Transport Protocols & TCP

CSE 3213

Fall 2007

TCP

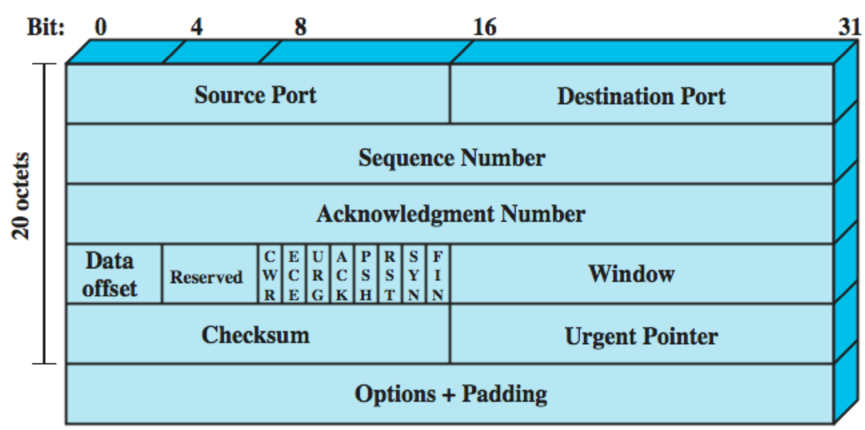
- Services
- Flow control
- Connection establishment and termination
- Congestion control

TCP Services

- 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

3

TCP Header



4

Issues

- ordered delivery,
- retransmission strategy,
- duplication detection,
- flow control,
- connection establishment & termination,
- crash recovery

- Note: since the underlying network is unreliable,
  - segments may get lost
  - segments may arrive out of order

5

Ordered Delivery

- segments may arrive out of order
- hence number segments sequentially
- TCP numbers each octet sequentially
- and segments are numbered by the first octet number in the segment

6

TCP Flow Control

Flow Control

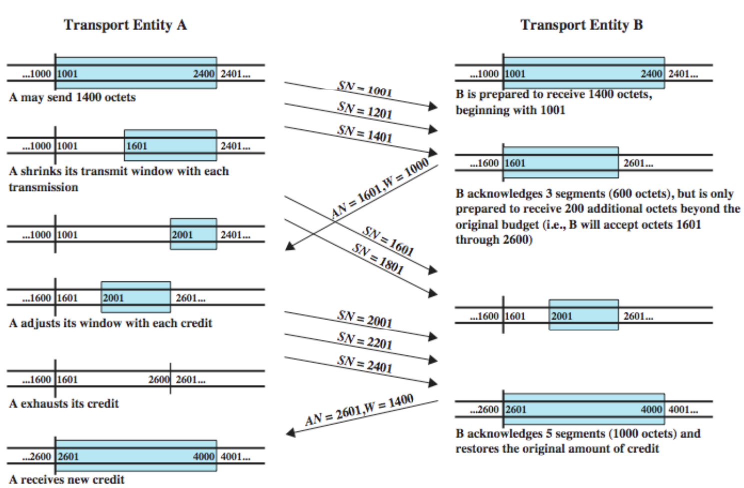
- Fixed sliding window approach
  - works well on reliable networks
  - does not work well on unreliable networks such as IP internet
- Credit scheme
  - more flexible
  - works for IP
  - used in TCP

Credit Scheme

- decouples flow control from ACK
- each octet has sequence number
- each transport segment has seq number (SN), ack number (AN) and window size (W) in header
- sends seq number of first octet in segment
- ACK includes (AN=i, W=j) which means
  - all octets through SN=i-1 acknowledged, want i next
  - permission to send additional window of W=j octets

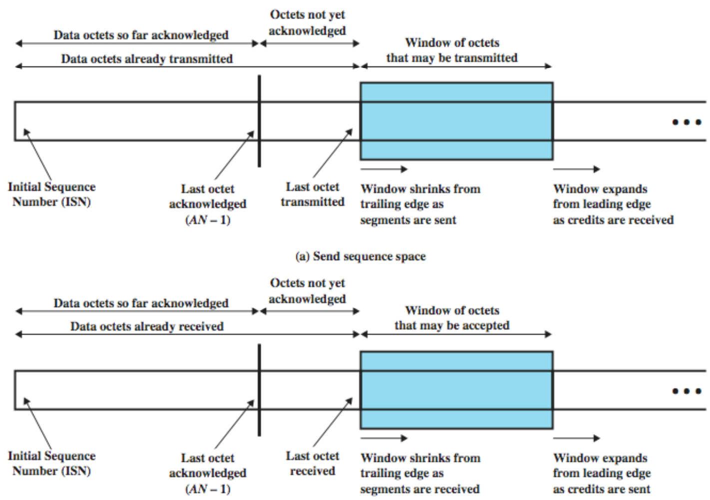
9

Credit Allocation



10

Sending and Receiving Perspectives



(b) Receive sequence space

11

Retransmission Strategy

- retransmission of segment needed because
  - segment damaged in transit
  - segment fails to arrive
- transmitter does not know of failure
- receiver must acknowledge successful receipt
  - can use cumulative acknowledgement for efficiency
- sender times out waiting for ACK triggers re-transmission

12

Retransmit Policy

- TCP has a queue of segments transmitted but not acknowledged
- will retransmit if not ACKed in given time
  - first only - single timer, send the front segment when timer expires, efficient, considerable delays
  - batch - single timer, send all segments when timer expires, has unnecessary retransmissions
  - individual - timer for each segment, lower delay, more efficient, but complex
- effectiveness depends in part on receiver's accept policy

13

Accept Policy

- segments may arrive out of order
- in order
  - only accept segments in order
  - discard out of order segments
  - simple implementation, but burdens network
- in windows
  - accept all segments within receive window
  - reduce transmissions
  - more complex implementation with buffering

14

Acknowledgement Policy

- immediate
  - send empty ACK for each accepted segment
  - simple at cost of extra transmissions
- cumulative
  - piggyback ACK on suitable outbound data segments unless persist timer expires
  - when send empty ACK
  - more complex but efficient

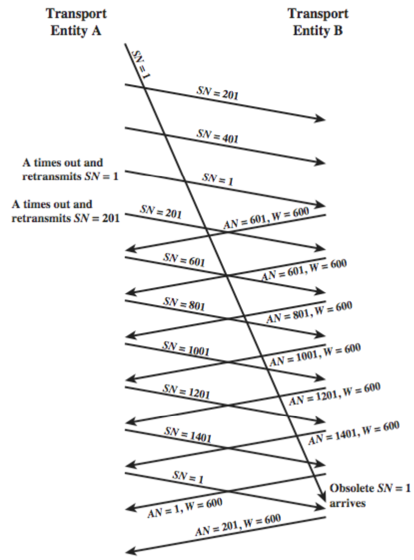
15

Duplication Detection

- if ACK lost, segment duplicated & re-transmitted
- receiver must recognize duplicates
- if duplicate received prior to closing connection
  - receiver assumes ACK lost and ACKs duplicate
  - sender must not get confused with multiple ACKs
  - need a sequence number space large enough to not cycle within maximum life of segment

16

Incorrect Duplicate Detection



17

Connection Establishment and Termination

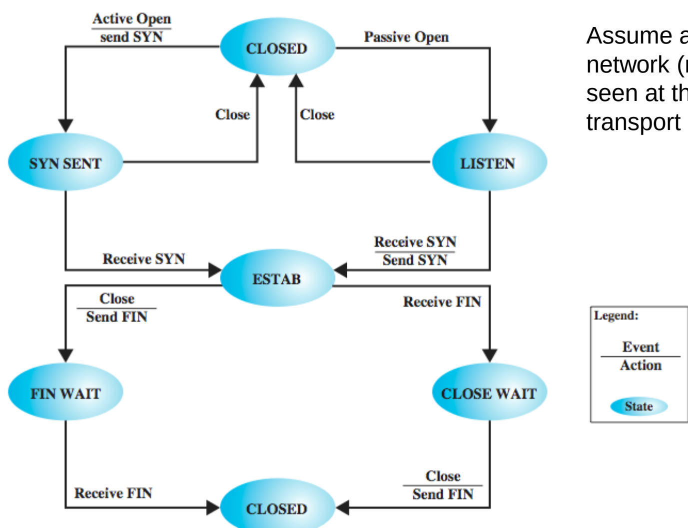
18

Connection Establishment and Termination

- required by connection-oriented transport protocols like TCP
- need connection establishment and termination procedures to allow:
  - each end to know the other exists
  - negotiation of optional parameters
  - triggers allocation of transport entity resources

19

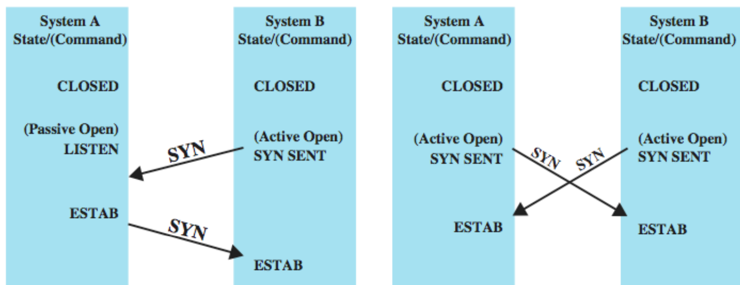
Connection State Diagram



Assume a reliable network (no loss seen at the transport layer).

20

Connection Establishment Diagram



(a) Active/Passive Open

(b) Active/Active Open

Assume a reliable network (no loss seen at the transport layer).

What if either SYN is lost? (discussed later)

21

Connection Termination

- either or both sides by mutual agreement
- graceful or abrupt termination
- if graceful, initiator must:
  - send FIN to other end, requesting termination
  - place connection in FIN WAIT state
  - when FIN received, inform user and close connection
- other end must:
  - when receives FIN must inform TS user and place connection in CLOSE WAIT state
  - when TS user issues CLOSE primitive, send FIN & close connection

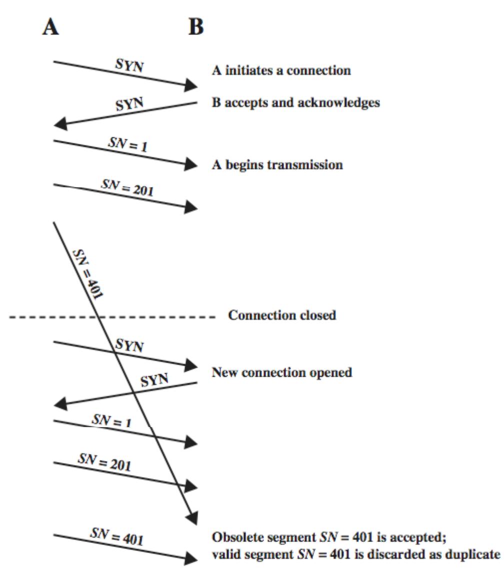
22

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
  - eg. segment from old connection

23

Two Way Handshake: Obsolete Data Segment

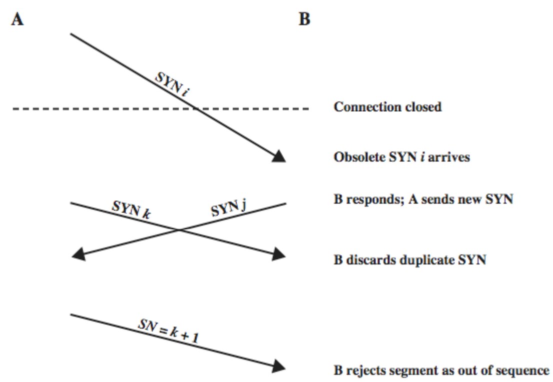


Solution: starting SN is far away from the last SN of the previous connection.

Use request of the form SYN***i*** where *i* +1 is the SN of the first data segment to be sent.

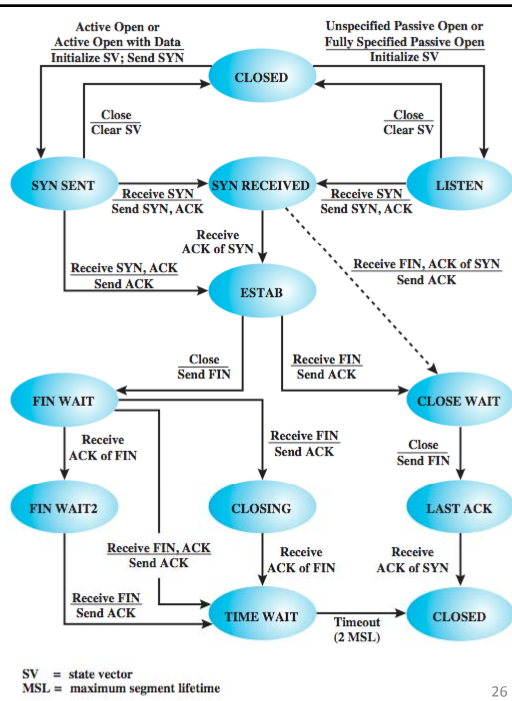
24

Two Way Handshake: Obsolete SYN Segment



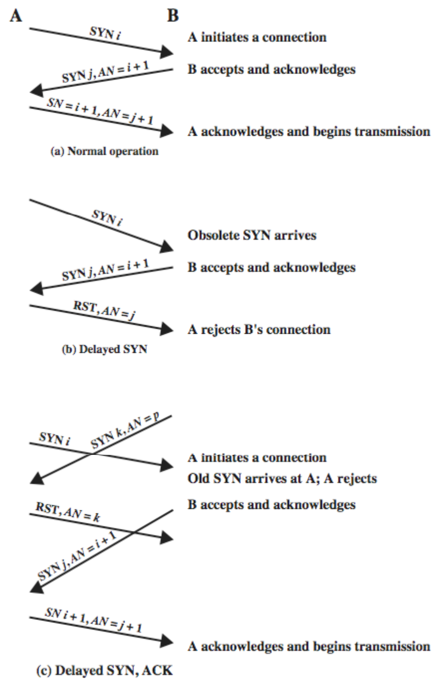
25

TCP Three Way Handshake: State Diagram



26

TCP Three Way Handshake: Examples



27

TCP Connection Establishment: Summary

- three way handshake
  – SYN, SYN-ACK, ACK
- connection determined by source and destination sockets (host, port)
- can only have a single connection between any unique pairs of ports
- but one port can connect to multiple ports

28

Connection Termination (2)

- also need 3-way handshake
- misordered segments could cause:
  - entity in CLOSE WAIT state sends last data segment, followed by FIN
  - FIN arrives before last data segment
  - receiver accepts FIN, closes connection, loses data
- need to associate sequence number with FIN
- receiver waits for all segments before FIN sequence number

29

Connection Termination: Graceful Close

- also have problems with loss of segments and obsolete segments
- need graceful close which will:
- send FIN i and receive AN i+1
- receive FIN j and send AN j+1
- wait twice maximum expected segment lifetime

30

TCP Congestion Control

TCP Congestion Control

- flow control also used for congestion control
  - recognize increased transit times & dropped packets
  - react by reducing flow of data
- RFC's 1122 and 2581 detail extensions
  - Tahoe, Reno and New Reno implementations
- two categories of extensions:
  - retransmission timer management
  - window management

Retransmission Timer Management

- static timer likely too long or too short
- estimate round trip delay by observing pattern of delay for recent segments
- set time to value a bit greater than estimated RTT
- simple average over a number of segments
- exponential average using time series (RFC793)

33

Computing RTT

- Simple average

$$r(K+1) = \frac{1}{K+1}\sum_{i=1}^{K+1} RTT(i)$$

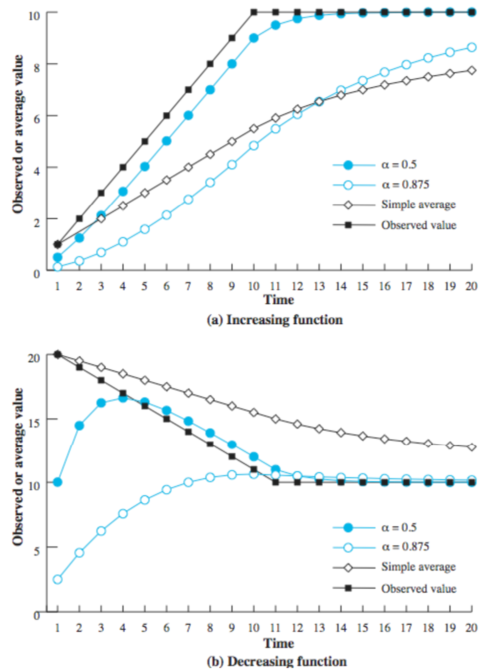
$$r(K+1) = \frac{K}{K+1} r(K) + \frac{1}{K+1} RTT(K+1)$$

- Exponential average

$$r(K+1) = a \times r(K) + (1-a) \times RTT(K+1)$$
$$0 < a < 1$$

34

Use of Exponential Averaging



Exponential RTO Backoff

- timeout probably due to congestion
  - dropped packet or long round trip time
- hence maintaining same RTO is not good idea
- better to increase RTO each time a segment is re-transmitted
  - RTO = q*RTO
  - commonly q = 2 (binary exponential backoff)
  - as in Ethernet CSMA/CD

36

Window Management

- slow start
  - larger windows cause problem on connection created
  - at start limit TCP to 1 segment
  - increase when data ACK, exponential growth
- dynamic windows sizing on congestion
  - when a timeout occurs perhaps due to congestion
  - set slow start threshold to half current congestion window
  - set window to 1 and slow start until threshold
  - beyond threshold, increase window by 1 for each RTT

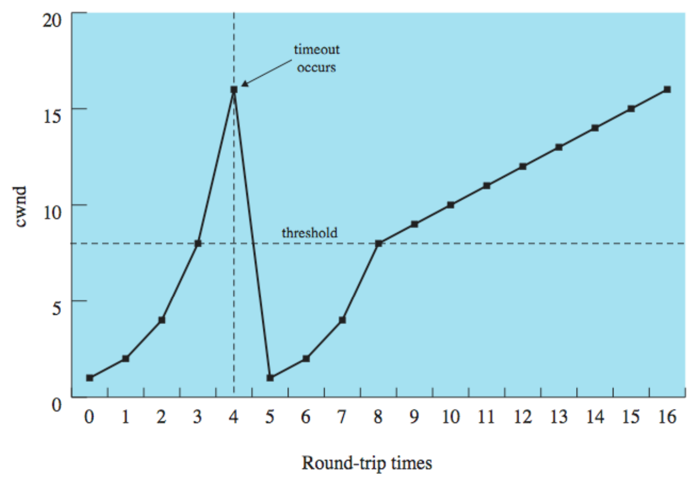
37

Summary

- Assigns a congestion window C_w:
  - Initial value of $C_w = 1$ (packet)
  - If trx successful, congestion window doubled. Continues until C_{max} is reached
  - After $C_w \geq C_{max}$, $C_w = C_w + 1$
  - If timeout before ACK, TCP assumes congestion
- TCP response to congestion is drastic:
  - A random backoff timer disables all transmissions for duration of timer
  - C_w is set to 1
  - C_{max} is set to $C_{max} / 2$
- Congestion window can become quite small for successive packet losses.

38

Window Management



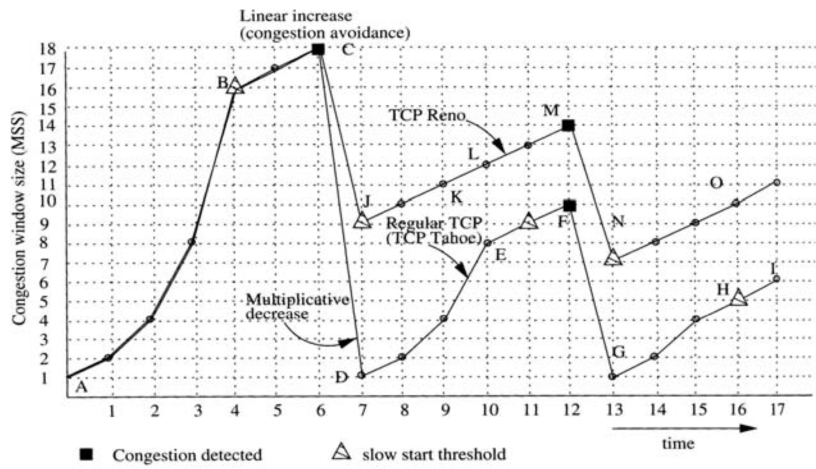
39

Fast Retransmit, Fast Recovery

- retransmit timer rather longer than RTT
- if segment lost TCP slow to retransmit
- fast retransmit
  - if receive a segment out of order, issue an ACK for the last in-order segment correctly received. Repeat this until the missing segment arrives.
  - if receive 4 ACKs for same segment then immediately retransmit (without waiting for timer) since it is likely to have been lost
- fast recovery
  - lost segment means some congestion
  - halve window then increase linearly
  - avoids slow-start

40

Window Management Examples



41

Reading

- Chapter 20, Stallings' book

42