CS118 Quiz 1, Spring 2021

Name: _____

Student ID: Solution

Notes:

1. This is an open-book, open-notes quiz. You have 2.5 hours to work on your quiz, scan or photo your paper copy, and upload to the Gradescope. You need to upload your PDF file, or scanned copy, or the photoed picture of your answer sheet to the Gradescope before the deadline. The submission deadline is strictly enforced, and no extension will be granted.

2. You are allowed to use your calculator. You are advised **not** to use the Internet to search for hints during the quiz. By submitting your quiz, you declare that **your work is solely done by yourself and you have not interacted with anyone else other than the instructor and proctors during the test**.

3. If you have any issues with the quiz, you can tune in the regular Lecture Zoom link and use the Chatroom there during the quiz time. We will provide clarifications there during the quiz.

4. Be **brief** and **concise** in your answers. Answer only within the space provided. If you need additional work sheets, use them but do NOT submit these sheets. If you wish to be considered for partial credit, show all your work.

5. **Budget your time properly**. You may not over-spend on multiple choices of Problem 1 and short Q/A of Problem 2. Leave time for Problems 3-6; you may start with them first.

6. You have total 9 pages (cover page, 6 problems in 7 pages, and 1-page Appendix).

PROBLEM	MAX SCORE	YOUR SCORE
1	40	
2	18	
3	12	
4	16	
5	8	
6	6	
TOTAL	100	

Problem 1 (40 points): Multiple choices (2 points each). Select **all** correct answers.

1. CD Which is (are) feature(s)of packet switching?

   - Your answer ____ (A) it performs worse than circuit switching for bursty computer traffic; (B) it offers throughput guaranteed services; (C) no reservation is needed in advance; (D) packets from the same app with identical source and destination might not always use the same path;

2. AB Which layers in the protocol stack are implemented at hosts but *not* at routers?

   - Your answer ____ (A) application layer; (B) transport layer; (C) network layer; (D) link layer.

3. D Which is (are) the limitation(s) of protocol layering?

   - Your answer ____ (A) One layer never overlaps functionality of another layer. (B) When a new implementation for a layer is released, other layers will continue to interoperate. (C) Every Internet router or end host runs every layer of the entire protocol stack. (D) It introduces overhead via the added protocol header at each layer.

4. BCD Which of the following protocol(s) will never use UDP?

   - Your answer ____ (A) DNS; (B) HTTP 1.1; (C) SMTP; (D) POP3.

5. BCD What would be the new feature(s) of a file-distribution application using the peer-to-peer (P2P) model rather than the client-server model?

   - Your answer ____ (A) All peers have to remain always on; (B) A peer can join at any time; (C) P2P model scales better than the client-server model in file downloading time; (D) A peer can serve certain peers while issuing requests to other peers as a client.

6. A Which protocol is **never** used when Bob uses his smartphone to browse the popular MSNBC News Web site?

   - Your answer ____ (A) SMTP; (B) TCP; (C) HTTP; (D) DNS.

7. AC What is (are) true on Dynamic Adaptive Streaming over HTTP (DASH) for video streaming?

   - Your answer ____ (A) The video is encoded into several versions, which have distinctive qualities with different bit rates; (B) It only allows clients with same Internet access speed to stream video; (C) It allows a client to dynamically adjust its streaming rate; (D) Most intelligence is placed at the server side once streaming starts.

8. C Which statement(s) about Internet process-to-process communication is (are) **correct**?

   - Your answer ____ (A) Internet applications might not use the process-to-process communication. (B) In P2P mode, there are no client and server processes in each peer. (C) One process can communicate with many processes at other hosts. (D) When two mail servers communicate with each other via SMTP, there are no client and server processes since both are servers.

9. (A)CD Which statement(s) is (are) **wrong**?

- Your answer ____ (A) The "Date:" field in the HTTP response message header indicates when the object in the response was last modified; (B) Two distinct image objects can be sent over the same persistent connection; (C) The HTTP response messages with status code 200 never have an empty message body; (D) With non-persistent connections between the browser and the server, it is possible for a single TCP segment to carry two distinct HTTP request messages.

10. D Joe has a UCLA account and reads his emails from this account via outlook application. Which protocols are used when he accesses emails sent by a friend alice@cs.ucb.edu?

    - Your answer ____ (A) DNS, HTTP; (B) HTTP, TCP; (C) UDP, POP3/IMAP; (D) SMTP, DNS, IMAP/POP3.

11. BD What mechanism(s) are used to allow a cache to verify its objects are up to date in HTTP?

    - Your answer ____ (A) Cookies; (B) conditional GET; (C) stateless HTTP server; (D) specify date of cached copy in HTTP request.

12. C Which of the following statement(s) on application protocols is (are) **correct**?

    - Your answer ____ (A) Internet Mail access protocol (IMAP) is a protocol for email exchanges between email servers; (B) SMTP uses pull model for data delivery; (C) DNS provides the translation service from the hostname to the IP address by using a distributed database; (D) DNS query and DNS reply messages use different DNS header format.

13. AC Which of the following statement(s) about DNS is (are) **true**?

    - Your answer ____ (A) A local DNS server may query the root DNS server; (B) DNS caching can never return outdated resolution results; (C) Some DNS queries can be iterative and others recursive, in the sequence of queries to translate a hostname; (D) DNS resolves names always through the root server.

14. B Which statement(s) is (are) **wrong**?

    - Your answer ____ (A) Hosts are running at the network edge; (B) Servers are at the network core but not at the network edge; (C) Access network connects end systems to edge router; (D) Packet switching uses store and forward in data delivery.

15. B Which field(s) in the UDP header are used in connectionless demultiplexing?

    - Your answer ____ (A) Source port number; (B) Destination port number; (C) Length; (D) Sequence number.

16. A,B Which statement(s) on UDP would be **correct**?

    - Your answer ____ (A) UDP provides connectionless service for the Internet. (B) UDP cannot control Internet congestion. (C) UDP uses the 4-tuple of (source-IP-address, destination-IP-address, source-port, destination-port) as its identifier. (D) UDP implements reliable data transfer.

17. A, B, C Which mechanism is required for reliable data transfer upon corrupted packets?

    - Your answer ____ (A) Error detection via checksum; (B) Sequence numbers for transmitted packets; (C) Retransmissions; (D) Pipelined transmission of data segments.

18. D Assume that the selective repeat protocol uses a window size of 20 segments at the sender side. How many unique sequence numbers do we need *at least*?

    - Your answer ____ (A) 2; (B) 20; (C) 21; (D) 40.

19. C Which mechanism is **not** absolutely required for reliable transfer upon data packet loss only?

    - Your answer ____ (A) Sequence numbers for transmitted packets; (B) Retransmissions; (C) Checksum; (D) Acknowledgment number for each received segment.

20. D In the Go-back-N Protocol, given the sender's window [401,402] and N =2, select which is **impossible** as being the receiver's next expected sequence number.

    - Your answer ____ (A) 401; (B) 402; (C) 403; (D) 404.

Problem 2 (18 points; 3 points each): Answer the following questions. Be brief and concise.

1. Explain how a hostname is resolved in DNS *recursive query* mode. What is the difference from the *iterated query*? *(Hint: note which DNS server makes the DNS query in each mode.)*

   First, local DNS server queries root, which then queries TLD, which queries the authoritative DNS severs, until it is answered. Key: it is the chain of DNS severs that sends query every time, not the local DNS server only.

2. A new video streaming startup *v-start* decides to use the third-party CDN provider *limelight* to scale and reduce streaming latency. Explain how to leverage DNS to intercept and redirect the user clicks on *http://video.v-start.com/* to the CDN servers deployed by *limelight*.

   Local DNS server sends DNS query to v-start DNS server; v-start DNS server has the limelight DNS server entry, and replies with the limelight DNS server; limelight DNS server returns its CND server's IP address; local DNS server gets the DNS response; contacts the CDN server.

3. Consider the queuing delay in a router buffer (preceding an outbound link). Suppose all packets are L bits, the transmission rate is R bps, and that all N packets simultaneously arrive at the buffer at time t=0. Find the average queuing delay for the N transmitted packets.

It waits for the transmission delay of i-1 packets for the i-th packet. Then the result is:

$$\frac{1}{N}\sum_{i=0}^{N}(i-1)\frac{L}{R} = \frac{N-1}{2} \times \frac{L}{R}.$$

4. A student group decides to develop a new Internet social networking app running on their computers, which are on and off at times. However, the app is expected to support a *large* number of users eventually. Which model the application should use, the client-server paradigm or the peer-to-peer one? Briefly justify your answer.

2 points: P2P
3 points: No reliance on always-on servers

5. A UDP receiver computes the Internet checksum for the received UDP segment and finds a *mismatch* with the checksum field. Can the receiver be absolutely sure that bit errors have occurred with **the received UDP data**? Assume it matches the value carried in the checksum field. Can the receiver be absolutely sure that no bit errors have occurred? Briefly justify your answer. (hint: a UDP segment includes the UDP header and the UDP data payload)

(1)No. The error might happen in the UDP header. (1.5 points deducted if yes).
(2)No, the receiver cannot be absolutely certain that no bit errors have occurred.
If bit errors transform the packet to another one with identical checksum value, then such errors will go undetected.

6. For the Go-Back-N protocol with the sender window size of 8, what is the minimum number of bits needed for the Sequence number field?

The number of sequence numbers needed is: 8+1=9. Therefore, 4-bit sequence number field is needed.

Problem 3 (12 points): Reliable Transfer Protocol In the following scenario, discuss how each reliable transfer protocol reacts:

- (3 points) In the Stop-and-Wait protocol, how does the receiver react if a duplicate data packet is received?

1. drop the duplicate data packet; 2. send back an ACK with the sequence number indicated by the duplicate data.

- (3 points) Assume cumulative acknowledgment is used and each data segment size is 100B. In the Selective Repeat protocol, how does the sender react when receiving an ACK segment (with acknowledgment number 900B) right after receiving a duplicate ACK segment (with acknowledgment number 800B)?

(1) Leap the window forward; this is indicated by ACK 900 (as this new ACK covers the previous duplicate ACK); (2) send two new data packets allowed by the window and associate the timer with each of two sent segments (per-segment timer) or the 1st segment in the window (per-window timer).

- (3 points) Assume cumulative acknowledgment is used. In the Go-back-N protocol, if timeout is triggered at the sender before the sender receives the third duplicate ACK, how does the sender react? Justify your answer.

1. retransmit all the packets in the current window; 2. restart the timer with the 1st packet in the window

- (3 points) Will premature timeout (i.e., the retransmission timer has a value smaller than RTT) affect the correctness of reliable transfer protocol? Briefly justify your answer.

  No. Premature timeouts only trigger unnecessary retransmissions, but the reliable data transfer is still ensured.

Problem 4 (16 points): Joe aims to visit www.google.com to search "news today" and then visit CNN (https://www.cnn.com/) to visit the news.

1. (2 points) When Joe enters the URL into his web browser, what application-layer and transport-layer protocols will be used?

   2 points. DNS (UDP), HTTP (TCP).

2. (5 points) Please specify what happens at the client side step by step?

   First, it runs DNS. It support TYPE=A nslookup and retrieves the IP address of www.google.com.
   Afterwards, it opens a TCP socket and connects with the server using the IP of www.google.com as destination IP and 80 as dest port number. Afterwards, it setups a TCP connection with Google.
   It sends a HTTP request (GET) and waits for HTTP response from Google.
   It receives the HTTP response and then click the link to another webpage.
   It repeats the above procedure to run DNS, TCP and HTTP to get the webpage.

3. (7 points) when the client visits cnn.com, it is fetching index.html which has 20 referenced objects (.jpeg images) from an Internet server. Assume this index.html and .jpeg files are extremely small, and the transmission time is negligible. Let us assume the RTT (round trip time) between

the client and server is 1 second. How long does it take to retrieve all the objects using persistent connection without pipelining? How long does it take to retrieve all the objects using non-persistent connection? How about the non-persistent HTTP with parallel connections (the maximum TCP connection is 5 at one time)? Which one is faster?

Persistent: $2r + r * 20 = 22r = 22$ seconds.
Non-Persistent: $2r + 2r * 20 = 42r = 42$ seconds.
Non-persistent HTTP with parallel connections: $2r + 2r\lceil k/c \rceil$ = 2r + 2r * 4 = 10r = 10 second.
Non-persistent HTTP with parallel connections.

4. (2 points) Joe next forwards one news article to his friend Bob. He sends it from his gmail to Bob's gmail. What application-layer and transport-layer protocols will be used in this case until Bob receives the news (List at least five protocols to get full points)?

2 points. DNS (UDP), HTTP (TCP), SMTP (TCP), HTTP (email access).

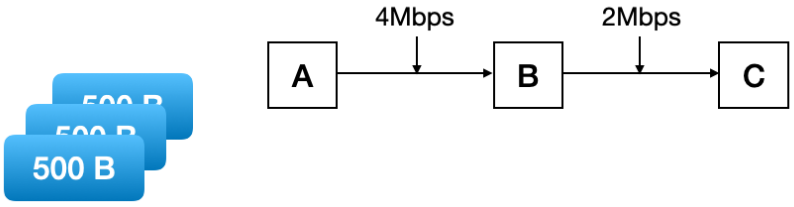
Problem 5 (8 points): Joe is writing programs with a TCP client and a TCP server using stream sockets. The following is the SERVER code that Joe wrote. Can you help Joe to fill in the blanks in his code? You can use the Appendix for references.

```
1  #include <server.h>  /* assume all headers are included correctly */
2  #define PORT 8080
3  int main(int argc, char const *argv[])
4  {
5      int server_fd, new_fd; /* listen on server_fd, new connection on new_fd */
6      struct sockaddr_in address;
7      int addrlen = sizeof(address);
8      char buffer[1024] = {0};
9      char *hello = "Hello from server"; /* hello message to be sent */
10
11     if ((server_fd = socket(AF_INET, _____, 0)) < 0) {
```

```
12          perror("socket␣failed");
13          exit(EXIT_FAILURE);
14      }
15      address.sin_family = PF_INET;
16      address.sin_addr.s_addr = INADDR_ANY;
17      address.sin_port = ntohs(PORT);
18
19      if (bind(server_fd, (struct sockaddr *)&address, addrlen) < 0) {
20          perror("bind␣failed");
21          exit(EXIT_FAILURE);
22      }
23      if (__________(server_fd, 3) < 0) {
24          exit(EXIT_FAILURE);
25      }
26      if ((new_fd = __________(server_fd, (struct sockaddr *)&address, &addrlen)) < 0) {
27          exit(EXIT_FAILURE);
28      }
29      int valread = read(server_fd, buffer, 1024);
30      printf("%s\n", buffer);
31      ____________________________________; /* send hello to socket new_fd */
32      printf("Hello␣message␣sent\n");
33  }
```

(2 points each) socket(AF_INET, SOCK_STREAM, 0)
listen(server_fd, 3) < 0)
accept(server_fd, (struct sockaddr *)&address, &addrlen)) ¡ 0)
write(new_fd, hello, sizeof(hello));

Problem 6 (6 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 4Mbps (4×10^6 bits per second), and link B-C's bandwidth is 2Mbps.



Assume A starts transmitting the first packet at time t = 0, and no packets are buffered at B at t = 0. When will C receive all 3 packets?

0.005 + 0.002*2 = 0.009 (9ms)
A → B = 500*8/4000000 = 0.001 second
B → C = 500*8/2000000 = 0.002 second.
The first packet arrives at C at t = 5ms = 1ms + 1ms + 2ms + 1ms The second packet starts at t = 1ms. It arrives at B at t = 3ms (1ms + 1ms + 1ms) but it can't be sent out immediately because it needs to wait for the first packet leaving B. It leaves B at t = 4ms and will arrive at C = 7 ms (4ms + 2ms + 1ms).
So the gap = 2ms. There are two gaps between three packets.
(A simpler answer is the bottleneck link is B-C. The second packet can be sent at B after the first packet leaves B. The gap is 2ms (B → C = 500*8/2000000 = 0.002 second).

Appendix. Socket Programming Function Calls.

- *struct in_addr { in_addr_t s_addr; /* 32-bit IP addr */}*
- *struct sockaddr_in {*
  short sin_family; / e.g., AF_INET */*
  ushort sin_port; / TCP/UDP port */*
  struct in_addr; / IP address */ }*
- *struct hostent* gethostbyaddr (const char* addr, size_t len, int family)*
  struct hostent gethostbyname (const char* hostname)*;
  char inet_ntoa (struct in_addr inaddr)*;
  int gethostname (char name, size_t namelen)*;
- *int socket (int family, int type, int protocol)*;
  [*family*: AF_INET (IPv4), AF_INET6 (IPv6), AF_UNIX (Unix socket); *type*: SOCK_STREAM (TCP), SOCK_DGRAM (UDP); *protocol*: 0 (typically)]
- *int bind (int sockfd, struct socketaddr* myaddr, int addrlen)*;
  [*sockfd*: socket file descriptor; *myaddr*: includes IP address and port number; *addrlen*: length of address structure=sizeof(struct sockaddr_in)]
  returns 0 on success, and sets *errno* on failure.
- *int sendto(int sockfd, char* buf, size_t nbytes, int flags, struct sockaddr* destaddr, int addrlen)*;
  [*sockfd*: socket file descriptor; *buf*: data buffer; *nbytes*: number of bytes to try to read; *flags*: typically use 0; *destaddr*: IP addr and port of destination socket; *addrlen*: length of address structure==sizeof(struct sockaddr_in)]
  returns number of bytes written or -1. Also sets *errno* on failure.
- *int listen (int sockfd, int backlog)*;
  [*sockfd*: socket file descriptor; *backlog*: bound on length of accepted connection queue]
  returns 0 on success, -1 and sets *errno* on failure.
- *int recvfrom (int sockfd, char* buf, size_t nbytes, int flags, struct sockaddr* srcaddr, int* addrlen)*;
  [*sockfd*: socket file descriptor; *buf*: data buffer; *nbytes*: number of bytes to try to read; *flags*: typically use 0; *destaddr*: IP addr and port of destination socket; *addrlen*: length of address structure==sizeof(struct sockaddr_in)]
  returns number of bytes read or -1, also sets *errno* on failure.
- *int connect(int sockfd, struct sockaddr* servaddr, int addrlen)*;
  [*sockfd*: socket file descriptor; *servaddr*: IP addr and port of the server; *addrlen*: length of address structure==sizeof(struct sockaddr_in)]
  returns 0 on success, -1 and sets *errno* on failure.
- *int close (int sockfd)*;
  returns 0 on success, -1 and sets *errno* on failure.
- *int accept (int sockfd, struct sockaddr* cliaddr, int* addrlen)*;
  [*sockfd*: socket file descriptor; *cliaddr*: IP addr and port of the client; *addrlen*: length of address structure==sizeof(struct sockaddr_in)]
  returns file descriptor or -1 sets *errno* on failure
- *int shutdown (int sockfd, int howto)*;
  returns 0 on success, -1 and sets *errno* on failure.
- *int write(int sockfd, char* buf, size_t nbytes)*;
  [*sockfd*: socket file descriptor; *buf*: data buffer; *nbytes*: number of bytes to try to write]
  returns number of bytes written or -1.
- *int read(int sockfd, char* buf, size_t nbytes)*;
  [*sockfd*: socket file descriptor; *buf*: data buffer; *nbytes*: number of bytes to try to read]
  returns number of bytes read or -1.
- *int select(int maxfdp1, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *tvptr)*;
  *FD_ZERO (fd_set *fdset)*;
  *FD_SET (int fd, fd_set *fdset)*;
  *FD_ISSET (int fd, fd_set *fdset)*;
  *FD_CLR (int fd, fd_set *fdset)*;