emg

Name_____ Student ID_____ Department/Year_____

2nd Examination

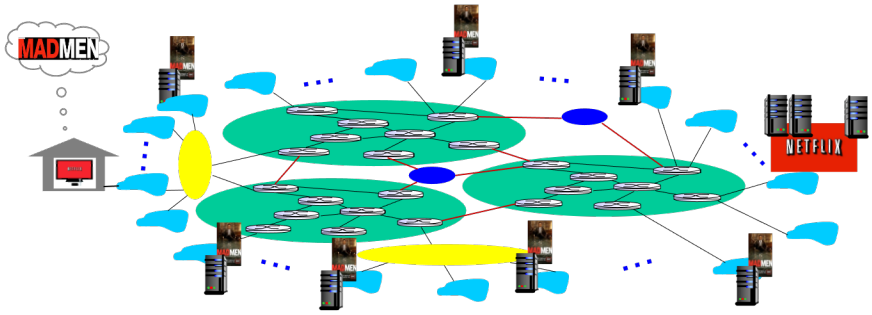
Introduction to Computer Networks (Online)
Class#: EE 4020, Class-ID: 901E31110
Fall 2021

10:20-12:10 Thursday
December 02, 2021

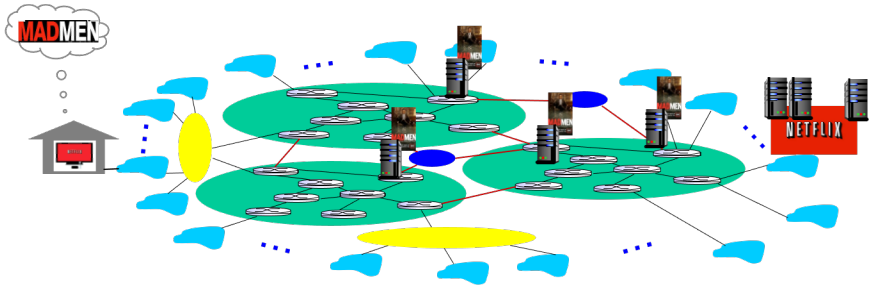
Cautions

1. There are in total 100 points to earn. You have 90 minutes to answer the questions. Skim through all questions and start from the questions you are more confident with.
2. Use only English to answer the questions. Misspelling and grammar errors will be tolerated, but you want to make sure with these errors your answers will still make sense.

1. (ch26, 8pt) Suppose Netflix's videos are distributed by two content distribution network (CDN) providers. Depicted below are where CDN (a) and (b) deploy their video servers. The original video servers are on the right and the client on the left.



(a)



(b)

(1) Which CDN is the "enter deep" type (1pt)?
(2) Which CDN is the "bring home" type (1pt)?
(3) Which type of CDN allows shorter delay to the viewers (1pt)?
(4) Which type of CDN tends to cost less (1pt)?
(5) Which type would you subscribe if you are just starting up a company? And why (2pt)?
(6) Which type would you subscribe if you are working for a large video content provider (such as Netflix) and looking to expand the viewership? Any why (2pt)?

Sample Solution:

(1) (a)
(2) (b)
(3) (a)
(4) (b)
(5) (b) any reason that makes sense. (e.g., to cut cost as there are not many viewers yet)
(6) (a) any reason that makes sense. (e.g., to provide a lower-level delay and therefore

better user experience to viewers. The viewer base is already large, the chance of getting a bulk rate deal is high.)

2. (ch26, 6pt) A viewer is watching a video on Netflix. The server streaming the video to the viewer pre-encodes the video to H.264 in different resolutions as follows.

Resolution	Bitrate
2160p (4K)	16 Mbps
1440p (2K)	9.6 Mbps
1080p	5 Mbps
720p	2.5 Mbps
480p	1.2 Mbps
360p	800 Kbps

Suppose server A and B both hold a copy of the video. The delay and available bandwidth from these servers to the viewer change often over time. Table (a) shows the measured delay and Table (b) shows the measured available bandwidth at time t1, t2, t3, and t4.

	t1	t2	t3	t4
Server A	10ms	50ms	5ms	10ms
Server B	20ms	20ms	20ms	50ms

(a) Delay

	t1	t2	t3	t4
Server A	3Mbps	5Mbps	1Mbps	2Mbps
Server B	10Mbps	8Mbps	9Mbps	10Mbps

(b) Available Bandwidth

(1) Suppose server A is assigned to the viewer. Tell the resolution of video the viewer is watching at t1. (1pt)

(2) Suppose server B is assigned to the viewer. Tell the resolution of video the viewer is watching at t2. (1pt)

(3) Suppose Netflix assigns the server of the shortest delay to the viewer. Tell the resolution of video the viewer is watching at t3. (2pt)

(4) Suppose Netflix assigns the server of the highest available bandwidth to the viewer. Tell the resolution of video the viewer is watching at t4. (2pt)

Sample Solution:

(1) 720p

(2) 1080p

(3) 360p

(4) 1440p

3. (ch31, 5pt) Recall the functionalities supported by the transport layer. For each of the following statements, tell whether it is true or false for UDP.
   (1) Bandwidth guarantee. (1pt)
   (2) Multiplexing and demultiplexing. (1pt)
   (3) Error detection. (1pt)
   (4) Reliable data transfer. (1pt)
   (5) Congestion control. (1pt)

Sample Solution:

F, T, T, F, F

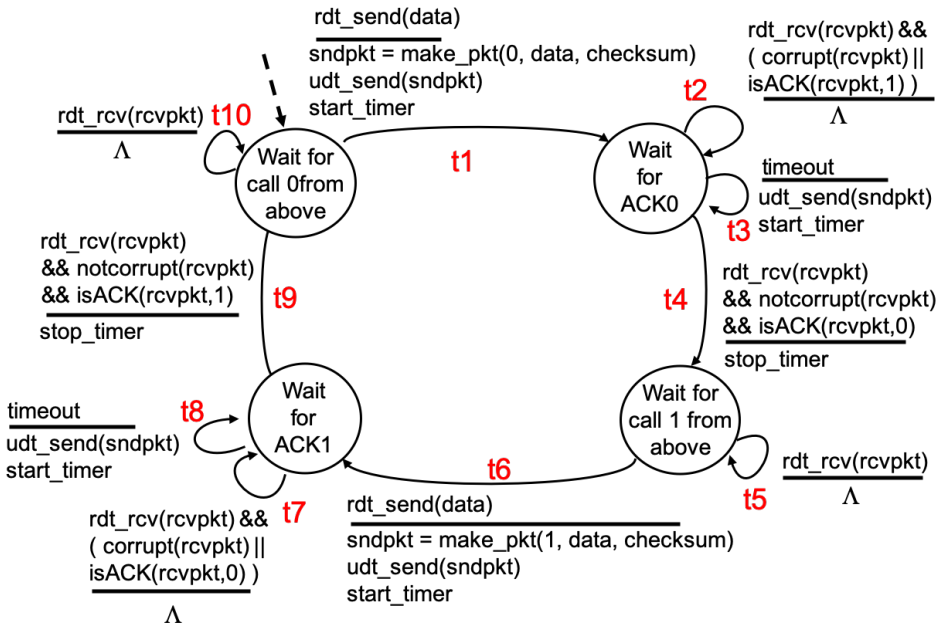
4. (ch31, 5pt) Recall the functionalities supported by the transport layer. For each of the following statements, tell whether it is true or false for TCP?
   (1) Bandwidth guarantee. (1pt)
   (2) Multiplexing and demultiplexing. (1pt)
   (3) Error detection. (1pt)
   (4) Reliable data transfer. (1pt)
   (5) Congestion control. (1pt)

Sample Solution:

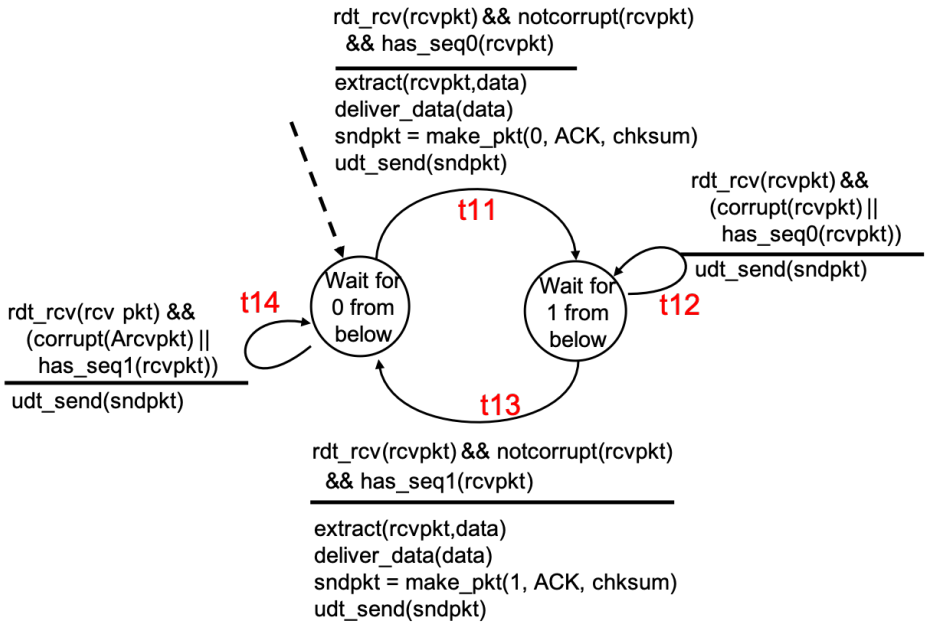
F, T, T, T, T

5. (ch34, 10pt) Provided below are the FSMs of rdt 3.0 sender and receiver. Indicate the order of the transitions (in terms of t1, t2, …, t14) taking place until the sender and receiver stabilize for the following scenarios.

rdt 3.0 sender:



rdt 3.0 receiver:



(1) Scenario 1: Both the sender and receiver start from the initial state. The sender gets a call from above to send just 1 data packet. There is no bit error and no loss in the data packet going to the receiver and all subsequent packet transmissions are fine, i.e., no bit error, no packet loss afterwards. (1%)

(2) Scenario 2: Continue from Scenario 1. The sender gets another call from above to send just 1 data packet. There is no bit error and no loss in the data packet going to the receiver and all subsequent packet transmissions are fine. (1%)

(3) Scenario 3: Both the sender and receiver start from the initial state. The sender gets a call from above to send just 1 data packet. There is a bit error in the data packet going to the receiver but all subsequent packet transmissions are fine, i.e., no bit error, no packet loss afterwards. (2%)

(4) Scenario 4: Continue from Scenario 3. The sender gets another call from above to send just 1 data packet. There is a bit error in the ACK 1 packet, but all subsequent packet transmissions are fine. (2%)

(5) Scenario 5: Both sender and receiver start from the initial state. The sender gets a call from above to send just 1 data packet. The data packet does not arrive at the receiver but all subsequent packet transmissions are fine. (2%)

(6) Scenario 6: Continue from Scenario 5. The sender gets another call from above to send just 1 data packet. The ACK 1 packet does not arrive back at the sender while all subsequent packet transmissions are fine. (2%)

Sample Solution:

(1) t1, t11, t4

(2) t6, t13, t9

(3) t1, t14, t2, t3, t11, t4

(4) t6, t13, t7, t8, t14, t9

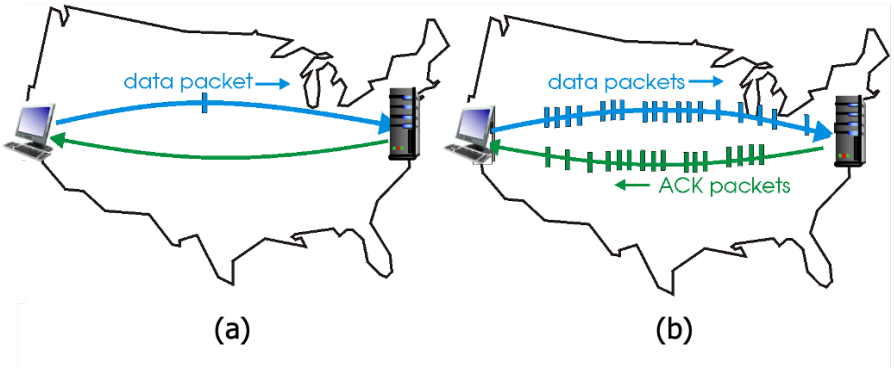
(5) t1, t3, t11, t4

(6) t6, t13, t8, t14, t9

Grading policy:

each correct transition (1pt), each wrong transition (-1pt)

6. (ch34, 7pt) Recall the following two types of data-ack exchange in a reliable data transfer (rdt) protocol. The sender and receiver of rdt (a) exchange 1 data packet and 1 ack packet in one round trip. The sender and receiver of rdt (b) exchange multiple data packets and multiple ack packets in 1 round trip.



(1) Recall and tell which one is referred to as a pipelined rdt protocol (1pt).
(2) Recall and tell which one is a stop-and-wait rdt protocol (1pt).
(3) Which rdt protocol offers a better level of utilization (1pt)?
(4) Let D be the data packet size (bits), R the transmission rate (bits per second), and L the client-server round-trip delay (second). To not exceed 100% utilization, how many data packets are allowed maximum in the round-trip delay? (2pt)
(5) To not exceed 100% utilization, the sending window size should be limited by the value derived in (4). Can you name another reason one might want to limit the sending window size? (2pt)

Sample Solution:

(1) (b)
(2) (a)
(3) (b)
(4) LR/D or LR/D + 1 (both accepted)
(5) any reason that makes sense
Example 1: To do flow control – receiver informing the sender to send in a rate slower than the rate the application can consume data.
Example 2: To do congestion control – receiver informing the sender of packet losses and the sender multiplicatively lower the sending window size.

7. (ch34, 9pt) Compare and contrast Go-Back-N (GBN), Selective Repeat (SR), and Transmission Control Protocol (TCP).
   (1) Which method(s) uses a dynamic window size? (1pt)
   (2) Which method(s) defines clearly the retransmission timeout interval? (1pt)
   (3) Which method(s) uses the least amount of network bandwidth to retransmit for losses? (1pt)
   (4) Which method(s) uses cumulative ack? (2pt)
   (5) Which method(s) uses at most one timer at the sender end? (2pt)
   (6) Which method(s) buffers out-of-order packets at the receiver end? (2pt)

Sample Solution:

(1) TCP
(2) TCP
(3) SR
(4) GBN, TCP
(5) GBN, TCP
(6) SR and TCP (ok to not list TCP since it's "officially" optional)

Grading policy:

(1)-(3) exact answer (1pt), else (0pt)
(4)-(6) each answer (1pt), each wrong/missing answer (-1pt)

8. (ch35, 8pt) Below is the pseudo-code describing the reliable data transfer part of the TCP sender. The mechanism to derive the retransmission timeout interval is missing. The timeout interval derivation starts by getting the SampleRTT. Let's try if we can find places in the pseudo-code where we add the mechanism to obtain the SampleRTTs.

```
1   NextSeqNum = InitialSeqNum; SendBase = InitialSeqNum
2   cwnd = InitialCwnd; ssthresh = InitialSsthresh
3   loop (forever) {
4       switch(event)
5       event: data received from application above
6           If (NextSeqNum + length(data) <= SendBase + wnd) {
7               create TCP segment with sequence number NextSeqNum
8               pass segment to IP
9               NextSeqNum = NextSeqNum + length(data)
10              if (timer currently not running)
11                  start timer
12          } else
13              refuse data
14      event: timer timeout
15          retransmit the smallest, not-yet-acknowledged segment
16          start timer
17      event: ACK received, with ACK field value of y
18          if (y > SendBase) {
19              SendBase = y
20              if (there are currently not-yet-acknowledged segments)
21                  (re)start timer
22              else
23                  cancel timer
24          } else {
25              increment count of dup ACKs received for y
26              if (count of dup ACKs received for y == 3) {
27                  resend segment with sequence number y
28                  count of dup ACKs received for y = 0
29              }
30          } /* end of if (y > SendBase)*/
31  }   /* end of loop forever */
```

A: lines 7–11
B: line 13
C: lines 15–16
D: lines 19–23
E: lines 25–29

A SampleRTT is calculated by taking the time difference between transmitting a data packet and receiving the corresponding ack. We need to be careful though. The SampleRTT of a data packet that has been retransmitted should be ignored because the time difference can potentially be longer than the actual RTT (including the retransmission timeout interval for example).

(1) Which code block(s) (i.e., A, B, C, D, E) transmits a new data packet? (1pt)
(2) Which code block(s) (i.e., A, B, C, D, E) retransmits a data packet? (1pt)
(3) Which code block(s) (i.e., A, B, C, D, E) to add the code that records the time transmitting a data packet? (2pt)
(4) Which code block(s) (i.e., A, B, C, D, E) to add the code that records the time receiving an ack? (2pt)
(5) Which code block(s) (i.e., A, B, C, D, E) to record that a packet's SampleRTT should be neglected? (2pt)

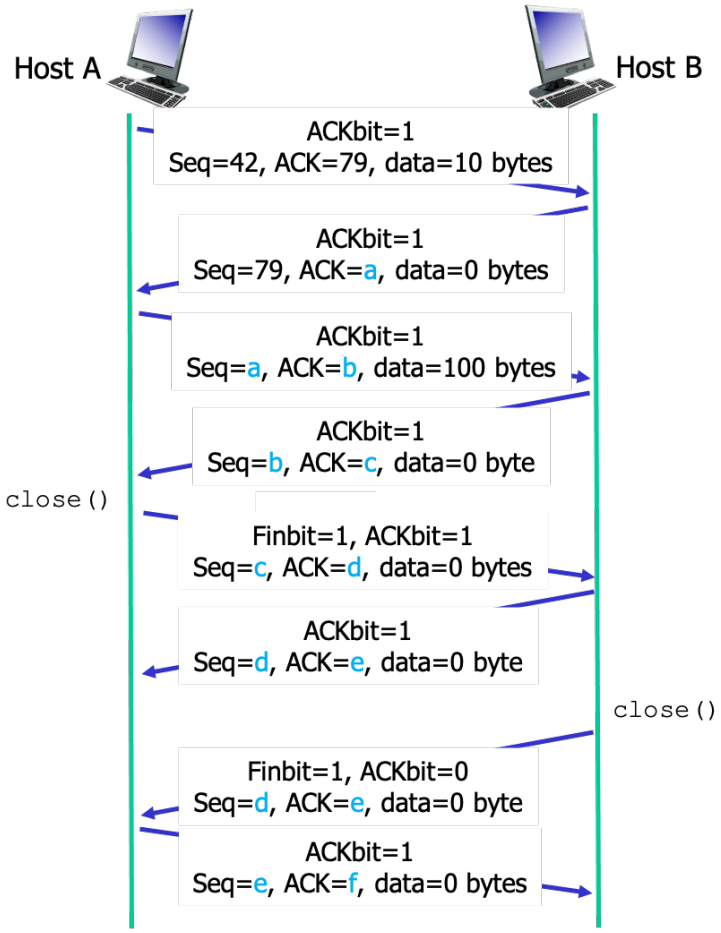
Sample Solution:

(1) A
(2) C, E
(3) any block that transmits a new data -- A
(4) any block that receives a new ack -- D
(5) any block that retransmit a packet -- C, E

Grading policy:

(1)-(2) exact answer (1pt)
(3)-(4) each answer (2pt), each wrong answer (-1pt)
(5) each answer (1pt), each wrong answer (-1pt)

9. (ch35, 6pt) Depicted below is a sequence of packet exchange in a TCP connection between Host A and B. Host A sends the last two data packets and receives the ACK packets from Host B. Then, Host A sends a FIN packet to initiate closing of the connection. Host B sends a FIN packet as well. Tell the value of a, b, c, d, e, f.



Sample Solution:

(a, b, c, d, e, f) = (52, 79, 152, 79, 153, 80)

10. (ch37, 10pt) In TCP's congestion control mechanism, the congestion window size (cwnd) and the slow start threshold (ssthresh) maybe adjusted in the following ways depending on the events occurring.

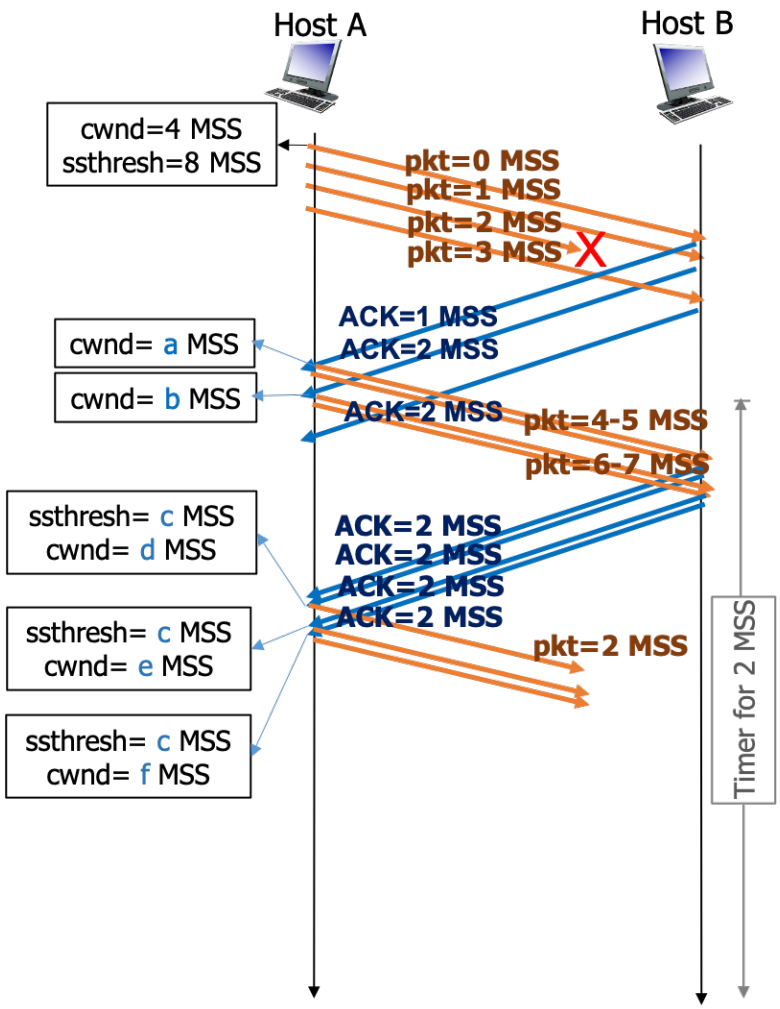
(a) cwnd = cwnd + 1MSS
(b) cwnd = cwnd + MSS * MSS/cwnd
(c) cwnd = 1MSS
(d) cwnd = cwnd/2
(e) cwnd = cwnd/2 + 3MSS
(f) ssthresh = 1MSS
(g) ssthresh = cwnd/2

(1) Tell the way cwnd is adjusted when a new ACK packet is received in slow start state. (1pt)
(2) Tell the way cwnd is adjusted when a new ACK packet is received in congestion avoidance state. (1pt)
(3) Tell the way cwnd is adjusted when timeout occurs in slow start state. (1pt)
(4) Tell the way cwnd is adjusted when timeout occurs in congestion avoidance state. (1pt)
(5) Tell the way cwnd is adjusted when triple duplicate ack is detected in slow start state. (1pt)
(6) Tell the way cwnd is adjusted when triple duplicate ack is detected in congestion avoidance state. (1pt)
(7) Tell the way ssthresh is adjusted when timeout occurs in slow start state. (1pt)
(8) Tell the way ssthresh is adjusted when timeout occurs in congestion avoidance state. (1pt)
(9) Tell the way ssthresh is adjusted when triple duplicate ack is detected in slow start state. (1pt)
(10) Tell the way ssthresh is adjusted when triple duplicate ack is detected in congestion avoidance state. (1pt)

Sample Solution:

(1) (a)
(2) (b)
(3)-(4) (c)
(5)-(6) (e)
(7)-(10) (g)

11. (ch37, 6pt) Below is the exchange of data and ack packets in a TCP connection. Tell the value of a, b, c, d, e, f.



Sample Solution:

(a, b, c, d, e, f) = (5, 6, 3, 6, 7, 8)

12. (PA, 10pt) Go on to the PA server and log in with the exam username and password. Create a subdirectory `exam2` If you have not yet done so. Go to the `exam2` subdirectory and work on the following.

    Let's see if you've picked up the basics of programing a server by implementing an echo server. In that, your server will just echo back whatever message it receives from the client. Assume the client will always be client-102.go as follows.

    client-102.go

```
package main

import "fmt"
import "bufio"
import "net"

func check(e error) {
    if e != nil {
        panic(e)
    }
}

func main() {
    conn, errc := net.Dial("tcp", "127.0.0.1:<your port#>")
    check(errc)
    defer conn.Close()

    writer := bufio.NewWriter(conn)
    len, errw := writer.WriteString("Hello World!\n")
    check(errw)
    fmt.Printf("Send a string of %d bytes\n", len)
    writer.Flush()

    reader := bufio.NewReader(conn)
    message, errr := reader.ReadString('\n')
    check(errr)
    fmt.Printf("Server replies: %s", message)
}
```

(1) Create e2-p12-1.go such that the server can receive the greeting from client-102.go, sleep for 3 seconds, and send the same greeting back to client-102.go. (4%)
(2) Create e2-p12-2.go such that the server can loop forever and echo the greeting from client-102.go one at a time. (2%)
(3) Create e2-p12-3.go such that the server can loop forever and echo greetings from multiple client-102.go concurrently. (4%).

Grading policy:

(1) Sleeping (2pt). sending same greeting back (2pt)
(2) Looping to send same greeting back (2pt)
(3) Looping to send same greeting back (2pt). Go routine (2pt)

13. (PA, 10pt) Go on to the PA server and log in with the exam username and password. Create a subdirectory `exam2` If you have not yet done so. Go to the `exam2` subdirectory and work on the following.

    Polly is running many servers on port 40001 to 41000 of the PA server. Among all these servers, one is special, called the bingo server. The end goal is to scan the ports between 50001 and 51000 and find the port number of the bingo server as quick as possible.

    Each of these servers takes a connection request and returns a simple string `"too low\n"`, `"bingo\n"`, or `"too high\n"`. The bingo server returns `"bingo\n"`. Those running on port numbers lower than the bingo server returns `"too low\n"`. Those running on port numbers higher than the bingo server returns `"too high\n"`.

    (1) Create e2-p13-1.go such that it is able to probe the server running on port 40001 and print the message from the server. (2pt)
    (2) Create e2-p13-2.go such that it is able to probe through the servers on port 40001 to 41000 sequentially, stop at the bingo server, and tell the port number of the bingo server. (4pt)
    (3) Create e2-p13-3.go such that it is able to binary search, find the bingo server within $\log_2 1000$ probes (i.e., 10 probes max), and tell the number of probes attempted. (4pt)

Grading Policy:

(1) Getting "too low" (2pt)
(2) Scanning (2pt). reaching the bingo port (2pt)
(3) Searching (2pt). reaching the bingo port (2pt)