CS 5/7-344

Computer Networks & Distributed Systems

Fall 2023

Practice Final Exam

Student Name: _____

SMU ID: _____

Exam instructions

You are allowed the following resources for this exam:

- Two sheets of notes (A4 – both sides)

Please put away all laptops and cell phones. You must use your own resources for this exam – sharing of resources is not allowed.

WHEN YOU HAVE COMPLETED THE EXAM, PLEASE SUBMIT YOUR EXAM AND NOTE SHEETS. PLEASE PUT YOUR NAME ON BOTH ITEMS.

For instructor use only

Student Name: _____

SMU ID: _____

Problem	Points	Score
1	20	
2	20	
3	20	
4	20	
5	20	
6	20	
Total	100	

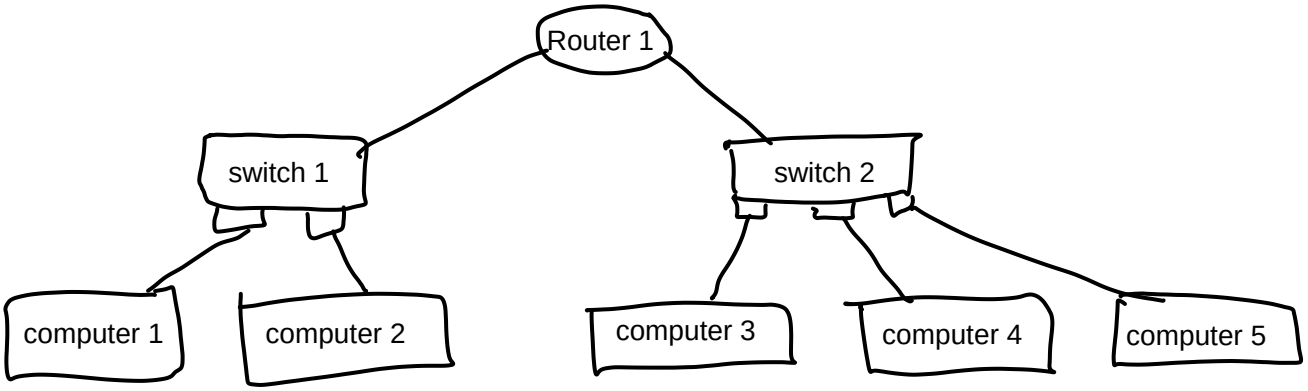
Any 5 of the 6 questions

TIME: 2 hours

Answer any 5 of the 6 questions.

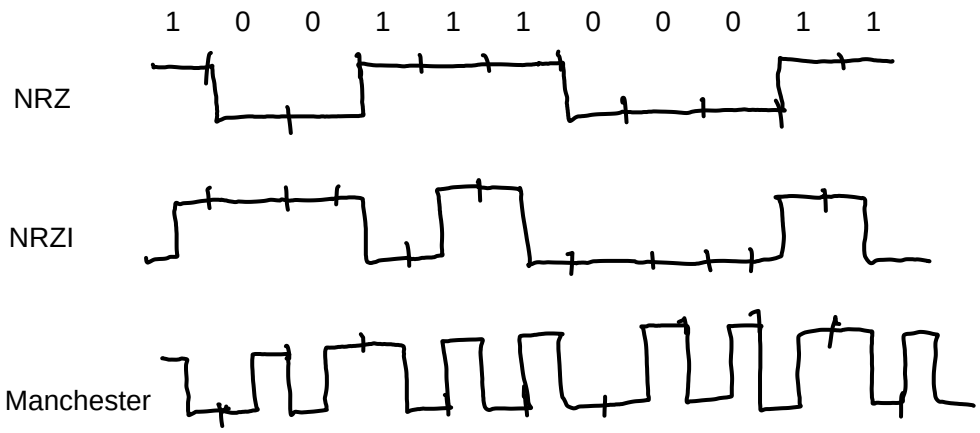
1)

A. (5 points) Suppose your house has two rooms. In the first room, 2 computers are connected with a LAN, and in the second room, 3 computers are connected with another LAN. And both of the LANs are connected with a router. Can you draw this network?



B. (5 points) Write the NRZ (non-return to zero), NRZI (non-return to zero invert), and Manchester forms for the following bit stream.

10011100011



C. (5 points) The following data fragment occurs in the middle of a data stream for which the byte-stuffing algorithm described in the text is used: A B ESC C ESC FLAG FLAG D. What is the output after stuffing?

After stuffing, we get A B ESC ESC C ESC ESC ESC FLAG ESC FLAG D

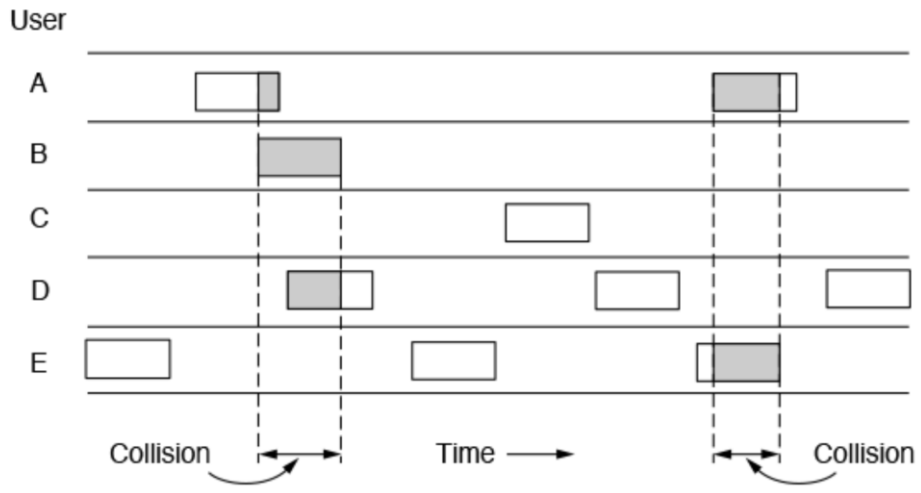
D. (5 points) Explain Piggybacking.

Although interleaving data and control frames on the same link is a big improvement over having two separate physical links, yet another improvement is possible. When a data frame arrives, instead of immediately sending a separate control frame, the receiver restrains itself and waits until the network layer passes the next packet. The acknowledgement is attached to the outgoing data frame (using the ack field in the frame header). In effect, the acknowledgement gets a free ride on the next outgoing data frame. The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as piggybacking.

2)

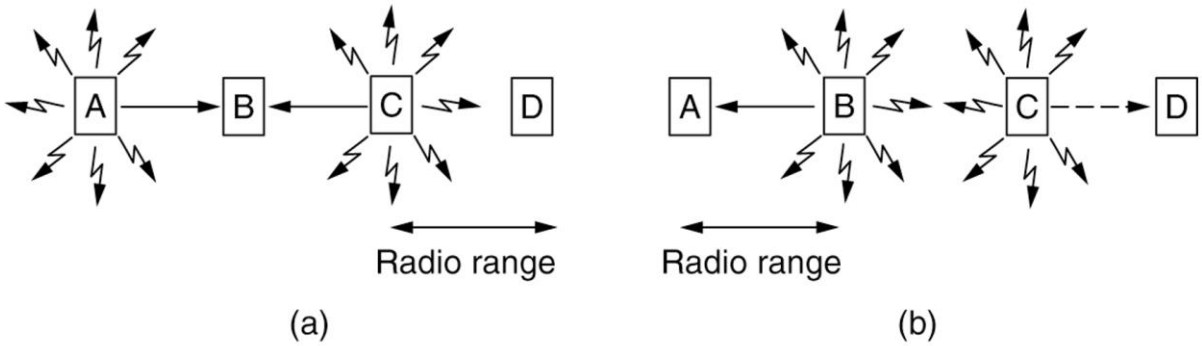
A. (10 points) How does pure ALOHA work? Why there is a chance of collision? Explain with a diagram.

The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged. Senders need some way to find out if this is the case.



From the above diagram we can see that whenever two frames try to occupy the channel at the same time, there will be a collision, and both will be garbled. If the first bit of a new frame overlaps with just the last bit of a frame that has almost finished, both frames will be totally destroyed, and both will have to be retransmitted later.

B. (10 points) What are hidden and exposed terminals? Explain with help of the following diagrams (a) and (b).

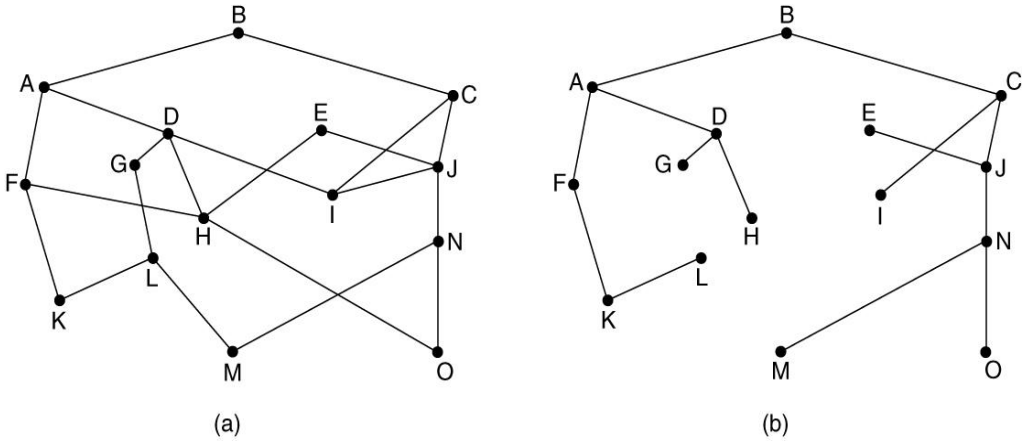


First consider what happens when *A* and *C* transmit to *B*, as depicted in (a). If *A* sends and then *C* immediately senses the medium, it will not hear *A* because *A* is out of its range. Thus, *C* will falsely conclude that it can transmit to *B*. If *C* does start transmitting, it will interfere with *B*, wiping out the frame from *A*. The problem of a station not being able to detect a potential competitor for the medium because the competitor is too far away is called the hidden terminal problem.

Now let us look at a different situation: *B* transmitting to *A* at the same time that *C* wants to transmit to *D,* as shown in (b). If *C* senses the medium, it will hear a transmission and falsely conclude that it may not send to *D* (shown as a dashed line). In fact, such a transmission would cause bad reception only in the zone between *B* and *C*, where neither of the intended receivers is located. The problem is called the exposed terminal problem.

3)

A. (10 points) What are optimality principle and sink tree? Define based on the following diagram.



a) A network. (b) A sink tree for router *B*

One can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle. It states that if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route. To see this, call the part of the route from I to J r1 and the rest of the route r2. If a route better than r2 existed from J to K, it could be concatenated with r1 to improve the route from I to K, contradicting our statement that r1r2 is optimal. As a direct consequence of the optimality principle, the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a sink tree. Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. In the above diagram (b) is a sink tree for router B.

B. (10 points) Write drawbacks of the following routing algorithms.

- Distance vector routing
- Link state routing
- Hierarchical routing within a network
- Broadcast routing

Distance vector routing: Although it converges to the correct answer, it may do so slowly.

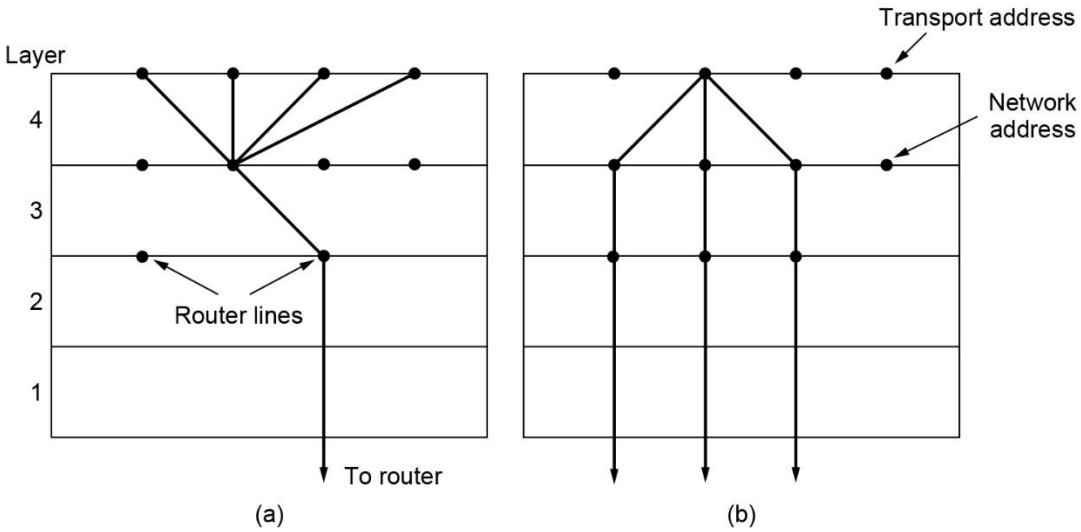
Link state routing: Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them, and more bandwidth is needed to send status reports about them.

Hierarchical routing within a network: There is a penalty to be paid: increased path length.

Broadcast routing: Each router must have knowledge of some spanning tree.

4)

A. (10 points) Explain multiplexing and inverse multiplexing according to the following diagram.

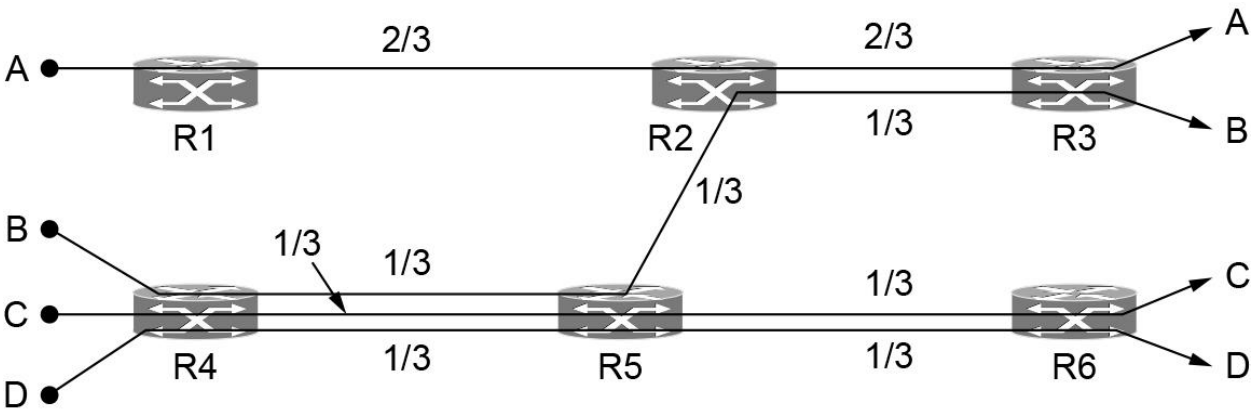


(a) Multiplexing. (b) Inverse multiplexing.

In the transport layer, the need for multiplexing can arise in a number of ways. For example, if only one network address is available on a host, all transport connections on that machine have to use it. When a segment comes in, some way is needed to tell which process to give it to. This situation, called multiplexing, is shown in the above figure. In this figure, four distinct transport connections all use the same network connection to the remote host.

Multiplexing can also be useful in the transport layer for another reason. Suppose, for example, that a host has multiple network paths that it can use. If a user needs more bandwidth or more reliability than one of the network paths can provide, a way out is to have a connection that distributes the traffic among multiple network paths on a round-robin basis, as indicated in the figure. This is called inverse multiplexing.

B. (10 points) Explain minmax fairness and use the following diagram as an example.



The form of fairness that is often desired for network usage is max-min fairness. An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation that is no larger. In the above diagram, each of the links between routers has the same capacity, taken to be 1 unit, though in the general case the links will have different capacities. If more of the bandwidth on the link between *R2* and *R3* is given to flow *B*, there will be less for flow *A*. This is reasonable as flow *A* already has more bandwidth. However, the capacity of flow *C* or *D* (or both) must be decreased to give more bandwidth to *B*, and these flows will have less bandwidth than *B*. Thus, the allocation is max-min fair.

5)

A. (10 points) Find differences between the following-

- Difference between HTTP and HTTPS
- Difference between static and dynamic content or object
- Difference between HTTP/1 and HTTP/1.1
- Difference between HTTP/1.1 and HTTP/2

Difference between HTTP and HTTPS: Both protocols HTTP and HTTPS (Secure HyperText Transfer Protocol) essentially retrieve objects in the same way. The HTTP standard to retrieve Web objects is evolving essentially independently from its secure counterpart, which effectively uses the HTTP protocol over a secure transport protocol called TLS (Transport Layer Security).

Difference between static and dynamic content or object: The page is a static content if it is a document that is the same every time it is displayed. In contrast, if it was generated on demand by a program or contains a program it is dynamic content.

Difference between HTTP/1 and HTTP/1.1: Establishing a separate TCP connection to transport each single icon became a very expensive way to operate. This observation led to HTTP/1.1, which supports persistent connections. With them, it is possible to establish a TCP connection, send a request and get a response, and then send additional requests and get additional responses. This strategy is also called connection reuse.

Difference between HTTP/1.1 and HTTP/2: Through a mechanism called server push, HTTP/2 allows the server to push out files that it knows will be needed but which the client may not know initially. Besides, in HTTP/2 the server can send smaller files first so that the browser can start displaying the files before having the larger files available. That is not allowed in HTTP/1.1.

B. (10 points) How do we transfer digital video through network? Write each step.

- The human eye has the property that when an image appears on the retina, the image is retained for some number of milliseconds before decaying.
- If a sequence of images is drawn at 50 images/sec, the eye does not notice that it is looking at discrete images.
- All video systems since the Lumière brothers invented the movie projector in 1895 exploit this principle to produce moving pictures.
- The simplest digital representation of video is a sequence of frames, each consisting of a rectangular grid of picture elements, or pixels.
- Most systems use 24 bits per pixel, with 8 bits each for the red, blue, and green (RGB) components.
- Red, blue, and green are the primary additive colors and every other color can be made from superimposing them in the appropriate intensity.

6)

A. (10 points) For each of the following applications, tell whether it would be (1) possible and (2) better to use a PHP script or JavaScript, and why:

- Displaying a calendar for any requested month since September 1752.
- Displaying the schedule of flights from Amsterdam to New York.
- Graphing a polynomial from user-supplied coefficients.

(a) There are only 14 annual calendars, depending on the day of the week on which 1 January falls and whether the year is a leap year. Thus, a JavaScript program could easily contain all 14 calendars and a small database of which year gets which calendar. A PHP script could also be used, but it would be slower.

(b) This requires a large database. It must be done on the server using PHP.

(c) Both works, but JavaScript is faster.

B. Explain the following Client-side socket programming.

```
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 8080                    /* arbitrary, but client & server must agree */
#define BUF_SIZE 4096                       /* block transfer size */


int main(int argc, char **argv)
{
  int c, s, bytes;
  char buf[BUF_SIZE];                       /* buffer for incoming file */
  struct hostent *h;                        /* info about server */
  struct sockaddr_in channel;               /* holds IP address */

  if (argc != 3) {printf("Usage: client server-name file-name0); exit(-1);}
  h = gethostbyname(argv[1]);               /* look up host's IP address */
  if (!h) {printf("gethostbyname failed to locate %s0, argv[1]); exit(-1;}

  s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (s <0) {printf("socket call failed0); exit(-1);}
  memset(&channel, 0, sizeof(channel));
  channel.sin_family= AF_INET;
  memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
  channel.sin_port= htons(SERVER_PORT);
  c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
  if (c < 0) {printf("connect failed0); exit(-1);}

  /* Connection is now established. Send file name including 0 byte at end. */
  write(s, argv[2], strlen(argv[2])+1);

  /* Go get the file and write it to standard output. */
  while (1) {
       bytes = read(s, buf, BUF_SIZE);      /* read from socket */
       if (bytes <= 0) exit(0);             /* check for end of file */
       write(1, buf, bytes);                /* write to standard output */
  }
}
```

The client code starts with some includes and declarations. Execution begins by checking to see if it has been called with the right number of arguments, where means the program was called with its name plus two arguments. Note that *argv*[1] contains the name of the server (e.g., *flits.cs.vu.nl*) and is converted to an IP address by *gethostbyname*. This function uses DNS to look up the name.

Next, a socket is created and initialized. After that, the client attempts to establish a TCP connection to the server, using *connect*. If the server is up and running on the named machine and attached to *SERVER_PORT* and is either idle or has room in its *listen* queue, the connection will (eventually) be established. Using the connection, the client sends the name of the file by writing on the socket. The number of bytes sent is one larger than the name proper, since the 0-byte terminating the name must also be sent to tell the server where the name ends.

Now the client enters a loop, reading the file block by block from the socket and copying it to standard output. When it is done, it just exits.