Computer Networks Laboratory (CSL58)

Lab SEE Question Bank

Implement the following in C/C++ orWireshark as suitable.

1. Write a program for error detection using CRC-CCITT (16-bits).
2. Write a program to generate Hamming Code for error detection and correction.
3. Trace Hypertext Transfer Protocol.
4. Trace Domain Name Server.
5. Trace FTP and TCP
6. Trace Internet Protocol and Internet Control Message Protocol.
7. Trace Dynamic Host Configuration Protocol.

PART-B

The following experiments shall be conducted using either NS3simulator/C/C++.

1. Simulate a three nodes point-to-point network with duplex links between them. Set the queue size vary the bandwidth and find the number of packets dropped.
2. Simulate simple Extended Service Set with transmitting nodes in wireless LAN and determine the performance with respect to transmission of packets.
3. Simulate a transmission of ping message over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
4. Write a client-server program using TCP/IP sockets in which client requests for a file by sending the file name to the server, and the server sends back the contents of the requested file if present.
5. Write a program to implement traffic policing using Leaky bucket algorithm.

Solution for Lab exercise

1. Write a program for error detection using CRC-CCITT ($x^{16}+x^{12}+x^5+1$).

```
#include<stdio.h>
#include<string.h>
#define N strlen(g)
char t[28],cs[28],g[]="10001000000100001";
inta,e,c;

voidxor(){
for(c = 1;c < N; c++)
cs[c] = (( cs[c] == g[c])?'0':'1');
}

voidcrc(){
for(e=0;e<N;e++)
cs[e]=t[e];
do{
if(cs[0]=='1')
xor();
for(c=0;c<N-1;c++)
cs[c]=cs[c+1];
cs[c]=t[e++];
}while(e<=a+N-1); }

int main() {
printf("\nEnter data : ");
scanf("%s",t);
printf("\n---------------------------------------");
printf("\nGeneratng polynomial : %s",g);
   a=strlen(t);
for(e=a;e<a+N-1;e++)
t[e]='0';
printf("\n---------------------------------------");
printf("\nModified data is : %s",t);
printf("\n---------------------------------------");
crc();
printf("\n CRC checksum is : %s",cs);
for(e=a;e<a+N-1;e++)
t[e]=cs[e-a];
printf("\n---------------------------------------");
printf("\nFinalcodeword transmitted is : %s",t);
printf("\n---------------------------------------");
printf("\nTest error detection 0(yes) 1(no)? : ");
scanf("%d",&e);
if(e==0)    {
do{
printf("\nEnter the position where error is to be inserted : ");
scanf("%d",&e);
}while(e==0 || e>a+N-1);
```

```
t[e-1]=(t[e-1]=='0')?'1':'0';
printf("\n----------------------------------------");
printf("\nErroneous data : %s\n",t);     }
crc();
for(e=0;(e<N-1) && (cs[e]!='1');e++);
if(e<N-1)            {
printf("\n CRC checksum is : %s",cs);
printf("\nError detected\n\n");            }
else
          {
printf("\n CRC checksum is : %s",cs);
printf("\nNo error detected\n\n");
          }
printf("\n----------------------------------------\n");
return 0;
}
```

Output 1:

```
Enter data : 101
---------------------------------------
Generatngpolynomial : 10001000000100001
---------------------------------------
Modified data is : 1010000000000000000
---------------------------------------
 CRC checksum is : 0101000010100101
---------------------------------------
Final codeword transmitted is : 1010101000010100101
---------------------------------------
Test error detection 0(yes) 1(no)? : 0
Enter the position where error is to be inserted : 3
---------------------------------------
Erroneous data : 1000101000010100101
CRC checksum is : 0001000000100001
Error detected
---------------------------------------
```

Output 2:

```
Enter data : 101
---------------------------------------
Generatngpolynomial : 10001000000100001
---------------------------------------
Modified data is : 1010000000000000000
---------------------------------------
 CRC checksum is : 0101000010100101
---------------------------------------
Final codeword transmitted is : 1010101000010100101
---------------------------------------
Test error detection 0(yes) 1(no)? : 1

 CRC checksum is : 0000000000000000
No error detected
---------------------------------------
```

2. Write a Program in C/ C++ for hamming code generation for error detection/correction

```
#include<stdio.h>

void main() {
int data[10];
intdataatrec[10],c,c1,c2,c3,i;
printf("Enter 4 bits of data one by one\n");
scanf("%d",&data[0]);
scanf("%d",&data[1]);
scanf("%d",&data[2]);
scanf("%d",&data[4]);
   //Calculation of even parity
data[6]=data[0]^data[2]^data[4];
data[5]=data[0]^data[1]^data[4];
data[3]=data[0]^data[1]^data[2];
printf("\nEncoded data is\n");
for(i=0;i<7;i++)
printf("%d",data[i]);

printf("\n\nEnter received data bits one by one\n");
for(i=0;i<7;i++)
scanf("%d",&dataatrec[i]);

   c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
   c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
   c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];
   c=c3*4+c2*2+c1 ;

if(c==0) {
printf("\nNo error while transmission of data\n");
   }
else {
printf("\nError on position %d",c);

printf("\nData sent : ");
for(i=0;i<7;i++)
printf("%d",data[i]);

printf("\nData received : ");
for(i=0;i<7;i++)
printf("%d",dataatrec[i]);

printf("\nCorrect message is\n");

      //if errorneous bit is 0 we complement it else vice versa
if(dataatrec[7-c]==0)
dataatrec[7-c]=1;
else
dataatrec[7-c]=0;
```

```
for (i=0;i<7;i++) {
printf("%d",dataatrec[i]);
        }
    }
}
```

Output 1:

```
Enter 4 bits of data one by one
1
0
1
0

Encoded data is
1010010

Enter received data bits one by one
1
0
1
0
0
1
0

No error while transmission of data
```

Output 2:

```
Enter 4 bits of data one by one
1
0
1
0

Encoded data is
1010010

Enter received data bits one by one
1
0
1
0
1
1
0

Error on position 3
Data sent : 1010010
Data received : 1010110
Correct message is1010010
```

Wireshark

3. HTTP

URL :http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html

1. List up to 10 different protocols that appear in the protocol column in the unfiltered packet-listing window.

   1. SCTP
   2. TCP
   3. UDP
   4. ICMP
   5. ARP
   6. OSPF
   7. GRE
   8. NetBIOS
   9. IPX
   10. VINES

2. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received? (By default, the value of the Time column in the packet listing window is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark*View* pull down menu, then select Time *Display Format*, then select*Time-of-day*.)

   Time of start is 13:25:18.818210
   Time of end is 13:25:19.196493
   Time of used is 0.37872 sec

3. What is the Internet address of the gaia.cs.umass.edu? What is the Internet address of your computer?

   IP address of my computer is 192.168.1.2
   IP address of gaia.cs.umass.edu Dst is 128.119.245.12

The Basic HTTP GET/response interaction

EXERCISE QUESTIONS :-

URL: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the serverrunning?
   browser is running HTTP version is 1.1
   server is running HTTP version is 1.1

2. What languages does your browser indicate that it can accept from the server?
   Accept-Language : en-US

3. What is the status code returned from the server to your browser?
   response code : 200 OK

4. When was the HTML file, that you are retrieving last modified at theserver?
   Last-Modified: Fri , 13 Jun 2008 04:26:01 GMT

5. How many bytes of content are being returned to yourbrowser?
   Content-Length : 128

The HTTP CONDITIONAL GET/response interaction

EXERCISE QUESTIONS :-

URL :http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html

1. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTPGET?

   There is no “IF-MODIFIED-SINCE” line in the HTTP GET.

2. Inspect the contents of the server response. Did the server explicitly returnthe contents of the file? How can youtell?

   Yes the server explicitly returned the contents of the file. The contents can be seen under line based text in the response message.

3. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:”header?

   YES, there is an “IF-MODIFIED-SINCE” field.
   “IF-MODIFIED-SINCE” header is Fri , 13 Jun 2008 04:45:01 GMT \r\n
   which is the same date and time as the last modified field of the previous response message.

4. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

   Response Code : 304 NOT MODIFIED, No the server did not explicitly return the contents of the file as there is no line based text.

Retrieving Long Documents

URL: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html

EXERCISE QUESTIONS :-

1. How many HTTP GET request messages were sent by your browser?

   One HTTP GET request message was sent by the browser

2. How many data-containing TCP segments were needed to carry the single HTTP Response ?

   5 TCP Segments.
   [Reassembled TCP Segments (4809 bytes): #8(309), #9(1452), #16(1452), #17(1452), #19(144)]
   Frame: 8, payload: 0-308 (309 bytes)
   Frame: 9, payload: 309-1760 (1452 bytes)
   Frame: 16, payload: 1761-3212 (1452bytes)
   Frame: 17, payload: 3213-4664 (1452bytes)
   Frame: 19, payload: 4665-4808 (144 bytes)

3. What is the status code and phrase associated with the response to the HTTP GET Request?

   Status code: 200
   Phrase: OK

4. Are there any HTTP status lines in the transmitted data associated with a TCP reassembled segments of a PDU?
   NO

HTML Documents with Embedded Objects

http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-

file4.html

EXERCISE QUESTIONS :

1. How many HTTP GET request messages were sent by your browser? To which Internet addresses were these GET requests sent?

   3 HTTP GET request messages.
   1. Internet addresses is gaia.cs.umass.edu(128.119.245.12)
   2. Internet addresses is manic.cs.umass.edu(128.119.245.12)
   3. Internet addresses is www.pearsonhighered.com(128.119.245.12)

2. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

   If the request for second image is sent after the downloading of first image, then the images are said to be downloaded serially otherwise they are downloaded parallelly. Use the time column to find the answer.

HTTP Authentication

EXERCISE QUESTIONS :

http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html

1. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?

   Response Code : 401 , HTTP/1.1 401 Authorization Required.

2. When your browser sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

   Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=\r\n
   Credentials: wireshark-students:network

4.DNS

EXERCISE QUESTIONS:
URL: http://www.ietf.org

1. Locate the DNS query and response messages. Are they sent over UDP or TCP?
They are sent over UDP

2. What is the destination port for the DNS query message? What is the source port of DNS response message?
The destination port for the DNS query is 53 and the source port of the DNS response is 53.

3. To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?
It's sent to 192.168.1.1 which is the IP address of one of the local DNS servers.

4. Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?
It's a type A Standard Query and it doesn't contain any answers.

5. Examine the DNS response message. How many "answers" are provided? What does each of these answers contain?

There were 2 answers containing information about the name of the host, the type of address, class, the TTL, the data length and the IP address.

```
Answers
www.ietf.org: type A, class IN, addr 209.173.57.180
    Name: www.ietf.org
    Type: A (Host address)
    Class: IN (0x0001)
    Time to live: 30 minutes
    Data length: 4
    Addr: 209.173.57.180
www.ietf.org: type A, class IN, addr 209.173.53.180
    Name: www.ietf.org
    Type: A (Host address)
    Class: IN (0x0001)
    Time to live: 30 minutes
    Data length: 4
    Addr: 209.173.53.180
```

6. Consider the subsequent TCP SYN packet sent by your host. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?
The first SYN packet was sent to 209.173.57.180 which corresponds to the first IP address provided in the DNS response message.

7. This web page contains images. Before retrieving each image, does your host issue new DNS queries?
No

EXERCISE QUESTIONS :nslookup –type=NS mit.edu

8. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?

It's sent to 192.168.1.1 which is the IP address of one of the local DNS servers.

9. Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?

This is a standard type NS query and it contains no answers.

10. Examine the DNS response message. What MIT name servers does the response message provide? Does this response message also provide the IP addresses of the MIT name servers?

The name servers in response message are :

mit.edunameserver = ns1-37.akam.net mit.edu
nameserver = ns1-173.akam.net mit.edu
nameserver = eur5.akam.net mit.edu
nameserver = use2.akam.net mit.edu
nameserver = use5.akam.net mit.edu
nameserver = usw2.akam.net mit.edu
nameserver = asia1.akam.net mit.edu
nameserver = asia2.akam.net

No it is not providing the IP address of the MIT name servers.

5.FTP

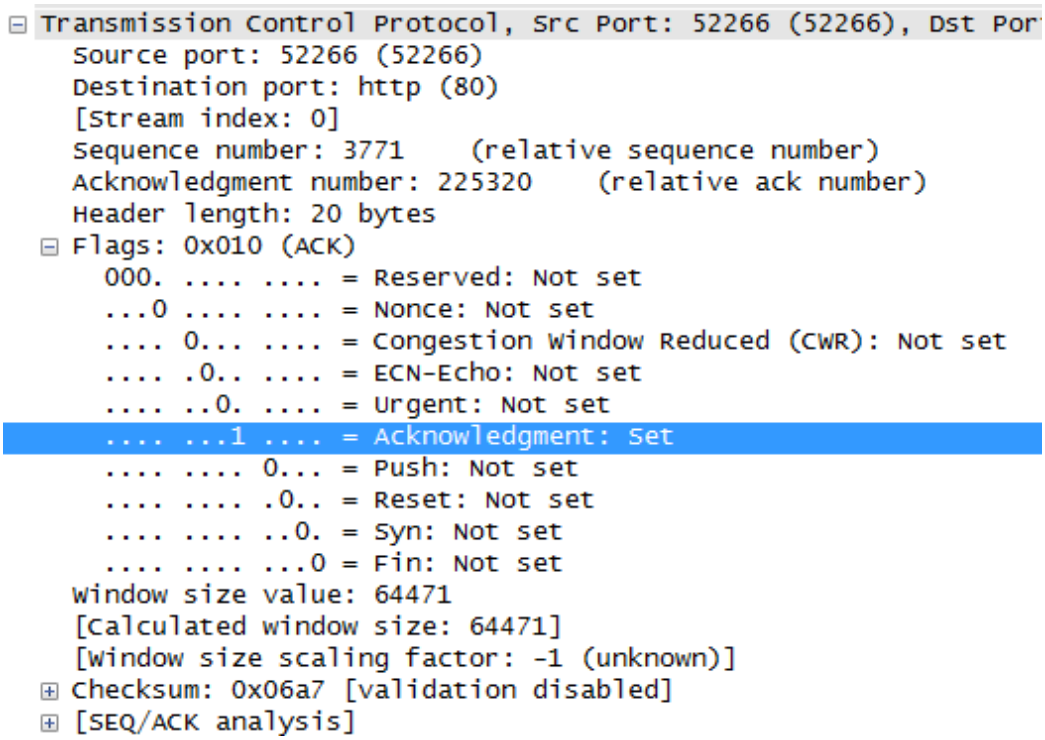
URL: ftp://ftp.cdc.gov/ (use firefox browser)

1. Identify the protocol that is used by transport layer for establishing FTP connection.
   TCP

2. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and ftp.cdc.gov?
   Sequence number is 0

3. What is the sequence number of the TCP SYN ACK segment that is used to initiate the TCP connection between the ftp.cdc.gov and client computer?
   Sequence number of TCP SYN ACK segment is also 0

4. List the values of all the fields in TCP ACK segment.



Write Acknowledgment flag value and Acknowledgment number

5. Identify the FTP commands and response codes during the FTP session.

574	23.765309	198.246.117.106	192.168.43.5	FTP	81 Response: 220 Microsoft FTP Service
575	23.765676	192.168.43.5	198.246.117.106	FTP	70 Request: USER anonymous
577	24.065273	0198.246.117.106	192.168.43.5	FTP	126 Response: 331 Anonymous access allowed, send identity (e-mail name) as password.
578	24.065845	0192.168.43.5	198.246.117.106	FTP	79 Request: PASS chrome@example.com
579	24.585119	0198.246.117.106	192.168.43.5	FTP	75 Response: 230 User logged in.
580	24.585554	0192.168.43.5	198.246.117.106	FTP	60 Request: SYST
592	24.941335	0198.246.117.106	192.168.43.5	FTP	70 Response: 215 Windows_NT
593	24.941953	0192.168.43.5	198.246.117.106	FTP	59 Request: PWD
608	25.540813	0198.246.117.106	192.168.43.5	FTP	85 Response: 257 "/" is current directory.
609	25.541200	0192.168.43.5	198.246.117.106	FTP	62 Request: TYPE I
612	25.851458	0198.246.117.106	192.168.43.5	FTP	74 Response: 200 Type set to I.
613	25.851953	0192.168.43.5	198.246.117.106	FTP	62 Request: SIZE /
642	27.287999	0198.246.117.106	192.168.43.5	FTP	78 Response: 550 Access is denied.
643	27.288415	0192.168.43.5	198.246.117.106	FTP	61 Request: CWD /
644	27.773769	0198.246.117.106	192.168.43.5	FTP	83 Response: 250 CWD command successful.
645	27.774296	0192.168.43.5	198.246.117.106	FTP	60 Request: PASV
646	28.099371	0198.246.117.106	192.168.43.5	FTP	107 Response: 227 Entering Passive Mode (198,246,117,106,222,58).
662	28.508213	0192.168.43.5	198.246.117.106	FTP	63 Request: LIST -l
675	29.015563	0198.246.117.106	192.168.43.5	FTP	108 Response: 125 Data connection already open; Transfer starting.
676	29.015728	0198.246.117.106	192.168.43.5	FTP	78 Response: 226 Transfer complete.
678	29.016049	0192.168.43.5	198.246.117.106	FTP	60 Request: QUIT
683	29.350525	0198.246.117.106	192.168.43.5	FTP	68 Response: 221 Goodbye.

Information

6. What is the username and password used to access FTP server?

Username:Anonymous

Password: chrome@example.com

7. Type this in filter “tcp and ip.addr==198.246.117.106” (hint: nslookup ftp.cdc.gov) and Identify the flags that are set to 1 on successful login.

PUSH and ACK flags are set to 1.

8. Identify the phrase and code of the Retrieve and Quit command response.

Phrase is RETR and QUIT

6.DHCP

1. Are DHCP messages sent over UDP orTCP?
   DHCP messages are sent over UDP (User Datagram Protocol).

2. What is the link-layer (e.g., Ethernet) address of yourhost?
   The Link Layer address of my workstation is: 00:90:4b:69:dd:34

3. What values in the DHCP discover message differentiate this message from the DHCP request message?
   The values which differentiate the Discover message from the Request message are in "Option 53: DHCP Message Type".

4. What is the value of the Transaction-ID in each of the first four (discover/Offer/Request/ACK) DHCP messages? What is the purpose of the Transaction-ID field?
   The value of the Transaction ID is 0xe6746a7d. A Transaction ID is used so that the DHCP server can differentiate between client requests during the request process

5. A host uses DHCP to obtain an IP address, among other things. But a host's IP address is not confirmed until the end of the four-message exchange! If the IP address is not set until the end of the four-message exchange, then what values are used in the IP datagrams in the four- message exchange? For each of the four DHCP messages (Discover/Offer/Request/ACK DHCP), indicate the source and destination IP addresses that are carried in the encapsulating IP datagram.
   The DHCP client and server both use 255.255.255.255 as the destination address. The client uses source IP address 0.0.0.0, while the server uses its actual IP address as the source address.

6. What is the IP address of your DHCP server?
   The IP address of the DHCP server is 192.168.243.1

7. What IP address is the DHCP server offering to your host in the DHCP Offer message? Indicate which DHCP message contains the offered DHCP address.
   The DHCP server offered the IP address 192.168.243.92 to my client machine. The DHCP message with "DHCP Message Type = DHCP Offer" contains the offered IP.

8. In the example screenshot in this assignment, there is no relay agent between the host and the DHCP server. What values in the trace indicate the absence of a relay agent? Is there a relay agent in your experiment? If so what is the IP address of the agent?

   The "Relay agent IP address" is 0.0.0.0, which indicates that there is no DHCP Relay used. There was no Relay Agent used in my experiment.

9. Explain the purpose of the lease time. How long is the lease time in your experiment?

The lease time is the amount of time the DHCP server assigns an IP address to a client. During the lease time, the DHCP server will not assign the IP given to the client to another client, unless it is released by the client. Once the lease time has expired, the IP address can be reused by the DHCP server to give to another client. In this experiment, the lease time is 8hours.

10. What is the purpose of the DHCP release message? Does the DHCP server issue an acknowledgment of receipt of the client's DHCP request?

The client sends a DHCP Release message to cancel its lease on the IP address given to it by the DHCP server. The DHCP server does not send a message back to the client acknowledging the DHCP Release message. If the DHCP Release message from the client is lost, the DHCP server would have to wait until the lease period is over for that IP address until it could reuse it for another client.

7.IP, ICMP

Run wireshark, Execute the command $ping <any website url> in the command prompt and capture the packets.

EXERCISE QUESTIONS :

1. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?

The IP address of my host is 192.168.1.101. The IP address of the destination host is 143.89.14.34.

2. Within the IP packet header, what is the value in the upper layer protocol field?

Within the header, the value in the upper layer protocol field is ICMP (0x01)

3. How many bytes are in the IP header? How many bytes are in the payload *of the IP datagram*? Explain how you determined the number of payload bytes.

There are 20 bytes in the IP header, and 56 bytes total length, this gives 36 bytes in the payload of the IP datagram

4. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

The more fragments bit = 0, so the data is not fragmented

5. Which fields in the IP datagram *always* change from one datagram to the next within this series of ICMP messages sent by yourcomputer?

Identification, Time to live and Header checksum always change.

6. Which fields stay constant?

The fields that stay constant across the IP datagrams are:

- *Version (since we are using IPv4 for all packets)*
- *header length (since these are ICMP packets)*
- *source IP (since we are sending from the same source)*
- *destination IP (since we are sending to the same dest)*
- *Differentiated Services (since all packets are ICMP they use the same Type of Service class)*
- *Upper Layer Protocol (since these are ICMP packets)*

7. Find the first ICMP Echo Request message that was sent by your computer after you changed the *Packet Size* to 2000 (Use command ping –s 2000 www.yahoo.com to change the MTU of the packet). Has that message been fragmented across more than one IP datagram.

Yes, this packet has been fragmented across more than one IP datagram

8. Write down the first fragment of the fragmented IP datagram. What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?

The Flags bit for more fragments is set, indicating that the datagram has been fragmented. Since the fragment offset is 0, we know that this is the first fragment. This first datagram has a total length of 1500, including the header.

9. What information in the IP header indicates that this is not the first datagram fragment? Are there more fragments? How can you tell?

We can tell that a fragment is not the first fragment, since the fragment offset is 1480 (not zero). We can tell if there are more fragments by looking at More Fragment flag value. If this value is 1, there are more fragments, if the value is 0 then this is the last fragment.

10. What fields change in the IP header between the first and second fragment?

The IP header fields that changed between the fragments are: total length, flags, fragment offset, and checksum.

NS3

1. **Problem statement: Three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.**

Expected learning outcome: NS3 basic simulation basics, on-off application (CBR), reading traces through flow monitor and display the network performance

Algorithm:

1. Create a simple 3 node topology using NodeContainer topology helper as no---n1---n2---n3. Use point to point links between two nodes.
   // Network topology
   //
   // 10.1.1.0 10.1.2.0
   // n0 ---------- n1----------n2
   // point-to-point point-to-point
   //
2. Install internet stack on all nodes.
3. Assign IP4 addresses to netdevice containing two nodes at a time
4. Set network address for interfaces
5. Populate the global routing table between all nodes
6. Install a UDP socket instance on Node0 as sender that will connect to Node3 as receiver.
7. Start the UDP application at time 1.0 at rate Rate1
8. Use the ns-3 tracing mechanism to record the network performance.
   // The output will consist of all the traced statistics collected at the network layer (by the flow monitor) and the application layer.
   // Finally, the number of packets dropped by the queuing discipline and
   // the number of packets dropped by the netdevice
9. vary the bandwidth of point-to-point link and observe the performance
10. Usegnuplot/matplotlib to visualise plots of bandwidth vs packet drop.
11. Conclude the performance from graph
12. Perform the above experiment for different topology conncetion.

Steps:

1. Open editor and write the program for the algorithm logic
2. Save in ns3.30/scratch directory
3. Compilation:

$./waf --run scratch/filenameWithoutExtention
*/

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/traffic-control-module.h"
#include "ns3/flow-monitor-module.h"

using namespace ns3;

int main ()
{
doublesimulationTime = 10; //seconds
std::stringsocketType="ns3::UdpSocketFactory";//"ns3::TcpSocketFactory";
```

```
NodeContainer nodes;
nodes.Create (3);

PointToPointHelper p2p;
  p2p.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));
  p2p.SetChannelAttribute ("Delay", StringValue ("2ms"));
  p2p.SetQueue ("ns3::DropTailQueue", "MaxSize", StringValue ("1p"));

NetDeviceContainer dev01;
  dev01= p2p.Install (nodes.Get(0),nodes.Get(1));
NetDeviceContainer dev12;
  dev12= p2p.Install (nodes.Get(1),nodes.Get(2));

InternetStackHelper stack;
stack.Install (nodes);

  Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer interfaces01 = address.Assign (dev01);

address.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer interfaces12 = address.Assign (dev12);

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  //Flow
uint16_t port = 7;
  Address localAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
PacketSinkHelperpsh (socketType, localAddress);
ApplicationContainersinkApp = psh.Install (nodes.Get (2));
sinkApp.Start (Seconds (0.0));
sinkApp.Stop (Seconds (simulationTime + 0.1));

OnOffHelperonoff (socketType, Ipv4Address::GetAny ());
onoff.SetAttribute ("OnTime",  StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
onoff.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute ("DataRate", StringValue ("50Mbps")); //bit/s
ApplicationContainer apps;

InetSocketAddressrmt (interfaces12.GetAddress (1), port);
AddressValueremoteAddress (rmt);
onoff.SetAttribute ("Remote", remoteAddress);
apps.Add (onoff.Install (nodes.Get (0)));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (simulationTime + 0.1));

FlowMonitorHelperflowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll();

  Simulator::Stop (Seconds (simulationTime + 5));
  Simulator::Run ();

Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
std::cout<<std::endl<< "*** Flow monitor statistics ***"  <<std::endl;
```

```
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iteratoriter = stats.begin (); iter != stats.end
(); ++iter)
    {
     Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);
std::cout<< "Flow ID: " <<iter->first <<" SrcAddr " <<t.sourceAddress<< " DstAddr "
<<t.destinationAddress<<std::endl;
std::cout<< "Tx Packets   = " <<iter->second.txPackets<<std::endl;
std::cout<< "Rx Packets   = " <<iter->second.rxPackets<<std::endl;
std::cout<< "Lost Packets = " <<iter->second.lostPackets<<std::endl;
std::cout<< "Throughput   = " <<iter->second.rxBytes * 8.0 / (iter-
>second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds()) / 1000000
<< " Kbps"<<std::endl;
    }

  Simulator::Destroy ();

return 0;
}


/*
Usage of the existing examples: (Lab1.cc--→ example/traffic-control/traffic-control.cc)

*/
```

2. **Simulate simple Extended Service Set with transmitting nodes in wireless LAN and determine the performance with respect to transmission of packets.**

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/flow-monitor-module.h"

// Default Network Topology
//
// Number of wifi or csma nodes can be increased up to 250
//                          |
//               Rank 0   |   Rank 1
// -------------------------|----------------------------
//   Wifi 10.1.3.0
//                AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                   point-to-point  |    |    |    |
//                                   ================
//                                     LAN 10.1.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

int
main (intargc, char *argv[])
{
uint32_tnCsma = 3;
uint32_tnWifi = 3;
doublesimulationTime = 10; //seconds
std::stringsocketType = "ns3::UdpSocketFactory";

CommandLinecmd;
cmd.Parse (argc,argv);

  // Check for valid number of csma or wifi nodes
  // 250 should be enough, otherwise IP addresses
  // soon become an issue
if (nWifi> 250 || nCsma> 250)
    {
std::cout<< "Too many wifi or csma nodes, no more than 250 each." <<std::endl;
return 1;
    }

NodeContainer p2pNodes;
  p2pNodes.Create (2);

PointToPointHelperpointToPoint;
```

```
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;
  p2pDevices = pointToPoint.Install (p2pNodes);

NodeContainercsmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);

CsmaHelpercsma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainercsmaDevices;
csmaDevices = csma.Install (csmaNodes);

NodeContainerwifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainerwifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelperphy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

WifiHelperwifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssidssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac", "Ssid", SsidValue (ssid), "ActiveProbing", BooleanValue (false));

NetDeviceContainerstaDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac","Ssid", SsidValue (ssid));

NetDeviceContainerapDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
                               "GridWidth", UintegerValue (3),
                               "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);
```

```
InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

  Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces;
  p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

/*  UdpEchoServerHelperechoServer (9);

ApplicationContainerserverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelperechoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainerclientApps =
echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
*/

 //Flow
uint16_t port = 7;
  Address localAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
PacketSinkHelperpacketSinkHelper (socketType, localAddress);
ApplicationContainersinkApp = packetSinkHelper.Install (csmaNodes.Get (nCsma));
sinkApp.Start (Seconds (0.0));
sinkApp.Stop (Seconds (simulationTime + 0.1));
uint32_tpayloadSize = 1448;
Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (payloadSize));
OnOffHelperonoff (socketType, Ipv4Address::GetAny ());
onoff.SetAttribute ("OnTime",  StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
onoff.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute ("PacketSize", UintegerValue (payloadSize));
onoff.SetAttribute ("DataRate", StringValue ("50Mbps")); //bit/s
ApplicationContainer apps;
AddressValueremoteAddress (InetSocketAddress (csmaInterfaces.GetAddress (nCsma), port));
onoff.SetAttribute ("Remote", remoteAddress);
apps.Add (onoff.Install (wifiStaNodes.Get (nWifi - 1)));
apps.Start (Seconds (1.0));
```

```
apps.Stop (Seconds (simulationTime + 0.1));
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
  Simulator::Stop (Seconds (10.0));
FlowMonitorHelperflowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll();

  Simulator::Run ();
// Print per flow statistics
monitor->CheckForLostPackets ();
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

for (std::map<FlowId, FlowMonitor::FlowStats>::const_iteratoriter = stats.begin (); iter != stats.end
(); ++iter)
    {
  Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);
      NS_LOG_UNCOND("Flow ID: " <<iter->first << " SrcAddr " <<t.sourceAddress<< " DstAddr "
<<t.destinationAddress);
      NS_LOG_UNCOND("Tx Packets = " <<iter->second.txPackets);
std::cout<< "Rx Packets   = " <<iter->second.rxPackets<<std::endl;
std::cout<< "Lost Packets = " <<iter->second.lostPackets<<std::endl;
std::cout<< "Throughput   = " <<iter->second.rxBytes * 8.0 / (iter-
>second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds()) / 1000000
<< " Kbps"<<std::endl;
    }
  Simulator::Destroy ();
return 0;
}
/*
Usage of the existing examples: (Lab3.cc--→ example/third.cc)
*/
```

3. **Simulate a transmission of ping message over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.**

```
// Network topology
//
//       n0    n1   n2   n3     n4      n5
//       |     |    |    |      |       |
//     ================================
//
//  node n0,n1,n3,n4,n5 pings to node n2
//  node n0 generates protocol 2 (IGMP) to node n3

#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-apps-module.h"
#include "ns3/internet-module.h"
#include "ns3/flow-monitor-module.h"
//#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("CsmaPingExample");


int
main (intargc, char *argv[])
{

CommandLinecmd;
cmd.Parse (argc, argv);
  Time interPacketInterval = Seconds (1.);
  // Here, we will explicitly create four nodes.
  NS_LOG_UNCOND ("Create nodes.");
NodeContainer c;
c.Create (6);

  // connect all our nodes to a shared channel.
  NS_LOG_UNCOND ("Build Topology.");
CsmaHelpercsma;
csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));
csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
csma.SetDeviceAttribute ("EncapsulationMode", StringValue ("Llc"));
NetDeviceContainerdevs = csma.Install (c);

  // add an ip stack to all nodes.
  NS_LOG_UNCOND ("Add ip stack.");
InternetStackHelperipStack;
ipStack.Install (c);
```

```
  // assign ip addresses
  NS_LOG_UNCOND ("Assign ip addresses.");
  Ipv4AddressHelper ip;
ip.SetBase ("192.168.1.0", "255.255.255.0");
  Ipv4InterfaceContainer addresses = ip.Assign (devs);

  NS_LOG_UNCOND ("Create Source");

InetSocketAddressdst = InetSocketAddress (addresses.GetAddress (3));
OnOffHelperonoff = OnOffHelper ("ns3::UdpSocketFactory", dst);
onoff.SetAttribute ("OnTime",  StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
onoff.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute ("PacketSize", UintegerValue (1100));
onoff.SetAttribute ("DataRate", StringValue ("50Mbps"));

ApplicationContainer apps = onoff.Install (c.Get (0));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));

  NS_LOG_UNCOND ("Create Sink.");
PacketSinkHelper sink = PacketSinkHelper ("ns3::UdpSocketFactory", dst);
apps = sink.Install (c.Get (3));
apps.Start (Seconds (0.0));
apps.Stop (Seconds (11.0));

  NS_LOG_UNCOND ("Create pinger");
  V4PingHelper ping = V4PingHelper (addresses.GetAddress (0));
// ping.SetAttribute ("Interval", TimeValue (interPacketInterval));
ping.SetAttribute ("Interval", TimeValue (interPacketInterval));
NodeContainerpingers;
pingers.Add (c.Get (3));
pingers.Add (c.Get (1));
pingers.Add (c.Get (2));
pingers.Add (c.Get (4));
pingers.Add (c.Get (5));
apps = ping.Install (pingers);
apps.Start (Seconds (2.0));
apps.Stop (Seconds (10.0));


 //Enable Tracing using flowmonitor
FlowMonitorHelperflowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll();

  Simulator::Stop (Seconds (10.0));

//Add visualization using Netanim
//  AnimationInterfaceanim ("ex5.xml");

  NS_LOG_UNCOND ("Run Simulation.");
  Simulator::Run ();

// Print per flow statistics
monitor->CheckForLostPackets ();
```

```
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

for (std::map<FlowId, FlowMonitor::FlowStats>::const_iteratoriter = stats.begin (); iter != stats.end
(); ++iter)
    {
  Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);
       NS_LOG_UNCOND("Flow ID: " <<iter->first << " SrcAddr " <<t.sourceAddress<< " DstAddr "
<<t.destinationAddress);
      NS_LOG_UNCOND("Tx Packets = " <<iter->second.txPackets);
      NS_LOG_UNCOND("Rx Packets = " <<iter->second.rxPackets);
std::cout<< "Lost Packets = " <<iter->second.lostPackets<<std::endl;
      NS_LOG_UNCOND("Throughput: " <<iter->second.rxBytes * 8.0 / (iter-
>second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds()) / 1024  << "
Kbps");
    }

  Simulator::Destroy ();
  NS_LOG_UNCOND ("Done.");
}


/*
Usage of the existing examples: (Lab5.cc--→ src/csma/examples/csma-ping.cc)
*/
```

Socket Program

4.Write a client-server program using TCP/IP sockets in which client requests for a file by sending the file name to the server, and the server sends back the contents of the requested file if present.

/*SERVER - Create a file called hello.txt in the current directory and pass that as the file name for server */

```
#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60

int i, j, tem;
char buff[4096], t;
FILE *f1;

int main(int afg, char *argv)
{
int sockfd, newsockfd, clength;
struct sockaddr_in serv_addr,cli_addr;
char t[MAX], str[MAX];
strcpy(t,"exit");
sockfd=socket(AF_INET, SOCK_STREAM,0);
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=INADDR_ANY;
serv_addr.sin_port=htons(SERV_TCP_PORT);
printf("\nBinded");
bind(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
printf("\nListening...");
listen(sockfd, 5);
clength=sizeof(cli_addr);
newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&clength);
close(sockfd);
read(newsockfd, &str, MAX);
printf("\nClient message\n File Name : %s\n", str);
f1=fopen(str, "r");
while(fgets(buff, 4096, f1)!=NULL) {
write(newsockfd, buff,MAX);
printf("\n"); }
fclose(f1);
printf("\nFile Transferred\n");
return 0;
}
```

//CLIENT

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60

int main(int arg,char*argv[])
{
int sockfd,n;
struct sockaddr_in serv_addr;
struct hostent*server;
char send[MAX],recvline[MAX],s[MAX],name[MAX];
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
serv_addr.sin_port=htons(SERV_TCP_PORT);
connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
printf("\nEnter the source file name : \n");
scanf("%s",send);
write(sockfd,send,MAX);
while((n=read(sockfd,recvline,MAX))!=0) {
printf("%s",recvline);
}
close(sockfd);
return 0;
}
```

4. Leacky Bucket Code

```
#include<stdio.h>
int main(){
    int incoming, outgoing, buck_capacity, n, store = 0;
    printf("Enter bucket capacity, outgoing rate and no of inputs: ");
    scanf("%d %d %d", &buck_capacity, &outgoing, &n);

    while (n != 0) { //loop over total number of inputs
        printf("Enter the number of incoming packets: ");
        scanf("%d", &incoming);
        printf("Incoming packet size %d\n", incoming);
  if((incoming-outgoing) <= (buck_capacity-store)) //it is possible to send without dropping
{
int sent = outgoing>=incoming?incoming:outgoing; //if incoming is more than outgoing, total sent
will be outgoing rest buff
if(sent < outgoing && store != 0)  //if incoming<outgoing, we can add values from the store to be
sent till op cap is reached
{
int remaining = outgoing-sent;
while(remaining > 0 && store != 0) //keeps adding one to sent until we run out of op cap or
nothing left in buff
{
remaining -= 1;
store -= 1;
sent += 1;
}

}
printf("%d packets sent out\n", sent);
if(outgoing<incoming)
{
store = store + (incoming - outgoing);
}
}
else //packets need to be dropped
{
int dropped = (incoming-outgoing)-(buck_capacity-store); //excess packets are the ones that after
sending cant be accomodated in buff
printf("%d packets dropped\n", dropped);
store = buck_capacity;
}
printf("%d out of %d space used in the buffer\n", store, buck_capacity);
        n--;
    }
}
```