Lab Session1: Socket programming

Refer to following links for more information:

1.https://youtu.be/eVYsIolL2gE
2.https://youtu.be/xfRdYrQUQeQ
3https://youtu.be/d9pmc7oObkw

Sample programs:

1. Write a c program to demonstrate the working of UDP echo Client/Server.

```
// server program for udp connection
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000

// Server code
int main()
{
        char buffer[100];
        int servsockfd, len,n;
        struct sockaddr_in servaddr, cliaddr;
        bzero(&servaddr, sizeof(servaddr));

        // Create a UDP Socket
        servsockfd = socket(AF_INET, SOCK_DGRAM, 0);
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(PORT);
        servaddr.sin_family = AF_INET;
```

```
        // bind server address to socket descriptor
        bind(servsockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));


        //receive the datagram
        len = sizeof(cliaddr);
        n = recvfrom(servsockfd, buffer, sizeof(buffer),0, (struct sockaddr*)&cliaddr,&len);
        buffer[n] = '\0';
        puts(buffer);
//Echoing back to the client
                        sendto(servsockfd, buffer, n, 0, (struct sockaddr*)&cliaddr, sizeof(cliaddr));
                getchar();


        // close the descriptor
        close(sockfd);
}
```

// udp client driver program

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

#define PORT 5000
#define MAXLINE 1000
```

```
// Driver code
int main()
{
        char buffer[100];
        char *message = "Hello Server";
        int sockfd, n,len;
        struct sockaddr_in servaddr, cliaddr;

        // clear servaddr
        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(PORT);
        servaddr.sin_family = AF_INET;

        // create datagram socket
        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)&servaddr,
sizeof(servaddr));
        len=sizeof(cliaddr);
        // waiting for response
        n=recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct
sockaddr*)&cliaddr,&len );
      buffer[n]='\0';
        printf("message fromser is %s \n",buffer);
      getchar();

        // close the descriptor
        close(sockfd);
}
```

2. Write a c program to demonstrate the working of TCP client server as follows: After connection set up client send a message. Server will reply to this. If server decides to close the program then it will send a message exit to client then closes itself. Client will close after receiving this message.. (Note: In each

program there is a function that handles the client and server function and main program is responsible for socket creation and connection setup.)

// TCP Server program

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// Function designed for chat between client and server.
void servfunc(int sockfd)
{
        char buff[MAX];
        int n;
        // infinite loop for chat
        for (;;) {
                bzero(buff, MAX);

                // read the message from client and copy it in buffer
                read(sockfd, buff, sizeof(buff));
                // print buffer which contains the client contents
                printf("From client: %s\t To client : ", buff);

                bzero(buff, sizeof(buff));
// Read server message from keyboard in the buffer
```

```
                n=0;
                while ((buff[n++] = getchar()) != '\n')
                        ;
// and send that buffer to client
                write(sockfd, buff, sizeof(buff));

                // if msg contains "Exit" then server exit and session ended.
                if (strncmp("exit", buff, 4) == 0) {
                        printf("Server Exit...\n");
                        break;
                }
        }
}

// Driver function
int main()
{
        int sockfd, connfd, len;
        struct sockaddr_in servaddr, cli;

        // socket create and verification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
                printf("socket creation failed...\n");
                exit(0);
        }
        else
                printf("Socket successfully created..\n");
        bzero(&servaddr, sizeof(servaddr));

        // assign IP, PORT
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
```

```
servaddr.sin_port = htons(PORT);

// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
}
else
        printf("Socket successfully binded..\n");

// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0) {
        printf("Listen failed...\n");
        exit(0);
}
else
        printf("Server listening..\n");
len = sizeof(cli);

// Accept the data packet from client and verification
connfd = accept(sockfd, (SA*)&cli, &len);
if (connfd < 0) {
        printf("server acccept failed...\n");
        exit(0);
}
else
        printf("server acccept the client...\n");

// Function for chatting between client and server
servfunc(connfd);

// After sending exit message close the socket
close(sockfd);
```

```
}
```

//TCP Client program

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void clifunc(int sockfd)
{
	char buff[MAX];
	int n;
	for (;;) {
		bzero(buff, sizeof(buff));
		printf("Enter the string : ");
		n = 0;
		while ((buff[n++] = getchar()) != '\n')
			;
		write(sockfd, buff, sizeof(buff));
		bzero(buff, sizeof(buff));
		read(sockfd, buff, sizeof(buff));
		printf("From Server : %s", buff);
		if ((strncmp(buff, "exit", 4)) == 0) {
			printf("Client Exit...\n");
			break;
		}
	}
```

```
}

int main()
{
	int sockfd, connfd;
	struct sockaddr_in servaddr, cli;

	// socket create and verification
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd == -1) {
		printf("socket creation failed...\n");
		exit(0);
	}
	else
		printf("Socket successfully created..\n");
	bzero(&servaddr, sizeof(servaddr));

	// assign IP, PORT
	servaddr.sin_family = AF_INET;
	servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
	servaddr.sin_port = htons(PORT);

	// connect the client socket to server socket
	if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
		printf("connection with the server failed...\n");
		exit(0);
	}
	else
		printf("connected to the server..\n");

	// function for client
	clifunc(sockfd);
```

```
        // close the socket
        close(sockfd);
}
```

3. TCP Concurrent Client\Server

```
//Server program
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>
int main()
{
        int sd,nd,n,len,reult;
        struct sockaddr_in seraddress, cliaddr;
        char buf[256];
        sd=socket(AF_INET, SOCK_STREAM,0);
        seraddress.sin_family=AF_INET;
        seraddress.sin_addr.s_addr=INADDR_ANY;
        seraddress.sin_port=htons(10200);
        bind(sd,(struct sockaddr*)&seraddress,sizeof(seraddress));
        listen(sd,5);
        len=sizeof(cliaddr);
        while(1)
        {
```

```
        nd=accept(sd,(struct sockaddr*)&cliaddr,&len);
        if (fork()==0){
                        close(sd);
        n=read(nd,buf,sizeof(buf));
        printf("message from client %s\n",buf);
        getchar();}
        close(nd);
}

//TCP Client program:
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>
int main()
{
        int sd,nd,n,len,reult,n1;
        struct sockaddr_in seraddress, cliaddr;
        char buf[256], buf1[256];
        sd=socket(AF_INET, SOCK_STREAM,0);
        seraddress.sin_family=AF_INET;
        seraddress.sin_addr.s_addr=INADDR_ANY;
        seraddress.sin_port=htons(10200);
        len=sizeof(seraddress);
        connect(sd,(struct sockaddr*)&seraddress,len);
```

```
    printf("enter the message tosen \n");
    gets(buf);
    n=write(sd,buf,strlen(buf));
  n1=read(sd,buf1,sizeof(buf1));
    printf("message from ser %s\n",buf1);
    getchar();


}
```

Exercise problems:

1. Write a UDP client-server program where client sends rows of a matrix to the server combines them together as a two dimensional matrix and display the same.

2. Write a TCP client which sends a string to a server program. Server displays the string along with client IP and ephemeral port number. Server then responds to the client by echoing back the string in uppercase. The process continues until one of them types “QUIT”.

3. Implement concurrent Remote Math Server To perform arithmetic operations in the server and display the result at the client. The client accepts two integers and an operator from the user and sends it to the server. The server will performs the operation on integers and sends result back to the client which is displayed in the client.

4. DayTime Server: Where client sends request to time server to send current time. Server responds by sending the current time . [Hint: read man pages of asctime() and localtime()] . Display server process id at client side along with time.

Lab Session 2: Packet Analysis with Wireshark

(Refer Lab Manual for details. Questions are numbered according to Lab Manual)

Refer to following links more more information:

1. https://youtu.be/DCqbOhWSFus
2. https://youtu.be/lb1Dw0elw0Q

PART-1 STUDY OF APPLICATION LAYER PROTOCOLS USING WIRESHARK.

Q 3.1. Retrieve web pages using HTTP. Use Wireshark to capture packets for analysis. Learn about most common HTTP messages . Also capture response messages and analyze them. During the lab session, also examine and analyze some HTTP headers.

Q 3.2 Use FTP to transfer some files, Use Wireshark to capture some packets. Show that FTP uses two separate connections: a control connection and a data-transfer connection. The data connection is opened and closed for each file transfer activity. Also show that FTP is an insecure file transfer protocol because the transaction is done in plaintext.

Q 3.7 Analyze the behavior of the DNS protocol. In addition to Wireshark [Several network utilities are available for finding some information stored in the DNS servers. Eg.dig utilities (which has replaced nslookup). Set Wireshark to capture the packets sent by this utility.]

PART-2 STUDY OF NETWORK DEVICES IN GNS3

Q 4.1 (a,b,c,d,e) and Q 4.3

Design network configuration shown in Figure 4.1 for all parts. Connect all four VMs to a single Ethernet segment via a single hub as shown in Figure 4.1. Configure the IP addresses for the PCs as shown in Table 4.1.

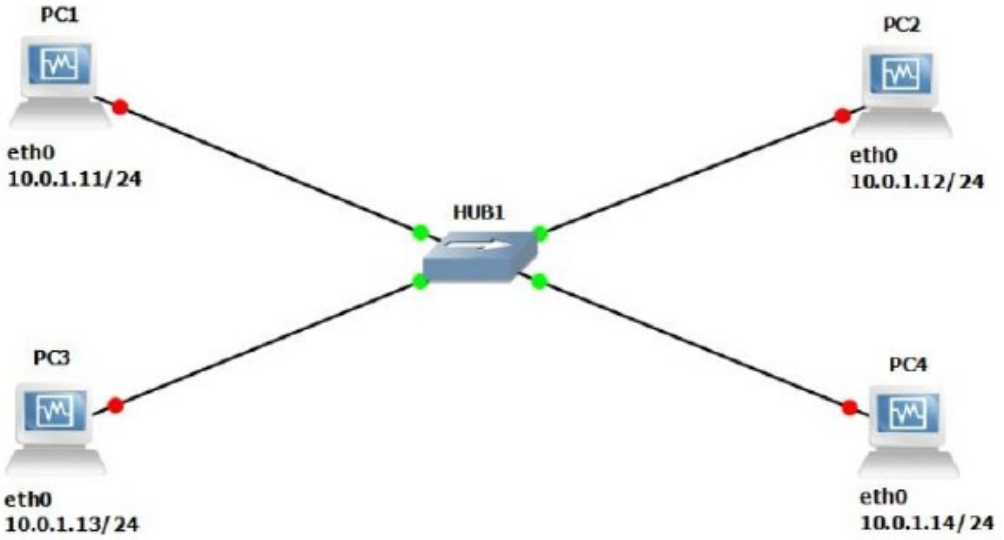


Figure 4.1: Network Design

VMS	IP Addresses of Ethernet Interface eth0
PC1	10.0.1.11 / 24
PC2	10.0.1.12 / 24
PC3	10.0.1.13 / 24
PC4	10.0.1.14 / 24

Table 4.1

a. On PC1, view the ARP cache with ***showarp***

b. Start Wireshark on PC1-Hub1 link with a capture filter set to the IP address ofPC2.

c. Issue a ping command from PC1toPC2:

PC1% ping 10.0.1.13 –c3

Observe the ARP packets in the Wireshark window. Explore the MAC addresses in the Ethernet headers of the captured packets.

Direct our attention to the following fields:

• The destination MAC address of the ARP Request packets.

• The Type Field in the Ethernet headers of ARP packets.

d. View the ARP cache again with the command ***arp -a***. Note that ARP cache

entries can get refreshed/deleted fairly quickly (~2minutes).

show arp

e. Save the results of Wireshark.

EXERCISES:

• What is the destination MAC address of an ARP Request packet?

• What are the different Type Field values in the Ethernet headers that you observed?

• Use the captured data to analyse the process in which ARP acquires the MAC

address for IP address10.0.1.12.

• Use your output data and ping results to explain what happened in each of the ping

commands.

• Which ping operations were successful, and which were unsuccessful? Why?

Q 4.2

To test the effects of changing the netmask of a network configuration. Design the configuration as Q4.1 and replace the hub with a switch, two hosts (PC2 and PC4) have been assigned different network prefixes. Setup the interfaces of the hosts as follows:

VPCS	IP Address of eth0	Network Mask
PC1	10.0.1.100 / 24	255.255.255.0
PC2	10.0.1.101 / 28	255.255.255.240
PC3	10.0.1.120 / 24	255.255.255.0
PC4	10.0.1.121 / 28	255.255.255.240

Run Wireshark on PC1-Switch1 link and capture the packets for the following scenarios

i. From PC1 ping PC3.

ii. From PC1 pingPC2.

iii. From PC1 pingPC4.

iv. From PC4 pingPC1.

v. From PC2 ping PC4.

vi. From PC2 ping PC3.

- Save the Wireshark output to a text file (using the “Packet Summary” option from “Print”) , and save the output of the ping commands. Note that not all of the above scenarios are successful. Save all the output including any error messages.
- When you are done with the exercise, reset the interfaces to their original values as

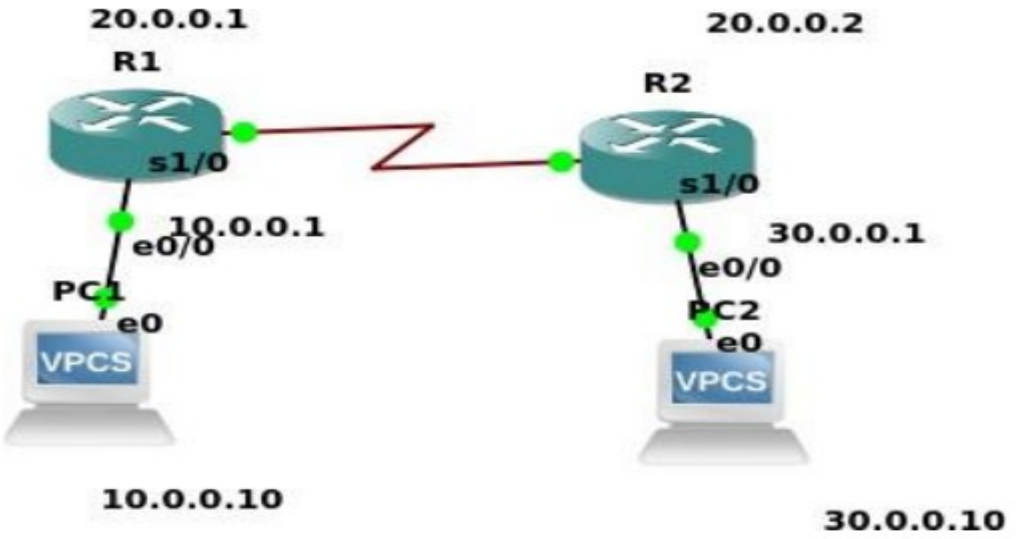
  Given Table4.1. (Note that /24 corresponds to network mask 255.255.255.0. and /28 to network mask 255.255.255.240).

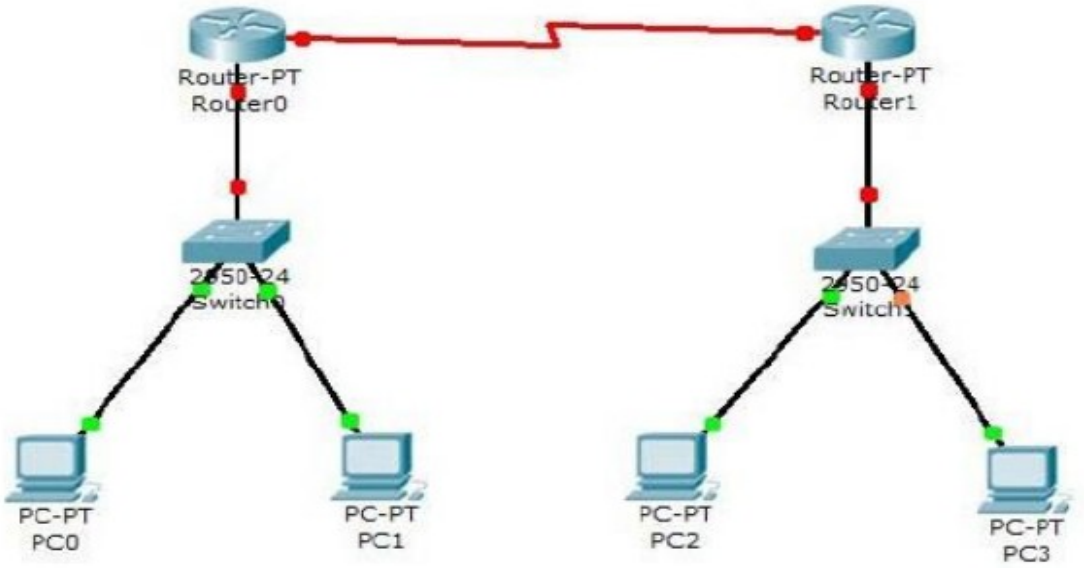
Q 4.6

Configure the below network topologies as shown in Figures below.

i. check the connectivity by pinging from PC1 toPC2.

ii. Analyse ARP exchanges between various network components.





Lab Session 3: Study of ARP,Subnetting and Supernetting using GNS3

(Refer Lab Manual for details. Questions are numbered according to Lab Manual)

PART-1 MID TERM EVALUATION

PART-2 STUDY OF ARP

Q 5.1

In this exercise you study how the network prefixes (netmasks) play a role when hosts determine if a datagram can be directly delivered or if it must be sent to a router. This part uses the network setup shown in Figure 5.1.

The network includes one router, four hosts and two hubs. The IP addresses of all devices are given in Figure 5.2. Here, each host has only a default route. In other words, the routing table at a host only knows about the directly connected networks and the default gateway.

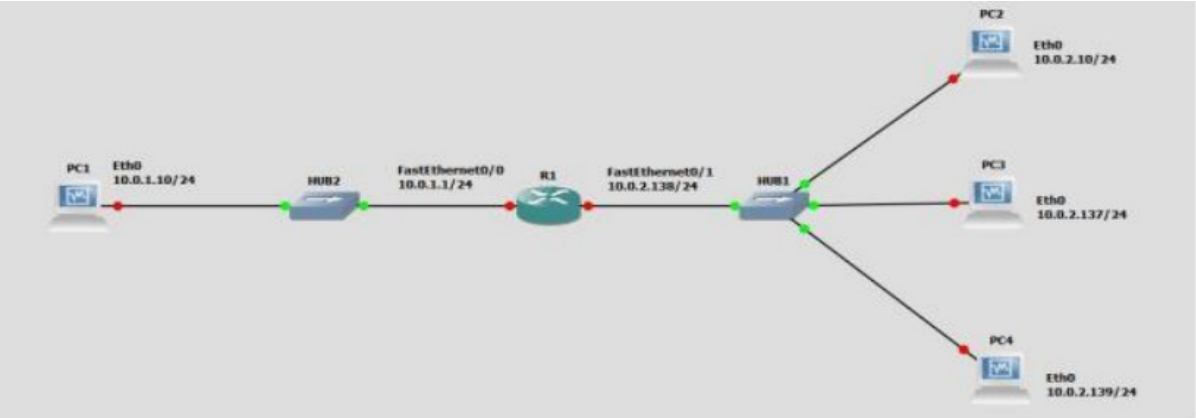


Figure 5.1: Network topology

Linux PC	Ethernet Interface eth0	Ethernet Interface eth1
PC1	10.0.1.10 / 24	Disabled
PC2	10.0.2.10 / 24	Disabled
PC3	10.0.2.137 / 29	Disabled
PC4	10.0.2.139 / 24	Disabled
Cisco Routers	FastEthernet0/0	FastEthernet0/1
Router1	10.0.1.1 / 24	10.0.2.138 / 24

Figure 5.2

Exploring the role of prefixes at hosts

In this exercise, you explore how hosts that are connected to the same local area network, but that have different netmasks, communicate or fail to communicate.

(1) Configure the hosts and the router to conform to the topology shown in Figure 5.2, using

the IP addresses as given in Figure 5.2. Note that PC2, PC3, and PC4 have different netmasks.

(2) Add Router1 as default gateway on all hosts. (PC1, PC2, PC3, andPC4).

(3) Issue ping commands from PC1

i) Clear the ARP table on all PCs.

ii) Start Wireshark on PC1 and on PC3, and set the capture filter to capture ICMP and

ARP packets only.

iii) Issue a ping command from PC1 to PC3 for at least two sends (-c2).

iv) Save the output of the ping command at PC1 and the output of Wireshark on PC1

and PC3.

(4) Save the ARP tables, routing tables, and routing caches of each host. Please note that

these are the tables entries from Step 3 after the ping commands are issued.

(5) Issue ping commands from PC3 toPC4

i) Clear the ARP table on all PCs.

ii) Start Wireshark on PC3, and set the capture filter to capture ICMP and ARP packets

only.

iii) Check the ARP table, routing table, and routing cache of each host.

Save the output.

Please note that these are the table entries from Step 4 before the ping is issued.

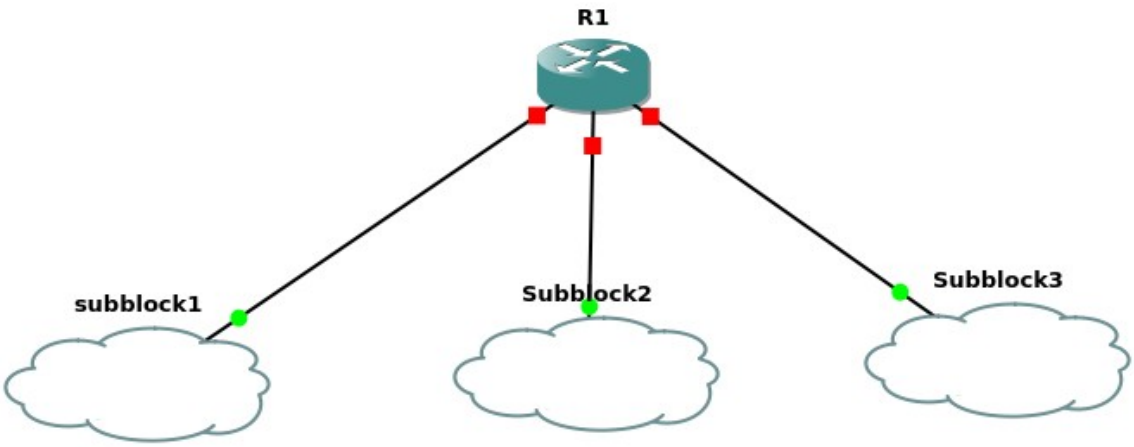
- Issue a ping command from PC3 to PC4 for at least three sends (-c 3). Save the output of the ping command and the output of Wireshark on PC3. Save the ARP table, routing table, and routing

cache of PC3. Please note that these are the table entries from Step 4 after the ping commands are issued.

- Repeat Step 4, but this time issues a ping from PC3 to PC2. Note that once an entry is made in the routing cache, you cannot repeat the previous experiment to obtain the same results. You have to wait until the routing cache is reset or you can delete all the routing caches on all devices.

PART 3 SUBNETTING

Q2. An organization is granted a block of addresses with the beginning address 14.24.74.0/24. The organization needs to have 3 subblocks of addresses to use in its three subnets: one subblock of 10 addresses, one subblock of 60 addresses, and one subblock of 120 addresses. Design the subblocks. Use the topology shown below.



PART-3 STUDY OF DHCP

Q 7.1

Configure two VMs that will be used to test connectivity from end to end and R1 will serve as a DHCP server to distribute IP addresses. The diagram below details the current setup:

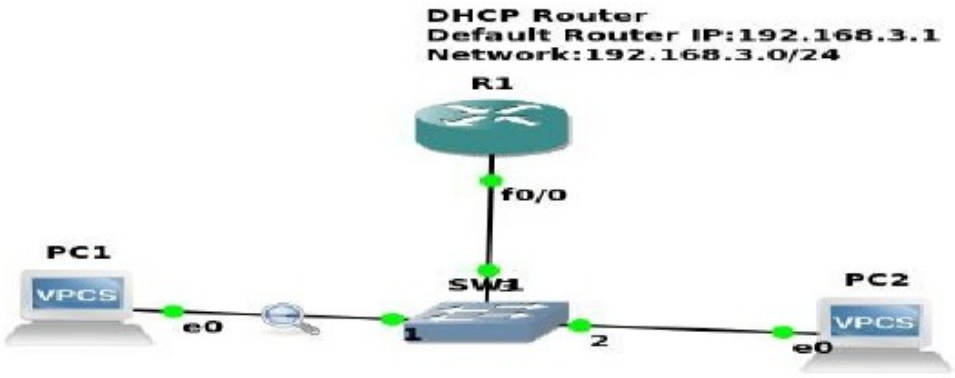


Figure 7.1 :Network Topology for DHCP Configuration

1. In order to configure our router as a DHCP server the following commands were used.

R1(config)#IP dhcp pool NAME

R1(dhcp-config)#Network 192.168.3.0 255.255.255.0

R1(dhcp-config)#Default-router 192.168.3.1

The commands above create a DHCP pool, adds the network that we want to assign IP addresses from, and specifies the default gateway for this subnet.

Note: There are many other parameters that go into configuring a DHCP server, but this will suffice for our test environment. That should be it for the DHCP configuration.

2.The next thing that you want to do is configure the fastethernet 0/0 interface which will connect to our switch.

R1(config)#Interface fastEthernet 0/0

R1(config-if)#No shutdown

R1(config-if)#ip address 192.168.3.1 255.255.255.0

The commands above will turn the interface on and assign an IP address.

3. Turn on the VPCS. In PC1 and PC2 type ***dhcp***

That is,

PC1>dhcp

PC2>dhcp

4. Let’s analyse some of the traffic patterns using Wireshark.

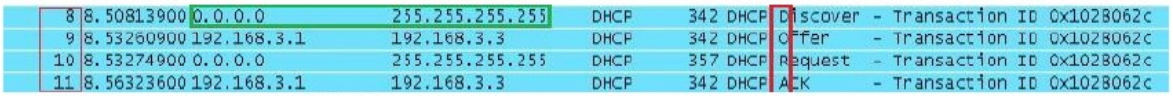


Figure 7.2: DHCP

We see a discover message followed by an offer, request, and an acknowledgement. This is

the process that clients go through in order to obtain an IP address via DHCP. The mnemonic for the steps above is DORA and it should help in memorizing the order of the steps.

Q 7.2 Configure DHCP server at R1 for the PART 2 Q2 Subnet configuration and topology.

Lab Session 4: DNS and VLAN

(Refer Lab Manual for details. Questions are numbered according to Lab Manual)

PART 1: STUDY OF DNS SERVER

Q 7.4

Configure the below topology to setup DNS server. R1 will use R2 as DNS server to make DNS resolutions.

First, let's begin with R1. We will setup hostname and IP related information.

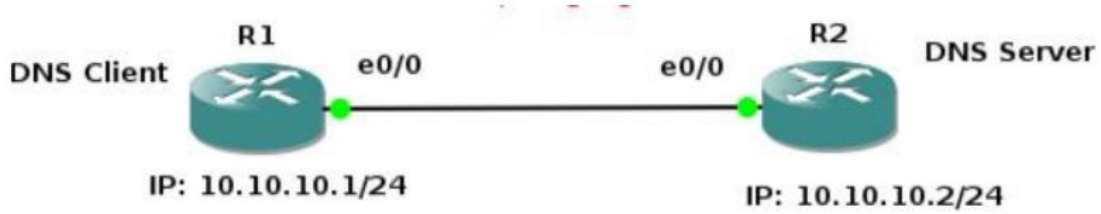


Figure 7.3 : Network Topology for DNS Configuration

R1 IP configurations:

#Enable

#configure terminal

#hostname R1

#interface e0/0

#ip address 10.10.10.1 255.255.255.0

#no shut

#do wr

#end

R2 IP and Hostname Configurations:

#enable

#config t

#hostname R2

#int e0/0

#ip address 10.10.10.2 255.255.255.0

#no shut

#do wr

#end

Setting up R2 as DNS Server

#config t

#ip dns server

#ip host loopback.R2.com 2.2.2.2

We mapped loopback.R2.com to ip address 2.2.2.2. Currently, we don't have 2.2.2.2, we could create loopback interface on R2 and assign ip 2.2.2.2.

#interface loopback 1

#ip address 2.2.2.2 255.255.255.255

#end

Let us verify that loop-back interface we just created is working. This will show us that the host name correctly setup locally on R2.

#ping loopback.R2.com

Now it's time to setup R1 to resolve hostnames using R2.On R1 type:

#config terminal

#ip domain lookup

#ip name-server 10.10.10.2

Set R1 to use R2 as default gateway to get to loopback interface on R2. So that after R1

resolve loopback.R2.com, it can reach 2.2.2.2 through its default route (R2).

on R1 type:

#config t

#ip route 0.0.0.0 0.0.0.0 10.10.10.2

#end

This tells our router that to get to any network not in its routing table, it is next hop is 10.10.10.2 which is our router R2.

Now on R1, do a ping to loopback.R2.com and you should get a success message.

#ping loopback.R2.com repeat 3

If you captured the traffic, you will see DNS query and Answer as shown in Wireshark capture screen shot below.

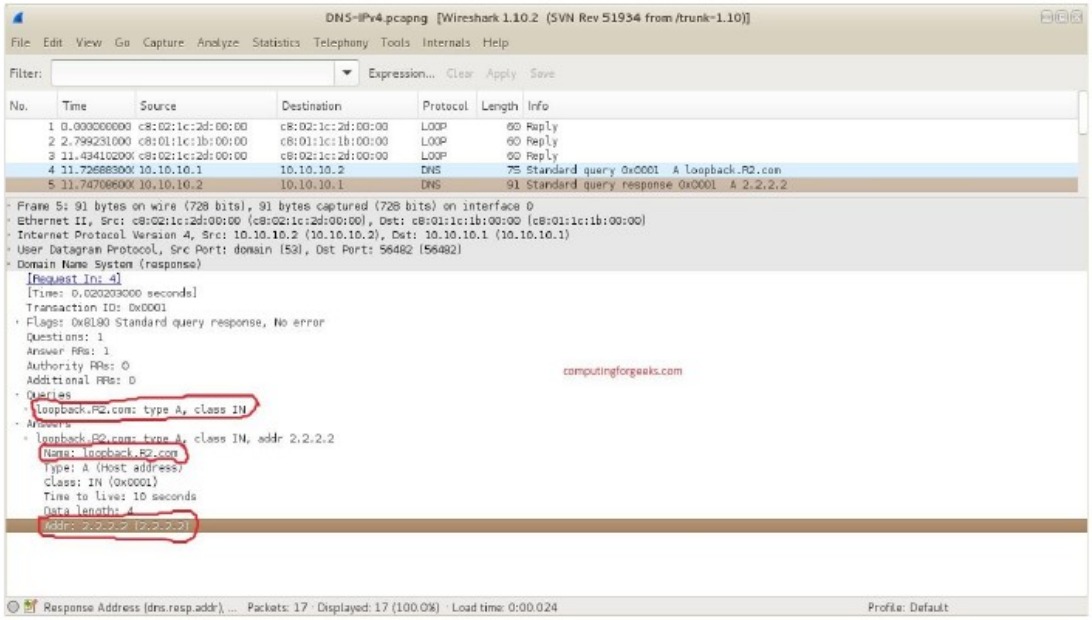
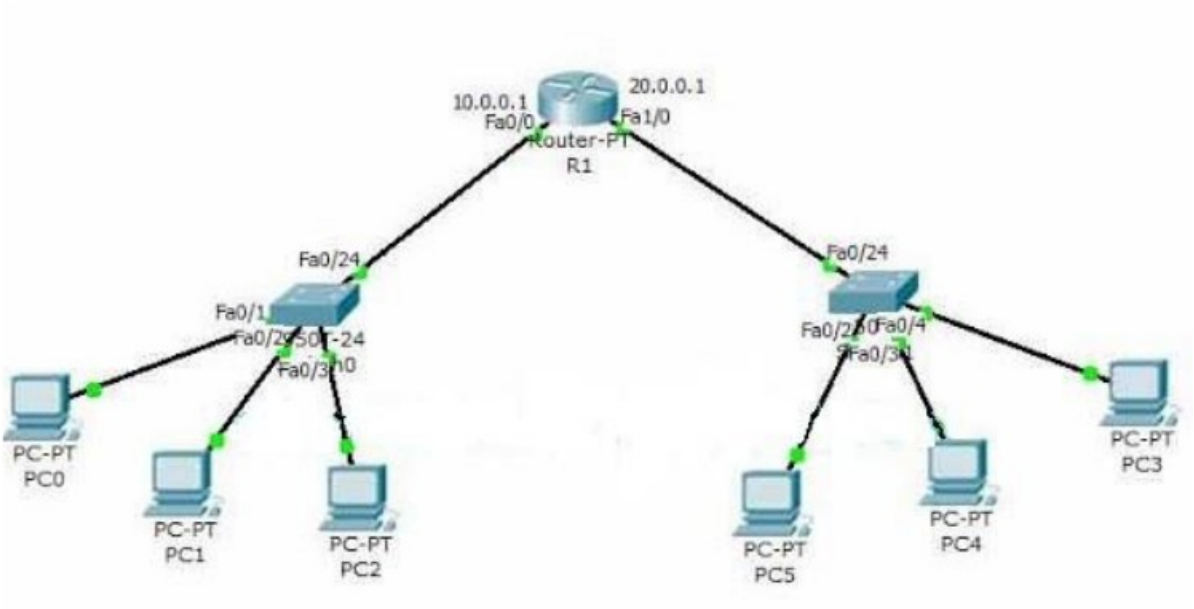


Figure 7.4 : Observation in WIRESHARK

Q 7.5

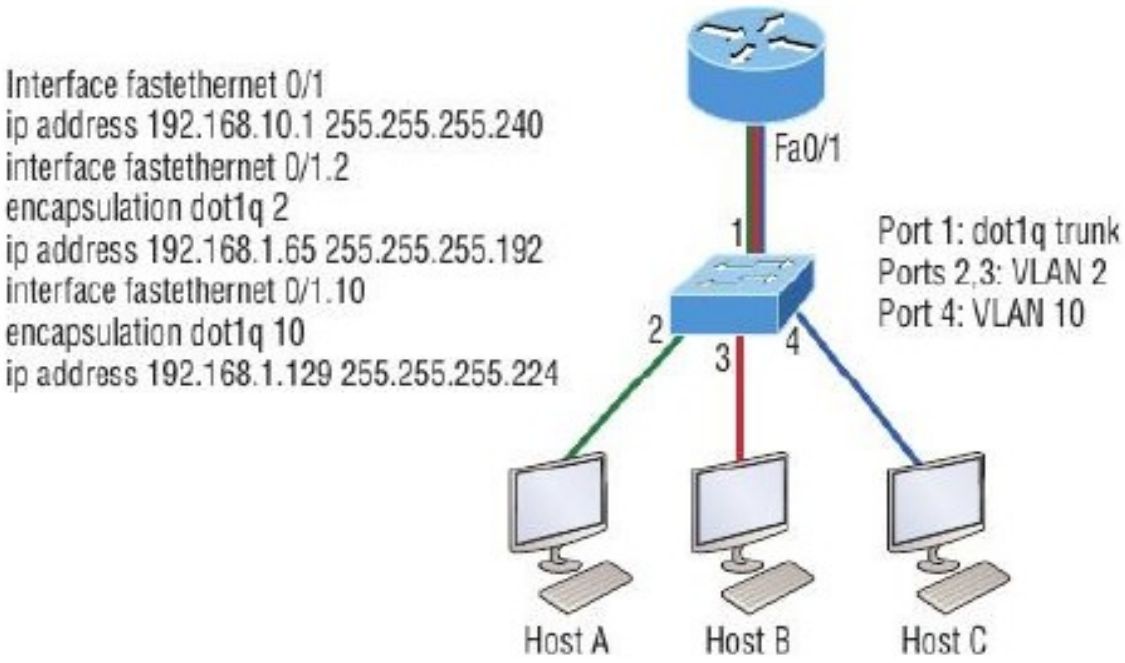
Configure the topology shown below DNS Server and DNS Client. Test the setup. Analyse the Interaction.



PART 2: STUDY OF VLAN

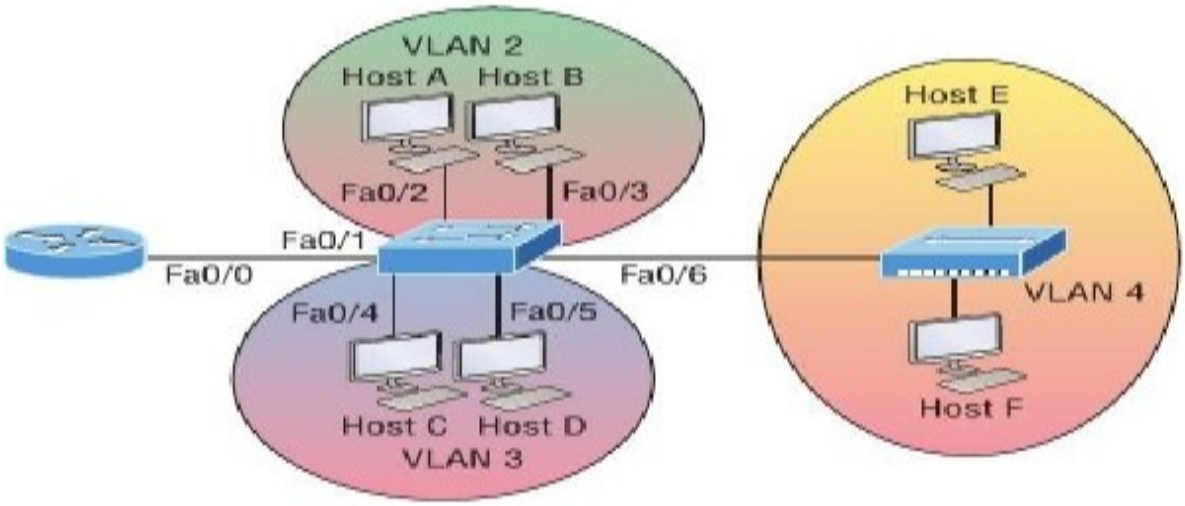
Q 8.1

Configure following inter-VLAN example in GNS3 and verify the working using Wireshark tool.



Q 8.2

Configure following inter-VLAN example in GNS3 and verify the working using Wireshark tool.


VLAN 2
Host A
Host B
Fa0/2
Fa0/3
Fa0/1
Fa0/0
Fa0/4
Fa0/5
Host C
Host D
VLAN 3
Fa0/6
Host E
VLAN 4
Host F