**CN LAB MANUAL**

**ROLL NO – 12012084**

**NAME – ASTHA**

**SEC – CSB5**

**LAB – 1**

Q1. Study the history of internet.

ANS. The initial days of internet backs to 1957. In the midst of the Cold War, October 4 1957, Soviets  launched the first man made satellite into space. this was alarming for Americans. So Americans  started to think more seriously about science and technology. US Administration funded various  agencies, one of them being ARPA. ARPA stands for Advanced Research Project Agency. It was a  Defence Department research project in Computer Science, a way for scientists and researchers to  share information, findings, knowledge, and communicate. Without ARPA the Internet would not  exist. It was because of this institution that the very first version of the Internet was created –  ARPANET. The goal now was resource sharing, whether that was data, findings, or applications. It  would allow people, no matter where they were, to harness the power of expensive computing that  was far away, as if they were right in front of them.

 By the end of 1969 a connection had been established between four nodes on the whole network  which included UCLA, SRI, UCSB (University of California Santa Barbara) and the University of Utah.  But the network grew steadily throughout the years and more and more universities joined. By 1973  there were even nodes connecting to England and Norway. ARPANET managed to connect these  supercomputing centers run by universities together into its network.

As time passed, more independent packet switched networks emerged that were not related to  ARPANET. These different networks had their own dialects, and their own standards for how data  was transferred. It was impossible for them to integrate into this larger network, To build an open  network of networks, a general protocol was needed. That is, a set of rules.

The rules for the Interconnection were:

• The independent networks were not required to change

• There was an effort to achieve communication

• Internal networks would exist in addition with gateways that would connect these networks. Their  job would be to translate between the networks. There would be one universal, agreed upon  protocol for that.

• There would be no central control, no one person or organization in charge.

The interconnected global network of networks was finally starting to happen. It was still mainly  used widely by researchers, scientists, and programmers to exchange messages and information.

Q2. Study different types of network cable.

ANS.

1. RJ45 connector

A registered jack (RJ) is a standardized physical network interface for connecting  telecommunications or data equipment. The physical connectors that registered jacks use  are mainly of the modular connector and 50-pin miniature ribbon connector types. The most  common twisted-pair connector is an 8-position, 8-contact (8P8C) modular plug and jack  commonly referred to as an RJ45 connector.

2. Twisted pair cable

This cable is a type of ordinary wiring which connects home and many business computers  to the telephone company. It is made by putting two separate insulated wires together in a  twisted pattern and running them parallel to each other, which helps to reduce crosstalk or  electromagnetic induction between pairs of wires.

Twisted pair cable is suitable for transferring balanced differential signals. The method of  transmitting signals dates back to the early days of the telegraph and radio. The advantages  of improved signal-to-noise ratio, crosstalk, and ground bounce that balanced signal  transmission brings are particularly valuable in wide bandwidth and high fidelity systems.

3. Co-axial cable

Coaxial cables have a single copper conductor at the center, while a plastic layer provides  insulation between the center conductor and braided metal shield. The metal shield blocks  outside interference from fluorescent lights, motors, and other computers.

The coaxial cable acts as a high-frequency transmission cable made up of a single solid  copper core and compared to twisted pair cable. It has 80 times or more transmission  capability. This kind of cable is mainly adopted in feedlines connecting radio transmitters  and receivers with their antennas, computer network connections, and distributing cable  television signals.

4. Fibre optic cable

Fiber optic cables possess a center glass core surrounded by multiple layers of protective  materials. They avoid electrical obstruction by transmitting light instead of electronic signals,  making them perfect for environments with large amounts of electrical interference. Fiber  optic cables have become the standard for connecting networks across buildings because of  their resistance to moisture and lighting.

**LAB – 2**

Q1. Study various network devices Switch, Hub, gateway, Bridge, Repeaters, Routers.  ANS.

 Switch : A switch is a multiport bridge with a buffer and a design that can boost its efficiency and  performance. A switch is a data link layer device. The switch can perform error checking before  forwarding data, which makes it very efficient as it does not forward packets that have errors and  forward good packets selectively to the correct port only.

Hub : A hub is basically a multiport repeater. A hub connects multiple wires coming from different  branches, for example, the connector in star topology which connects different stations. Hubs  cannot filter data, so data packets are sent to all connected devices. In other words, domain of all  hosts connected through Hub remains one

Gateway : A gateway is a network node used in telecommunications that connects two networks  with different transmission protocols together. Gateways serve as an entry and exit point for a  network as all data must pass through or communicate with the gateway prior to being routed.

Bridge : A bridge operates at the data link layer. A bridge is a repeater, with add on the functionality  of filtering content by reading the MAC addresses of source and destination. It is also used for  interconnecting two LANs working on the same protocol. It has a single input and single output port,  thus making it a 2 port device.

Repeater : A repeater operates at the physical layer. Its job is to regenerate the signal over the same  network before the signal becomes too weak or corrupted so as to extend the length to which the  signal can be transmitted over the same network. An important point to be noted about repeaters is  that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit  and regenerate it at the original strength.

Router : A router is a device like a switch that routes data packets based on their IP addresses. The  router is mainly a Network Layer device. Routers normally connect LANs and WANs together and  have a dynamically updating routing table based on which they make decisions on routing the data  packets. Router divide broadcast domains of hosts connected through it.

Q2. Study different types of addresses- IP-address, port-address, and physical  ANS.

1. IP-Address : An IP address is a unique address that identifies a device on the internet or a  local network. IP stands for "Internet Protocol," which is the set of rules governing the  format of data sent via the internet or local network. Physical addresses are not adequate in  an internetwork environment where different networks can have different address formats.  A universal addressing system is needed in which each host can be identified uniquely,  regardless of the underlying physical network. No two publicly addressed and visible hosts  on the Internet can have the same IP address.

2. Port-Address :

the main objective of the internet is the process to process communications. For this  purpose, it is necessary to label or name a specific process. Thus the process needs  addresses. The label that is allocated to a process is known as the port address. It is a 16 bit  address field .

3. Physical Address :

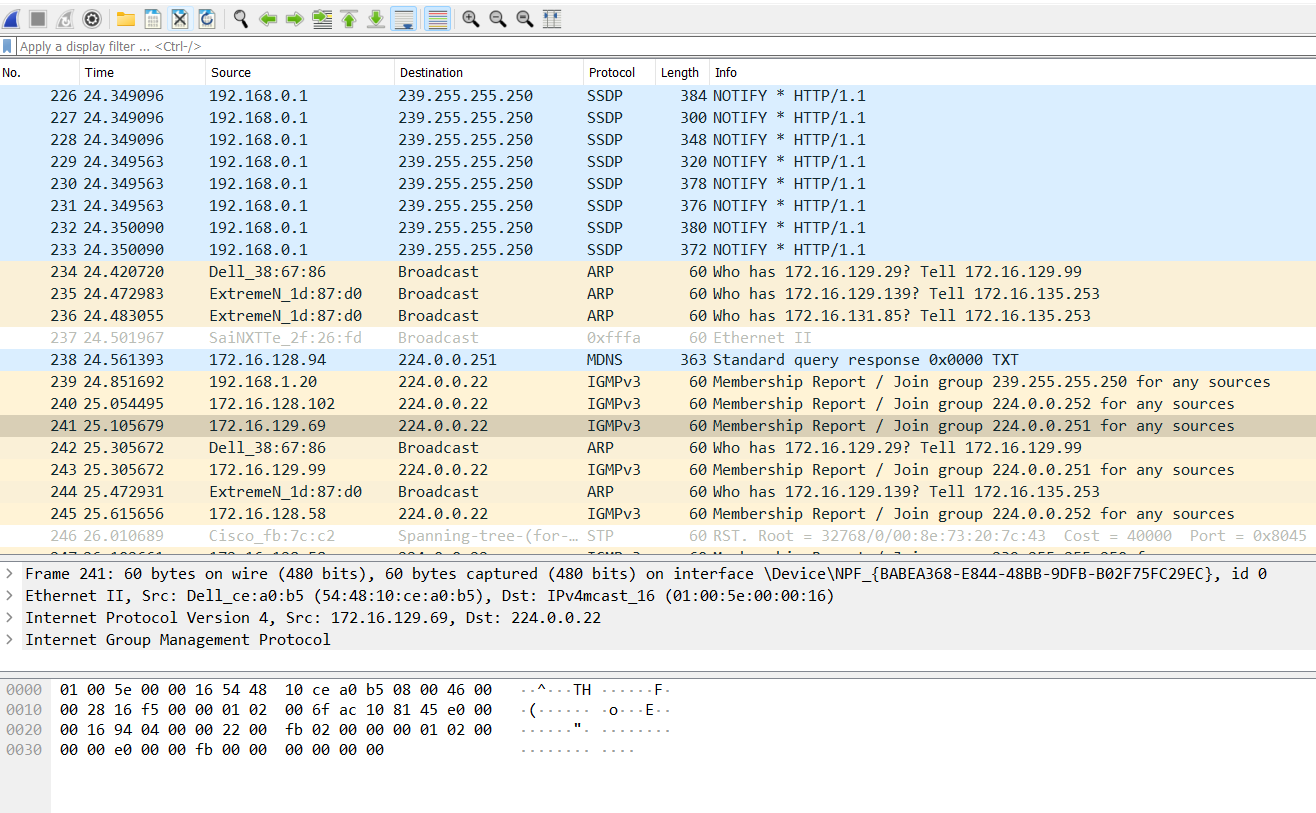
The physical address, also known as the link address, is the address of a node as defined by  its LAN or WAN. It is included in the frame used by the data link layer. It is the lowest-level  address. The size and format of these addresses vary depending on the network. For  example, Ethernet uses a 6-byte (48-bit) physical address that is imprinted on the network  interface card (NIC).

**LAB – 3 & 4**

1. Each line in the top pane of the Wireshark window corresponds to a single packet

seen on the network. Learn Wireshark color coding from help

documents.



2. To locate specific packets related to individual requests or responses from a within

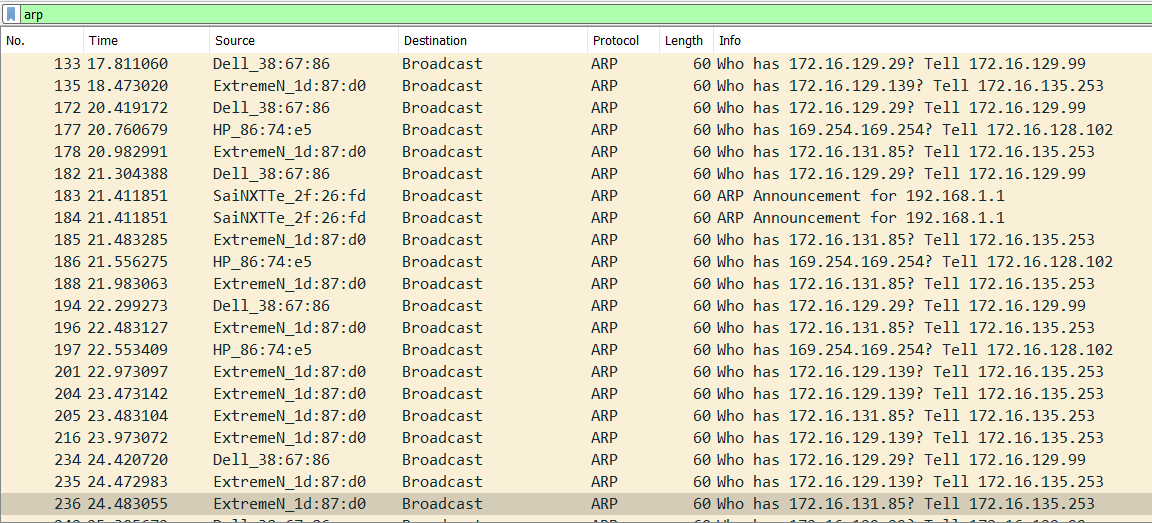
larger capture containing more traffic, we can perform even more specific filtering

using a variety of expressions relating to various header fields and their contents.

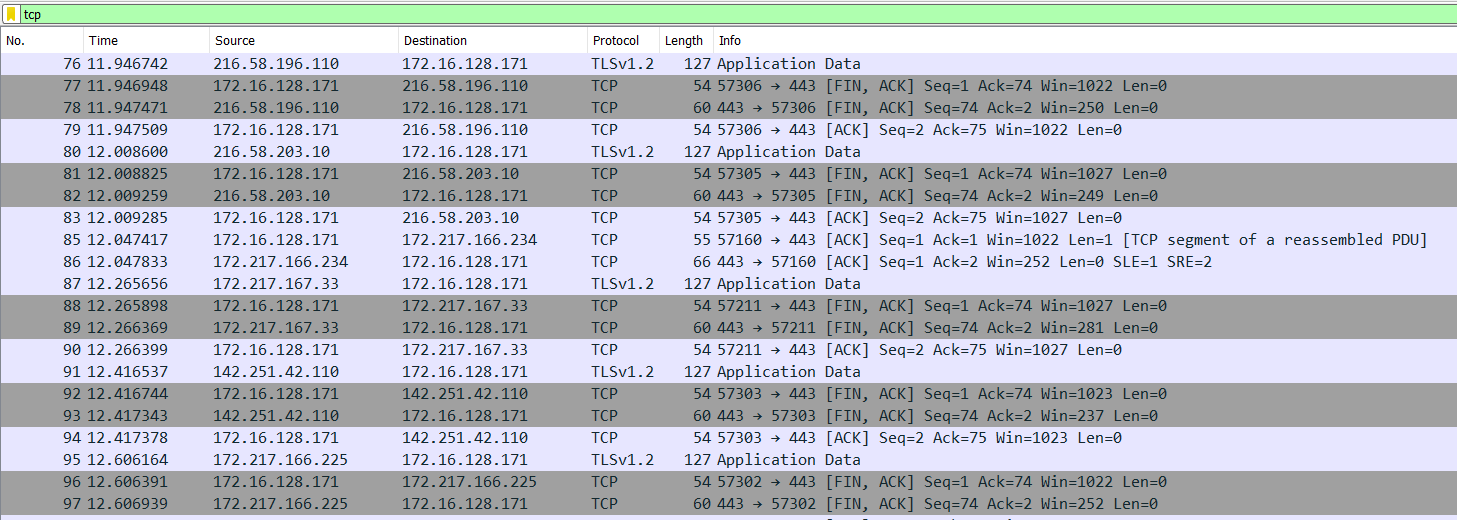
Study how to applying a display filter to the captured packets.

Using display filter we can apply various filter on the captured packets.

1. applying ARP filter

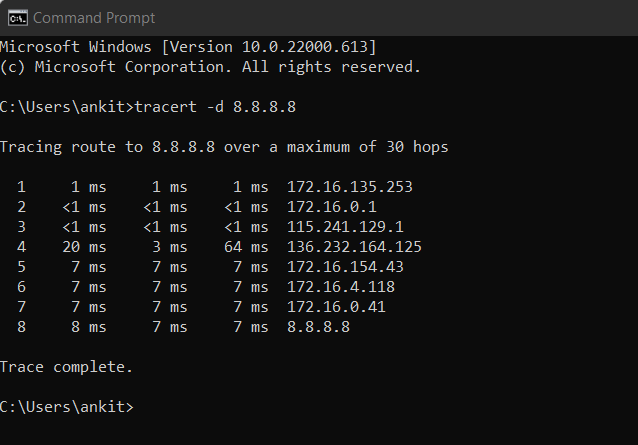


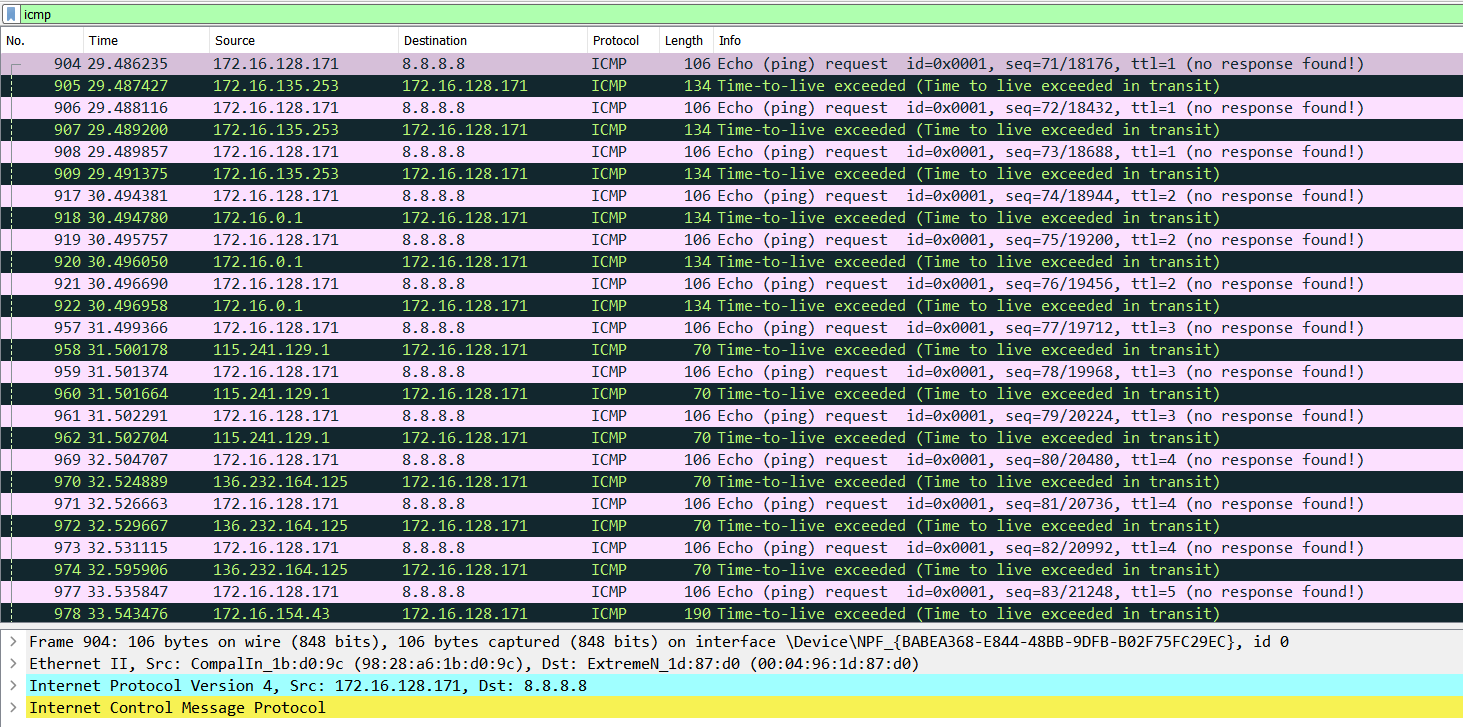
1. applying TCP filter

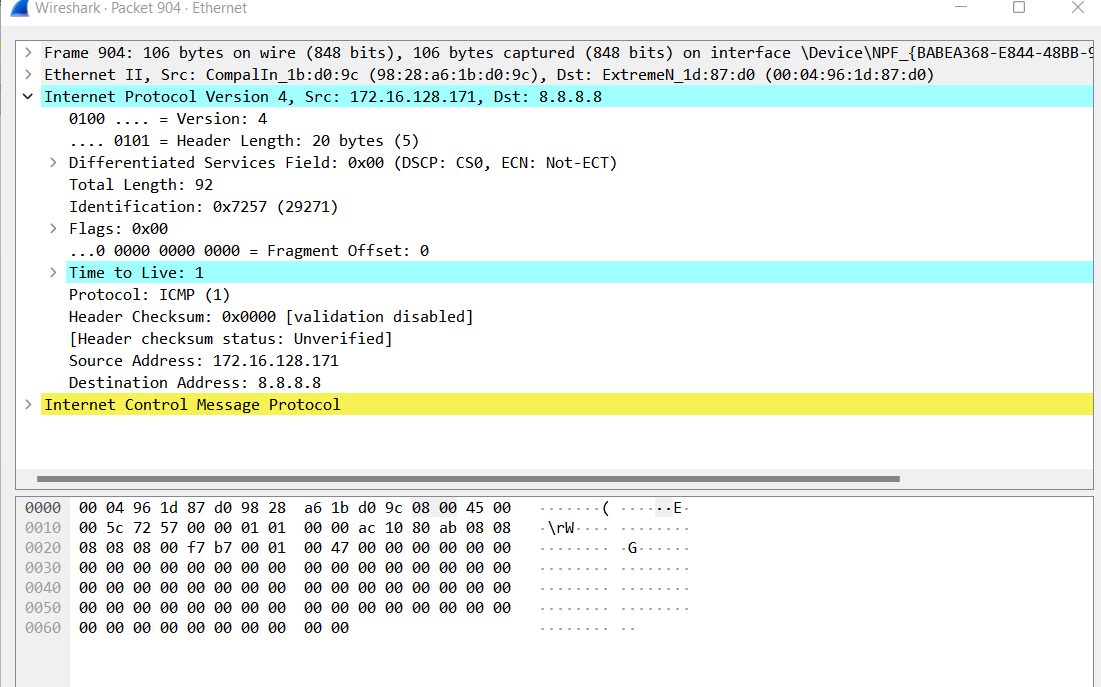


4. Using Wireshark explores the use of the ICMP protocol in the ping and traceroute

commands.





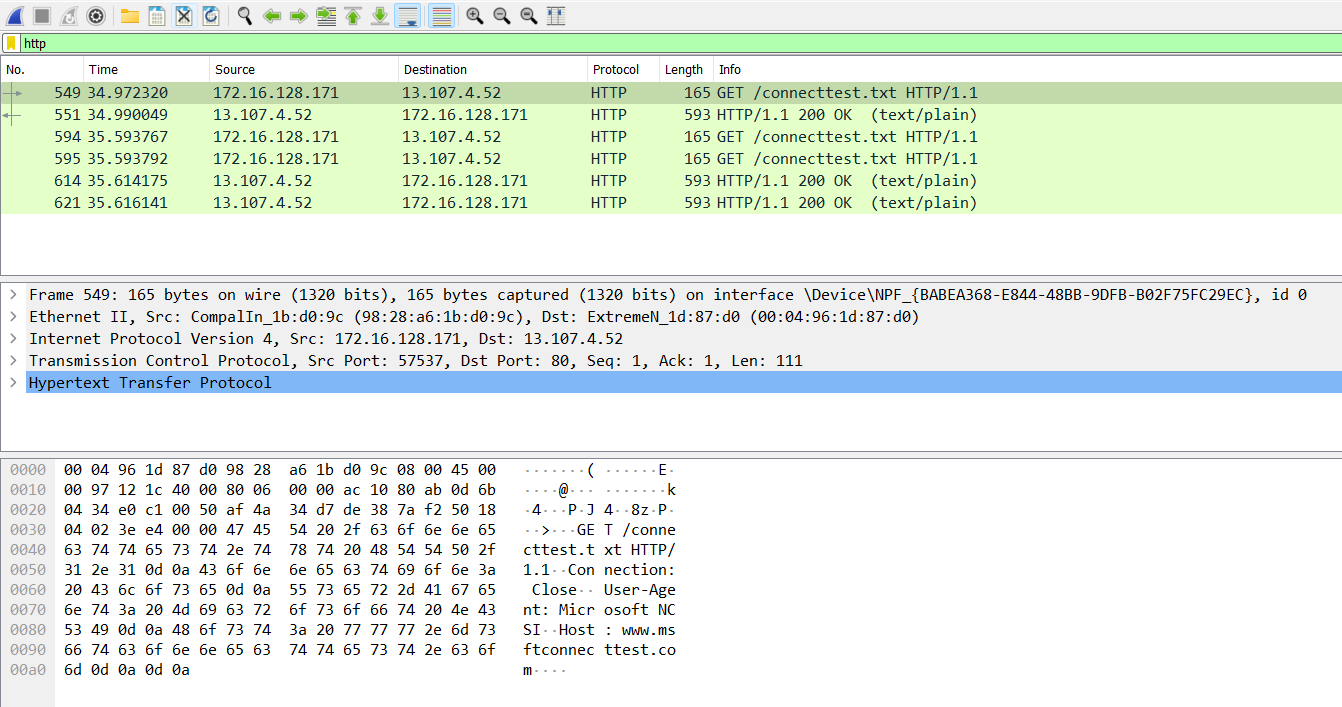


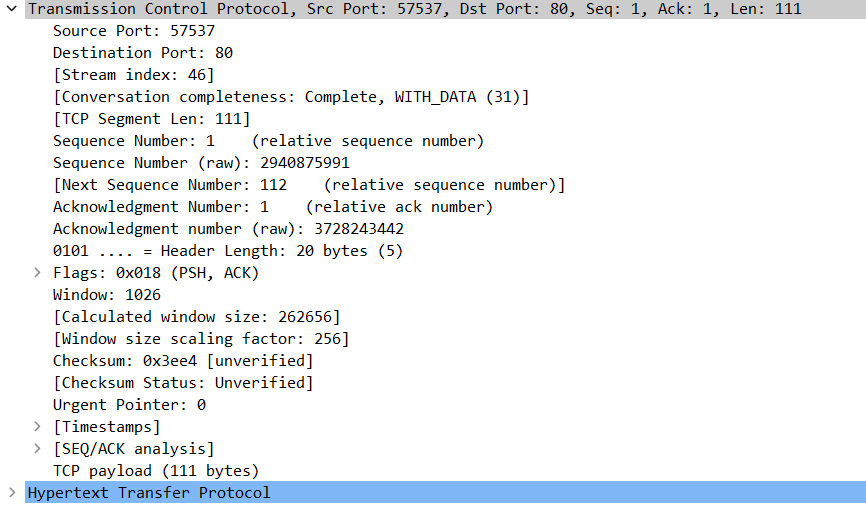
5. Using Wireshark explore several aspects of the HTTP protocol: the basic GET/reply

interaction, HTTP message formats, retrieving large HTML files, retrieving HTML

files with embedded URLs, persistent and non-persistent connections, and HTTP

authentication and security.

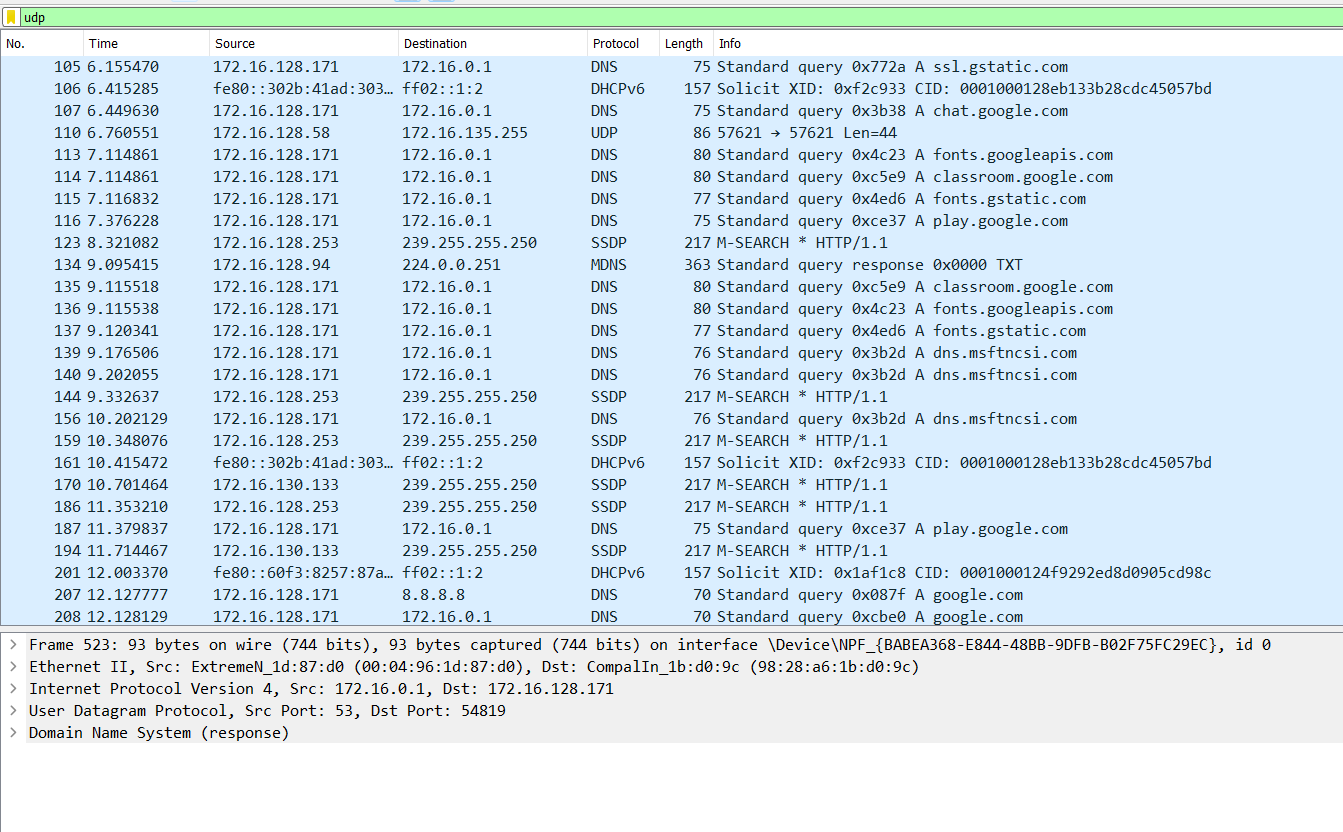


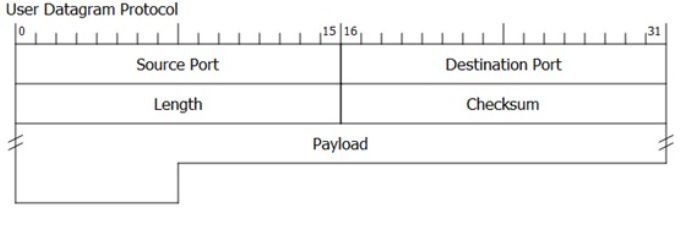


6. Using Wireshark perform a packet capture and analysis of an application that uses

UDP (for example, DNS or a multimedia application such as Skype). Also,

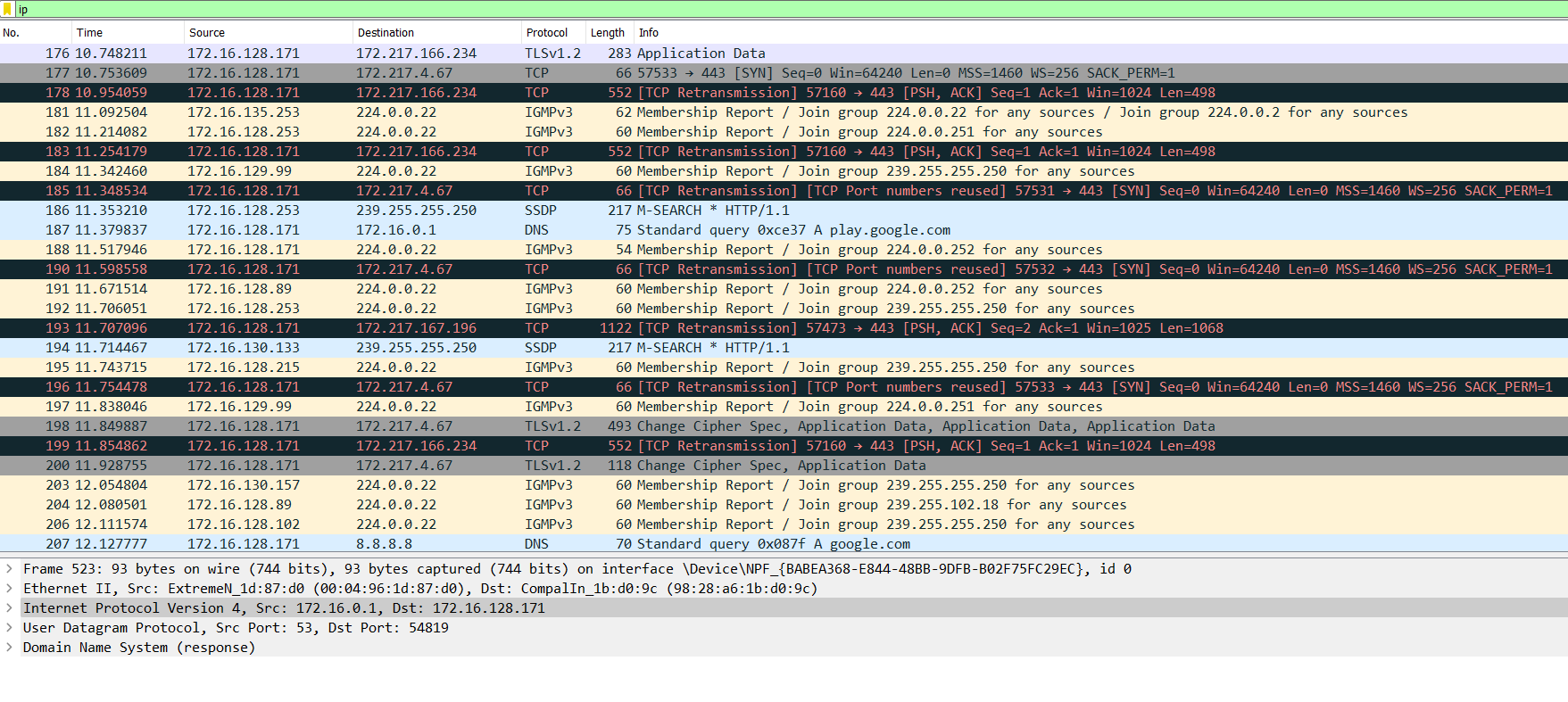
investigate the header fields in the UDP segment as well as the checksum calculation.

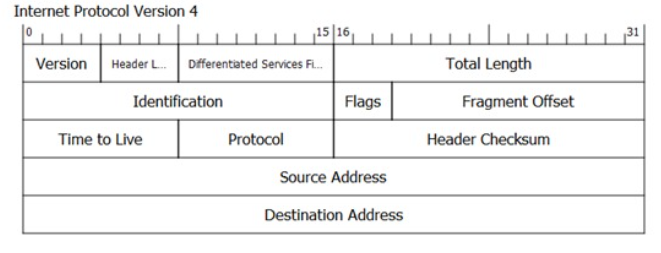




7. Using Wireshark examines the operation of the IP protocol, and the IP datagram

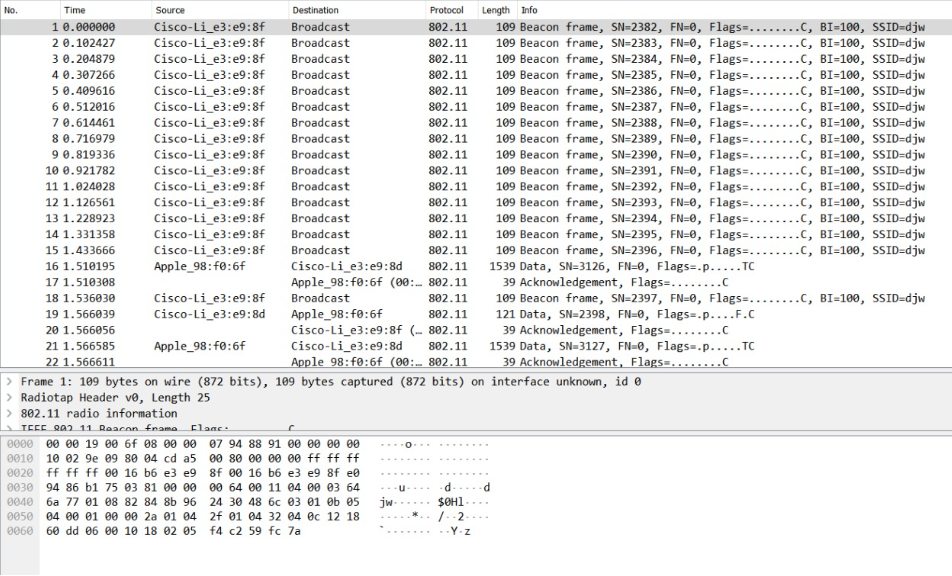
format.



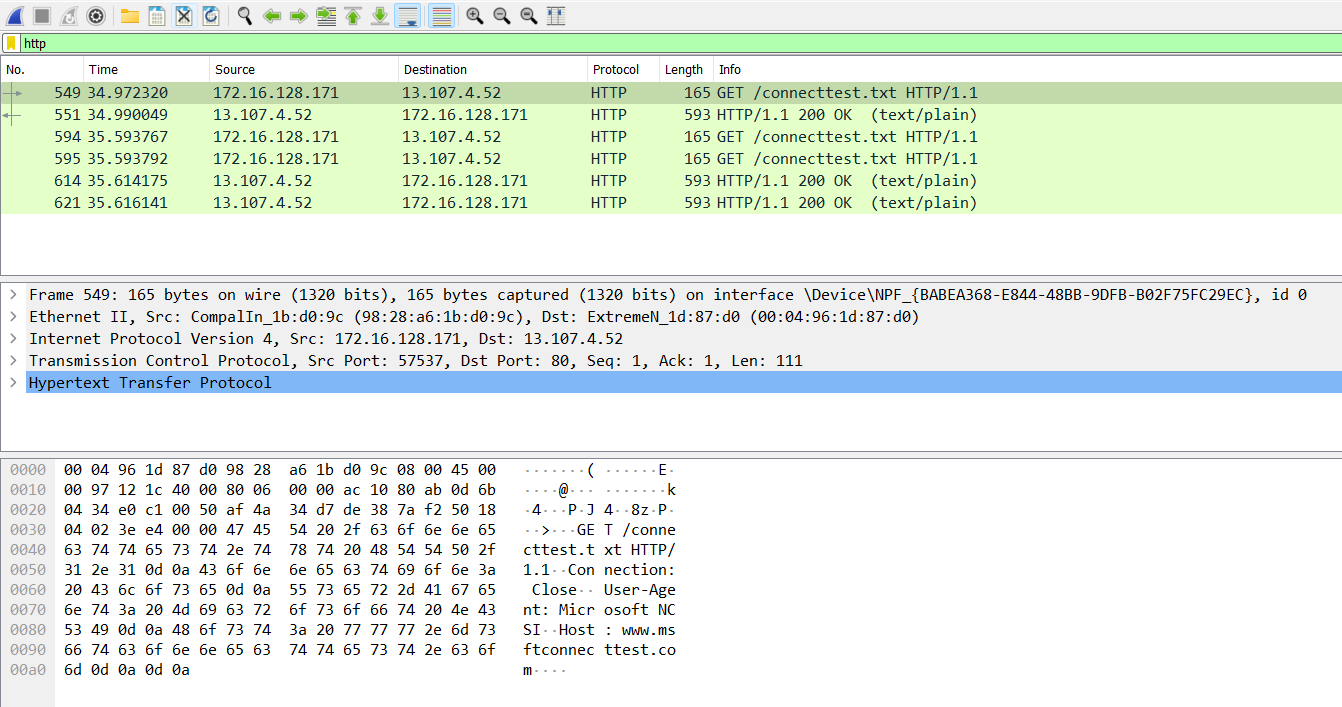


8. Using Wireshark lab perform capture and study the 802.11 frames exchanged

between a wireless laptop and an access point.



**LAB – 5**

****

**LAB – 6 & 7**

**1)  Using NS2 simulate the scenario of Question 1 & 2 of Experiment 2 from the lab manual.**

**Q1) CODE:**

set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

set nf [open out2a.nam w]

$ns namtrace-all $nf

proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec nam out2a.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient right

set udp [new Agent/UDP]

$ns attach-agent $n0 $udp

set null [new Agent/Null]

$ns attach-agent $n1 $null

$ns connect $udp $null

$udp set fid\_ 2

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

$cbr set rate\_ 1.5mb

$cbr set random\_ false

$ns at 0.5 "$cbr start"

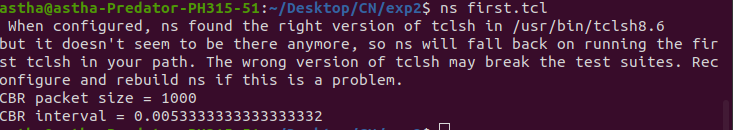
$ns at 4.5 "$cbr stop"

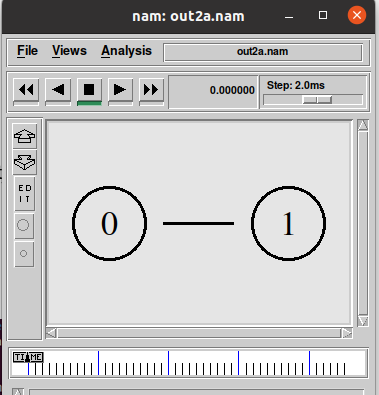
$ns at 5.0 "finish"

puts "CBR packet size = [$cbr set packet\_size\_]"

puts "CBR interval = [$cbr set interval\_]"

$ns run





**Q2) CODE:**

# Create a simulator object

set ns [new Simulator]

# Define different colors

# for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

# Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

# Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

# Close the NAM trace file

close $nf

# Execute NAM on the trace file

exec nam out.nam &

exit 0

}

# Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

# Create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 1 Mb 10ms DropTail

# Set Queue Size of link (n2-n3) to 10

$ns queue-limit $n2 $n3 10

# Give node position (for NAM)

$ns duplex-link-op $n0 $n2 orient left-down

$ns duplex-link-op $n1 $n2 orient left-up

$ns duplex-link-op $n2 $n3 orient right

# Monitor the queue for link (n2-n3). (for NAM)

$ns duplex-link-op $n2 $n3 queuePos 0.5

# Setup a TCP connection

set tcp [new Agent/TCP]

$tcp set class\_ 2

$ns attach-agent $n1 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

# Setup a FTP over TCP connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type\_ FTP

# Setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n0 $udp

set null [new Agent/Null]

$ns attach-agent $n3 $null

$ns connect $udp $null

$udp set fid\_ 2

# Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

$cbr set rate\_ 1mb

$cbr set random\_ false

# Schedule events for the CBR and FTP agents

$ns at 0.1 "$cbr start"

$ns at 0.5 "$ftp start"

$ns at 4.5 "$ftp stop"

$ns at 4.5 "$cbr stop"

# Call the finish procedure after

# 5 seconds of simulation time

$ns at 5.0 "finish"

# Print CBR packet size and interval

puts "CBR packet size = [$cbr set packet\_size\_]"

puts "CBR interval = [$cbr set interval\_]"

# Run the simulation

$ns run

**2)  Implement the following concepts using c++/Java:**

* **bit stuffing**
* **CRC error correction**

**BIT STUFFING:**

**CODE:**

#include <bits/stdc++.h>

#include <iostream>

using namespace std;

void bitStuffing(int N, int arr[])

{

    int brr[30];

    int i, j, k;

    i = 0;

    j = 0;

    int count = 1;

    while (i < N)

    {

    if (arr[i] == 1)

    {

    brr[j] = arr[i];

    for(k = i + 1; arr[k] == 1 && k < N && count < 5;

    k++)

    {

    j++;

    brr[j] = arr[k];

    count++;

    if (count == 5)

    {

    j++;

    brr[j] = 0;

    }

    i = k;

    }

    }

    else

    {

    brr[j] = arr[i];

    }

    i++;

    j++;

    }

    for(i = 0; i < j; i++)

    cout << brr[i];

cout<<”/n”;

}

int main()

{

    int N,i;

    cout<<"enter the number of bits in the msg : ";

    cin>>N;

    int arr[N];

    cout<<"enter the data :";

    for(i=0; i<N;i++)

    {

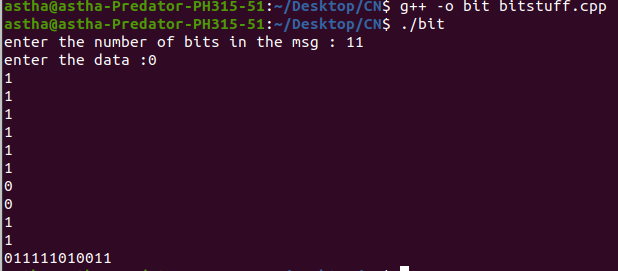
    cin>>arr[i];

    }

    bitStuffing(N, arr);

    return 0;

}



**Cyclic Redundancy Check :**

**CODE:**

#include <iostream>

#include <math.h>

#include <cstring>

using namespace std;

char exor(char a,char b)

{

if(a==b)

return '0';

else

return '1';

}

void crc(char data[], char key[])

{

int datalen = strlen(data);

int keylen = strlen(key);

for(int i=0;i<keylen-1;i++)

data[datalen+i]='0';

data[datalen+keylen-1]='\0';

int codelen = datalen+keylen-1;

char temp[20],rem[20];

for(int i=0;i<keylen;i++)

rem[i]=data[i];

for(int j=keylen;j<=codelen;j++)

{

for(int i=0;i<keylen;i++)

temp[i]=rem[i];

if(rem[0]=='0')

{

     for(int i=0;i<keylen-1;i++)

         rem[i]=temp[i+1];

}

else

{

     for(int i=0;i<keylen-1;i++)

         rem[i]=exor(temp[i+1],key[i+1]);

}

if(j!=codelen)

     rem[keylen-1]=data[j];

else

     rem[keylen-1]='\0';

}

for(int i=0;i<keylen-1;i++)

data[datalen+i]=rem[i];

data[codelen]='\0';

cout<<"CRC="<<rem<<"\nDataword="<<data;

}

int main()

{

char key[20],data[20];

cout<<"Enter the data:";

cin>>data;

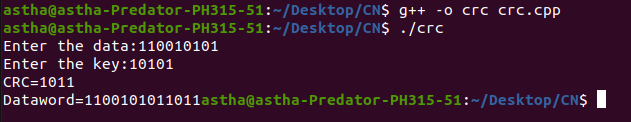
cout<<"Enter the key:";

cin>>key;

crc(data,key);

return 0;

}



**LAB – 8**

**1)  Using NS2 simulate the scenario of Question 3 & 4 of Experiment 2 from the lab manual.**

**Q3) CODE:**

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

#Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

set outfile [open  "bytesReceived.xg"  w]

# procedure to plot the bytesReceived window

proc plotWindow {tcpSource outfile} {

   global ns

   set now [$ns now]

   set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called bytesReceived.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

   puts  $outfile  "$now $cwnd"

   $ns at [expr $now+0.1] "plotWindow $tcpSource  $outfile"

}

proc finish {} {

     global ns nf

     $ns flush-trace

     #Close the NAM trace file

     close $nf

     #Execute NAM on the trace file

     exec nam out.nam &

     exec xgraph bytesReceived.xg -geometry 500x500 &

     exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n1 10Mb 10ms DropTail

#Setup a TCP connection $n0->$n1

set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type\_ FTP

#schedule

$ns  at  0.0  "plotWindow $tcp  $outfile"

$ns at 1.0 "$ftp start"

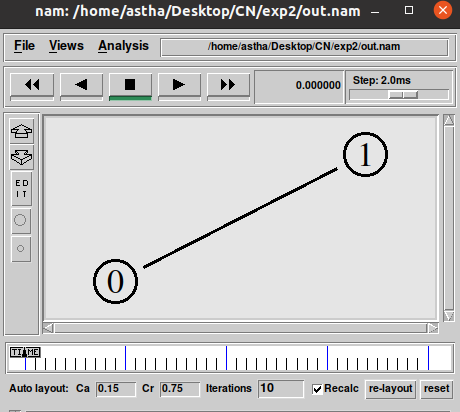
$ns at 4.0 "$ftp stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 6.0 "finish"

#Run the simulation

$ns run



**Q4) CODE:**

set ns [new Simulator]

set namfile [open ex\_01.nam w]

$ns namtrace-all $namfile

set tracefile [open ex\_01.tr w]

$ns trace-all $tracefile

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

$ns duplex-link $n0 $n4 1Mb 10ms DropTail

$ns duplex-link $n1 $n4 1Mb 10ms DropTail

$ns duplex-link $n4 $n3 1Mb 10ms DropTail

$ns duplex-link $n4 $n2 1Mb 10ms DropTail

$ns duplex-link-op $n4 $n0 orient left-up

$ns duplex-link-op $n4 $n1 orient left-down

$ns duplex-link-op $n4 $n2 orient right-up

$ns duplex-link-op $n4 $n3 orient right-down

set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n2 $null

$ns connect $udp $null

$udp set class\_ 1

$ns color 1 Blue

$tcp set class\_ 2

$ns color 2 Red

set cbr [new Application/Traffic/CBR]

$cbr set packetsize\_ 500

$cbr set interval\_ 0.005

$cbr attach-agent $udp

$ns at 0.0 "$cbr start"

$ns at 0.0 "$ftp start"

$ns at 9.0 "$cbr stop"

$ns at 9.0 "$ftp stop"

proc finish {} {

global ns namfile tracefile

$ns flush-trace

close $namfile

close $tracefile

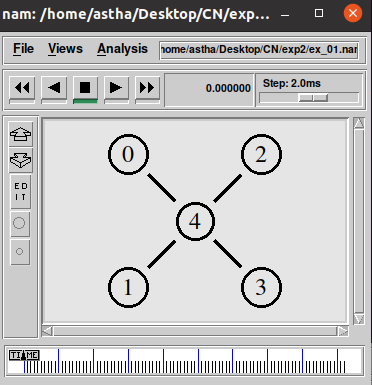
exec nam ex\_01.nam &

exit 0

}

$ns at 10.0 "finish"

$ns run



**2)  Implement the following concepts using c++/Java:**

* **Go-back-N Protocol**
* **Select Repeat ARQ protocol**

**GO BACK N**

**CODE:**

#include<bits/stdc++.h>

#include<iostream>

#include<ctime>

#include<cstdlib>

using namespace std;

int main()

{

 int nf,N;

 int no\_tr=0;

 srand(time(NULL));

 cout<<"Enter the number of frames : ";

 cin>>nf;

 cout<<"Enter the Window Size : ";

 cin>>N;

 int i=1;

 while(i<=nf)

 {

  int x=0;

  for(int j=i;j<i+N && j<=nf;j++)

  {

      cout<<"Sent Frame "<<j<<endl;

      no\_tr++;

  }

  for(int j=i;j<i+N && j<=nf;j++)

  {

      int flag = rand()%2;

      if(!flag)

          {

              cout<<"Acknowledgment for Frame "<<j<<endl;

              x++;

          }

      else

          {   cout<<"Frame "<<j<<" Not Received"<<endl;

              cout<<"Retransmitting Window"<<endl;

              break;

          }

  }

  cout<<endl;

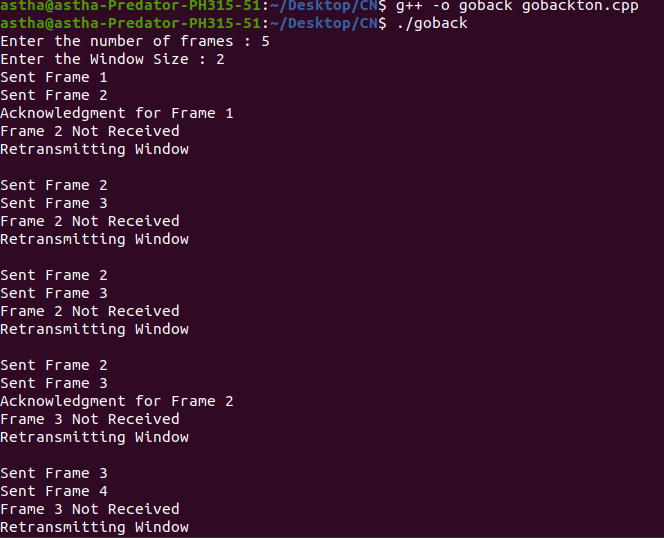
  i+=x;

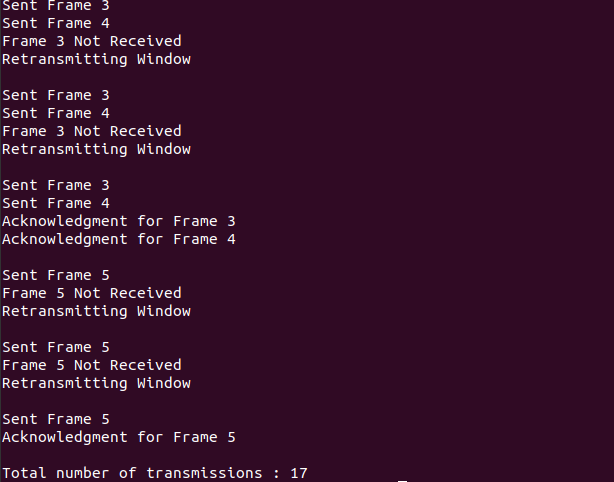
 }

 cout<<"Total number of transmissions : "<<no\_tr<<endl;

 return 0;

}





**SELECT REPEAT ARQ PROTOCOL**

**CODE:**

#include<iostream>

using namespace std;

#include<time.h>

#include<math.h>

#define TOT\_FRAMES 500

#define FRAMES\_SEND 10

class sel\_repeat

{

private:

int fr\_send\_at\_instance;

int arr[TOT\_FRAMES];

int send[FRAMES\_SEND];

int rcvd[FRAMES\_SEND];

char rcvd\_ack[FRAMES\_SEND];

int sw;

int rw;   //tells expected frame

public:

void input();

void sender(int);

void receiver(int);

};

void sel\_repeat::input()

{

int n; //no. of bits for the frame

int m; //no. of frames from n bits

int i;

cout<<"Enter the no. of bits for the sequence no. : ";

cin>>n;

m=pow(2,n);

int t=0;

fr\_send\_at\_instance=(m/2);

for(i=0;i<TOT\_FRAMES;i++)

{

arr[i]=t;

t=(t+1)%m;

}

for(i=0;i<fr\_send\_at\_instance;i++)

{

send[i]=arr[i];

rcvd[i]=arr[i];

rcvd\_ack[i]='n';

}

rw=sw=fr\_send\_at\_instance;

sender(m);

}

void sel\_repeat::sender(int m)

{

for(int i=0;i<fr\_send\_at\_instance;i++)

{

if(rcvd\_ack[i]=='n')

cout<<"SENDER : Frame "<<send[i]<<" is sent\n";

}

receiver(m);

}

void sel\_repeat::receiver(int m)

{

time\_t t;

int f;

int j;

int f1;

int a1;

char ch;

srand((unsigned)time(&t));

for(int i=0;i<fr\_send\_at\_instance;i++)

{

if(rcvd\_ack[i]=='n')

{

f=rand()%10;

//if f=5 frame is discarded for some reason

//else frame is correctly recieved

if(f!=5)

{

for(int j=0;j<fr\_send\_at\_instance;j++)

if(rcvd[j]==send[i])

{

cout<<"reciever:Frame"<<rcvd[j]<<"recieved correctly\n";

rcvd[j]=arr[rw];

rw=(rw+1)%m;

break;

}

int j;

if(j==fr\_send\_at\_instance)

cout<<"reciever:Duplicate frame"<<send[i]<<"discarded\n";

a1=rand()%5;

//if al==3 then ack is lost

//else recieved

if(a1==3)

{

cout<<"(acknowledgement "<<send[i]<<" lost)\n";

cout<<"(sender timeouts-->Resend the frame)\n";

rcvd\_ack[i]='n';

}

else

{

cout<<"(acknowledgement "<<send[i]<<" recieved)\n";

rcvd\_ack[i]='p';

}

}

else

{int ld=rand()%2;

//if =0 then frame damaged

//else frame lost

if(ld==0)

{

cout<<"RECEIVER : Frame "<<send[i]<<" is damaged\n";

cout<<"RECEIVER : Negative Acknowledgement "<<send[i]<<" sent\n";

}

else

{

cout<<"RECEIVER : Frame "<<send[i]<<" is lost\n";

cout<<"(SENDER TIMEOUTS-->RESEND THE FRAME)\n";

}

rcvd\_ack[i]='n';

}

}

}

for(int j=0;j<fr\_send\_at\_instance;j++)

{

if(rcvd\_ack[j]=='n')

break;

}

int i=0;

for(int k=j;k<fr\_send\_at\_instance;k++)

{

send[i]=send[k];

if(rcvd\_ack[k]=='n')

rcvd\_ack[i]='n';

else

rcvd\_ack[i]='p';

i++;

}

if(i!=fr\_send\_at\_instance)

{

for(int k=i;k<fr\_send\_at\_instance;k++)

{

send[k]=arr[sw];

sw=(sw+1)%m;

rcvd\_ack[k]='n';

}

}

cout<<"Want to continue";

cin>>ch;

cout<<"\n";

if(ch=='y')

sender(m);

else

exit(0);

}

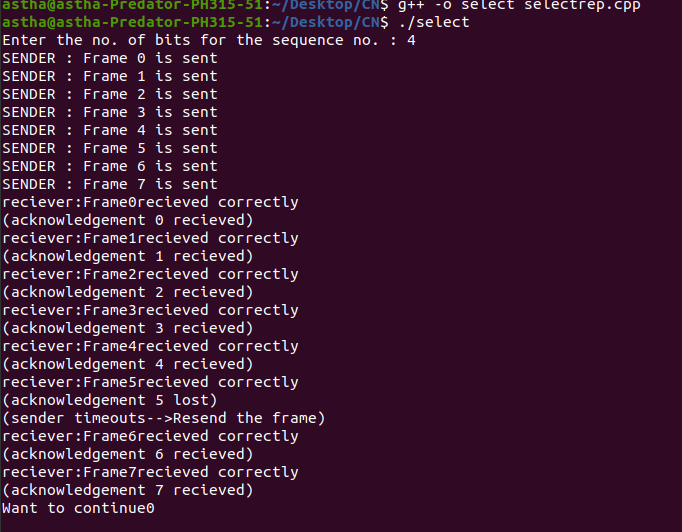
int main()

{

sel\_repeat sr;

sr.input();

}



**LAB – 9 & 10**

Using NS2 simulate the scenario of Question 1 & 2 of Experiment 3 from the lab manual.

Q1.

**CODE:**

set ns [new Simulator]  
set namfile [open ex\_03.nam w]  
$ns namtrace-all $namfile  
  
set tracefile [open ex\_03.tr w]  
$ns trace-all $tracefile    
  
set n0 [$ns node]  
set n1 [$ns node]  
set n2 [$ns node]  
set n3 [$ns node]  
set n4 [$ns node]  
set n5 [$ns node]  
set n6 [$ns node]  
  
set lan [$ns newLan "$n0 $n1 $n2 $n3 $n4 $n5 $n6" 100Mb 0.5ms LL Queue/DropTail Mac/802\_3 Channel Phy/WiredPhy]  
  
set udp [new Agent/UDP]  
$ns attach-agent $n0 $udp  
   
set null [new Agent/Null]  
$ns attach-agent $n6 $null  
  
$ns connect $udp $null  
  
set cbr [new Application/Traffic/CBR];  
$cbr set packetSize\_ 500  
$cbr set interval\_ 0.005  
$cbr attach-agent $udp  
  
$ns at 0.0 "$cbr start"  
$ns at 9.0 "$cbr stop"  
  
proc finish {}  
{      
global ns namfile tracefile  
$ns flush-trace close  
$namfile close  
$tracefile exec nam ex\_03.nam &  
exit 0  
}  
$ns at 0.25 "finish" $ns run

Q2.

**CODE:**

set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

set nf [open out2a.nam w]

$ns namtrace-all $nf

proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec nam out2a.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient right

set udp [new Agent/UDP]

$ns attach-agent $n0 $udp

set null [new Agent/Null]

$ns attach-agent $n1 $null

$ns connect $udp $null

$udp set fid\_ 2

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

$cbr set rate\_ 1.5mb

$cbr set random\_ false

$ns at 0.5 "$cbr start"

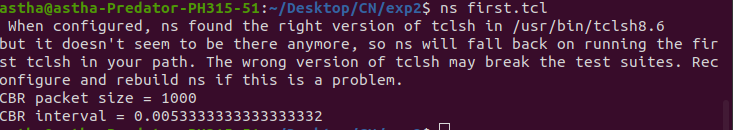
$ns at 4.5 "$cbr stop"

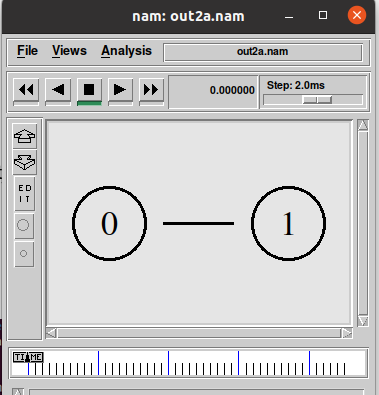
$ns at 5.0 "finish"

puts "CBR packet size = [$cbr set packet\_size\_]"

puts "CBR interval = [$cbr set interval\_]"

$ns run





Q) Configuring switch using Cisco Packet Tracer

