Brief Review of TCP

Dr. Chengzhi Li

March 8, 2005

Transmission Control Protocol (TCP)

- A connection oriented, end-to-end reliable transport protocol
  - Using acknowledgement/retransmission for reliability
- Key assumption
  - Packet loss only due to congestion (buffer overflow)
- End-to-end semantics
  - ACK is sent by the receiver to TCP sender to confirm successful delivery only after the data is obtained

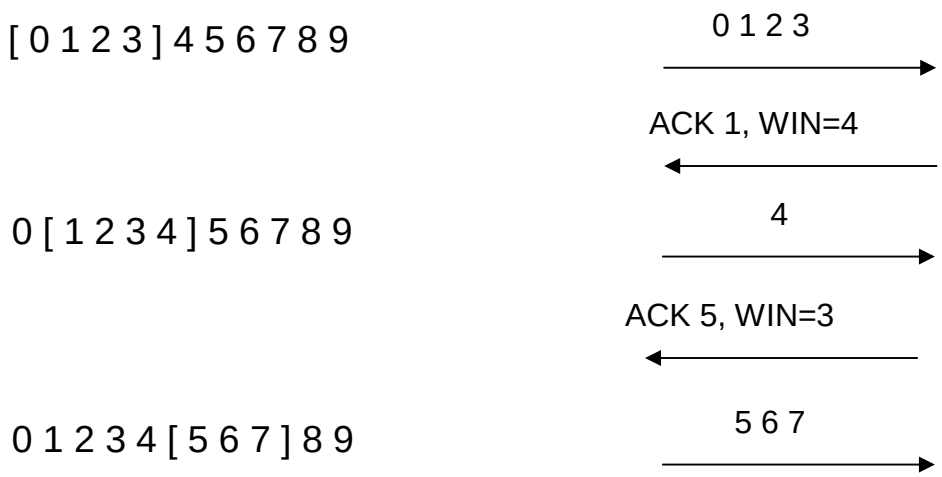
Basic Concepts, Mechanisms, Algorithms and Parameters

- TCP assigns byte sequence numbers for each segment
- Cumulative acknowledgements
  - An ACK acknowledgements bytes up to the first missing byte in the stream
  - A new cumulative acknowledgement is generated only on receipt of a new in-sequence packet
  - A duplicate ACK is generated whenever an out-of-order segment arrives at the receiver
- Indications of packet loss
  - Retransmission time out (RTO)
  - 3 Duplicate ACKs

Basic Concepts, Mechanisms, Algorithms and Parameters

- Sliding window control
  - Maximum range of data sent but not acknowledged
- Average Throughput
  - Average sliding window size/average RTT
- Sliding window size = Minimum { receiver's advertised window, congestion window}
  - receiver's advertised window
    - determined by available buffer space at the receiver
  - congestion window
    - determined by the sender, based on feedback from the network

An Example: Initial Sliding Window Size = 4 bytes



Basic Concepts, Mechanisms, Algorithms and Parameters

- Four parameters
  - Congestion window "cwnd"
    - initial value 1 segment
  - Receiver advertised window
    - Based on receiver buffer size
    - Bound cwnd always
  - Slow start threshold "ssthresh"
    - initial value 64 KB (A. S. Tanenbaum) or 65535 bytes (RFC 2001)
  - Retransmission timer "RTO"
    - RTO = RTT + 4*D
    - RTT = a*pre_RTT + (1-a)*M
    - M = the time taken by ACK
    - a = 7/8
    - D = a*D + (1-a)*|RTT – M|

Basic Concepts, Mechanisms, Algorithms and Parameters

- Four algorithms
  - Slow start
  - Congestion avoidance
  - Fast transmit
  - Fast recovery
- Tahoe TCP
  - Slow start + congestion avoidance + fast transmission
- Reno TCP
  - Slow start + congestion avoidance + fast transmission + fast recovery

Slow Start

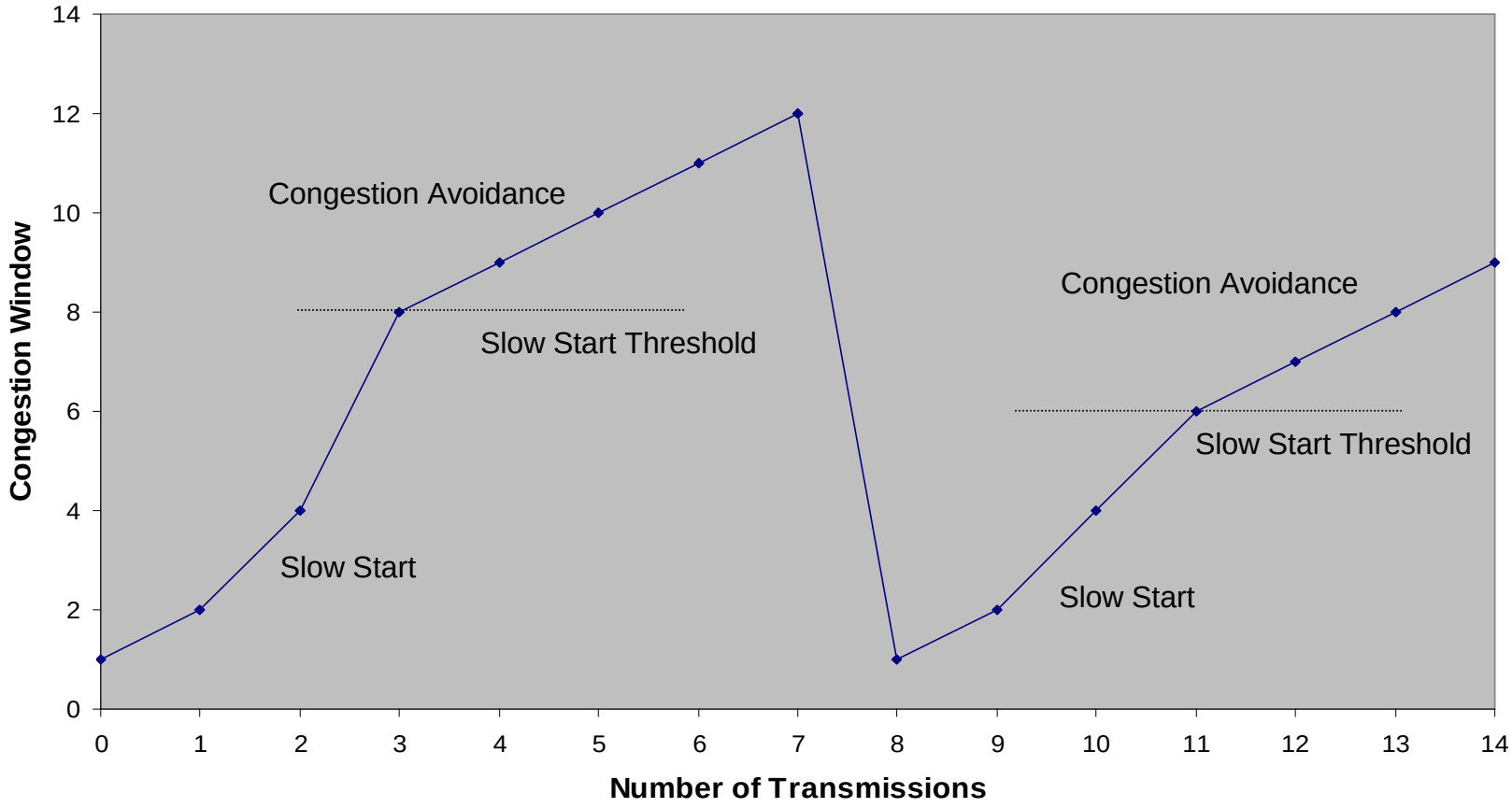
- Goal: to fully exploit network resources
- Slow start procedure is triggered at
  - the beginning of the TCP connection
  - each time a packet loss is detected
- Congestion window "cwnd"
  - set by sender to one segment at the beginning
  - increased by one segment for every ACK (exponential growth of cwnd) till cwnd = ssthresh (entering congestion avoidance)
- Exponential backoff
  - For every timeout, RTO = 2 * RTO, upto 64 sec

Congestion Avoidance

- Goal: to prevent network from being overloaded
- When an ACK is received and cwnd >ssthresh
  - cwnd is increased by segmentsize*segmentsize/cwnd (linear growth of cwnd)
- When congestion occurs (timeout or 3 duplicate ACKs)
  - ssthresh = 0.5*cwnd
  - slow start is triggered if timeout occurs (Tahoe TCP & Reno TCP) or 3 duplicate ACKs (Tahoe TCP)
- How to determine TCP in slow start or congestion avoidance
  - If cwnd <=ssthresh **è** slow statr
  - Otherwise **è** congestion advoidance

Example for Slow Start and Congestion Advoidance

TCP Tahoe, Reno

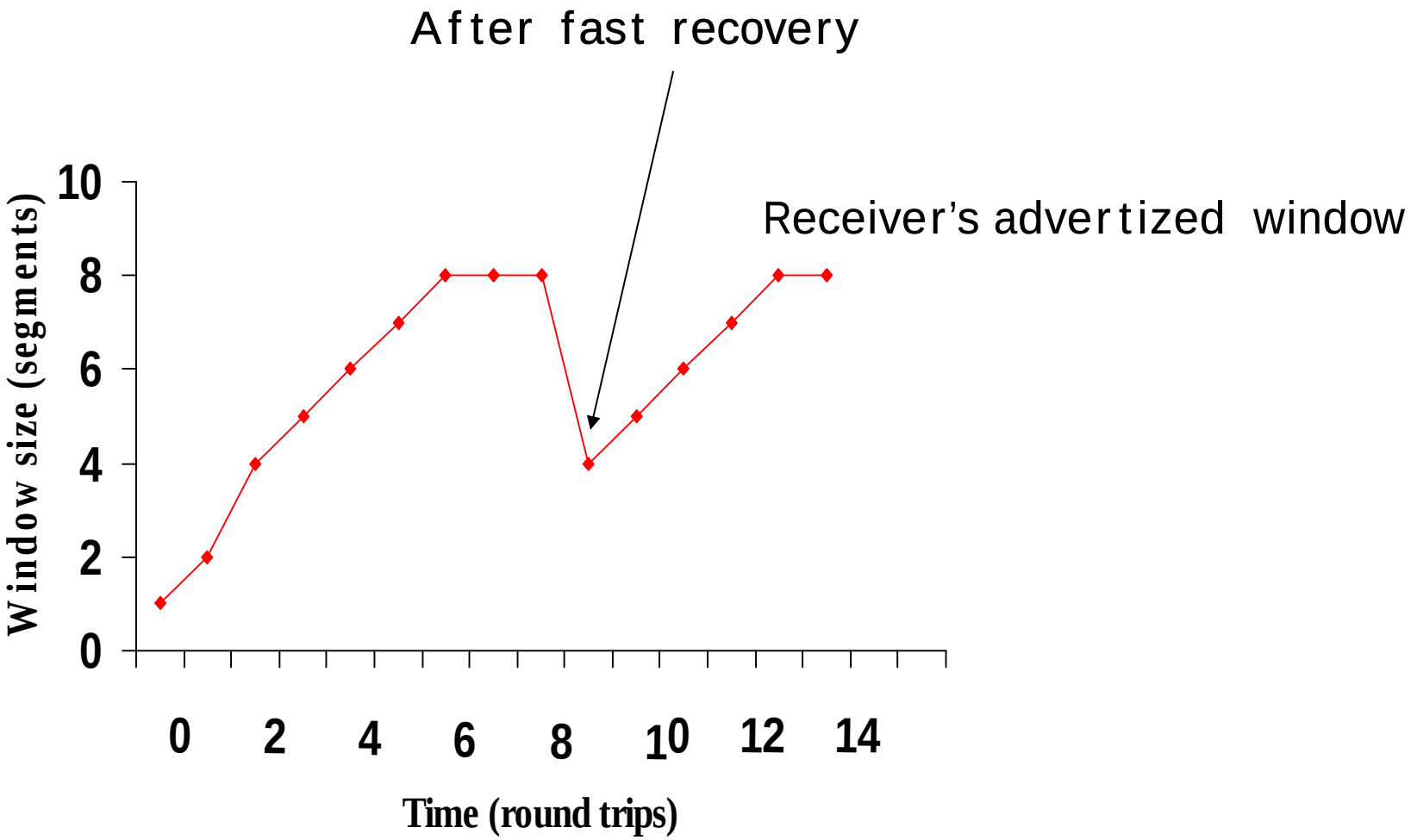


Fast Retransmission

- Goad: quickly response to packet loss
- After 3 duplicate ACKs are received, retransmission for the missing segment is performed without waiting for RTO timeout
- For Tahoe TCP, slow start is triggered
- For Reno TCP, fast recovery is triggered

Fast Recovery

- Goad: to allow high throughput under minor or moderate congestion
- After fast retransmission
  - ssthresh = 0.5*cwnd
  - cwnd = ssthresh + 3 segment size
  - Each time another duplicate ACK arrives, cwnd = cwnd + 1 segment size
  - When an ACK for new data arrives
    - cwnd = ssthresh
    - begin congestion avoidance procedure



After fast retransmit and fast recovery window size is reduced in half

Other Variations of TCP

- TCP New-Reno
  - stay in fast recovery until all packet losses in window are recovered
  - can recover 1 packet loss per RTT without causing a timeout
- Selective Acknowledgements (SACK)
  - provides information about out-of-order packets received by receiver
  - can recover multiple packet losses per RTT

References

1. RFC 2001 – TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery
2. “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP”, K. Fall and S. Floyd, Computer Communication Review, 1995