|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| C:\Users\saif\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\final design.jpg | **Course:** | **Computer Networks** | **Course Code:** | **CL-307** |
| **Program:** | **BS (Computer Science)** | **Semester:** | **Spring 2018** |
| **Duration:** | **2 hr 45 min** | **Total Marks:** | **30** |
| **Paper Date:** | **11-May-2018** | **Weight** | **30 %** |
| **Section:** | **A** | **Page(s):** | **5** |
| **Exam:** | **Final** | **Reg. No.** |  |
| **Instructions:** | **READ ALL INSTRUCTIONS CAREFULLY.**  1. Understanding the question paper is also part of the exam, so do not ask any clarification. Make suitable ASSUMPTIONS.  2**.** Final Submissionsshould be done in your respective section folder on **sandata/xeon/Spring2018/Computer Network**. Each question related files must be in **separate folder (Question 1, Question 2, Question 3)** and all the separate folders must be in a single zip file. **Zip file must be renamed after your roll number e.g., “14L-4125”. Multiple submissions are not allowed** **(if done, only first one will be considered)**  3. You are immediately disqualified from the exam if:  i. You are seen talking, whispering, borrowing or looking at someone’s PC  ii. A USB is found attached to your PC  iii. You are caught accessing internet. | | | |

**Part 1 TCP SOCKET PROGRAMMING (Marks: 6)**

**\*\*\*\*Submission: You have to submit your (Roll-No\_Client.c) and (Roll-No\_Server.c) files in a folder named Question 1\*\*\*\***

Write the searching program for phone directory in which you are required to write a TCP based server client code in C language.

**Server Side**:

* Each client can request data multiple times, until the client sends **‘Exit’** command. Your server utilizes multithreading to deal with infinite number of clients
* Assigns a new thread for each client that connects
* The server has access to a phone directory (**phonebook.txt**) in which all the contact information of people are saved.
* Upon receiving the request form a client, your program should fetch the required information from the text file and send it to the client.

**Client Side**:

Each client logs in to the server and can ask for contact details of a particular person (i.e house address, phone number etc) from server by entering the **Name** of the person.

**For Example:**

**Name:** Ahmed

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Name** | **Address** | **Phone Number** |
| 1. | Ahmed Arshad | ABC Street C-Block Lahore | 042-3555111 |
| 2. | Ahmed | XYZ house Lahore | 042-3786667 |

If the directory includes more than one person with same name, the detail of all the people with same name should be displayed.

On client side the information returned by the server to the client should be displayed on the terminal as well as saved in the text file in **proper format** as mentioned in the example above.

**Note: Each field in text file is spaced using tab.**

**Multithreading Socket Programming Syntax**

**Socket Programming:**

* int socket(int domain, int type, int protocol);
* domain = AF\_INET, AF\_INET6
* type = SOCK\_STREAM, SOCK\_DGRAM
* protocol = 0(preferred), IPPROTO\_TCP, IPPROTO\_UDP, IPPROTO\_ICMP
* int bind(int socket, struct sockaddr \*name, int namelen)
* struct sockaddr\_in {

short sin\_family; // e.g. AF\_INET, AF\_INET6

unsigned short sin\_port; // e.g. htons(3490)

struct in\_addr sin\_addr; // see struct in\_addr, below

char sin\_zero[8]; // zero this

};

* struct in\_addr {

unsigned long s\_addr; // load with inet\_addr()

};

* struct sockaddr {

unsigned short sa\_family; // address family, AF\_xxx

char sa\_data[14]; // 14 bytes of protocol address

};

* int listen(int socket, int backlog)
* int accept(int socket, struct sockaddr \*addr, int \*addrlen)
* int connect(int socket, struct sockaddr \*addr, int addrlen)
* int send(int socket, const void \*buf, int buflen, int flags);
* int recv(int socket, void \*buf, int buflen, int flags);
* int sendto(int socket, const void \*buf, int buflen, int flags, struct sockaddr\* to, int tolen);
* int recvfrom(int socket, void \*buf, int buflen, int flags, struct sockaddr\* from, int \*fromlen);
* int close(int socket)

**Multithreading:**

* int pthread\_create(pthread\_t \*thread, pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg);
* void pthread\_exit(void \*value\_ptr);
* int pthread\_join(pthread\_t thread, void \*\*value\_ptr);

**Part 2 Network Simulator 2 (Marks: 10)**

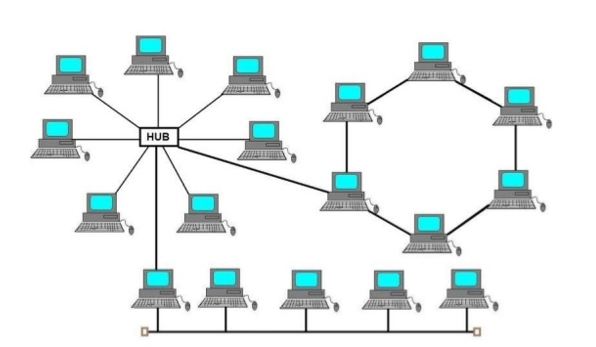
**\*\*\*\*Submission: You have to submit your (Roll-No.tcl) file in a folder named Question 2. You should provide screen shots of your working code along with the tcl file \*\*\*\***

You will have to create a hybrid topolgy as given in the diagram below using statements in correct format from ns2 to implement the Distance vector routing protocol. Assume all the devices in the topology as nodes (including HUB) and all the wires as duplex links having a capacity of **1.5Mb** and a propagation delay of **10ms** with a stochastic fair queue scheduling algorithm. You have to orient the nodes as shown in the topology below. You will have to send TCP data from **N3** to **N11**. Also you will have to send UDP data with a rate of **500 packets/sec** with a single packet having a size of **500 Bytes** from **N19** to **N7**.

**Note: Implement the task using less number of statements to get the full marks**

**Scheduling Events:**

* TCP Data starts at 0.2 and stops at 1.8
* UDP Data starts at 0.4 and stops at 1.6
* Bring the link between **N9** and **N10** down at 0.7 and bring it back up at 1.0
* Bring the link between **N1** and **N7** down at 0.9 and bring it back up at 1.3
* Stop the simulation at 2.0



**NS2 Syntax:**

**Create Simulation:** set ns [new Simulator]

**Trace Files for NAM**: set nf [open out.nam w]

$ns namtrace-all $nf

**Finish Procedure:** proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec nam out.nam &

exit 0

}

**Routing Algorithm**: $ns rtproto <protocol\_name>; <protocol\_name>: DV

**Node creation**: set <node\_name> [$ns node]

**Links Creation**: $ns <link\_type> <node1> <node2> <Bandwidth> <Delay> <queue\_type>

<link\_type>: simplex-link, duplex-link; <queue\_type>: DropTail, SFQ

**Graphical Settings (NAM)**: $ns <type> <node1> <node2> <option> <args>

<type> : simplex-link-op, duplex-link-op; <option> : orient, queuePos

**Transport Layer**: set <layer\_name> [new Agent/<agent\_type>]

<agent\_type>: UDP,TCP,Null,TCPSink

**Attaching Transport layer:** $ns attach-agent <node\_name> <layer\_name>

**Connecting Transport layer:** $ns connect <layer\_name> <layer\_name>

**File Transfer Protocol:** set <ftp\_name> [new Application/FTP]

**FTP Attach Agent:** <ftp\_name> attach-agent <layer\_name>

**Constant Bit Rate:** set <cbr\_name> [new Application/Traffic/CBR]

**CBR Attach Agent:** <cbr\_name> attach-agent <layer\_name>

**CBR Parameters:** <cbr\_name>set <parameter> <parameter\_value>

<parameter>: packetSize\_, interval\_, rate\_

**Event Scheduling:** $ns at <time\_frame\_value> “<cbr\_name>/<ftp\_name> <time\_event>”

<time\_event>: start, stop

**Ending Simulation:** proc finish { } { Finish Procedure Commands }

$ns at <time\_frame\_value> “finish”

**Run Simulation:** $ns run

**Link Up/Down:** $ns rtmodel-at <time\_frame\_value> <function> <node1> <node2>

<function>: up,down

**Part 3 CISCO PACKET TRACER (Marks: 14)**

**\*\*\*\*\*Submission: You have to submit your (Roll-No.pkt) file along with the screen shots of the CLI Code (Only for the configuration of RIPv2) of all the routers in a folder named Question 3 \*\*\*\*\***

There are three startup companies named **CMP X**, **CMP Y** and **CMP Z**. You have to design a network solution for merger of these three companies (i.e., all the PCs in each company must be able to communicate with each other). All the three startup companies have **three rooms** with each room having **2 PCs** each. The IP regulating company has assigned following network addresses to three companies:

**CMP X: 184.86.92.0/23** **CMP Y: 122.72.0.0/13** **CMP Z: 220.91.121.160/27**

All the PCs in single room must be on same sub network and all the rooms of a single company must be on a different sub-network which will be assigned after sub-netting the above given **network address** only for the relevant company (no outside network or the network of other company will be accepted) e.g., each room for CMP X will be assigned a different subnetwork after sub-netting the address of 184.86.92.0/23 only and not any other network address. The number of subnets in your design should be limited rather than the number of hosts per subnet.

All the three companies have also agreed to buy one network address for the communication between the routers which will be connecting different Inter-Company and Intra-Company subnets. The communication between different routers must be accomplished by sub-netting the given network address below:

**Routers Serial Communication: 220.91.121.80/28**

Make sure that you optimally design the network considering the number of devices (switches, routers etc.) used and how you are assigning the IP addresses to different subnets in your design.

1. Use Straight Through wires, Cross Over cables or Serial DCE wires where necessary and applicable
2. Use **Generic (Empty)** Router, **2950-24** switches and **Generic** PCs
3. You will have to add interfaces (Fast Ethernet/Serial) manually in your router
4. You can add a maximum of 2 Fast Ethernet Interfaces and a maximum of 2 Serial Interfaces in a single router to be used with copper media (no other interface should be used)
5. You have to assign IPs to the machines using **Static IP allocation**
6. Although you have to use GUI of the router to configure its interfaces but you must use CLI of the router to configure the **RIPv2 protocol** for **Classless addressing** and attach screen shots of the CLI code (You can use snipping tool to take screen shots or you can attach a text file having the CLI code of each router).
7. Clearly mention each subnet address using comments.

Make your design as neat as possible and properly add comments in your design to get the full credit.