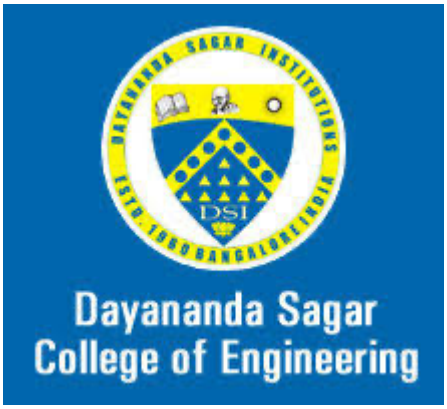
DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute Affiliated to VTU, Belagavi)

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



RECORD BOOK

NAME OF STUDENT: ……Gautam Kumar…….

BRANCH: ……Computer Science & Engineering……

USN: ……1DS18CS710……...

SECTION: ……E……..

SUBJECT: ……Computer Networks Laboratory with Mini-project……..

SUBJECT CODE: ……18CS5DLCNL……..

EXPERIMENT - 1

1) Analyse VLAN communication using CPT
a) Sketch and simulate 3 VLANS
b) Setup an extended VLAN using Trunk Interface
c) Inter VLAN Routing

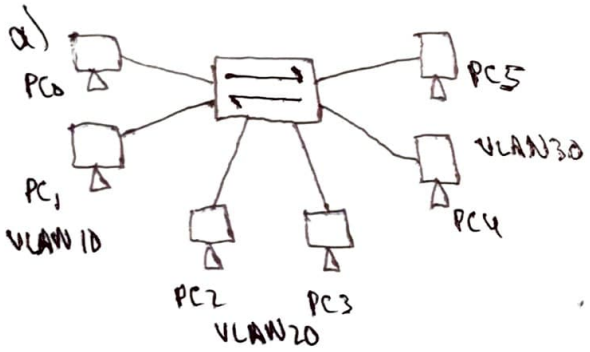
Design

A VLAN is a subnetwork which can group together the collections of devices on separate physical VLAN.

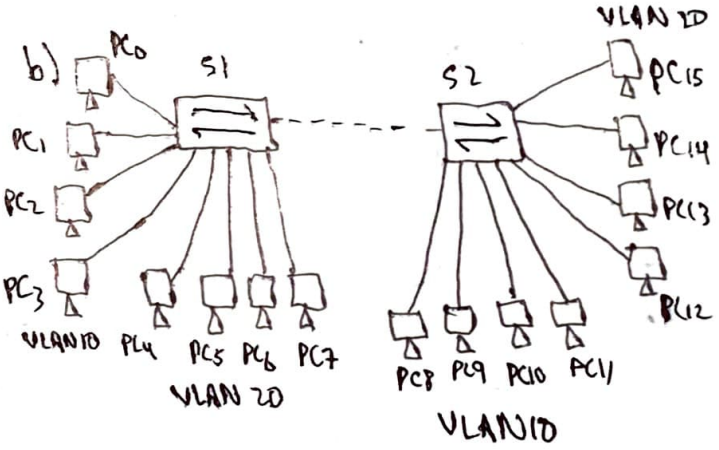
Advantages of VLAN:
- Improved security
- Easier fault management, etc.

Topology:

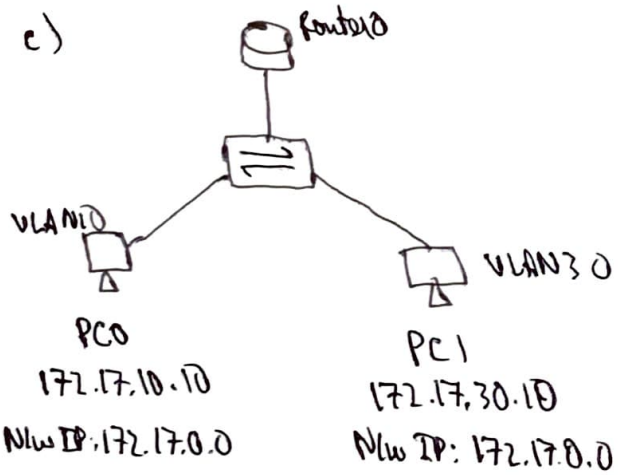
a)



b)



c)



Configuration:

a) switch > enable
switch # config t
switch (config) # vlan 10.
switch (config-vlan) # name green
switch (config-vlan) # exit
switch (config) # vlan 20
switch (config-vlan) # name red
switch (config-vlan) # exit
switch (config) # vlan 30
switch (config-vlan) # name red
switch (config-vlan) # exit
switch (config) # interface range fa0/1-2
switch (config-if-range) # switchport mode access
switch (config-if-range) # switchport access vlan 10
switch (config-if-range) # exit
switch (config) # interface range fa0/3-4
switch (config-if-range) # switchport mode access
switch (config-if-range) # switch access vlan 20
switch (config-if-range) # exit
switch (config) # interface range fa0/5-6
switch (config-if-range) # switchport mode access
switch (config-if-range) # switch access vlan 30
switch (config-if-range) # exit

b) trunk interface at s1:
switch > enable
switch # config t
switch (config) # interface fa0/15
switch (config-if) # switchport mode trunk
switch (config-if) # exit
trunk interface at s2:
switch > enable
switch # config
switch (config) # interface fa0/15
switch (config-if) # switchport mode trunk
switch (config-if) # exit

c) S1(config)# vlan 10 (Switch Configuration)
S1(config-vlan)# vlan 30
S1(config-vlan)# interface fa0/5
S1(config-if)# switchport mode trunk
S1(config-if)# end

R1(config)# interface g0/0.10
R1(config-subif)# encapsulation dot1q 10
R1(config-subif)# ip address 172.17.10.1 255.255.255.0
R1(config-subif)# interface g0/0.30
R1(config-subif)# encapsulation dot1q 30
R1(config-subif)# ip address 172.17.30.1 255.255.255.0
R1(config)# interface g0/0
R1(config-if)# no shutdown

Output Observed:

VLAN communications were analysed by sketching and simulating 3 different VLANs. An extended VLAN was setup to the existing VLAN and communicated using trunk interface. Inter VLAN routing was achieved by setting up two different VLANs and communicated by a single router.

EXPERIMENT-2

2) Setup a Router based wide area network using Dynamic routing (RIP, EIGRP, OSPF).

a) Set up a network of 3 and 4 routers. Configure routing in each router and test the network.

Design:

Dynamic Routing: Process where a router can forward data via different route on given destination based on the current conditions of communications circuits within the system.

i) RIP (Routing Information Protol): One of the oldest distance-vector protocols which employ hop count as a routing metric. The largest number of hops allow are is for RIP.

RIP Routing table in routers:

Router0: 10.0.0.0
192.168.1.0

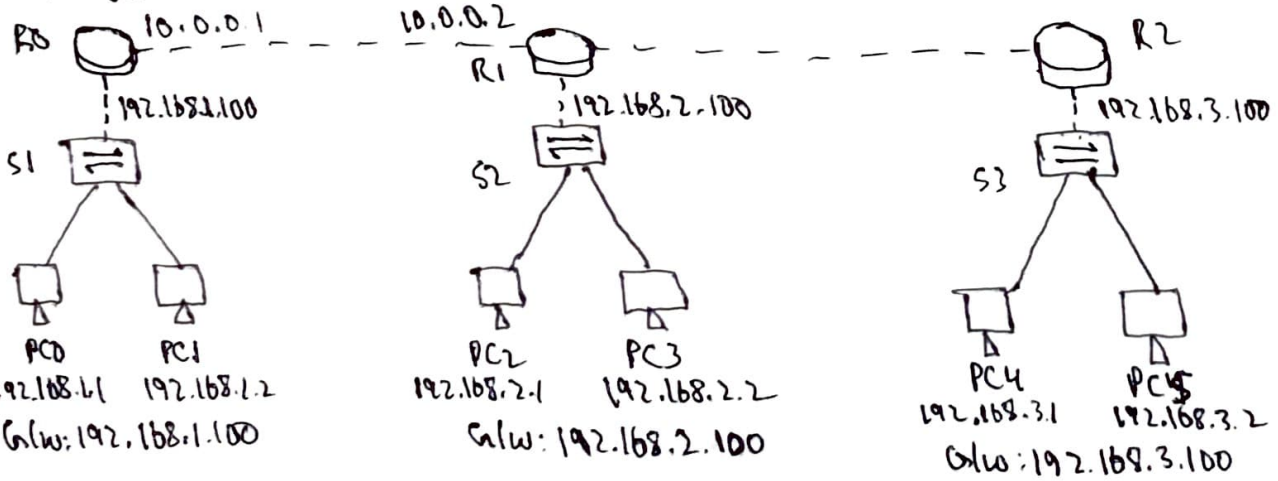
Router1: 10.0.0.0
11.0.0.0
192.168.2.0

Router2: 11.0.0.0
192.168.3.0

ii) EIGRP (Enhanced Interior Gateway Routing Protol): An advanced distance vector protocol that is used on a computer network for automating routing decisions and configuration

iii) OSPF (Open Shortest Path First): A routing protocol for Internet Protocol networks. It uses link state routing (LSR) algorithm and falls into group of interior gateway protocols (IGPs). Operating within a single autonomous systems (AS).

Topology:



Configurations:

i) RIP

Router 0

```
Router>enable
Router#config t
Router(config)#router rip
Router(config-router)#version 2
Router(config-router)#network 192.168.1.0
Router(config-router)#network 10.0.0.0
Router(config-router)#end
```

Router 1

```
Router>enable
Router#config t
Router(config)#router rip
Router(config-router)#version 2
Router(config-router)#network 192.168.2.0
Router(config-router)#network 10.0.0.0
Router(config-router)#network 11.0.0.0
Router(config-router)#end
```

Router 2

```
Router>enable
Router#config t
Router(config)#router rip
Router(config-router)#version 2
Router(config-router)#network 192.168.3.0
Router(config-router)#network 11.0.0.0
Router(config-router)#end
```

ii) EIGRP

Router 0

```
Router>enable
Router#config
Router(config)#router eigrp 1
Router(config-router)#network 192.168.1.0
Router(config-router)#network 10.0.0.0
Router(config-router)#end
```

Router 1
Router> enable
Router# config t
Router(config)# router eigrp 1
Router (config-router)#network 192.168.2.0
Router (config-router)#network 10.0.0.0
Router (config-router)#network 11.0.0.0
Router (config-router)# end

Router 2
Router> enable
Router# config t
Router (config)# router eigrp 1
Router (config-router)# network 192.168.3.0
Router (config-router)# network 11.0.0.0
Router (config-router)# end

iii) OSPF

Router 0
Router> enable
Router# config t
Router (config)# router ospf 1
~~Router (config)#~~ Router (config-router)#network 192.168.1.0 0.0.0.255 area 1
Router (config-router)# network 10.0.0.0 0.255.255.255 area 1
Router (config-router)# end

Router 1
Router> enable
Router# config t
Router (config)# router ospf 2
Router (config-router)# network 192.168.2.0 0.0.0.255 area 0
Router (config-router)# network 10.0.0.0 0.255.255.255 area 1
Router (config-router)# network 11.0.0.0 0.255.255.255 area 2
Router (config-router)# end

Router 2

```
Router > enable
Router # config t
Router (config) # router ospf 1
Router (config-router) # network 192.168.3.0 0.0.0.255 area 2
Router (config-router) # network 11.0.0.0 0.255.255.255 area 2
Router (config-router) #end
```

Output Observed:

Packets are being routed even through the systems are in same network but different subnets.

EXPERIMENT - 3

3) Practice IP addressing principles

a) Set up a subnet (N1) comprising 4 Nodes. Change the subnet masks of some of the nodes and test the network. Set up another subnet (N2) of 4 nodes. Connect these two subnets using a router

b) Create 4 equal sized subnets in the subnet N1 and test the network

Design:

Subnetting: Process of dividing a network into multiple smaller networks.

Advantages:

- Converting host bits into network bits, i.e., converting 0's into 1's.
- Minimizing the wastage of IP address.

a) Subnet N1

	Network	Subnet	Host
Network IP	192.168.1.0	0 0000000/25	= 192.168.1.0
First IP	192.168.1.0	0 0000001/25	= 192.168.1.1
Last IP	192.168.1.0	0 1111110/25	= 192.168.1.126
Broadcast IP	192.168.1.0	0 1000000/25	= 192.168.1.127

Subnet N2

Network IP	192.168.1.1	10000000/25	= 192.168.1.128
First IP	192.168.1.1	10000001/25	= 192.168.1.129
Last IP	192.168.1.1	10111111/25	= 192.168.1.254
Broadcast IP	192.168.1.1	11111111/25	= 192.168.1.255

Subnet mask - 255.255.255.128

b) 4 Subnets

Subnet N1	192.168.1.0	0000000/26	= 192.168.1.0
First IP	192.168.1.0	0000001/26	= 192.168.1.1
Last IP	192.168.1.0	0011111/26	= 192.168.1.62
Broadcast IP	192.168.1.0	0111111/26	= 192.168.1.63

Subnet N2

Network IP	192.168.1.0	1000000/26	= 192.168.1.64
First IP	192.168.1.0	1000001/26	= 192.168.1.65
Last IP	192.168.1.0	1111110/26	= 192.168.1.126
Broadcast IP	192.168.1.0	1111111/26	= 192.168.1.127

Subnet N3

Network IP	192.168.1.1	0000000/26	= 192.168.1.128
First IP	192.168.1.1	0000001/26	= 192.168.1.129
Last IP	192.168.1.1	0111110/26	= 192.168.1.190
Broadcast IP	192.168.1.1	0111111/26	= 192.168.1.191

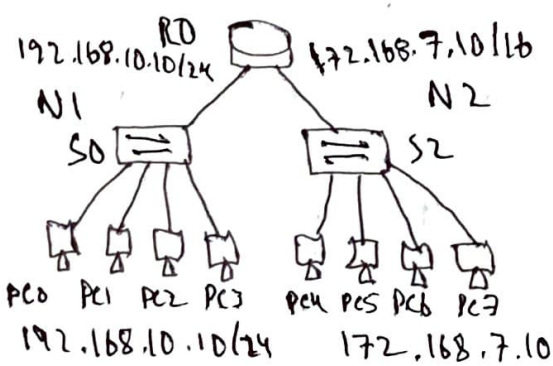
Subnet N4

Network IP	192.168.1.1	1000000/26	= 192.168.1.192
First IP	192.168.1.1	1000001/26	= 192.168.1.193
Last IP	192.168.1.1	1111110/26	= 192.168.1.254
Broadcast IP	192.168.1.1	1111111/26	= 192.168.1.255

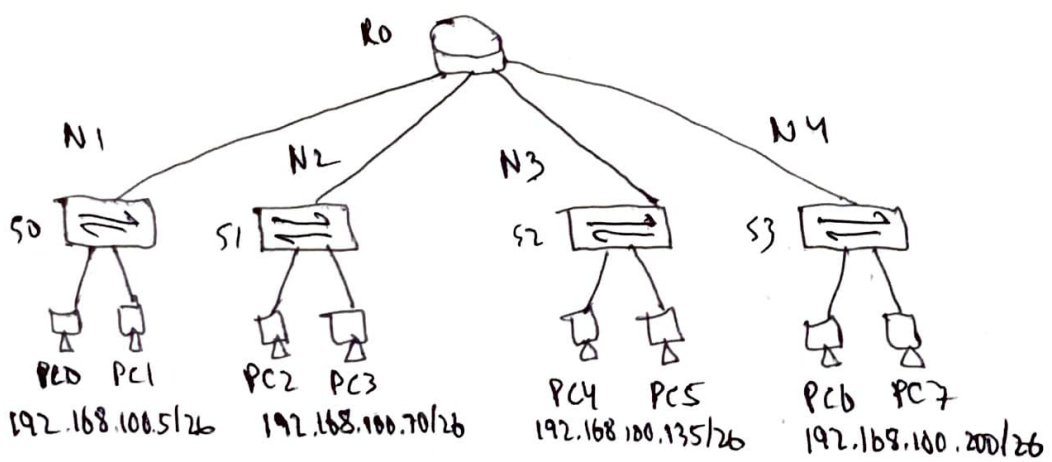
Subnet mask - 255.255.255.192

Topology

a)



b)



Output Observed:

The packets sent from one subnet are being routed by the router to its respective destination even through all the system are in the same network.

EXPERIMENT-4

4) Implement DHCP and DNS

a) A client, single DNS server and a Web Server

b) A client, two DNS server and a Web Server

c) A client, a hierarchy of DNS and a Web Server

Design:

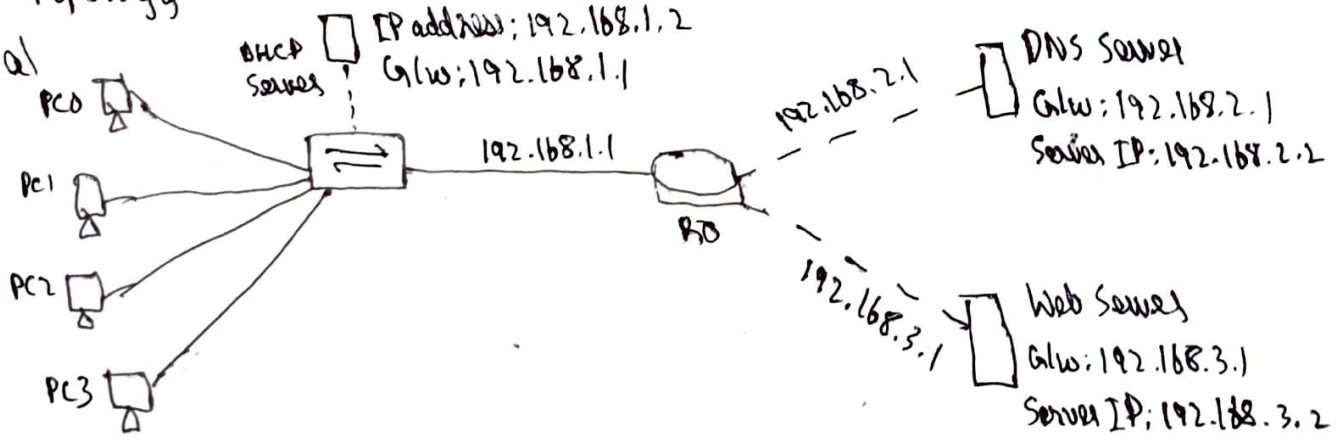
DNS: A type of name server that manages, maintains and other processes internet domain names & their associated records. In other words, DNS server is primary component that implements DNS protocol & provisions domain name ~~resolution~~ resolution services to we hosts & clients on an IP based networks.

Root DNS: A DNS server for the ~~configuration~~ DNS of the internet. They publish root zone file contents which are responsible for DNS functionality.

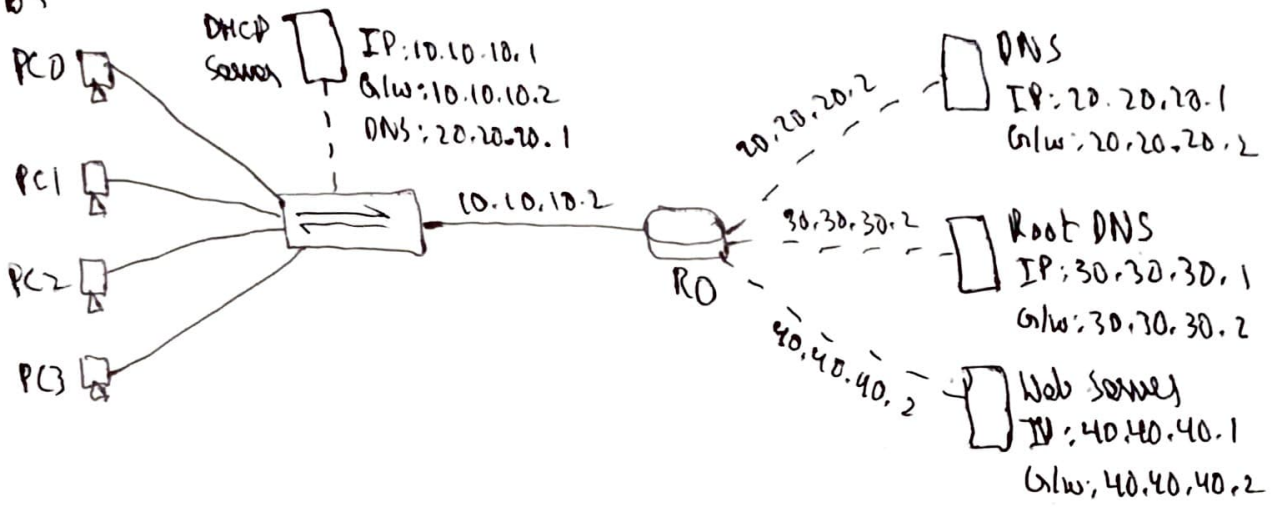
DNS Hierarchy: DNS uses hierarchy to manage its distributed database system. Also called Domain Name Space, it is an inverted tree structure The Hierarchy is as follows:

Authority Server
↓
Top Level Domain
↓
Root DNS
↓
Local DNS

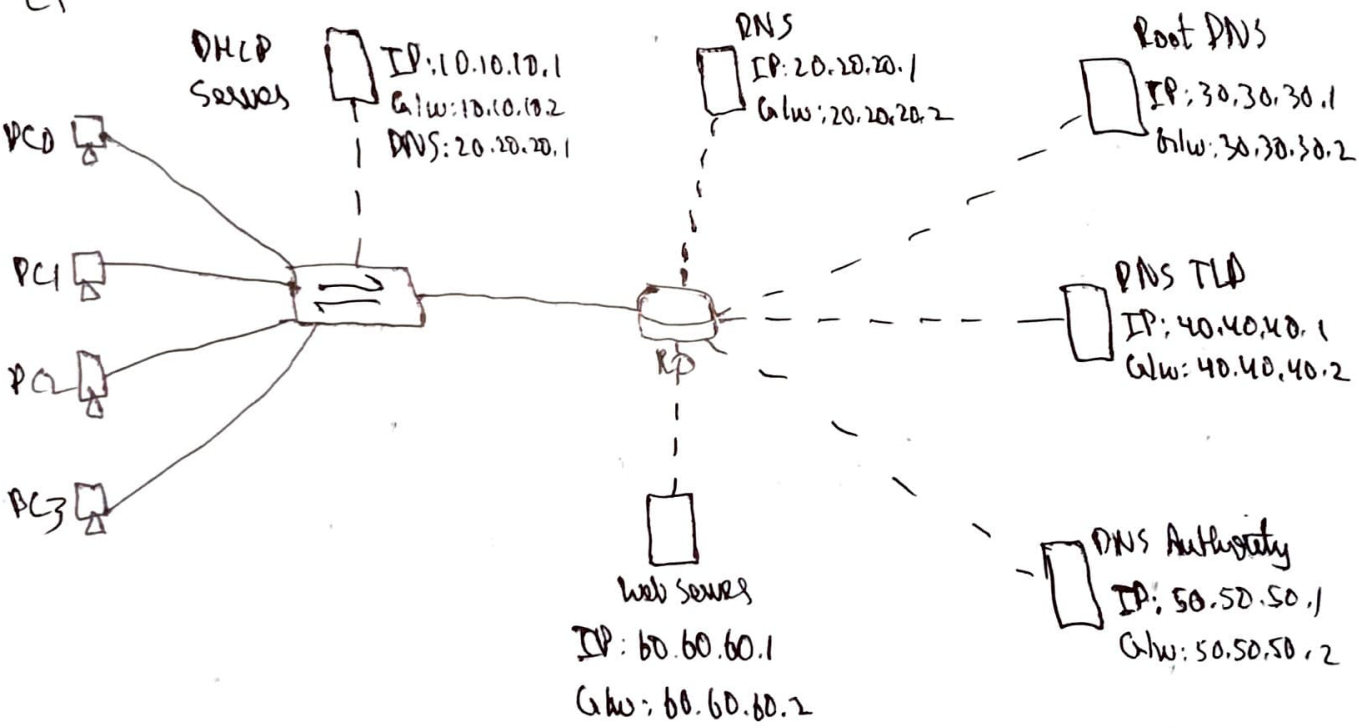
Topology:



b)



c)



Configurations:

a)

No	Name	Type	Detail
0	www.cisco.com	A Record	192.168.3.2

b) In Local DNS → Services → DNS

No	Name	Type	Detail
0	. root	A Record	30.30.30.1
1	www.cisco.com	NS	root

In Root DNS → Services → DNS

No	Name	Type	Detail
0	www.cisco.com	A Record	40.40.40.1

c) In local DNS

No	Name	Type	Detail
0	root	ARecord	30.30.30.1
1	www.google.com	NS	root

In Root DNS

No	Name	Type	Detail
0	tld	ARecord	40.40.40.1
1	www.google.com	NS	tld

In TLD DNS

No	Name	Type	Detail
0	authority	ARecord	50.50.50.1
1	www.google.com	NS	authority

In Authority DNS

No	Name	Type	Detail
0	www.google.com	ARecord	60.60.60.1

Output Observed

a) When a website is searched in a web browser it is routed to the DNS server. The DNS server returns the IP address for a given string by searching its routing table. Then the request is routed to the webserver. The webserver responds with the HTML page.

b) When PC0 sends request to local DNS when the given website is not available in local DNS the request is routed to root DNS server. The root DNS server has the IP address of the given website so the request is routed to the IP address returned by the root DNS which then responds with the webpage.

c) When PC0 sends a request, it goes to local DNS if the IP is not present in local DNS the request is routed to root DNS, then to TLD DNS, then to authority DNS where the IP is present. Using that IP address the request is routed to webserver which responds with the HTML page.

5)

EXPERIMENT-5

Implement Static NAT, Dynamic NAT and PAT.

Design:

NAT: Network Address Translation is a method of remapping one IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.

Static NAT: Designed to allow one-to-one mapping between local and global addresses.

Dynamic NAT: Designed to map an unregistered IP address to a registered IP address from out of a pool of registered IP addresses.

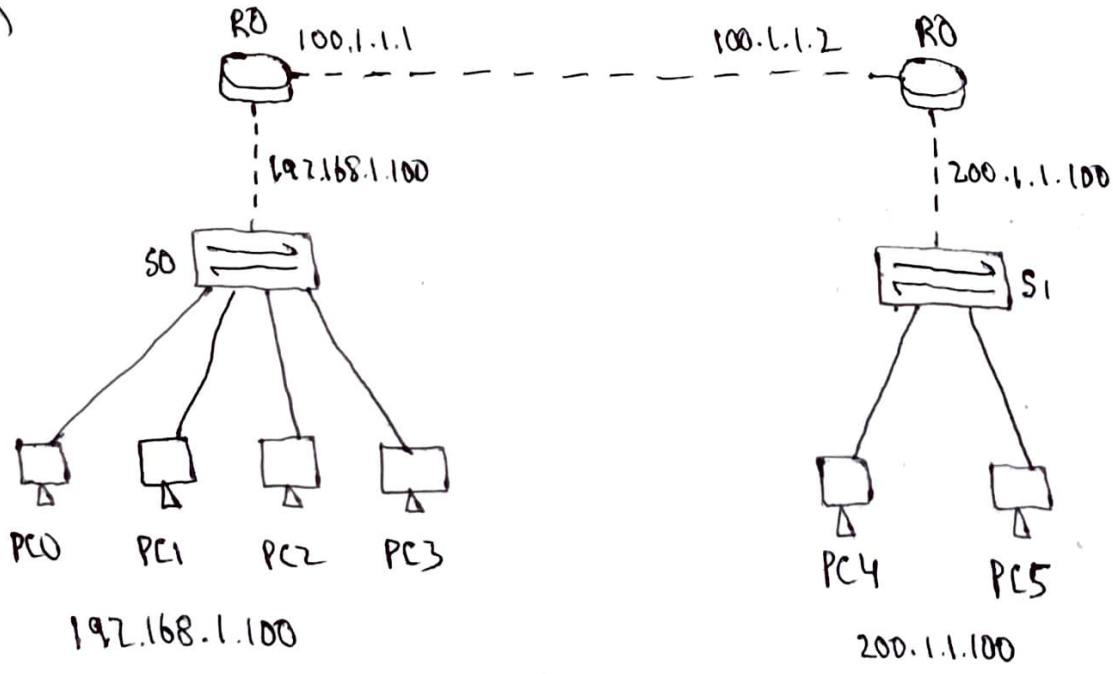
PAT (Port Address Translation): Designed to permit multiple devices on a LAN to be mapped to a single IP address. It is used to conserve IP addresses.

NAT Terminology:

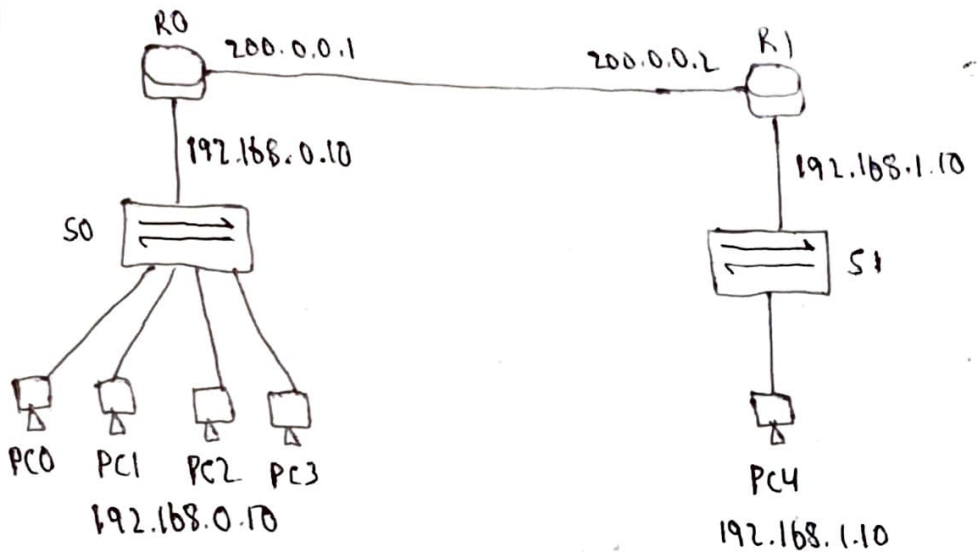
Inside Local Addresses - Name of inside source address before translation
Inside Global Addresses - Name of inside host after translation.
Outside Local Addresses - Name of destination host before translation.
Outside Global Addresses - Name of outside destination host after translation

Topology:

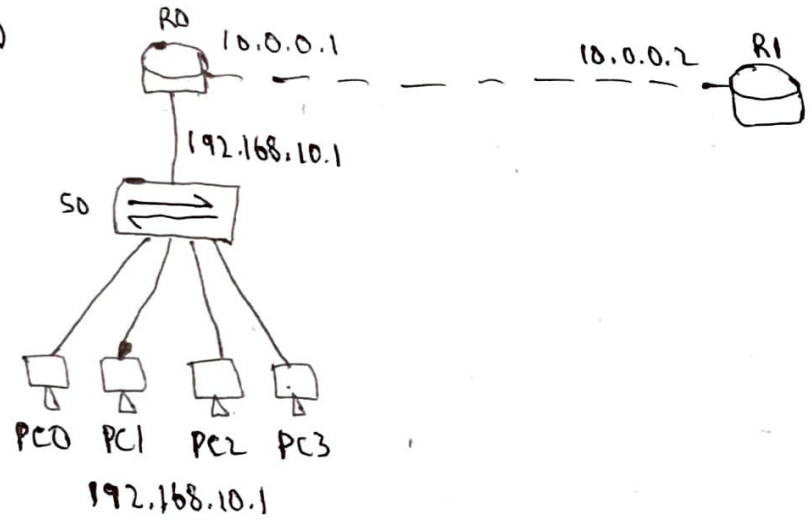
a)



b)



c)



Configuration:

a) NAT

1) Construct Topology

2) Assign IP address

3) Static routing on both routers

R0(config)# ip route 0.0.0.0 0.0.0.0 100.0.0.2

R1(config)# ip route 0.0.0.0 0.0.0.0 100.0.0.1

4) Static NAT Configuration

R0(config)# ip nat inside source static 192.168.1.1 50.1.1.1

R0(config)# ip nat inside source static 192.168.1.2 50.1.1.2

5) Verify NAT Translations

R0# show ip nat translations

6) NAT inside & outside configuration at Router0

R0(config)# interface gigabitethernet 0/0

~~R0(config-if)# interface~~ R0(config-if)# ip nat inside

R0(config-if)# exit

```
R0(config)# interface gigabitEthernet 0/1
R0(config-if)# ip nat outside
R0(config-if)# exit
```

b) Dynamic NAT

```
R0(config)# access-list 1 permit 192.168.0.0 0.0.0.255
R0(config)# ip nat pool CCNA 50.0.0.1 50.00.200 netmask 255.255.255.0
R0(config)# ip nat inside source list 1 pool CCNA
R0(config)# interface gigabitEthernet 0/0
R0(config-if)# ip nat inside
R0(config-if)# exit
R0(config)# interface Serial 0/0/0
R0(config-if)# ip nat outside
R0(config-if)# exit
```

c) PAT

```
R0(config)# access-list 1 permit 192.168.10.0 0.0.0.255
R0(config)# ip nat inside source list 1 interface gigabitEthernet 0/1 overload
R0(config)# interface gigabitEthernet 0/0
R0(config-if)# ip nat inside
R0(config-if)# exit
R0(config)# interface gigabitEthernet 0/1
R0(config-if)# ip nat outside
R0(config-if)# exit
```

Output Observed:

In each of the cases in static NAT, dynamic NAT and PAT, the private IP addresses are mapped and translated to their respective public IP addresses. The public IP addressed are shown in place of the private IP addresses when the device is communicated.

EXPERIMENT-6

6) Write a C/C++ program to implement the data link layer framing methods. A) Bit stuffing B) Character Stuffing

a. Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
  int a[20], b[30], i, j, k, count, n;
  printf("Enter frame size: ");
  scanf("%d", &n);
  printf("Enter the frame in the form of 0 and 1:");
  for(i=0; i<n; i++)
    scanf("%d", &a[i]);
  i=0;
  count=1;
  j=0;
  while(i<n)
  {
    if(a[i]==1)
    {
      b[j]=a[i];
      for(k=i+1; a[k]==1 && k<n && count<5; k++)
      {
        j++;
        b[j]=a[k];
        count++;
        if(count==5)
        {
          j++;
          b[j]=0;
        }
      } i=k;
```

```
        else
        {
            b[j]=a[i];
        }
        i++;
        j++;
    }
    printf("After Bit Stuffing: ");
    for(i=0;i<j;i++)
        printf("%d",b[i]);
    printf("\n");
    return 0;
}
```

Output:

Enter frame size: 10

Enter the frame in the form of 0 and 1: 1 0 1 0 1 1 1 1 1 1

After Bit Stuffing: 1010111111

6) Program

```
#include <stdio.h>
#include <string.h>

int main()
{
  char a[30], fs[50]=" ", t[3], sd, ed, x[3], s[4], d[3], y[3];
  int i, j, n=0, q=0;
  printf("Enter characters to be stuffed: ");
  scanf("%s", a);
  printf("Enter a character that represents starting delimiter:");
  scanf("%c", &sd);
  printf("Enter a character that represents ending delimiter: ");
  scanf("%c", &ed");
  x[0]=s[0]=s[1]=sd;
  x[1]=s[2]='\0';
  y[0]=d[0]=d[1]=ed;
  d[2]=y[1]='\0';
  strcat(fs,x);
  for(i=0; i<strlen(a); i++)
  {
    t[0]=a[i];
    t[1]='\0';
    if(t[0]==sd)
      strcat(fs,s);
    else if(t[0]==ed)
      strcat(fs,d);
    else
      strcat(fs,t);
  }
```

```
        strcat(fs,y);
        printf("After Stuffing: %s", fs);
        printf("\n");
        return 0;
}
```

Output:

Enter characters to be stuffed: goodday
Enter a character that represents starting delimeter: t
Enter a character that represents ending delimeter: c
After Stuffing: tgoodayc

EXPERIMENT-7

7) Write C/C++ program to implement Distance Vector Routing Algorithm

Program:

```
#include <stdio.h>
int dist[50][50],temp[50][50],n,i,j,k,x;
void dvr()
{
  for(i=0; i<n; i++)
    for(j=0; j<n; j++)
      for(k=0; k<n; k++)
        if(dist[i][k] + dist[k][j] < dist[i][j])
        {
          dist[i][j] = dist[i][k]+dist[k][j];
          temp[i][j]= k;
        }
        for(i=0; i<n; i++)
        {
          printf("\n\n State value for Router %d is:", i+1);
          for(j=0; j<n; j++)
            printf("\t\n Node %d via %d Distance %d", j+1, temp[i][j]+1, dist[i][j]);
        }
        printf("\n\n");
}
```

```
int main()
{
  printf("Enter the number of nodes: ");
  scanf("%d",&n);
  printf("Enter the distance matrix :\n");
  for(i=0;i<n;i++)
  {
    for(j=0;j<n;j++)
    {
      scanf("%d",&dist[i][j]);
      dist[i][i]=0;
      temp[i][j]=j;
    }
  }
  div();
  return 0;
}
```

Output:

Enter the number of nodes: 3

Enter the distance matrix:

0 2 7

2 0 1

7 1 0

State value for Router 1 is:

Node 1 via 1 Distance 0

Node 2 via 2 Distance 2

Node 3 via 2 Distance 3

State value for Router 2 is:

Node 1 via 1 Distance 2

Node 2 via 2 Distance 0

Node 3 via 3 Distance 1

State value for Router 3 is:

Node 1 via 2 Distance 3

Node 2 via 2 Distance 1

Node 3 via 3 Distance 0

EXPERIMENT 8

8) Write a C/C++ Program to implement Stop and Wait Flow Control Protocol

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main()
{
   int i,j,noframes,x,x1=10,x2;
   for(i=0;i<200;i++)
      srand(1);
   noframes = rand()%200;
   i=1;
   j=1;
   noframes = noframes+5;
   printf("\nNumber of frames is %d",noframes);
   while(noframes>0)
   {
      printf("\nSending frame %d",i);
      srand(x1++);
      x = rand()%10;
      if(x%2==0)
      {
         for(x2=1;x2<2;x2++)
         {
            printf("\nwaiting for %d seconds",x2);
            sleep(x2);
         }
         printf("\nSending frame %d",i);
         srand(x1++);
         x = rand()%10;
      }
      printf("\nAcknowledgement for frame %d",j);
      noframes-=1;
      i++;
      j++;
   }
```

```
    printf("\n End of stop and wait protocols");
    return 0;
}
```

Output:

No of frames is 6
Sending frame 1
Acknowledgement for frame 1
Sending frame 2
Acknowledgement for frame 2
Sending frame 3
Acknowledgement for frame 3
Sending frame 4
Acknowledgement for frame 4
Sending frame 5
Acknowledgement for frame 5
Waiting for 1 second
Sending frame 6
Acknowledgement for frame 6
waiting for 1 second
End of stop and wait protocol

EXPERIMENT 9

9) Write C/C++ program for ERROR detecting code using CRC-CCITT (16-bit)

Program:

```
#include <stdio.h>
int main(void)
{
  int data[50], div[16], rem[16], datalen, divlen, i, j, k, ch;
  printf("Enter the data: ");
  i=0;
  while((ch=fgetc(stdin))!='\n')
  {
    if(ch=='1')
      data[i]=1;
    else
      data[i]=0;
    i++;
  }
  datalen=i;
  printf("Enter the divisor: ");
  i=0;
  while((ch=fgetc(stdin))!='\n')
  {
    if(ch=='1')
      div[i]=1;
    else
      div[i]=0;
    i++;
  }
  divlen=i;
  for(i=datalen; i<datalen+divlen-1; i++)
    data[i]=0;
  datalen=datalen+divlen-1;
```

```
for(i=0;i<divlen;i++)
    rem[i]=data[i]; k=divlen-1;
while(k<datalen)
    if(rem[0]==1)
    {
        for(i=0;i<divlen;i++)
            rem[i]=rem[i]^div[i];
    }
    else
    {
        if(k==datalen-1)
            break;
        for(i=0;i<divlen-1;i++)
        {
            rem[i]=rem[i+1];
            printf("%d\n",rem[i]);
        }
        rem[i]=data[++k];
        printf("%d\n",rem[i]);
    }
    j=1;
    for(i=datalen-divlen+1;i<datalen;i++)
    {
        data[i]=rem[j++];
    }
    printf("The data to be sent is\n");
    for(i=0;i<datalen;i++)
        printf("%d",data[i]);
    printf("\n");
    return 0;
}
```

Output:

```
Enter the data: 1010111
Enter the divisor: 1011
0011
0111
1111
1001
0100
0010
0110
The data to be sent is
10101111110
```

EXPERIMENT 10

10) Write a C/C++ program for Congression control using Leaky Bucket Algorithm.

Program:

```
#include <stdio.h>
#include <stdlib.h>
struct packet
{
   int item;
   int size;
} p[50];
int main()
{
   int i,n,m,k=0;
   int bsize, bfilled, outrate;
   printf("Enter the number of packets:");
   scanf("%d",&n);
   printf("Enter the packets in the order of their arrival time\n");
   for(i=0;i<n;i++)
   {
      printf("Enter the time and size:");
      scanf("%d%d",&p[i].time,&p[i].size);
   }
   printf("Enter the bucket size:");
   scanf("%d",&bsize);
   printf("Enter the output rate:");
   scanf("%d",&outrate);
   m=p[n-1].time;
   i=1;
   k=0;
   bfilled=0
```

```
while (i<=n || bfilled!=0)
{
  printf("\n\n At time %od", i);
  if(p[k].time==i)
  {
    if(bsize >=bfilled +p[k].size)
    {
      bfilled =bfilled + p[k].size)
      printf ("\n%od byte packet is inserted", p[k].size);
      k=k+1;
    }
    else
    { printf("\n%od byte packet is discarded", p[k].size);
      k=k+1;
    }
  }
  if(bfilled ==0)
    printf("\n\No packets to transmit");
  else if( bfilled >=outrate)
  {
    bfilled =bfilled-outrate;
    printf("\n%d bytes transferred", outrate);
  }
  else
  {
    printf("\n%od bytes transferred", bfilled);
    bfilled =0;
  }
  printf("\nPackets in the bucket %d bytes\n", bfilled);
  i++;
}
return 0;
}
```

Output:

Enter the number of packets: 3
Enter packets in the order of their arrival time
Enter the time and size: 1 100
Enter the time and size: 2 400
Enter the time and size: 3 600
Enter the bucket size: 500
Enter the output rate: 200

At time 1
100 byte packet is inserted
100 bytes transfered
Packets in the bucket 0 byte

At time 2
400 byte packet is inserted
200 bytes transfered
Packets in the bucket 200 byte

At time 3
600 byte packet discarded
200 bytes transfered
Packets in the bucket 0 byte