University Of Central Punjab

Faculty of Information Technology

FINAL TERM EXAMINATION

Course Title: Computer Communications and Networks Lab [CSNC2411]	**Semester: Fall 2022**
Time Allowed: 90 Minutes (1.5 HOURS)	**Total Marks: 50**

Name: _____Reg No: _____ Sec__

Instructions

1. Write your Name, Registration number and section on the Word file.
2. Submit only one docx file.
3. Attach the screenshot and code where required.
4. Calculators are not allowed.
5. Attempt all questions.
6. There are two submissions ;one for windows and another for Ubuntu.

Q1. **You are required to create an concurrent TCP client-server connection with the following functionality. (Helping material code with all relevant networking-calls syntaxes and libraries is given at the end of the sheet)** **[25]**

Write a C program that takes a sentence as input and prints the number of words in it.

Client 1

1- The Client will ask the user to type any string (e.g How are you)

2- The Client will send this string to the Server.

Client 2

1- The Client will ask the user to type any string (e.g Welcome to final exam)

2- The Client will send this string to the Server.

Server side:

1- The Server will initially wait for the Client to receive string

2- After receiving the string, the Server will display the number of words in the both string

e.g(Number of words are 7)

Q. No.2**: Packet Tracer**

- Attach the screenshot below the highlighted questions [25]

Objectives

Part 1: Examine HTTP Web Traffic

Background

This simulation activity is intended to provide a foundation for understanding the TCP/IP protocol suite and the relationship to the OSI model. Simulation mode allows you to view the data contents being sent across the network at each layer.

As data moves through the network, it is broken down into smaller pieces and identified so that the pieces can be put back together when they arrive at the destination. Each piece is assigned a specific name (protocol data unit [PDU]) and associated with a specific layer of the TCP/IP and OSI models. Packet Tracer simulation mode enables you to view each of the layers and the associated PDU. The following steps lead the user through the process of requesting a web page from a web server by using the web browser application available on a client PC.

Even though much of the information displayed will be discussed in more detail later, this is an opportunity to explore the functionality of Packet Tracer and be able to visualize the encapsulation process.

Part 1: Examine HTTP Web Traffic

In Part 1 of this activity, you will use Packet Tracer (PT) Simulation mode to generate web traffic and examine HTTP.

Step 1: Switch from Realtime to Simulation mode.

In the lower right corner of the Packet Tracer interface are tabs to toggle between **Realtime** and **Simulation** mode. PT always starts in **Realtime**mode, in which networking protocols operate with realistic timings. However, a powerful feature of Packet Tracer allows the user to “stop time” by switching to Simulation mode. In Simulation mode, packets are displayed as animated envelopes, time is event driven, and the user can step through networking events.

a. Click the **Simulation** mode icon to switch from **Realtime** mode to **Simulation** mode.

b. Select **HTTP** from the **Event List Filters**.

1) HTTP may already be the only visible event. Click **Edit Filters** to display the available visible events. Toggle the **Show All/None** check box and notice how the check boxes switch from unchecked to checked or checked to unchecked, depending on the current state.

2) Click the **Show All/None** check box until all boxes are cleared and then select **HTTP**. Click anywhere outside of the **Edit Filters** box to hide it. The Visible Events should now only display HTTP.

Step 2: Generate web (HTTP) traffic.

Currently the Simulation Panel is empty. There are six columns listed across the top of the Event List within the Simulation Panel. As traffic is generated and stepped through, events appear in the list. The **Info** column is used to inspect the contents of a particular event.

Note: The Web Server and Web Client are displayed in the left pane. The panels can be adjusted in size by hovering next to the scroll bar and dragging left or right when the double-headed arrow appears.

a. Click **Web Client** in the far left pane.

b. Click the **Desktop** tab and click the **Web Browser** icon to open it.

c. In the URL field, enter **www.osi.local** and click **Go**.

Because time in Simulation mode is event-driven, you must use the **Capture/Forward** button to display network events.

d. Click **Capture/Forward** four times. There should be four events in the Event List.

Look at the Web Client web browser page. Did anything change?

Step 3: Explore the contents of the HTTP packet.

a. Click the first colored square box under the **Event List** > **Info** column. It may be necessary to expand the **Simulation Panel** or use the scrollbar directly below the **Event List**.

The **PDU Information at Device: Web Client** window displays. In this window, there are only two tabs (**OSI Model** and **Outbound PDU Details**) because this is the start of the transmission. As more events are examined, there will be three tabs displayed, adding a tab for **Inbound PDU Details**. When an event is the last event in the stream of traffic, only the **OSI Model** and **Inbound PDU Details** tabs are displayed.

b. Ensure that the **OSI Model** tab is selected. Under the **Out Layers** column, ensure that the **Layer 7** box is highlighted.

What is the text displayed next to the **Layer 7** label?

What information is listed in the numbered steps directly below the **In Layers** and **Out Layers** boxes?

c. Click **Next Layer**. Layer 4 should be highlighted. What is the **Dst Port** value?

d. Click **Next Layer**. Layer 3 should be highlighted. What is the **Dest. IP** value?

e. Click **Next Layer**. What information is displayed at this layer?

f. Click the **Outbound PDU Details** tab.

Information listed under the **PDU Details** is reflective of the layers within the TCP/IP model.

Note: The information listed under the **Ethernet II** section provides even more detailed information than is listed under Layer 2 on the **OSI Model** tab. The **Outbound PDU Details** provides more descriptive and detailed information. The values under **DEST MAC** and **SRC MAC** within the **Ethernet II** section of the **PDU Details** appear on the **OSI Model** tab under Layer 2, but are not identified as such.

What is the common information listed under the **IP** section of **PDU Details** as compared to the information listed under the **OSI Model** tab? With which layer is it associated?

What is the common information listed under the **TCP** section of **PDU Details**, as compared to the information listed under the **OSI Model** tab, and with which layer is it associated?

What is the **Host** listed under the **HTTP** section of the **PDU Details**? What layer would this information be associated with under the **OSI Model** tab?

g. Click the next colored square box under the **Event List** > **Info** column. Only Layer 1 is active (not grayed out). The device is moving the frame from the buffer and placing it on to the network.

h. Advance to the next HTTP **Info** box within the **Event List** and click the colored square box. This window contains both **In Layers** and **Out Layers**. Notice the direction of the arrow directly under the **In Layers** column; it is pointing upward, indicating the direction the information is travelling. Scroll through these layers making note of the items previously viewed. At the top of the column the arrow points to the right. This denotes that the server is now sending the information back to the client.

Comparing the information displayed in the **In Layers** column with that of the **Out Layers** column, what are the major differences?

i. Click the **Outbound PDU Details** tab Scroll down to the **HTTP** section.

What is the first line in the HTTP message that displays?

j. Click the last colored square box under the **Info** column. How many tabs are displayed with this event and why?

Code:

In the client side add the ip address (127.0.0.1) at the time of run
e.g gcc echoClient.c -o echoClient
./echoClient 127.0.0.1

Server side:
gcc echoServer.c -o echoServer
./echoServer

Client Code:

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


#define MAXLINE 4096 /*max text line length*/
#define SERV_PORT 3000 /*port*/

int
main(int argc, char **argv)
{
 int sockfd;
 struct sockaddr_in servaddr;
 char sendline[MAXLINE], recvline[MAXLINE];

 //basic check of the arguments
 //additional checks can be inserted
 if (argc !=2) {
  perror("Usage: TCPClient <IP address of the server");
  exit(1);
 }

 //Create a socket for the client
 //If sockfd<0 there was an error in the creation of the socket
 if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) <0) {
  perror("Problem in creating the socket");
  exit(2);
 }

 //Creation of the socket
 memset(&servaddr, 0, sizeof(servaddr));
```

```
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr= inet_addr(argv[1]);
servaddr.sin_port =  htons(SERV_PORT); //convert to big-endian order

//Connection of the client to the socket
if (connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr))<0) {
 perror("Problem in connecting to the server");
 exit(3);
}

while (fgets(sendline, MAXLINE, stdin) != NULL) {

 send(sockfd, sendline, strlen(sendline), 0);

 if (recv(sockfd, recvline, MAXLINE,0) == 0){
  //error: server terminated prematurely
  perror("The server terminated prematurely");
  exit(4);
 }
 printf("%s", "String received from the server: ");
 fputs(recvline, stdout);
}

 exit(0);
}
```

Server Code:

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


#define MAXLINE 4096 /*max text line length*/
#define SERV_PORT 3000 /*port*/
#define LISTENQ 8 /*maximum number of client connections*/

int main (int argc, char **argv)
{
 int listenfd, connfd, n;
 pid_t childpid;
```

```
socklen_t clilen;
char buf[MAXLINE];
struct sockaddr_in cliaddr, servaddr;

//Create a socket for the soclet
//If sockfd<0 there was an error in the creation of the socket
if ((listenfd = socket (AF_INET, SOCK_STREAM, 0)) <0) {
 perror("Problem in creating the socket");
 exit(2);
}


//preparation of the socket address
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(SERV_PORT);

//bind the socket
bind (listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));

//listen to the socket by creating a connection queue, then wait for clients
listen (listenfd, LISTENQ);

printf("%s\n","Server running...waiting for connections.");

for ( ; ; ) {

 clilen = sizeof(cliaddr);
 //accept a connection
 connfd = accept (listenfd, (struct sockaddr *) &cliaddr, &clilen);

 printf("%s\n","Received request...");

 if ( (childpid = fork ()) == 0 ) {//if it's 0, it's child process

   printf ("%s\n","Child created for dealing with client requests");

   //close listening socket
   close (listenfd);

   while ( (n = recv(connfd, buf, MAXLINE,0)) > 0)  {
    printf("%s","String received from and resent to the client:");
    puts(buf);
    send(connfd, buf, n, 0);
   }
```

```
    if (n < 0)
      printf("%s\n", "Read error");
    exit(0);
  }
  //close socket of the server
  close(connfd);
 }
}
```