CS118 Quiz 1, Spring 2020

Name: _____

Student ID: Solution _____

Notes:

1. This is an open-book, open-notes quiz. You have two hours to work on your quiz, scan or photo your paper copy, and upload to the Gradescope.
2. You need to upload your 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.
3. 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**.
4. 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.
5. 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.
6. If you wish to be considered for partial credit, show all your work.
7. **Budget your time properly**. You may not over-spend your time on multiple-choice questions of Problem 1. Leave enough time for Problems 2-5; you may start with them first.
8. You have 5 problems in 6 pages, plus half-page bonus survey, one-page Appendix and this page.

PROBLEM	MAX SCORE	YOUR SCORE
1	45	
2	15	
3	8	
4	8	
5	14	
Survey	3 (bonus)	
TOTAL	90+3	

Problem 1 (45 points): Multiple choices (3 points each). Select **all** correct answers.

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

   - Your answer ____ (A) it performs better than circuit switching for bursty data traffic; (B) it offers delay guaranteed services; (C) no reservation is needed in advance; (D) different packets with the same source and destination will always use the same path; (E) Packets might be lost/dropped during delivery via statistical multiplexing.

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

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

3. BCD Which device(s) is (are) end hosts?

   - Your answer ____ (A) Network gateways operated by Time Warner Cable; (B) Servers deployed by Google to provide Google Map service; (C) Laptop at home that is online via Time Warner Cable; (D) iphone that turns on its mobile data plan; (E) Home wireless routers bought and run by AT&T DSL service.

4. AD Which is (are) the potential **drawback(s)** of protocol layering?

   - Your answer ____ (A) One layer may duplicate lower-layer functionality. (B) When a new version of implementation for a layer is released, other layers will not continue to interoperate. (C) Every Internet equipment (e.g., routers and end hosts) has to run every layer of the entire protocol stack. (D) It introduces overhead via the added protocol header at each layer. (E) It achieves the goal of divide and conquer via separation of layers.

5. BCDE Which of the following protocol(s) will use TCP?

   - Your answer ____ (A) DNS; (B) HTTP 1.0; (C) SMTP; (D) HTTP 1.1; (E) Telnet.

6. BDE What would be the unique 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) A peer cannot leave even after its file downloading completes; (D) P2P model scales better than the client-server model in file downloading time; (E) A peer can act as a client by issuing requests while serving other peers.

7. AD Which protocol is **not** used when Bob uses his smartphone to browse the popular ESPN sports Web site?

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

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

   - Your answer ____ (A) The video is encoded into several different versions, which have different qualities but the same bit rate; (B) It allows clients with different Internet access rates to stream video; (C) The client selects different chunks once each time with HTTP PUT request messages; (D) It does not allow a client to dynamically adjust its streaming rate; (E) Most intelligence is *not* placed at the server side once streaming starts.

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

   - Your answer ____ (A) Internet applications use the process-to-process communication. (B) In P2P mode, there are no client and server processes since all are equal peers. (C) When two mail servers communicate with each other via SMTP, there are no client and server processes since both are servers. (D) One process can communicate with many processes at other hosts. (E) One process can communicate with only one process.

10. ABDE 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) The HTTP response messages never have an empty message body; (C) Two distinct image objects can be sent over the same persistent connection; (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; (E) HTTP is using the push operation mode.

11. (A)(B)C(D)(E) Joe has a UCLA account and reads his emails from this account via outlook. 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) SMTP, DNS, IMAP/POP3; (D) UDP, POP3/IMAP; (E) DNS, HTTP.

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

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

13. CE 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) HTTP is a stateful protocol on top of TCP; (C) SMTP uses push model for data delivery; (D) DNS provides the translation service from the hostname to the IP address by using a centralized database; (E) Both DNS query and DNS reply messages use the same DNS header format.

14. A Which of the following statement(s) about DNS is (are) **wrong**?

    - Your answer ____ (A) A local DNS server will never query the root DNS server; (B) DNS caching is used to improve performance; (C) Some DNS queries can be iterative and others recursive, in the sequence of queries to translate a hostname; (D) DNS follows hierarchical design approach; (E) DNS do not use large centralized database.

15. BE 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) Routers in the network core find routes and forward packets; (E) Packet switching never uses store and forward in data delivery.

Problem 2 (15 points; 3 points each): Answer the following questions. Be brief and concise.

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

   First, local DNS server queries root, and then TLD and then authoritative DNS severs until it is answered. Key: it is the local DNS sever that sends query every time.

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. Compared with HTTP, SMTP requires authorization phase. Why does SMTP needs this while HTTP does not?

   Email is private information for one user. Emails should be not public to anyone but web page is public given the url. To protect the privacy, SMTP requires authorization.

4. 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 queuing delay of the packet that is transmitted last.

   It takes N-1 packets to complete their transmission: $(N-1) \times \frac{L}{R}$

5. A student group decides to develop a new Internet social networking application running on their own computers, which are on and off all the time. However, they expect their application 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.

P2P. No reliance on always-on servers

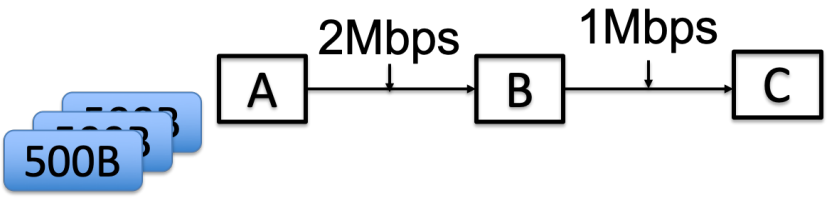
Problem 3 (8 points): Joe is writing programs with a TCP client and a TCP server using stream sockets. The following is the SERVER code that Joe wrote. Can you help Joe to find four errors (there can be more) in his code? You need to answer which line of code is wrong and the specific error. You can use the Appendix for references.

```
#include <server.h>  /* assume all headers are included correctly */
#define PORT 8080
int main(int argc, char const *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, SOCK_STREAM, 0)) > 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    address.sin_family = PF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = ntohs(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, addrlen) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (accept(server_fd, 3) < 0) {
        exit(EXIT_FAILURE);
    }
    if ((new_fd = listen(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
        exit(EXIT_FAILURE);
    }
    int valread = read(server_fd, buffer, 1024);
    printf("%s\n", buffer);
    sendto(new_fd, hello, strlen(hello), 0);
    printf("Hello message sent\n");
}
```

Should be socket(AF_INET, SOCK_STREAM, 0) < 0;
Forget byte ordering conversion htnos(INADDR_ANY);

Forget byte ordering conversion htnos(PORT);
Listen first then accept.
API sendto should be send;
Should read on the new socket read(new_fd, buffer, 255);

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 ($2x10^6$ 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)

4 point: A → B = 500*8/2000000 = 0.002 second
?B → C = 500*8/1000000 = 0.004 second.
The first packet arrives at C at t = 8ms = 2ms + 1ms + 4ms + 1ms The second packet starts at t = 2ms. It arrives at B at t = 5ms (2ms + 1ms + 2ms) but it can't be sent out immediately because it needs to wait for the first packet leaving B. It leaves B at t = 7ms and will arrive at C = 12 ms (7ms + 4ms + 1ms).
So the gap = 4ms.
(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 4ms (B → C = 500*8/1000000 = 0.004 second).

2. (4 points) When will C receive all 3 packets?

0.008 + 0.004*2 = 0.016 (16ms)

Problem 5 (14 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?

   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. (5 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)?

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

Survey (3 bonus points).

1. (0.5 point) How much networking knowledge have you learned from this course so far, compared with your expectation?

   ◯ **below** your expectation

   ◯ **match** with your expectation

   ◯ **above** your expectation

2. (0.5 point) Rate course assignment workload (homework, programming, and reading):

   ◯ **Light**

   ◯ **Average**

   ◯ **Heavy**

3. (0.5 point) Rate the current progress:

   ◯ **Slow** (You can surely speed up)

   ◯ **Average**

   ◯ **Fast** (You should slow down)

4. (0.5 point) Do you want to proctor future quizzes over Zoom (i.e., you start Zoom video during the quiz for the proctors to check)?

   ◯ **Yes**

   ◯ **No**

5. **Your suggestions (1 point).** What would you recommend to improve our online Zoom learning experience for the instructor? What would you recommend for your TA?

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)*;