Unit 3

Transmission Control Protocol (TCP)

- Review
- Congestion control
- Flow control
- Error control

Before Going through

- Two types of transport services
  - Connection-oriented → reliable
  - Connectionless/datagram → unreliable
- Protocols are introduced in an ***evolutionary*** fashion

To Provide Reliable Service on Reliable Network Service

- Four issues need to be addressed
  - Addressing
  - Multiplexing
  - Flow control
  - Connection establishment/termination

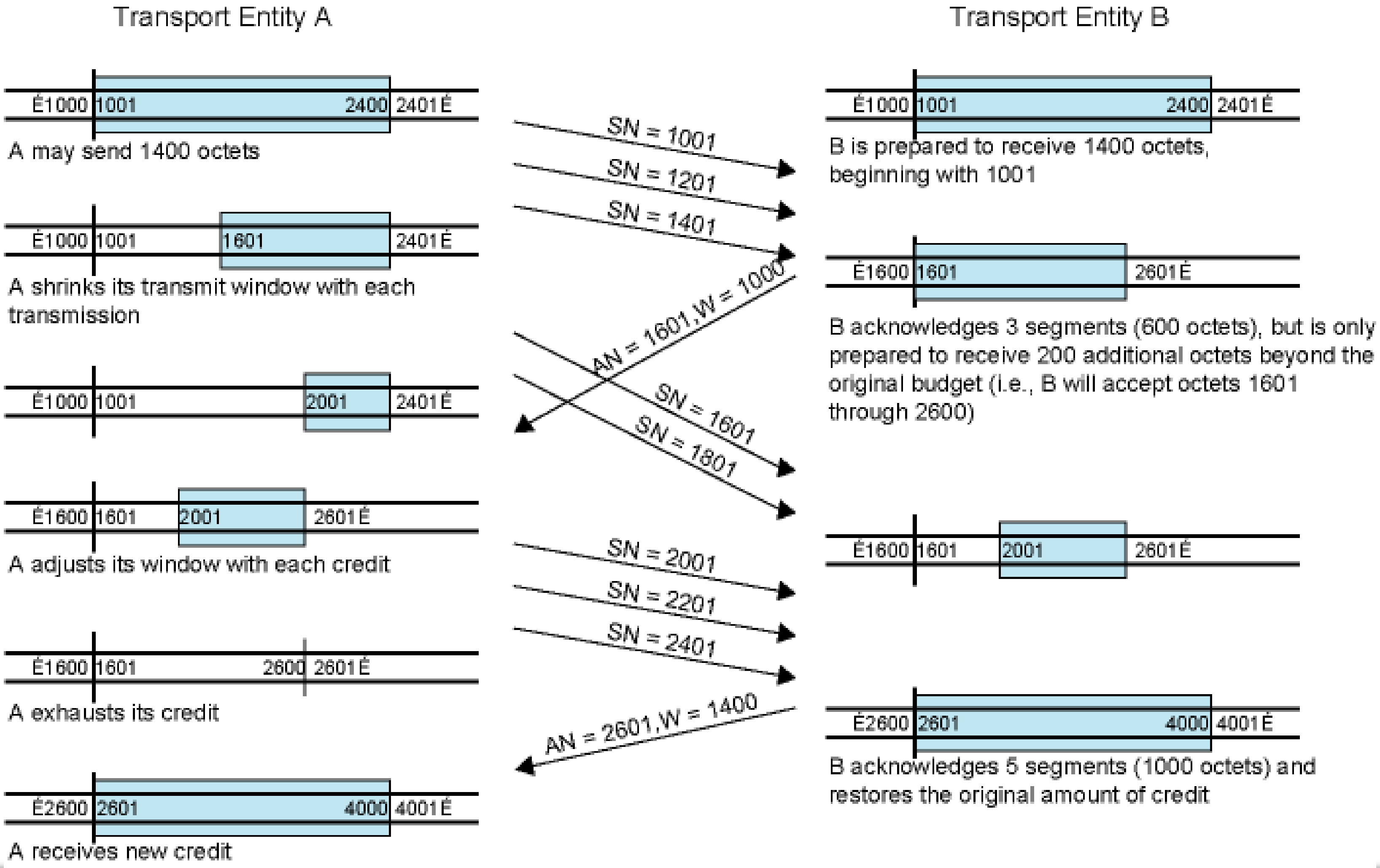
Flow Control

- Approaches
  - Do nothing: segments discarded.
    - What’s next?
    - Is that reasonable?
  - Refuse to accept further segments from the network service: a backpressure mechanism.
  - Use a credit scheme: used in TCP.

Flow Control (contd.)

- Credit scheme
  - Sequence number (SN), acknowledgment number (AN), window (W)
  - SN (in data segment): the ***first*** octet in the segment data field
  - AN=*i* (in ACK, or piggyback in data segment): all octets through sequence number SN=*i*-1 are acknowledged; the expected next has SN=*i*
  - W=*j* (in ACK, or piggyback in data segment): permitted to send an addition *j* octets of data

Flow Control (contd.)



A may send 1400 octets

B is prepared to receive 1400 octets, beginning with 1001

A shrinks its transmit window with each transmission

B acknowledges 3 segments (600 octets), but is only prepared to receive 200 additional octets beyond the original budget (i.e., B will accept octets 1601 through 2600)

A adjusts its window with each credit

A exhausts its credit

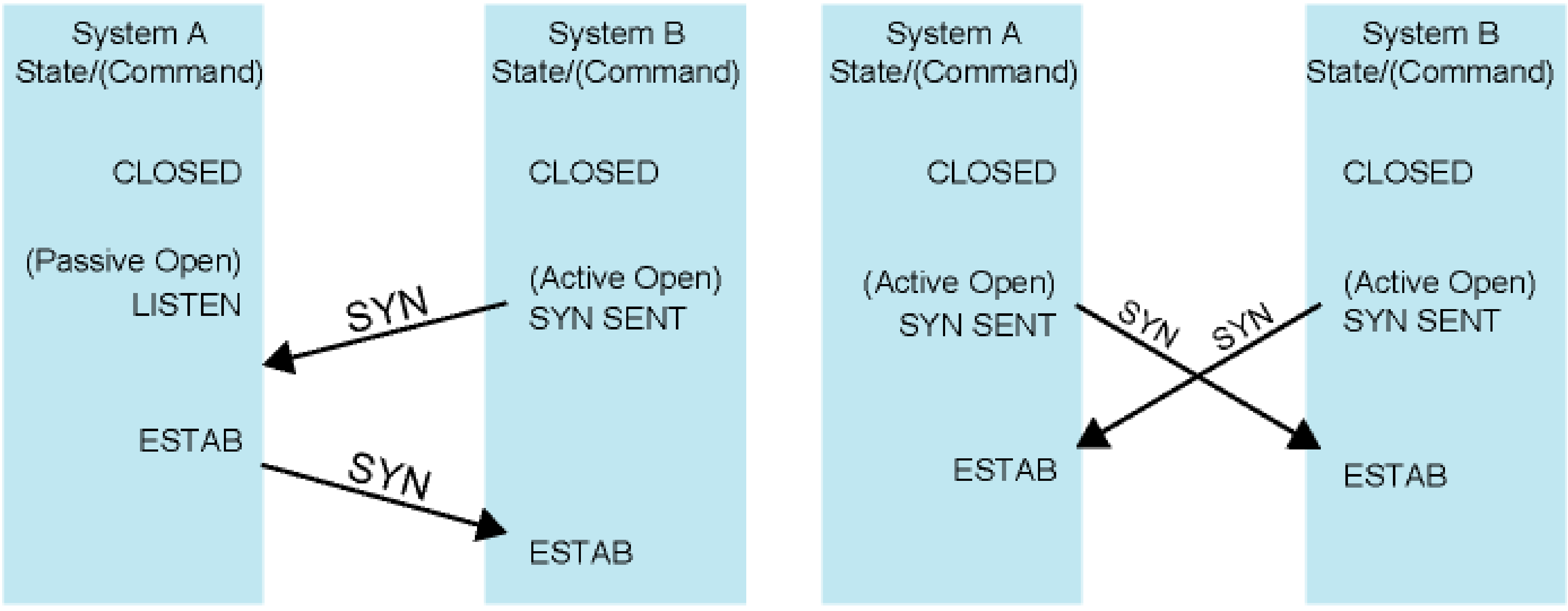
B acknowledges 5 segments (1000 octets) and restores the original amount of credit

A receives new credit

Connection Establishment and Termination

- Purposes
  - Existence of the other end
  - Optional parameter negotiation or exchange (max. segment size, max. window size, QoS)
  - Resource allocation (buffer space)

Connection Establishment and Termination (contd.)



(a) Active/Passive Open

(b) Active/Active Open

To Provide Reliable Service on Unreliable Network Service

- Seven issues need to be addressed
  - Ordered delivery
  - Retransmission strategy
  - Duplication detection
  - Flow control
  - Connection establishment
  - Connection termination
  - Failure recovery

Ordered Delivery

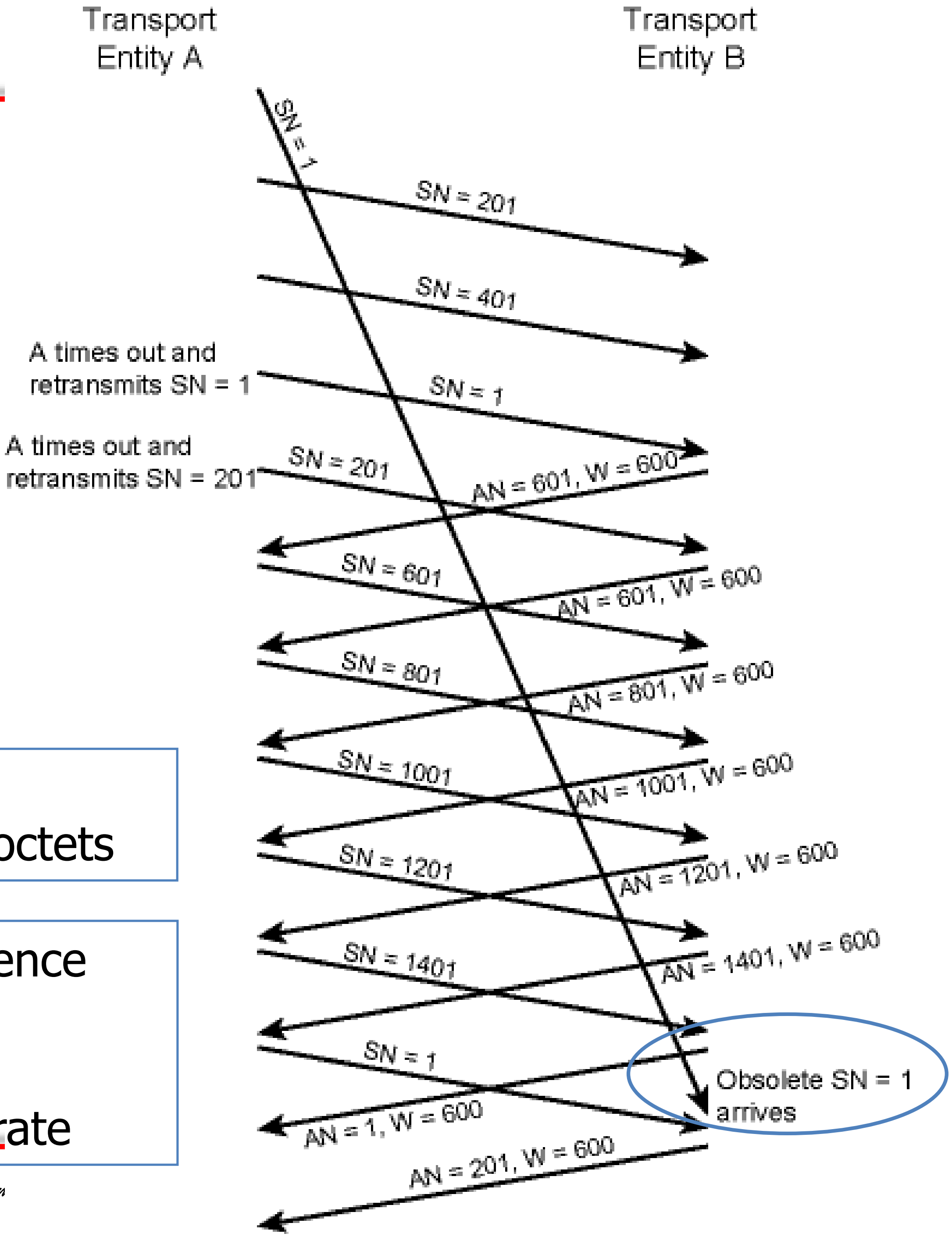
- Segments arrive out of order.
- Number transmitted segments in increasing order.

Retransmission Strategy

- When?
  - Damaged segment
  - Fail-to-arrive segment
- How to do?
  - Positive acknowledgment + timer
- Efficiency consideration → cumulative acknowledgment

Duplicate Detection

- Segment loss → segment retransmission → no confusion
- Segment successfully delivered → ACK loss → duplicate detection triggered
  - A duplicate is received prior to the close of the connection.
    - What should be dealt with?
      - For receiver: ACK loss, ack again.
      - For sender: cannot get confused when receiving multiple ACKs to the same segment.
      - Large sequence number space
  - A duplication is received after the close of the connection.
    - See "connection establishment"



Sequence space: 1600 octets
Initial credit window size: 600 octets

How to choose a suitable sequence space?

(1) Max. packet lifetime
(2) Segment transmission rate

Flow Control

- Segment (AN=*i*, W=*j*)
  - Acknowledge all octets through number *i*-1;
  - Grant credit for an additional *j* octets beginning with octet *i*.
- ACK segment loss
  - Data segment timeout → retransmission → new ACK segment
  - Receiver replies (AN=*i*, W=0) to temporarily close the window;
  - Then receiver sends (AN=*i*, W=*j*) but lost;
  - Deadlock occurs → window timer
  - Window timer is set with each outgoing segment containing AN and W fields.

Connection Establishment

- SYN exchange: two-way handshake
  - What's the problem?
    - Either side SYN loss
    - SYN timer (retransmit-SYN timer)
    - Duplicate SYN detection
    - Connection termination and reestablishment soon

Connection Establishment (contd.)

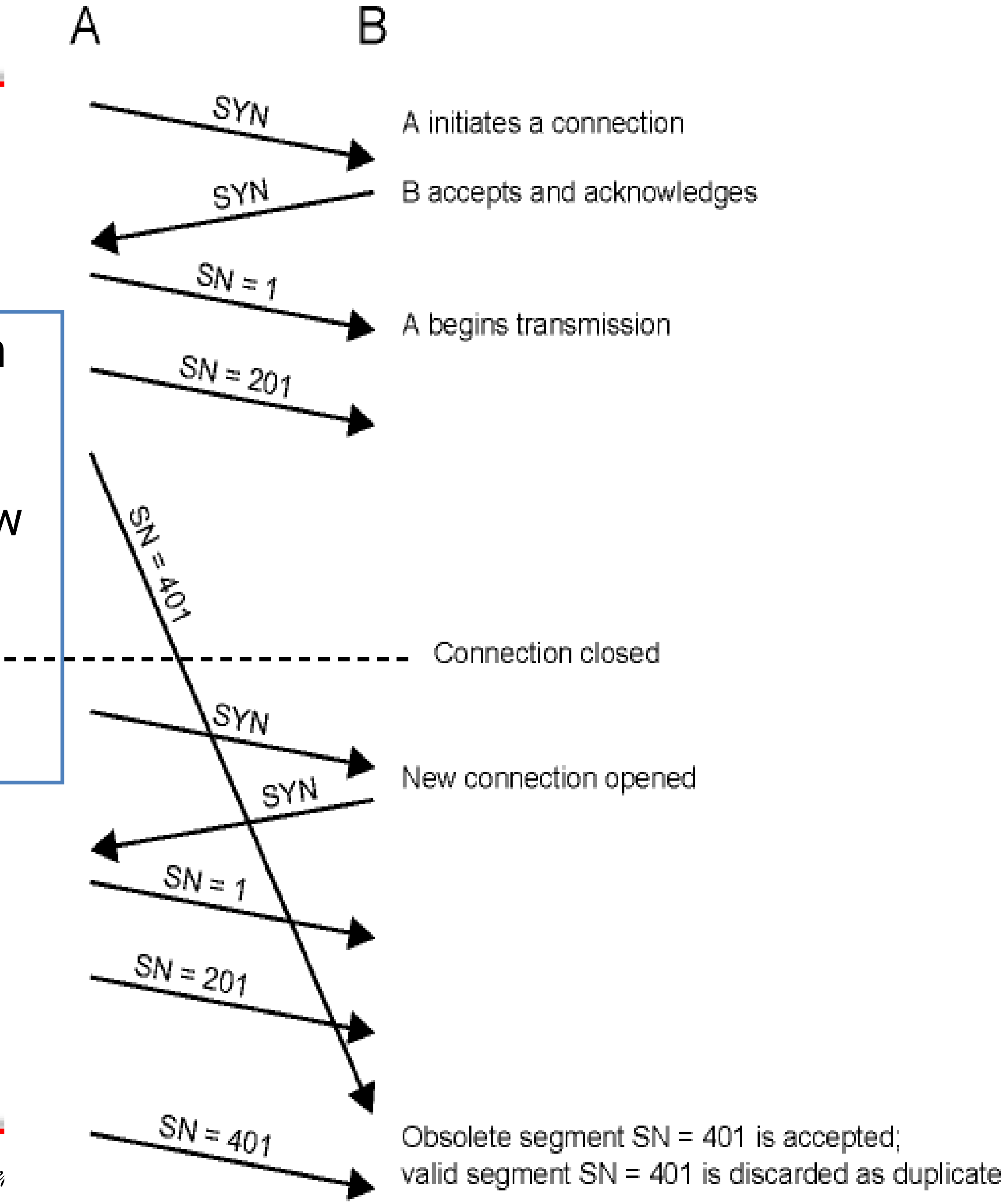
Two-way handshake: problem with obsolete data segment

Initial: data segment sequence number of each new connection: 1

Solution:

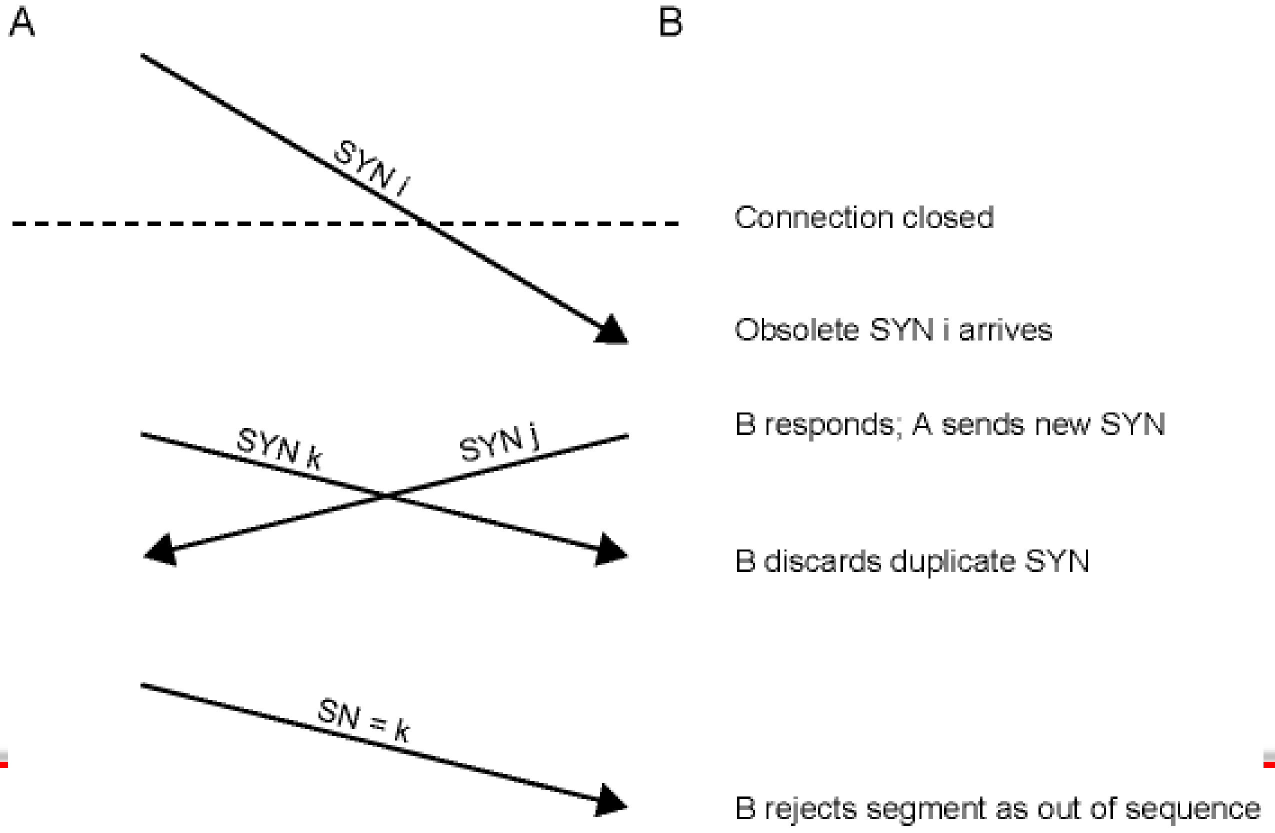
SYN (i);

Data segment: i+1



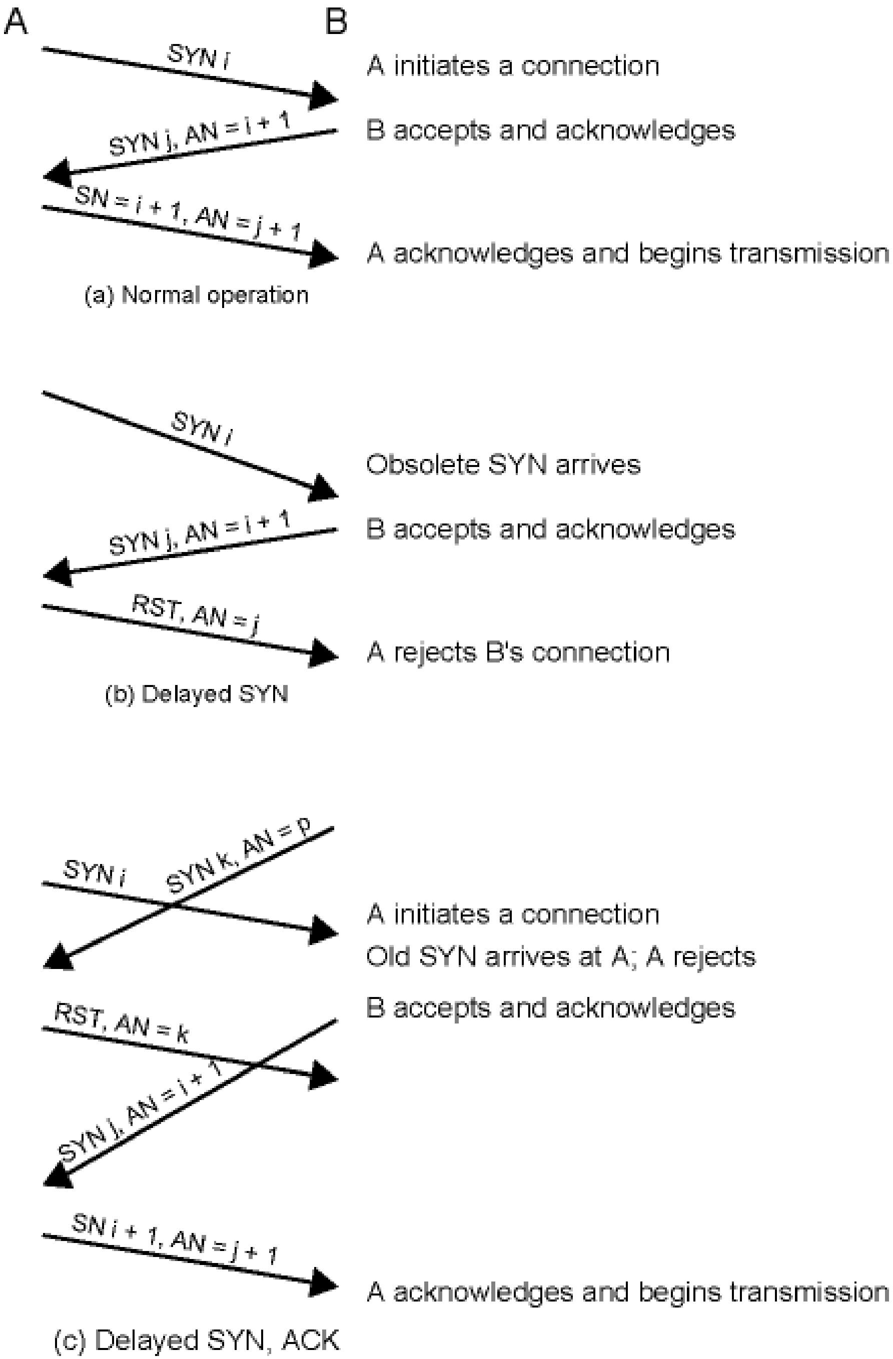
Connection Establishment (contd.)

Two-way handshake: problem with obsolete SYN segments
Solution: three-way handshake



Connection Establishment (contd.)

Examples of three-way handshake



A initiates a connection

B accepts and acknowledges

A acknowledges and begins transmission

(a) Normal operation

Obsolete SYN arrives

B accepts and acknowledges

A rejects B's connection

(b) Delayed SYN

A initiates a connection

Old SYN arrives at A; A rejects

B accepts and acknowledges

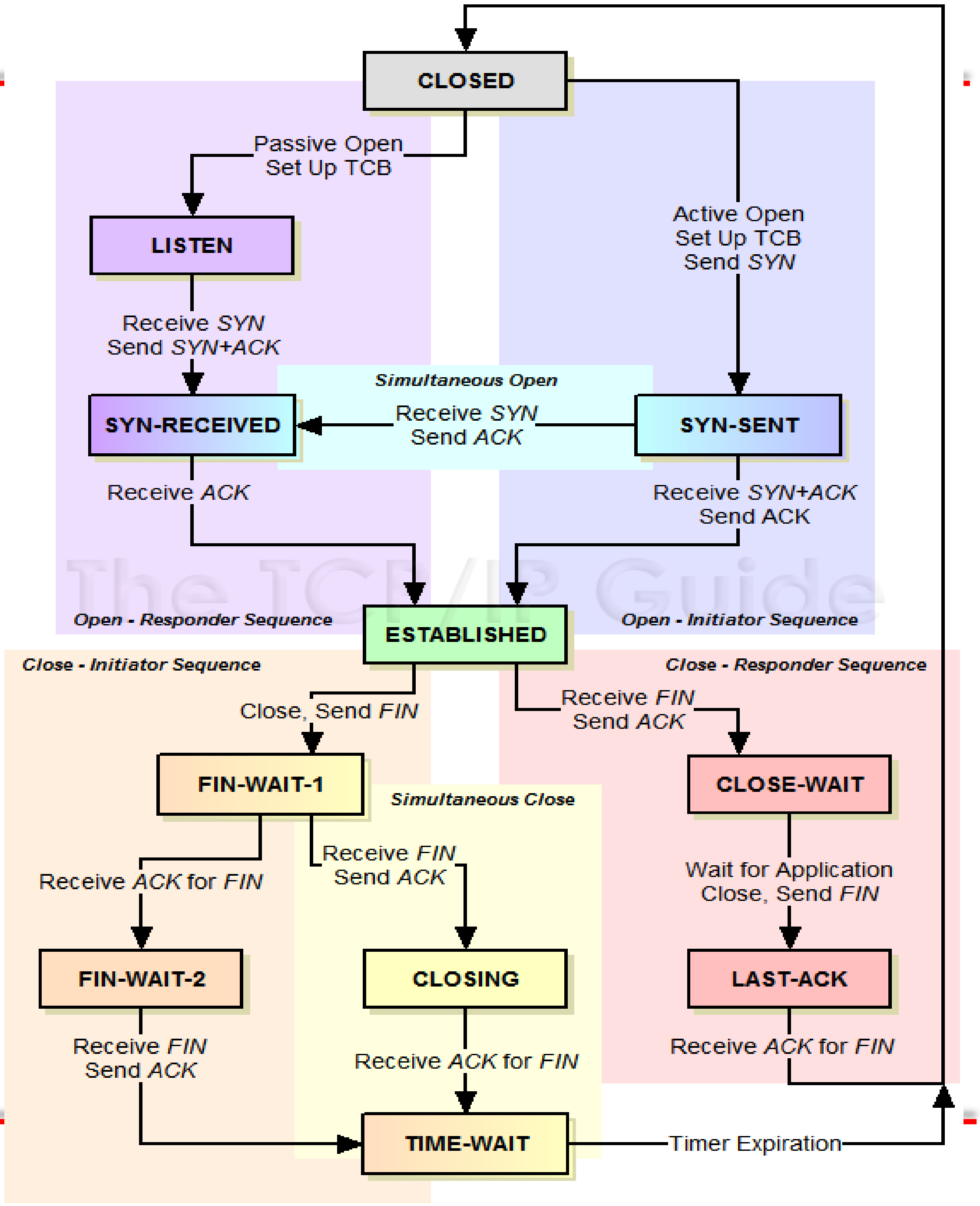
A acknowledges and begins transmission

(c) Delayed SYN, ACK

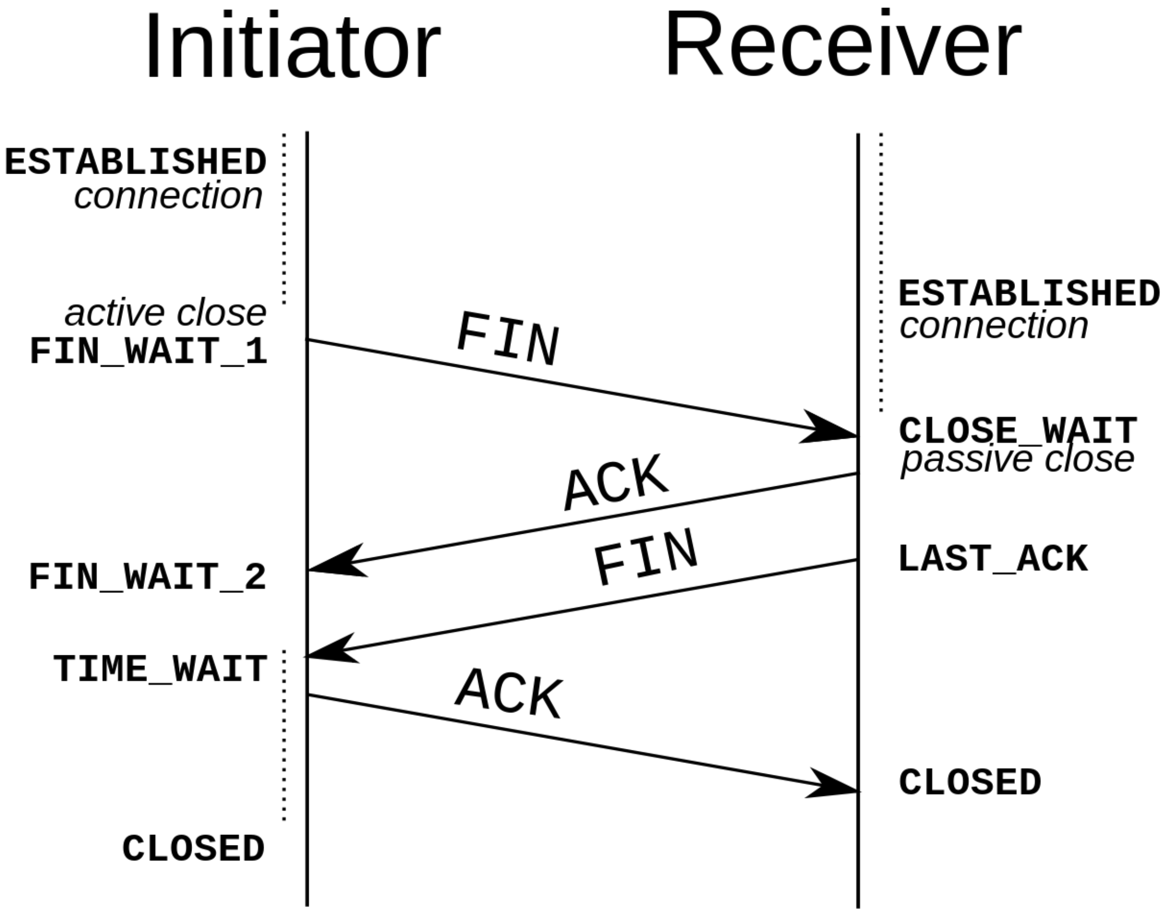
Connection Termination

- FIN
- CLOSE WAIT
- FIN is with a sequence number to deal with the problem of late-arriving segment.

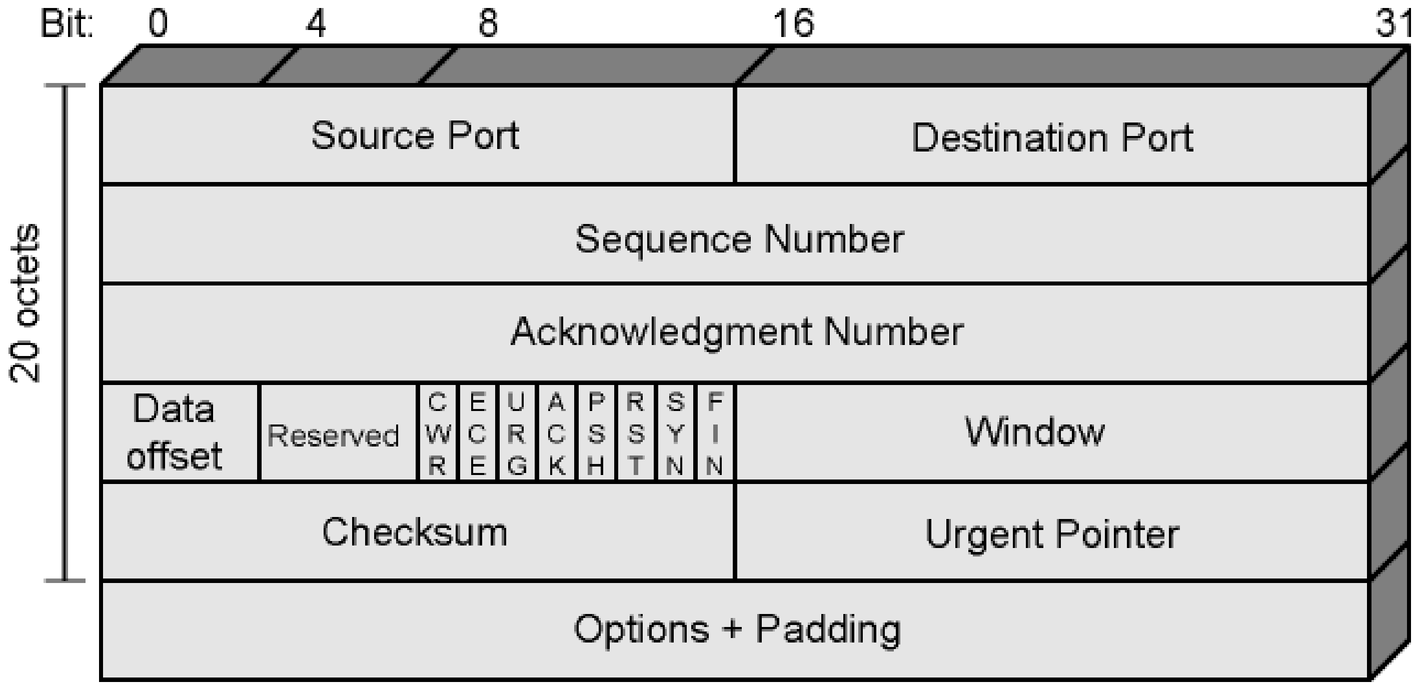
TCP Finite State Machine



TCP Finite State Machine



TCP Header Format



Bit: 0 4 8 16 31

20 octets

Source Port | Destination Port

Sequence Number

Acknowledgment Number

Data offset | Reserved | CWR | ECE | URG | ACK | PSH | RST | SYN | FIN | Window

Checksum | Urgent Pointer

Options + Padding

Options

(1) Max. segment size
(2) Window scale
(3) Sack-permitted
(4) Sack
(5) Timestamps

TCP Mechanisms

- Connection establishment
  - Three-way handshake
  - A connection is uniquely determined by source and destination sockets (host, port).
- Data transfer
  - Each octet is numbered.
  - Segment sequence number is the first octet’s number.
  - No meaning segment triggers RST segment.
- Connection termination
  - Graceful close: FIN
  - Abrupt abort: RST

TCP Implementation Policy Options

- Send policy
  - No transmission till a certain amount of data.
  - PUSH
- Delivery policy
  - Similar to send policy.
- Accept policy
  - In-order only
  - In-window

TCP Implementation Policy Options (contd.)

- Retransmission policy
  - First-only
    - One retransmission timer for the entire queue.
    - Received ACK triggers resetting timer.
    - Timer expires → retransmit the segment at the front of the queue → reset timer.
  - Batch
    - One retransmission timer for the entire queue.
    - Received ACK triggers resetting timer.
    - Timer expires → retransmit the all segments in the queue → reset timer.
  - Individual
    - Each segment has a timer.
    - Timer expires → retransmit the segments → reset timer.

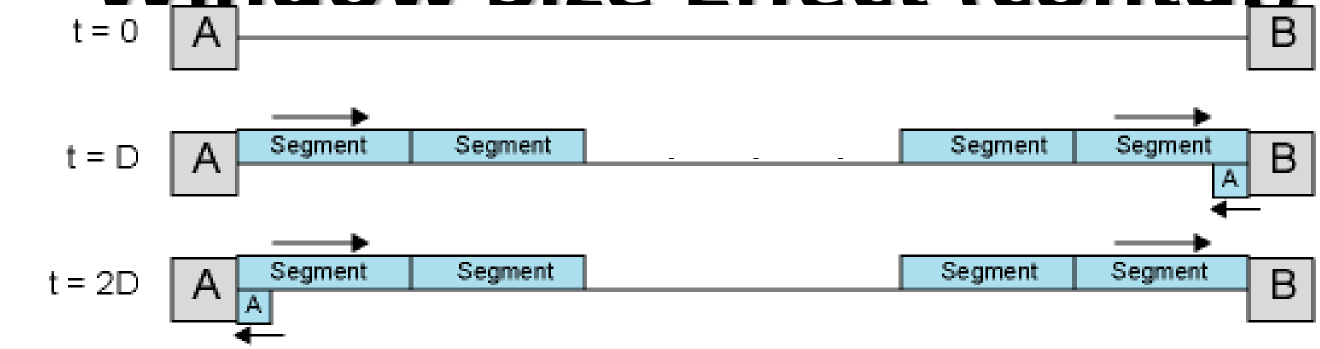
TCP Implementation Policy Options (contd.)

- Acknowledgment policy
  - Immediate
  - Cumulative
    - Timer setting is required.

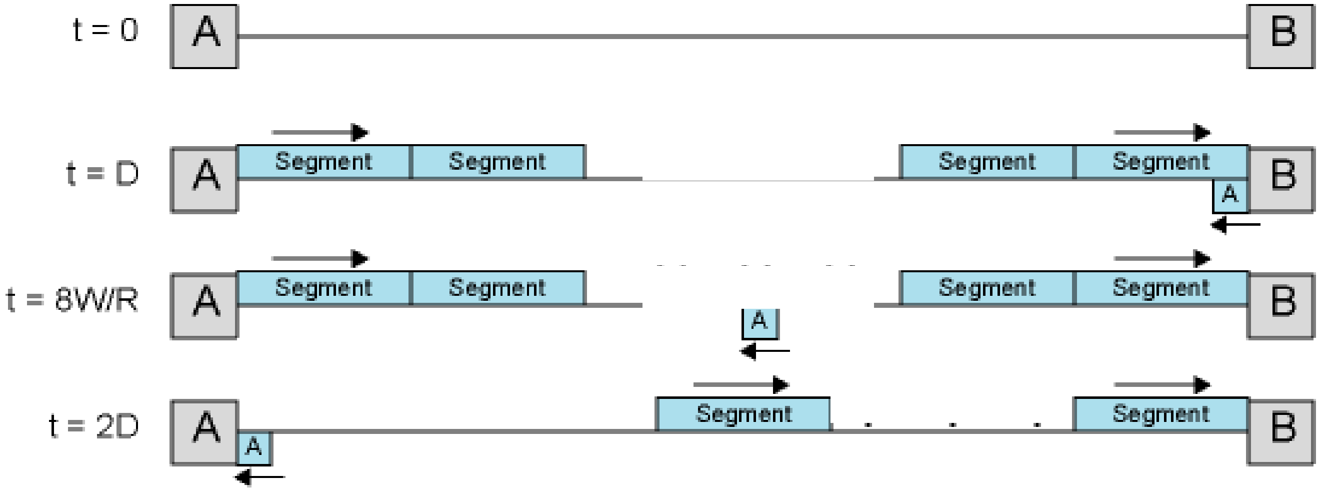
Window Size Effect

- W = TCP window size (octets)
- R = Data rate (bps) at TCP source
- D = Propagation delay (seconds)
- After TCP source begins transmitting, it takes D seconds for first octet to arrive, and D seconds for acknowledgement to return.
- TCP source could transmit at most 2RD bits, or RD/4 octets.

Window Size Effect (contd.)



(a) W > RD/4 (8W/R > 2D)



(b) W < RD/4 (8W/R < 2D)

Retransmission Strategy

- TCP relies on positive acknowledgements.
  - Retransmission on timeout
- No explicit negative acknowledgement
- Retransmission required when:
  - Segment arrives damaged.
    - Checksum error
    - Receiver discards
  - Segment fails to arrive.

TCP Congestion Control

- Dynamic routing can alleviate congestion by spreading load more evenly.
- But only effective for unbalanced loads and brief surges in traffic.
- Congestion can only be controlled by limiting total amount of data entering the Internet.
- ICMP Source Quench message is crude and not effective.
- RSVP may help but not widely implemented.

TCP Flow and Congestion Control

- The rate at which a TCP entity can transmit is determined by rate of incoming ACKs to previous segments with new credit.
- Rate of ACK arrival determined by round-trip path between source and destination.
- Bottleneck may be destination or internet.
- Sender cannot tell which.
- Only the internet bottleneck can be due to congestion.

Window Management

- Slow start (RFC 2581)
- Dynamic window sizing on congestion (RFC 2581)
- Fast retransmit (RFC 2581)
- Fast recovery (RFC 2581)
- Limited transmit (RFC 3042)

Slow Start

- awnd = MIN[credit, cwnd]

  where

  – awnd = allowed window in segments

  – cwnd = congestion window in segments

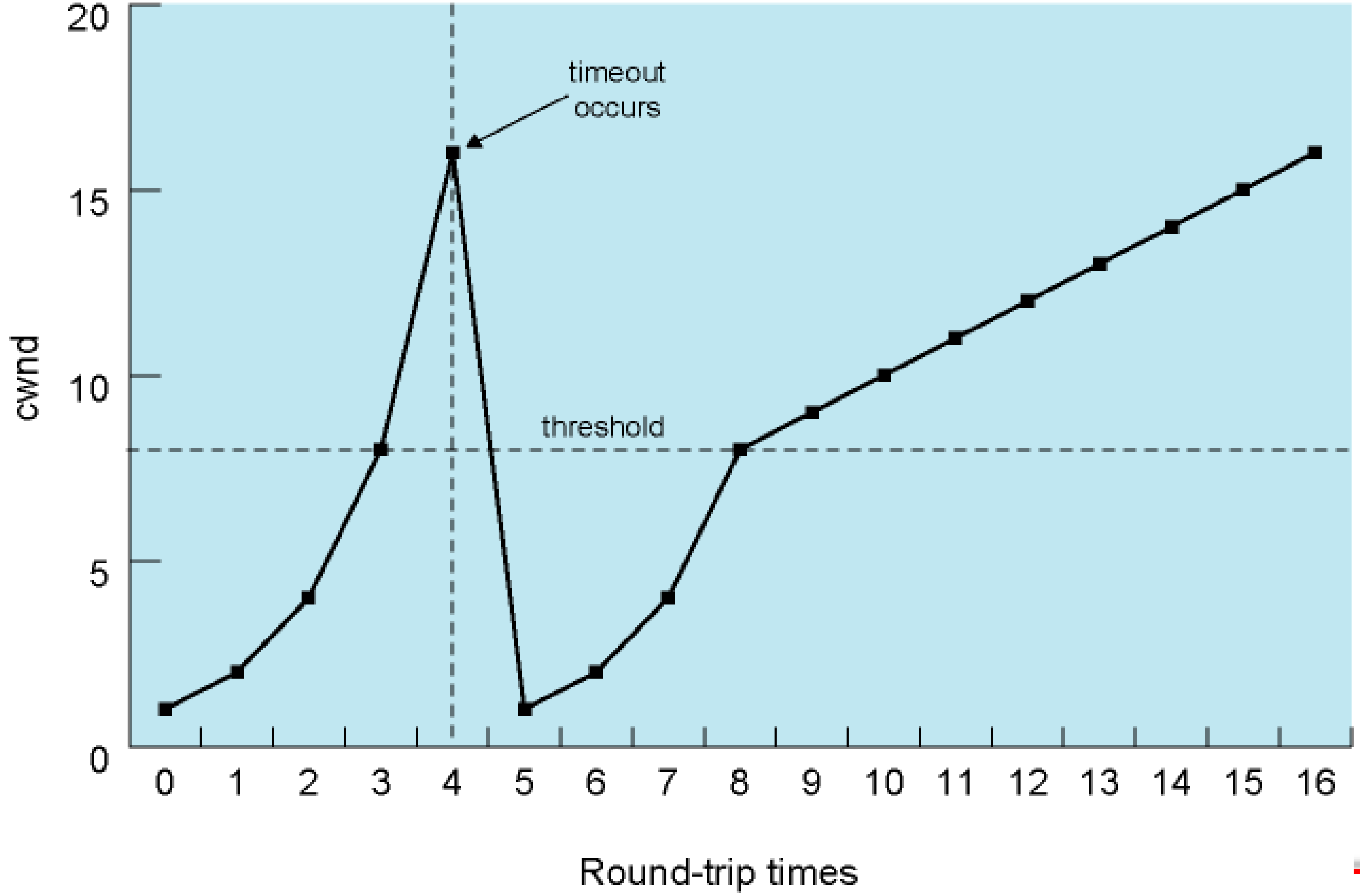
  – credit = amount of unused credit granted in most recent ACK in segments (=window/segment size)

- cwnd = 1 for a new connection and increased by 1 for each ACK received, up to a maximum.

Dynamic Window Sizing on Congestion

- A lost segment indicates congestion.
- Prudent to reset cwsd = 1 and begin slow start process.
- May not be conservative enough: “ easy to drive a network into saturation but hard for the net to recover”.
- Instead, first slow start then followed by linear growth.

Illustration of Slow Start and Congestion Avoidance


cwnd
timeout occurs
threshold
Round-trip times