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

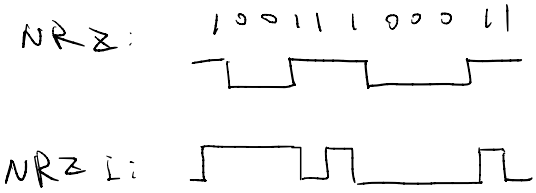
Answer any 5 of the 6 questions.

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?

A B ESC C ESC FLAG FLAG D

A B ESC ESC C ESC ESC ESC FLAG ESC FLAG D

D. (5 points) Explain Piggybacking.

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. 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.

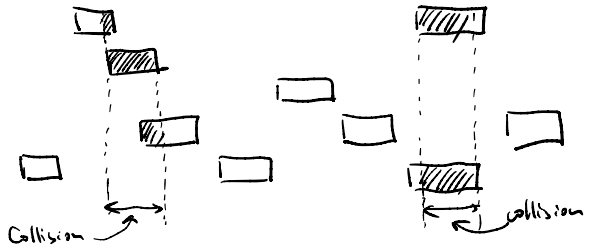
2)

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

Pure ALOHA: send the frame when ready, resend if the frame is destroyed

User

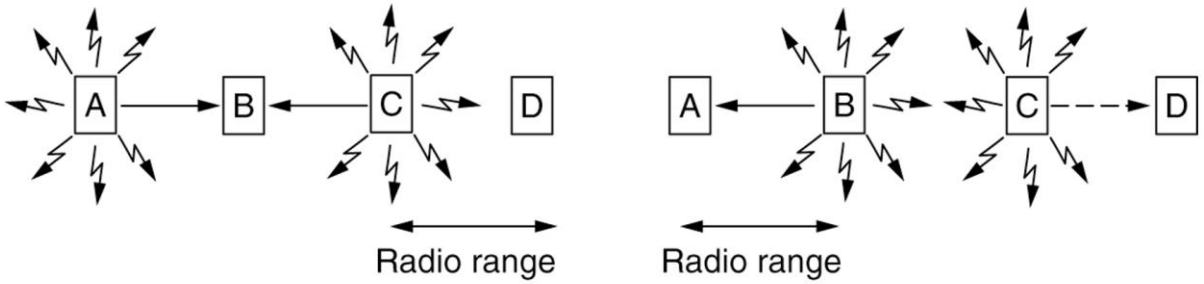
A
B
C
D
E



pure ALOHA is free.

Whenever two frames try to occupy the channel at the same time, there will be a collision (as see in the figure) 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 (i.e. have incorrect checksums) and both will have to be retransmitted later. The checksum does not (and should not) distinguish between a total loss and and a near miss. Bad is bad.

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

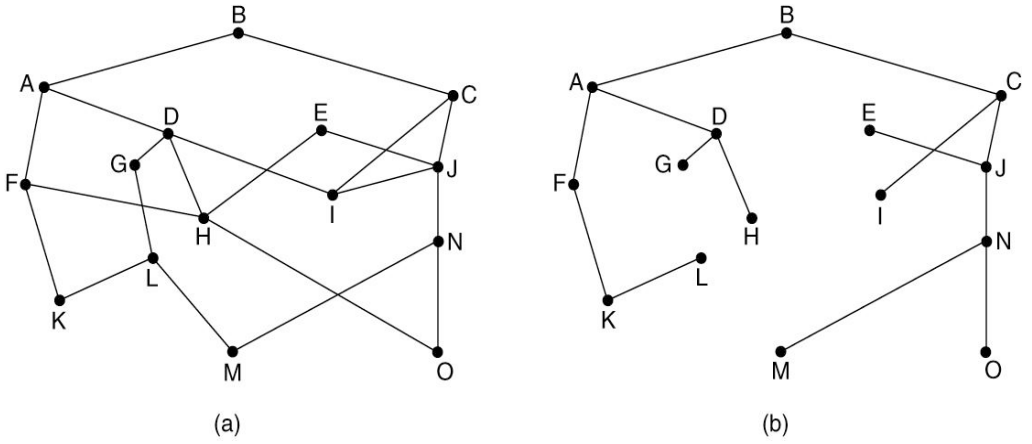


(a) A and C are hidden terminals to B, when transmit to B.
If A sends and then C immediately senses the medium, it will not hear A because A is out of its range. Thus C will falsely conclude that it can transmit to B. If C does starting transmitting, it will interfere at B, wiping out the frame from A. (We assume here that no CDMA-type scheme is used to provide multiple channels, so collisions garble the signal and destroy both frames.) We want a MAC protocol that will prevent this kind of collision from happening because it wastes bandwidth. 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.

(b) B and C are exposed terminals when transmitting to A and D.
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. We want a MAC protocol that prevents this kind of deferral from happening because it waste bandwidth. 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*

Sol: • One can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle.

e.g. If a route from node A to node B is optimal, then every sub-path of that route must also be optimal. That is, if the path A to B is the best path from A to C, then the path from B to C must also be the best path from B to C.

• As a direct consequence of the optimality, 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.

e.g (b) represents a sink tree for router B, which means that for every node in the network, the path from B to that node in the sink tree is as short as possible. It's a subset of the network that highlights the optimal paths from the sink (or root) to all other nodes.

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.

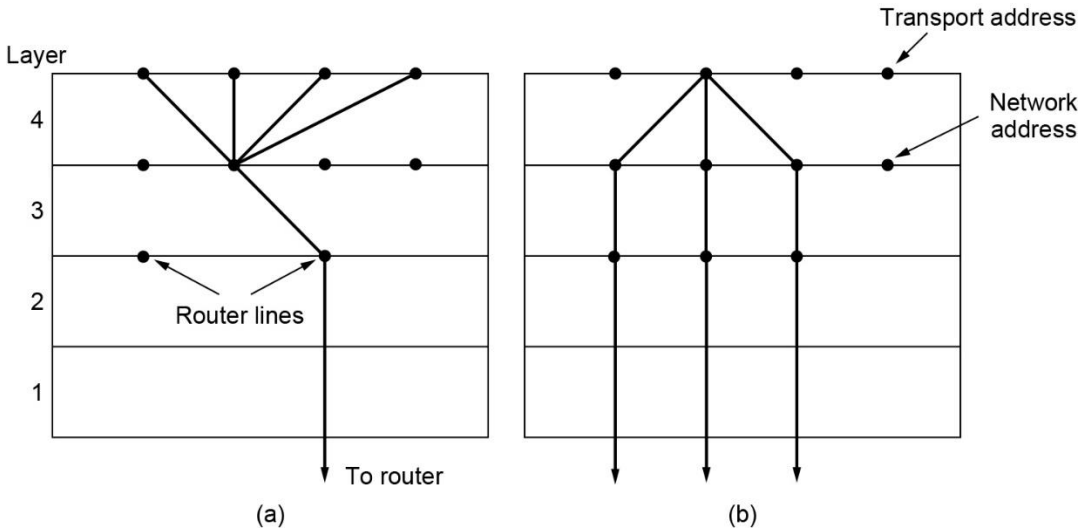
Link state routing: Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them, and more bandwidth is needed to send status reports about them.

Hierarchical routing within a network: There is a penalty to be paid: increased path length. For example, the best route from 1A to 5C is via region 2, but with hierarchical routing all traffic to region 5 goes via region 3, because that is better for most destinations in region 5.

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.

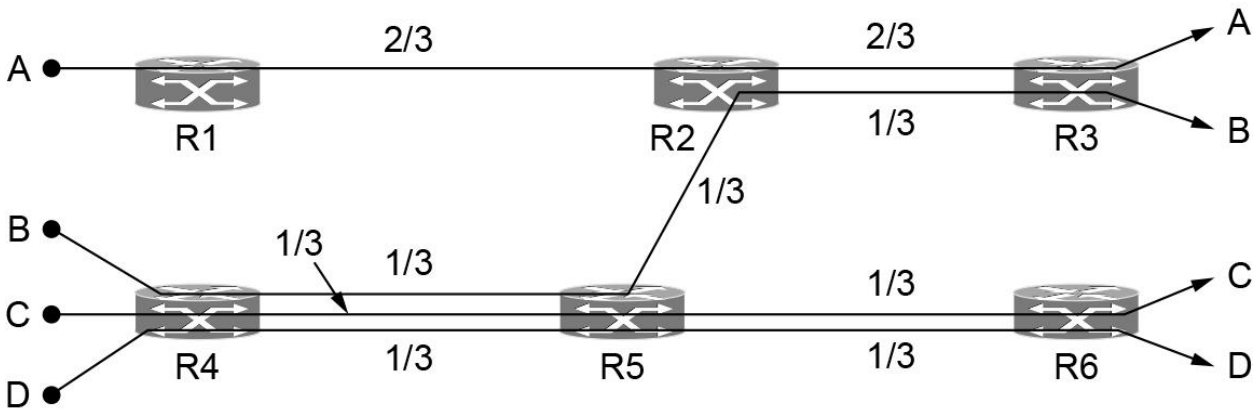


(a) Multiplexing. (b) Inverse multiplexing.

(a) 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 Fig (a). In this figure, four distinct transport connections all use the same network connection. (eg. IP address) to the remote host.

(b) 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. 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.
- 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

•A max-min fair allocation is shown for a network with four flows, A, B, C, and D. Each of the links between routers has the same capacity, taken to be 1 unit, though in the general case the link will have different capacities, taken to be 1 unit, though in the general case the links will have different capacities. Three flows compete for the bottom-left link between router R4 and R5. Each of these flows therefore gets 1/3 of the link. The remaining flow, A competes with B on the link from R2 to R3. Since B has an allocation of 1/3, A gets the remaining 2/3 of the link. Notice that all of the other links have spare capacity. However, this capacity cannot be given to any of the flows without decreasing the capacity of another, low flow. 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

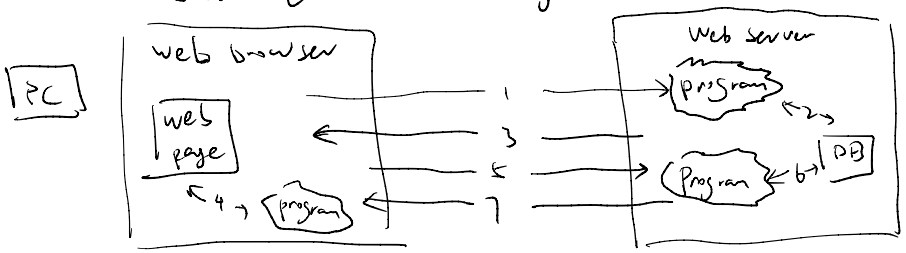
HTTP vs HTTPS: The protocol that is used to transport all the information between Web servers and clients is HTTP. Both protocols HTTP and HTTPS (Secure HyperText Transfer Protocol) essentially retrieve objects in the same way.

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)

Static vs dynamic content or object:

static objects: files sitting on a server, presenting themselves in the same way each time they are fetched and viewed.
e.g logo, style sheets, header and footer.
Generally amenable to caching.

dynamic: This is content that can change based on user interaction, location, permission etc.



The requests and responses happen in the background; the user may not may not even be aware of them because the page URL and title typically do not change. By including client-side programs, the page can present a more responsive interface than with server-side programs alone.

HTTP.1 vs HTTP1.1

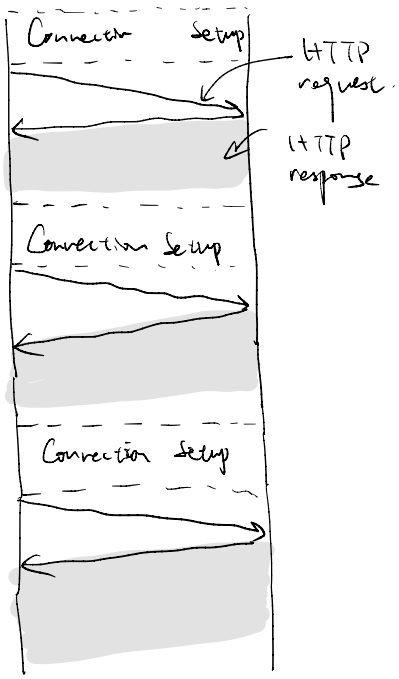
HTTP.1: Supports only one request per connection and does not support persistent connections, leading to more bandwidth waste and less efficient cache management.

HTTP1.1: 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.

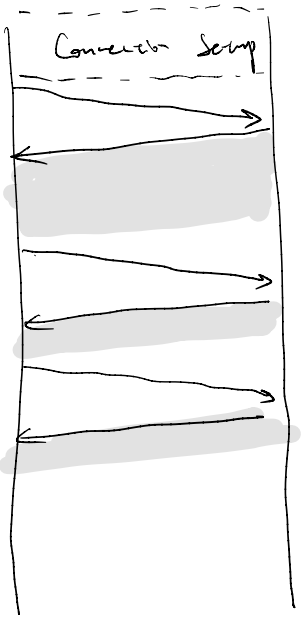
HTTP1.1 vs HTTP2:

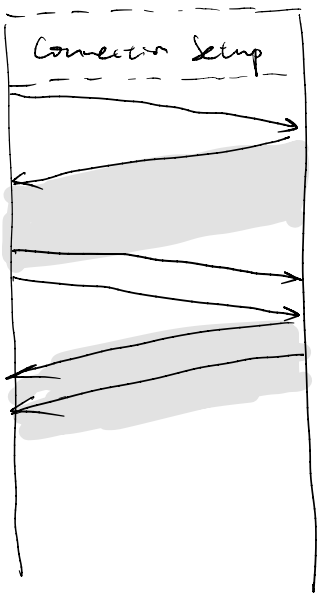
HTTP 1.1: It is characterized by loading resources one after the other, which can lead to one resource blocking another if there are issues with loading.

HTTP2: It allows a single TCP connection to send multiple streams of data at once, Server push, which sends content to clients before they request it, and HPack header compression for faster loading.

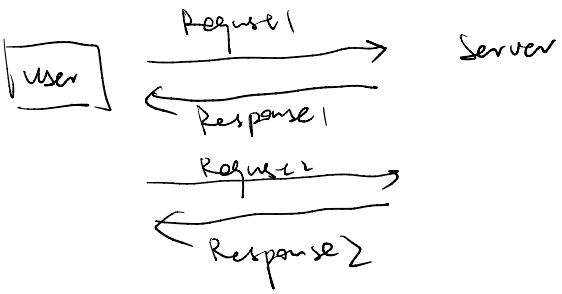


HTTP

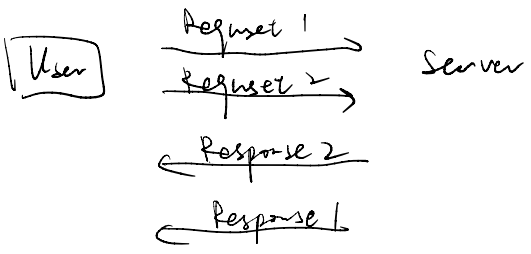




HTTP1.1

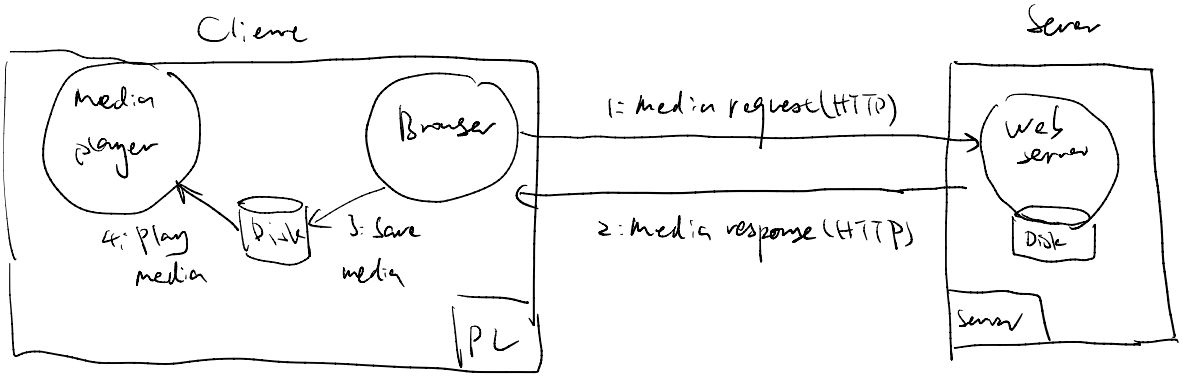


HTTP1.1



HTTP2

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



Step 1: it sends an HTTP request for the movie to the Web server to which the movie is linked.

step 2: The server fetches the movie (which is just a file in MP4 or some other format) and sends it back to browser

step 3: The browser then saves the entire movie to a scratch file on disk in step 3.

step 4: Finally, in step 4 the media player starts reading the file and play the movie.

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.

① JavaScript. Because 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 with calendar.

② PHP: This requires a large database. It must be done on the server by using PHP.

③ Both work, 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 */
  }
}
```

1. Including Necessary Libraries: The program begins by including headers like '<sys/types.h>'... These provide essential functions and definitions for network operations, standard input/output and standard library functions.

2. Defining Constants: Constants like 'SERVER_PORT' (the port number to connect to) and 'BUF_SIZE' (the size of the buffer for data transfer) are defined.

3. Main Function: The program's execution starts in the 'main' function. It expects command-line arguments ('argc', 'argv') specifying the server name and the file name to request.

4. Validating arguments: The program checks if the correct number of arguments is provided. If not, it exits with an error message.

5. Resolving the Server's IP address: The 'gethostbyname' function is used to convert the server's hostname (provided as an argument) into its IP address. If the hostname cannot be resolved, the program exits with an error.

6. Creating a socket: A socket is created using the 'socket' function, which returns a socket descriptor. This socket is used for TCP communication ('SOCK_STREAM')

7. Setting up Server Address information: The 'sockaddr_in' structure is set up with the server's IP address and port number. ; The IP address is obtained from the 'hostent' structure, and the port number is set using the 'SERVER_PORT' constant.

8. Connecting to the Server: The 'connect' function initiates a connection to the server using the created socket and the 'sockaddr_in' structure.

9. Sending Data to the Server: The data to the Server: the program sends the filename (from command line arguments) to the server using 'write' function.

10. Receiving Data from the Server: The program enters a loop, continuously reading data from the server using the 'read' function and writing it to standard output. The loop continues until 'read' returns 0 (or less), indicating the end of the file or an error.

11. Closing the Connection: After the file transfer is complete, the connection is usually closed. However, in this provided code, the connection is not explicitly closed, which is typically done using the 'close' function.