CS118 Discussion 1B

TCP, Quiz 1 Review

Week 5

Zhiyi Zhang (zhiyi@cs.ucla.edu)

@myself record

Contents

- TCP timer
- TCP connection
- Quiz 1 review
- TCP congestion control (if we have time)

TCP Timer

TCP Timer Calculation

- EstimatedRTT, DevRTT, and TimeoutInterval
  - EstimatedRTT = (1- a)*EstimatedRTT + a*SampleRTT
    - Alpha = 0.125
  - DevRTT = (1-b)*DevRTT + b*|SampleRTT-EstimatedRTT|
    - Beta = 0.25
  - TimeoutInterval = EstimatedRTT + 4*DevRTT

- For every received packet
  - 1. Calculate SampleRTT
  - 2. Calculate Estimated RTT
  - 3. Calculate DevRTT
  - 4. Calculate TimeoutInterval

How TCP Timer works?

- Loop forever:
  - Event: socket API receives data from application
    - Start timer for this segment if it’s not running now
  - Event: receive ACK from the network
    - Start timer for the next not-yet-acked segment if any
  - Event: time out
    - Start timer for the same segment again
- Reading: textbook 3.33 Pseudo code of TCP timer

Homework 4 Q1

Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 326. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 327, the source port number is 40200, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A. Fill in the blanks for questions (a) – (c) directly; work out the diagram in the box for question (d).

(a) In the second segment sent from Host A to B, the sequence number is ______, source port number is ______, and destination port number is ______.

(b) If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, the ACK number is ______, the source port number is ______, and the destination port number is ______.

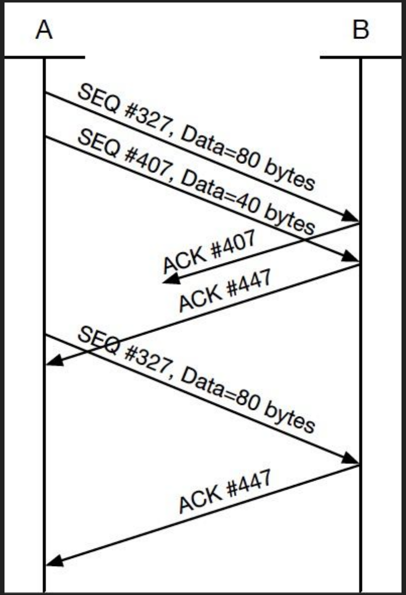
(c) If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, the ACK number is ______.

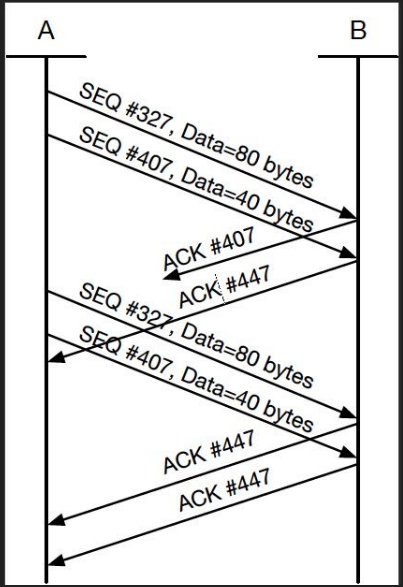
(d) Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after A's timeout intervals right after the time out interval of the first segment. Draw a timing diagram in the box below, showing these segments and acknowledgments until A receives all the acknowledgments of re-transmitted packets. Assume no additional packet loss. For each segment in your diagram, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the ACK number.

- Original (d)'s statement "after the time out interval for both the first and second packet" is incorrect
- Because there is only one timer for the oldest data and won't retransmit the second segment

Homework 4 Q1

- Therefore the right figure is impossible in TCP. (but possible in GBN)
- Reading: Textbook section 3.5.3 last subsection
  - Go-Back-N or Selective Repeat? Is TCP GBN or SR?





Sequence Number Calculation

Why we need sequence number?

- To avoid duplicate re-transmission
- Go-back-N:
  - Sender window: n; Receiver window: 1; # of sequences required: n+1
  - Reason: When A sends n pkts, B receives them all but all acks are lost, A will retransmit seq 1 to n and B is expecting seq = n+1
    - If # of sequence == n, B cannot distinguish the duplicate pkt because n+1 = 1
    - If # of sequence == n+1, B can distinguish because n+1 != 1
- Selective Repeat (Similar to TCP)
  - Sender window: n; Receiver window: m (m<=n); # of sequences required: n+m
  - Reason: When A sends n pkts, B receives n pkts but all acks are lost, A will retransmit seq 1 to n and B is expecting seq = n+1 to n+m
    - If # of sequence == n + m -1, B cannot distinguish the duplicate pkt because n+m = 1
    - If # of sequence == n + m, B can distinguish the duplicate pkt because n+m != 1

Quiz 1

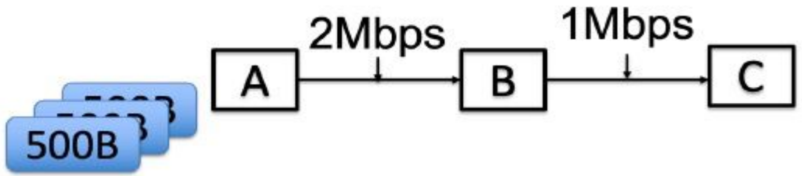
Debug the code: Question 3

```
#include <server.h> /* assume all headers are included correctly */
#define PORT 8080
int main(int argc, const char *argv[])
{
int server_fd, new_fd; /* listen on server_fd, new connection on new_fd */
struct sockaddr_in address;
int addrlen = sizeof(address);
char buffer[1024] = {0};
char *hello = "Hello from server";
if ((server_fd = socket(AF_INET => PF_INET, SOCK_STREAM, 0)) > 0) {
perror("socket failed");
exit(EXIT_FAILURE);
}
address.sin_family = PF_INET; => AF_INET
address.sin_addr.s_addr = INADDR_ANY; => htonl(INADDR_ANY)
address.sin_port = ntohs(PORT); => htons(PORT)
setzero(rest of address)
if (bind(server_fd, (struct sockaddr *)&address, addrlen) < 0) {
perror("bind failed");
exit(EXIT_FAILURE);
}
if (accept(server_fd, 3) < 0) { => listen
exit(EXIT_FAILURE);
}
if ((new_fd = listen(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) { =>accept
exit(EXIT_FAILURE);
}
int valread = read(server_fd, buffer, 1024); => new_fd
printf("%s\n", buffer); valread into
sendto(new_fd, hello, strlen(hello), 0);
printf("Hello message sent\n");
}
```

Question 4

- Tx delay
  - A: 2ms
  - B: 4ms
- Prop delay
  - A-B: 1ms
  - B-C: 1ms

Problem 4 (8 points): Consider sending 3 packets from Node A to Node C via Node B (see the figure below). The packet length is 500 bytes each. The propagation delay of both Link A-B and link B-C is 1 msec (0.001 second). Link A-B's bandwidth is 2Mbps (2×10^6 bits per second), and link B-C's bandwidth is 1Mbps. Assume A starts transmitting the first packet at time t = 0, and no packets are



buffered at B at t = 0.

1. (4 points) What is the time gap between the first and second packets when they arrive at C? (i.e., the time gap between receiving the last bit of the first packet and the last bit of the second packet)

- Packet 1
  - Leave A: 0+2=2ms
  - Arrive B: 2+1=3ms
  - Leave B: 3+4=7ms
  - Arrive C: 7+1=8ms
- Packet 2
  - Leave A: **2**+2=4ms
  - Arrive B: 4+1=5ms
  - Leave B: **7**+4=11ms
  - Arrive C: 11+1=12ms
- Packet 3
  - Leave A: **4**+2=6ms
  - Arrive B: 6+1=7ms
  - Leave B: **11**+4=15ms
  - Arrive C: 15+1=16ms

TCP Connection Management

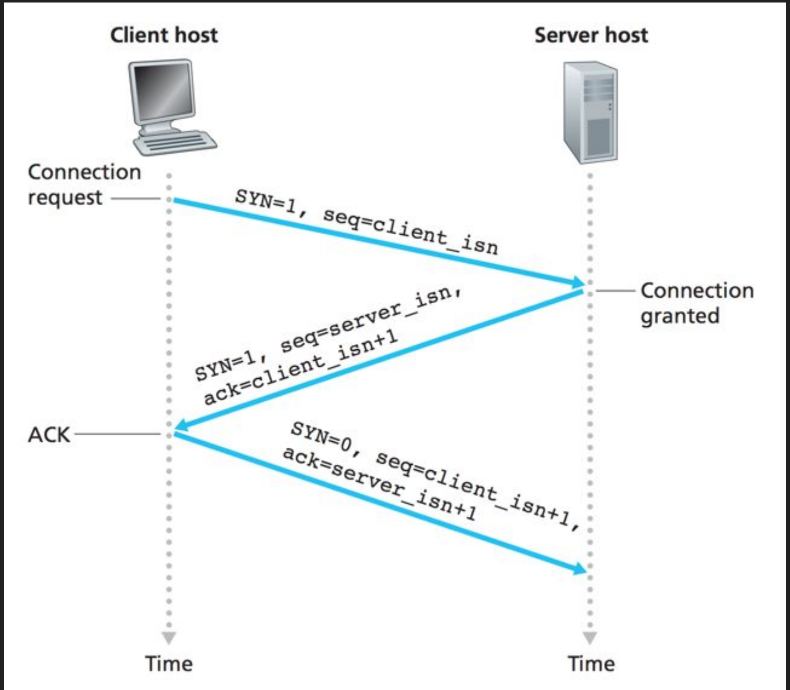
TCP Connection Management

- Three-way handshake
  - Client => Server: SYN + initial Seq #
  - Server => Client: SYN + ACK + initial seq
  - Client => Server: ACK + optional data
- Four-way teardown
  - Each side sends FIN and the other side sends ACK

TCP Connection Setup

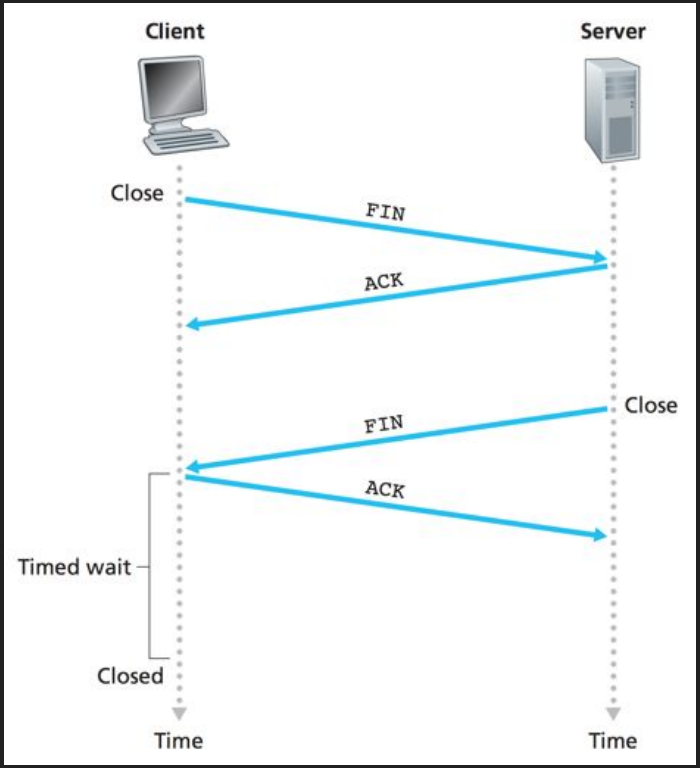
- Client
  - Choose client's initial seq number (ISN)
- Server
  - Choose server's initial seq number (ISN)
  - Ack client's SYN (Client ISN + 1)
- Client
  - Ack server's SYN (Server ISN + 1)
  - Can start to send data

- How to select ISN?
  - For better security, can use random numbers



TCP Connection Teardown

- When a side A finishes its data sending, it sends FIN.
- The other side B will Ack the FIN and continue to receive data for a while
  - Because though A sends out all the data, B may not receive them all yet
  - FIN ACK can get lost
- Reading: RFC 793's section 3.5
  - https://tools.ietf.org/html/rfc793#section-3.5



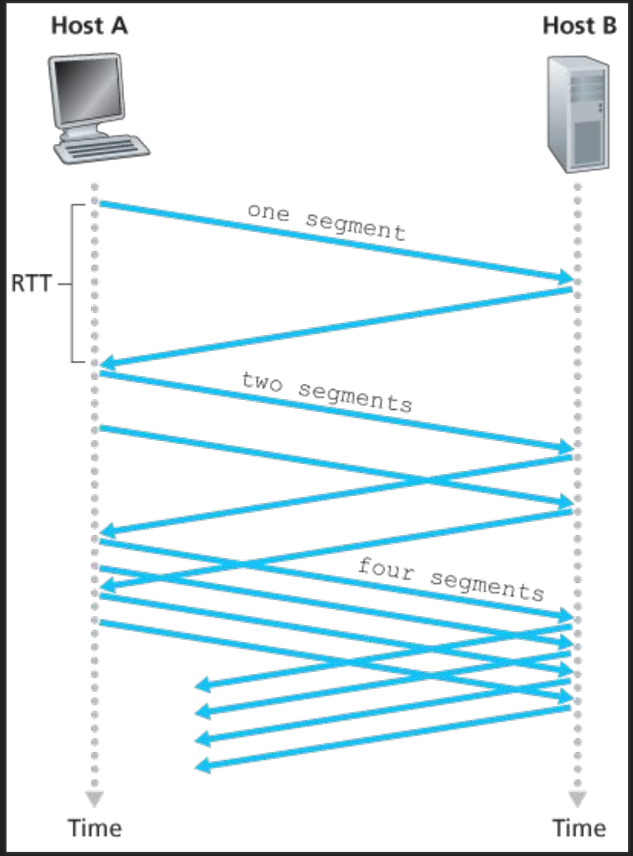
TCP Congestion Control

TCP Congestion Control

- Three main principles
  - Slow start
  - Congestion avoidance
  - Fast recovery

Slow Start

- Start with 1MSS and increase 1MSS for each ACK (not for each RTT)(so it’s not slow at all 😂)
- Change to fast recovery
  - If fast retransmission occurs
- Change to congestion avoidance
  - If congestion window >= slow start threshold



Congestion Avoidance

- Increase cwnd by 1 MSS every RTT (this is truly slow)
- Change to slow start
  - If retransmission timer goes off
- Change to fast recovery
  - If fast retransmission occurs

Fast recovery

- The transition state from multiplicative reduction to congestion avoidance. Congestion window += 1 for each duplicate ACK.
- Change to slow start
  - If timeout
- Change to congestion avoidance
  - If new ACK

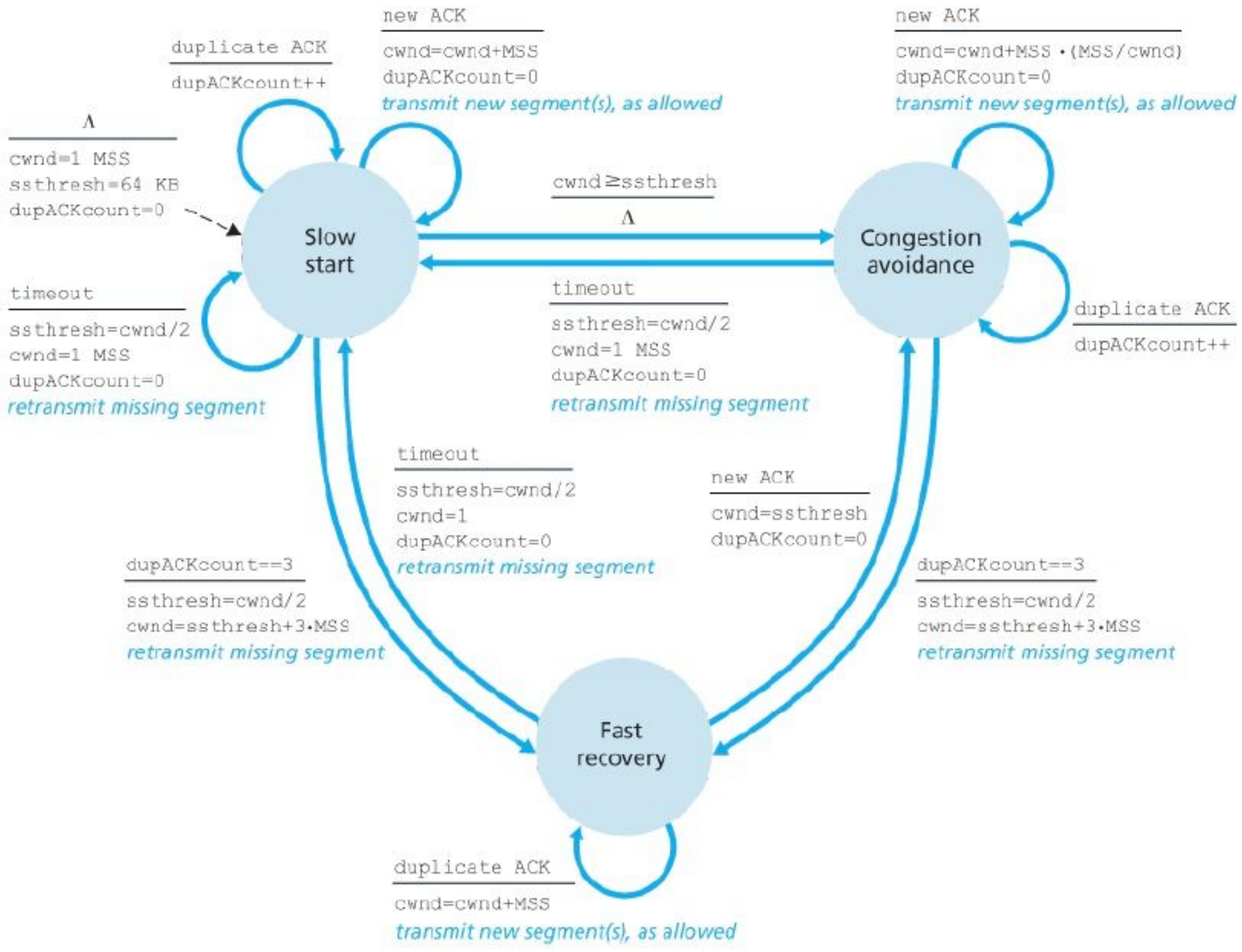


Figure from the textbook

(Note that the lecture is different from the textbook. Use slides as reference.)