

TCP

Kai Shen

10/17/2011 CSC 257/457 - Fall 2011 1

TCP: Overview

- **Connection-oriented:**
 - handshaking (exchange of control msgs) to initialize sender, receiver state before data exchange
- **Reliable data transfer:**
 - guaranteed arrival, no error, in order
- **Flow controlled:**
 - sender does not overwhelm receiver
- **Congestion controlled:**
 - sender does not overwhelm the network
- **No delay or bandwidth guarantee.**

10/17/2011 CSC 257/457 - Fall 2011 2

TCP Segment Structure

The diagram shows a 32-bit TCP segment structure. The header is 20 bytes long and contains the following fields:

- source port # (16 bits)
- dest port # (16 bits)
- sequence number (32 bits)
- acknowledgement number (32 bits)
- header flags: head, not used, U, A, P, R, S, F (9 bits)
- Receive window (16 bits)
- checksum (16 bits)
- Urg data pointer (16 bits)
- Options (variable length)
- application data (variable length)

Annotations on the left side:

- URG: urgent data (generally not used)
- ACK: ACK # valid
- PSH: push data now (generally not used)
- RST, SYN, FIN: connection estab (setup, teardown commands)
- Internet checksum

Annotations on the right side:

- counting by bytes of data (not segments!)
- # bytes willing to receive

10/17/2011 CSC 257/457 - Fall 2011 3

Maximum Segment Size (MSS)

- MSS is the maximum TCP segment that'd fit into the link layer frame
- Local MSS
- Path MSS
 - Try and error probing

10/17/2011 CSC 257/457 - Fall 2011 4

TCP Reliable Data Transfer

- TCP provides reliable data transfer service on top of IP's unreliable service
 - Pipelined transmissions
 - Cumulative ACKs
 - When the receiver receives out-of-order segments, it buffers them and re-ACKs the last in-order data
 - Retransmit a single segment at each timeout
 - The sender retransmits at timeout or receiving duplicate ACKs
- Somewhere between Go-back-N and Selective Repeat, closer to which?

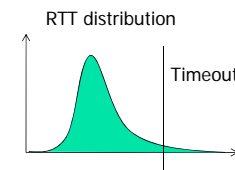
10/17/2011

CSC 257/457 - Fall 2011

5

TCP Timeout

- Q: principles for setting transmission timeout value?
- too short: premature timeout and unnecessary retransmissions
 - too long: slow reaction to segment loss
 - longer than normal RTT (round trip time)
 - one challenge is that RTT varies



10/17/2011

CSC 257/457 - Fall 2011

6

Expected Round Trip Time

Derive expected RTT from past RTT measurement.

- Basic measurement: measured time from segment transmission until ACK receipt
- **Stability:** RTT fluctuates, we want to avoid instability (premature reaction to short-term spikes)
 - average several recent measurements, not just current RTT
- **Agility:** in case things do change, we want to adjust quickly
 - give more recent measurements higher weight

10/17/2011

CSC 257/457 - Fall 2011

7

EWMA - Exponentially Weighted Moving Average

- influence of past samples decreases exponentially fast

$$\text{ExpectedRTT} = \frac{\text{SampleRTT}_1 + \alpha \cdot \text{SampleRTT}_2 + \alpha^2 \cdot \text{SampleRTT}_3 + \dots}{1 + \alpha + \alpha^2 + \dots}$$

SampleRTT₁ is RTT for the most recent data segment, SampleRTT₂ is RTT for the next recent data segment, etc.

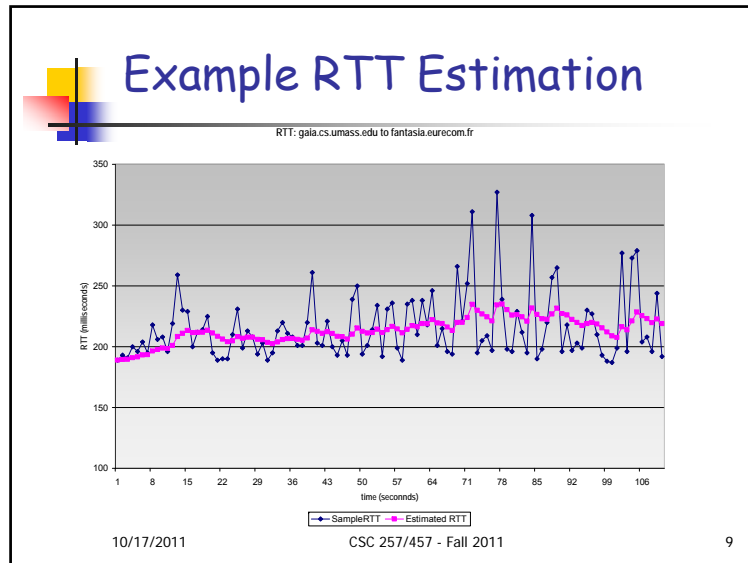
$$\Rightarrow \text{ExpectedRTT} = \alpha \cdot \text{ExpectedRTT}_{\text{last}} + (1 - \alpha) \cdot \text{SampleRTT}_1$$

- typical value in TCP: $\alpha = 0.875$

10/17/2011

CSC 257/457 - Fall 2011

8



TCP Timeout

Setting the timeout:

- ExpectedRTT plus "safety margin"
 - larger variation in RTT → larger safety margin
 - $\text{TimeoutInterval} = \text{ExpectedRTT} + 4 * \text{DevRTT}$
- DevRTT also estimated using EWMA of past measurements

10/17/2011 CSC 257/457 - Fall 2011 10

TCP Sender Events and Processing

Data ready to send:

- create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer
- timeout value: we just decided it!!

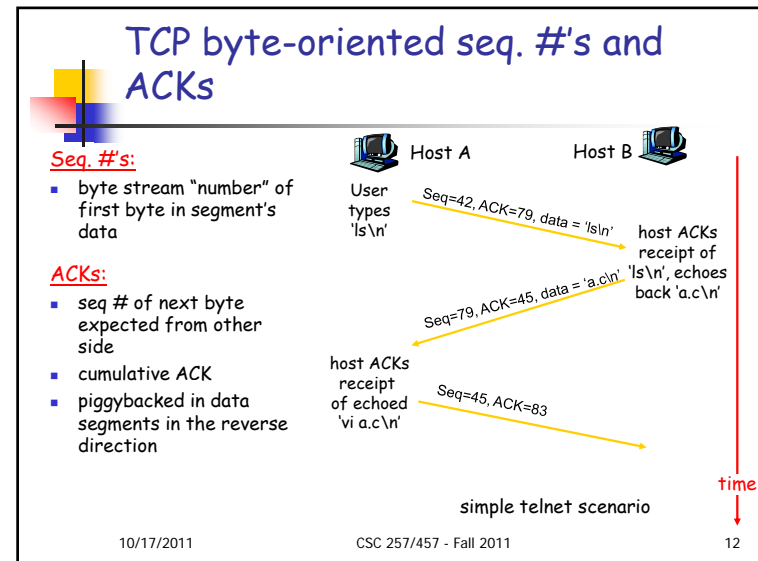
Timeout:

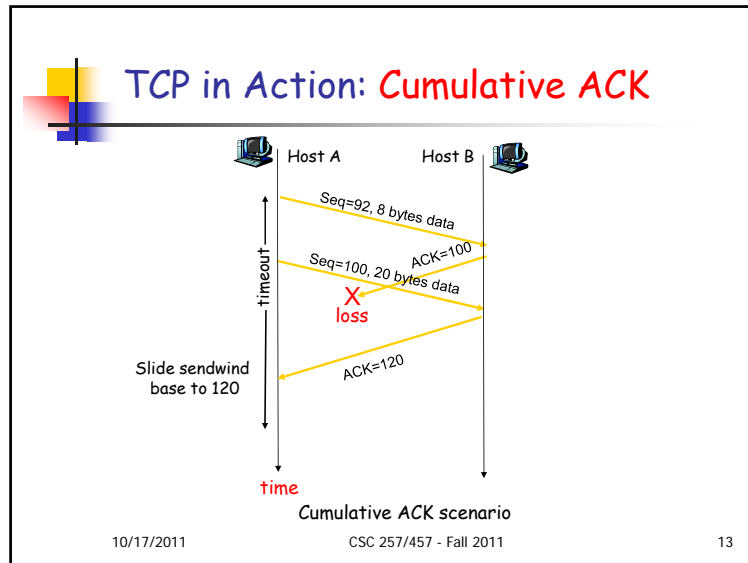
- retransmit segment that caused timeout
- restart timer

ACK rcvd:

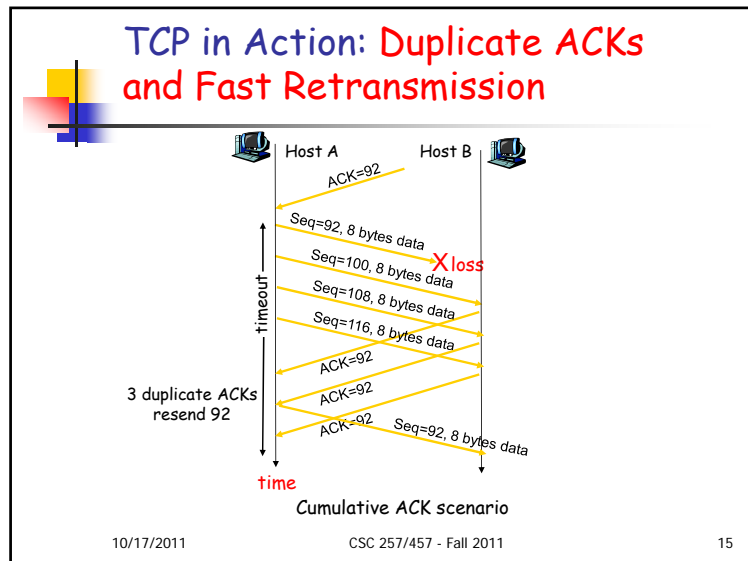
- slide sender window if acknowledges previously unacked segments
- retransmit if 3 duplicate ACKs

10/17/2011 CSC 257/457 - Fall 2011 11





- ### Fast Retransmission
- Time-out period often relatively long:
 - long delay before resending lost packet
 - When receiver receives out-of-order segments, it re-ACKs the last in-order byte
 - If sender receives 3 ACKs for the same data, it supposes that segment after ACKed data was lost:
 - **fast retransmission**: resend segment before timer expires, restart timer
- 10/17/2011 CSC 257/457 - Fall 2011 14



- ### Outline
- segment structure
 - reliable data transfer
 - **flow control**
 - connection management
- 10/17/2011 CSC 257/457 - Fall 2011 16

TCP Flow Control

- Receive side of TCP connection has a receive buffer:
- App process may be slow at reading from buffer.

The diagram shows a horizontal bar representing the receive buffer. It is divided into two sections: a blue section on the left labeled 'spare room' and a hatched section on the right labeled 'TCP data in buffer'. Above the bar, a double-headed arrow labeled 'RevWindow' spans the entire width. Below the bar, a double-headed arrow labeled 'RevBuffer' also spans the entire width. An arrow labeled 'data from IP' points into the 'spare room' section from the left. An arrow labeled 'application process' points out from the 'TCP data in buffer' section to the right.

Flow Control:

- Receiver advertises spare room by including value of `RecvWindow` in segments
- Sender limits unACKed data to `RecvWindow`, therefore guarantees receive buffer doesn't overflow

10/17/2011 CSC 257/457 - Fall 2011 17

TCP Connection Management

- Establishment:**
 - TCP sender, receiver establish "connection" before exchanging data segments
 - initialize TCP variables: starting seq. #s, MSS, buffers, flow control info (e.g. `RecvWindow`)
- Teardown:**
 - free up resources after mutually close

10/17/2011 CSC 257/457 - Fall 2011 18

TCP Connection Establishment

Three-way handshake:

- Step 1:** client (active open) sends TCP SYN segment to server
 - specifies initial seq #
 - no data
- Step 2:** server (passive open) host receives SYN, replies with SYNACK segment
 - server allocates buffers
 - specifies server initial seq. #
- Step 3:** client receives SYNACK, replies with ACK segment, which may contain data

The diagram shows a vertical line for the 'client' and a vertical line for the 'server'.
 1. A yellow arrow labeled 'SYN, seq=x, no data' goes from the client to the server.
 2. A yellow arrow labeled 'SYN, seq=y, ACK=x+1, no data' goes from the server to the client.
 3. A yellow arrow labeled 'seq=x+1, ACK=y+1, maybe data' goes from the client to the server.
 A 'connection request' label is placed near the first arrow.

10/17/2011 CSC 257/457 - Fall 2011 19

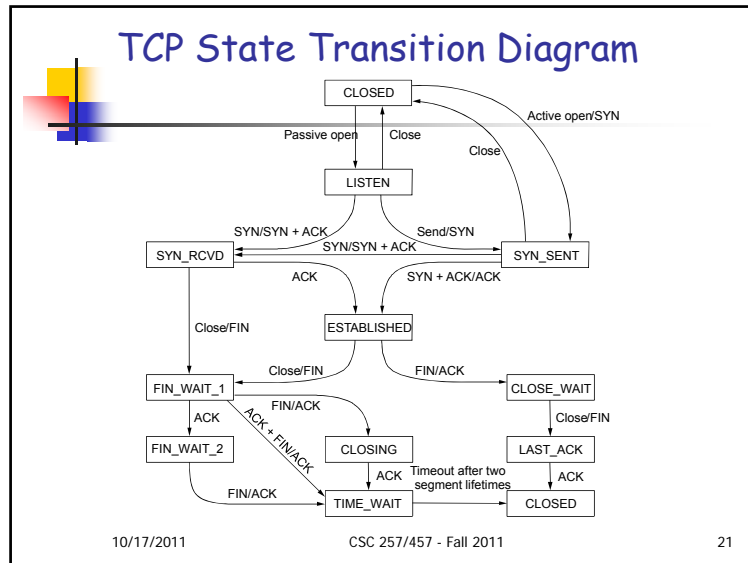
TCP Connection Teardown

Closing a connection:
close socket: `close(sockfd)`

- Step 1:** A (active closing host) sends TCP FIN control segment.
- Step 2:** B (passive closing host) receives FIN, replies with ACK. Closes connection, sends FIN.
- Step 3:** A receives FIN, replies with ACK. Enters "timed wait" - resend ACK in case it is lost.
- Step 4:** B receives ACK. Connection closed.

The diagram shows two vertical lines for hosts 'A' and 'B'.
 1. A yellow arrow labeled 'FIN' goes from A to B.
 2. A yellow arrow labeled 'ACK' goes from B to A.
 3. A yellow arrow labeled 'FIN' goes from B to A.
 4. A yellow arrow labeled 'ACK' goes from A to B.
 A vertical bracket on the A side is labeled 'timed wait'.
 The state of A is labeled 'close' at the top and 'closed' at the bottom. The state of B is labeled 'close' at the top and 'closed' at the bottom.

10/17/2011 CSC 257/457 - Fall 2011 20



Disclaimer

- Parts of the lecture slides contain original work of James Kurose, Larry Peterson, and Keith Ross. The slides are intended for the sole purpose of instruction of computer networks at the University of Rochester. All copyrighted materials belong to their original owner(s).

10/17/2011 CSC 257/457 - Fall 2011 22