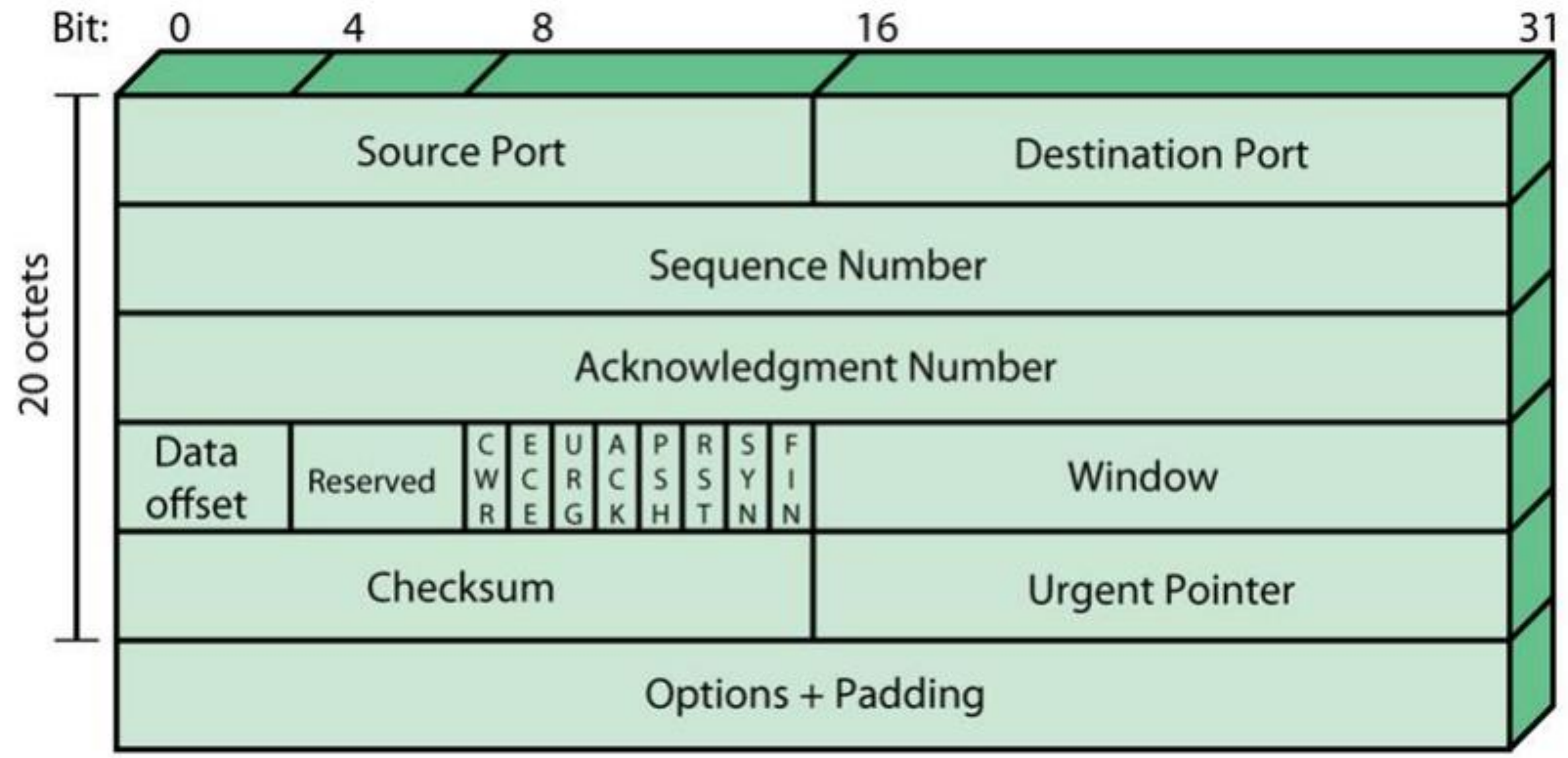
– TCP Services Request Primitives



Primitive	Parameters	Description
Unspecified Passive Open	source-port, [timeout], [timeout-action], [precedence], [security-range]	Listen for connection attempt at specified security and precedence from any remote destination.
Fully Specified Passive Open	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security-range]	Listen for connection attempt at specified security and precedence from specified destination.
Active Open	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security]	Request connection at a particular security and precedence to a specified destination.
Active Open with Data	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security], data, data-length, PUSH-flag, URGENT-flag	Request connection at a particular security and precedence to a specified destination and transmit data with the request.
Send	local-connection-name, data, data-length, PUSH-flag, URGENT-flag, [timeout], [timeout-action]	Transfer data across named connection.
Allocate	local-connection-name, data-length	Issue incremental allocation for receive data to TCP.
Close	local-connection-name	Close connection gracefully.
Abort	local-connection-name	Close connection abruptly.
Status	local-connection-name	Query connection status.

Note: Square brackets indicate optional parameters.

– TCP Header Format



TCP Mechanisms

- Can be grouped into:
- connection establishment
  - Always uses a three-way handshake
  - Connection is determined by host and port
- Data transfer
  - Viewed logically as consisting of a stream of octets
  - Flow control is exercised using credit allocation
- Connection termination
  - Each TCP user must issue a CLOSE primitive
  - An abrupt termination occurs if the issues an ABORT primitive

TCP Implementation Policy Options (1)

- Send policy
  - If no pushed data, a sending TCP entity transmits at its own convenience in credit allocation
  - May construct segment per batch of data from user : quick response but higher overheads
  - May wait for certain amount of data : slower response but lower overheads
- Deliver policy
  - In absence of push, can deliver data at own convenience
  - May deliver from each segment received : higher O/S overheads but more responsive
  - May buffer data from multiple segments : less O/S overheads but slower

TCP Implementation Policy Options (2)

- Accept policy
- If segments arrive out of order the receiving TCP entity has two options:
- In-order
  - Accept only segments that arrive in order; any segment that arrives out of order is discarded
  - Makes for simple implementation but places a burden on the networking facility
  - If a single segment is lost in transit, then all subsequent segments must be retransmitted
- In-window
  - Accept all segments that are within the receive window
  - Requires a more complex acceptance test and a more sophisticated data storage scheme

TCP Implementation Policy Options (3)

- Retransmit policy (three strategies)
- First only
  - Maintain one retransmission timer for entire queue
  - If timer expires, retransmit the segment at the front of the queue
  - Efficient traffic generation, but can have considerable delays
- Batch
  - Maintain one retransmission timer for entire queue
  - If timer expires, retransmit all segments in the queue
  - May result in unnecessary retransmissions

TCP Implementation Policy Options (3)

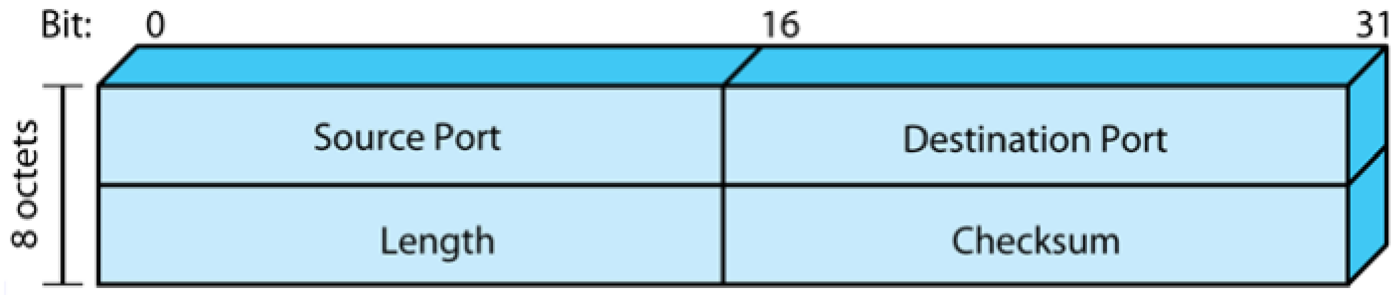
- Individual
  - Maintain one timer for each segment in the queue
  - More complex implementation

TCP Implementation Policy Options (4)

- Acknowledge policy (timing of ack.)
- Immediate
  - Immediately transmit an empty segment containing the appropriate acknowledgment number
  - Simple and keeps the remote TCP fully informed
  - Limits unnecessary retransmissions
  - Can cause a further load on the network
- Cumulative
  - Wait for an outbound segment with data on which to piggyback the acknowledgement
  - Typically used
  - Requires more processing at the receiving end and complicates the task of estimating round-trip time

UDP

- Connectionless service specified in RFC 768
  - Delivery & duplication control not guaranteed
- Reduced overhead in transmission
- Least common denominator service
- Uses:
  - Data collection & dissemination
  - Request-response
  - Real time application


Bit:
0
16
31
8 octets
Source Port
Destination Port
Length
Checksum