EXPERIMENT-06

QUESTION: Write a program to implement the ARP protocol using socket programming.

CODE:

Client side:

```
import java.io.*;
import java.net.*;
class arp.client
{
    public static void main (String[]args)throws IoException
    {
     Socket s = new Socket("localhost", 55);
    DataInputStream in = new DataInputStream(s.getInputStream));
    DataOutputStream out = new DataOutputStream(s.getOutputputStream));
    DataInputStream sysin = new DataOaputStream(System.in));
    System.out.println("Enter IPaddress:");
     String str = sysin.readLine();
     out.writeBytes( str+"\n");
     System.out.println(" The MAC address is: "+in.readLine());
    }
}
```

—> Server side:

```
import java.net.*;
import java.io.*;
public class arpserver
{
 public static void main (String[ ]args)
 { ServerSocket ss = new ServerSocket(55);
   Socket ss = s.accept();
   DataInputStream in = new DataInputStream(s.getInputStream);
   DataOutputStream out= new DataOutputStream(s.getOutputStream());
   String iparr[]={"10.0.1.45","172.16.5.21", "172.16.5.22"};
   String macarr[] = {"00-0c-6e-5c-3c-63","02-11-B6-F3-EF-21",
                   "03-12-B3-F3-EF-18"}};
   String str = in.readLine();
   System.out.println(" Ip received "+str);
   int flag=0;
   for(int i=0; i<3; i++)
   {  if(str.equals (iparr[i]==true)
      {
         flag = 1;
         String str1 = macarr[i];
         out.writeBytes(str+"\n");
         break;
      }
   }  if(flag==0)
      {
        System.out.println(" IP not in network");
      }
      s.close();
 }
}
```

EXPERIMENT-07

→ QUESTION: Write a program to implement the FTP protocol using socket programming.

→ CODE:-

- Client side:

```
import java.io.*;
import java.net.*;
public class ftpClient {
   public static void main(String [] args)
 {
 Socket s = new Socket(InetAddress.getLocalHost(),5555);
 DataInputStream s1 = new DataInputStream(s.getInputStream());
 DataInputStream inp = new DataInputStream(System.in);
 DataOutputStream so = new DataOutputStream(s.getOutputStream());
 System.out.println("Enter path:\n");
 String str = inp.readLine();
 FileOutputStream fos = new FileOutputStream("output.txt");
 int str1;
 while ((str1 = s1.read())! = -1)
           fos.write((char)str1);

 System.out.println("File received\n");
  s1.close();
  so.close();
  inp.close();
  s.close();
  }
}
```

Server Side:

```
import java.net.*;
import java.io.*;
Class ftpserver
{
  public static void main (String[]args)
  {
   ServerSocket ss =new ServerSocket(5555);
   Socket S = ss.accept();
   DataOutputStream dos= new DataOutputStream(S.getOutputStream());
   DataInputStream din = new DataInputStream(S.getInputStream());
   String s1;
   S1 = din.readLine();
   FileInputStream fin = new FileInputStream(s1);
   int str1;
  while((str1 = fin.read())!= -1)
          dos.writeBytes("(char) str1);
  System.out.println("File Sent");
  dos.close();
  din.close();
  S.close();
 }
}
```

MODULE-4
WIRESHARK TOOL

→ INTRODUCTION:

- Wireshark formerly known as Ethereal is one of the most powerful tools in a network security analysts kit. Wireshark can peer inside the network and examine the details of traffic at a variety of levels using connection level information and to the bits comprising a single packet.

- Features of Wireshark:
  1. Available in both UNIX and Windows.
  2. Ability to capture live packets from various interfaces
  3. Filters packet with many criteria.
  4. Can save & merge captured prokets.

- The flexibility and depth of inspection allows the valuable tool to analyze security events and troubleshoot network security device issues.

- The installation of this is easily available as we can find the source code at www.wireshark.org and we have the links that are compatible with Linux and x32 bit and x64 bit systems.

MODULE-5
NS2 SIMULATOR

EXPERIMENT-01
THREE NODE POINT TO POINT NETWORK.

AIM: To simulate a three node point to point network with duplex links between them. Set queue size and vary the bandwidth and find number of packets dropped.

TCL file:

```
Set ns [new Simulator]
set nf [open PA1.nam w]
$ns namtrace-all $tf
proc finish { } {
 global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam PA1.nam &
exit 0
}
sent n0 [$nsnode]
 set n2 [$nsnode]
$ns duplex-link $n0 $n2 200mb 10ms DropTail
$ns duplex-link $n2 $n3 1mb 1000ms DropTail
$ns queue-limit $n0 $n2 10
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_500
```

```
$cbr0 set-interval_0.005
$ cbr0 attach-agent $udp0
set null0 [new Agent/NULL)
$ns attach agent $n3 $null0
$ns connect $udp0 $null0
$ ns at 0.1 "$cbr0 start"
$ns at 1.0 "finish"
$ ns run.
```

→ Awk file:

```
BEGIN {c=0;}
{
 if ($1=="d")
 { c++;
   printf("%s\t%s\n", $5,$11);
 }
 END{
 printf(" The number of packets dropped =%d\n",c);
    }
```

EXPERIMENT-02

TRANSMISSION OF PING MESSAGE

→ AIM: To simulate transmission of ping message over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

→ TCL file:

```
set ns [new Simulator]
set nf [open lab4.nam w]
$nf nam-trace-all $nf
set tf[" open lab4.tr w]
$ns trace-all $tf
set n0[$ns node]
set n1[$ns node]
set n2[$ns node]
set n3[$nsnode]
set n4[$nsnode]
set n5[$ns node]
$ns duplexlink $n0 $n4 $005mb 1ms DropTail
$ns duplex-link $n1 $n4 50mb 1ms DropTail
$ns duplex-link $n2 $n4 2000mb 1ms DropTail
$ns duplex-link $n3 $n4 200mb 1ms DropTail
$ns duplex link $n4 $n5 1mb 1ms DropTail
set p1 [new Agent/ping]
$ns attach-agent $n0 $n1
$p1 packet_size_50000
$p1 set_interval_0.0001
set p2 [new Agent/Ping]
$ns attach-agent $n2 $p3
```

```
$p3 set packeSize _30000
$p3 set_interval_0.00001
set p4 [new Agent/Ping]
ns attach-agent $n3 $p4
set p5[new Agent/Ping]
$ns queue limit $n0 $n4 5
$ns queue limit $n2 $n4 3
$ns queue limit $n4 $n5 2
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node[$node_id] from $from with $rtt msec.
$ns connect $p1 $p5
$ns connect $p3 $p4
proc finish {} {
global ns tf nf
$ns flush-trace
close $nf
close $tf
exec nam lab4.nam &
exit 0
}
$ns at 0.1 "$p1 send"
$ns at 0.2 "$p1 send"
      :
$ns at 0.9 "$p1 send"
$ns at 1.0 "$p1send"
$ns at 1.1 "$p1 send"
      :
$ns at 1.9 "$p1send"
```

```
$ns at 2.0 "$p3 send"
$ns at 2.1 "$p3 send"
         :
$ns at 2.9 "$p3 send"
$ns at 0.1 "p3 send"
         :
$ns at 0.9 "p3 send"
$ns at 1.0 "p3 send"
$ns at 1.1 "p3 send".
         :
$ns at 2.0 "p3 send"
$ns at 2.1 "p3 send"
         :
$ns at 2.9 "p3 send".
$ns at 3.0 "finish".
$ns run.
```

→ AWK file:

```
BEGIN{
drop=0;
}{
   if($1=="d")
   {
     drop++;
   }
   END{
     printf("Total %s packets dropped=%d \n",$5,drop);
   }
```

EXPERIMENT-03

ETHERNET LAN USING n nodes and set multiple traffic nodes and plot congestion window.

→ AIM: To simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source/destination.

→ TCL file:-

```
set ns [new Simulator]
set tf [open p7.tr w]
$ns trace-all $tf
set nf [open p7.nam w]
$nam trace-all $nf
set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"
set n1 [$ns node]
set n2 [$ns node]
$n2 color "magenta"
$n2 label "src2"
set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"
set n4 [$ns node]
set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100mb 100ms LL Queue/DropTail MAC/802_3
```

```
$ns duplex-link $n4 $n5 1mb 1ms DropTail
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/TCP]
$ ftp0 attach-agent tcp0
$ ftp0 set packet_size 500
$ ftp set interval 0.0001
sent sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$tcp2 attach$file2
$tcp0 trace cwnd_
cwnd_
proc finish{} {
 global ns nf tf
$ns flush-trace
close $tf
close $nf
exec nam p7.nam &
exit 0
}
```

```
$ns at 0.1 "$ftp0 start"
$ns at 5 "$ftp0 stop"
$ns at 7 "$ftp0 start"
$ns at 0.2 "$ftp2 start"
$ns at 8 "$ftp2 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp2 start"
$ns at 15 "$ftp2 stop"
$ns at 16 "finish"
$ns run.
```

→ AWK file :

```
BEGIN{
  {
    if($6=="cwnd_")
    printf("%f\t %f\t \n",$1,$7);
  }
  END{
  }
```

EXPERIMENT-04

SIMPLE ESS WITH WIRELESS LAN.

→ AIM : To simulate simple ESS with transmitting nodes in wireless LAN.

→ TCL file:

```
set ns [new Simulator]
set tf [open lab8.tr w]
$ns trace-all $tf
set topo [new topography]
$ topo load_flatgrid 1000 1000
set nf [open lab8.nam w]
$ns nametrace-all-wireless $nf 1000 1000
$ns node-config -adhocROUTING DSDV\
-llType LL
-macType Mac/802-11\
-ifqTypequeue/DropTail
-ifqlen50\
-phyType Phy/WirelessPhy\
-chanelType/Wireless channel\
-antType antenna /Omni Antenna
-topoInstance $topo\
-agentTrace ON\
-routerTrace ON
create-god 3
set no[$nsnode]
set n1[$nsnode]
set n2[$nsnode]
$n0 label "tcp0"
$n1 label "sink-1/tcp1"
```

```
$set n0 X_50
$n0 set Y_50
$n1 set X_100
$ n1 set Y-100
$ n2 set X_600
$ n2 set Y_600
$n0 set Z_0
$ n1 set Z_0
$ n2 set Z_0
$ ns at 0.1 "$n0 setdest 50 50 15"
$ ns at 0.1 "$n1 setdest 100 100 25"
$ ns at 0.1 "$n2 setdest 600 600 25"
set tcp0[new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0[new-Application/FTP]
$ns connect $tcp0 $ sink 1
set tcp1 [new Agent/TCP]
$ns attach-agent $1 $ tcp1
set ftp1[new-Application/FTP]
$ ftp 1 attach_agent $tcp 1
set sink 2 [new_Agent/TCPSink]
$ns attach-agent $n2 $sink2
$ ns at 5 "$ftp0 start"
$ ns at 5 " $ftp1 start"
$ns at 100 "$n1 setdest 550 550 15"
$ ns at 190 " $n1 setdest 70 70 15"
proc finish { } {
 global ns nf tf
 $ns flush-trace
 exec nam lab8.nam &
 close $tf
 exit 0
}
 $ ns at 250 "finish"
 $ns run.
```

→ AWK FILE :

```
BEGIN {
  count1 = 0
  count 2 = 0
  pack 1 = 0
  pack 2 = 0
  time 1 = 0
  time 2 = 0
      }
 { if ($ = = "r" && $3 = = "1" && $4 = = "AGT")
   {  count 1++
      pack 1 = pack1 + $8
      time 1 = $2
   }
   if ($1 = "r" && $3 = = "2" && $4 = "AGT")
   {   count2 ++
       pack 2 = pack2 + $8
       time 2 = $2
   }
 }
END {
printf(" Throughput from no to n1 : %f mbps \n",
      ((count1 * pounk1 * 8) / (time1 * 1000000));
printf(" Throughput from n1 to n2 : %f mbps \n",
      ((count2 * pack 2 * 8) / (time2 * 1000000));
```