Laboratory Manual for

EC-702 –Data & Computer Communications

B. Tech.

SEM. VII (EC)

Department of Electronics & Communication
Faculty of Technology
Dharmsinh Desai University
Nadiad

TABLE OF CONTENTS

PART – 1 LABORATORY MANUAL

Sr No.	Title	Page No.
1.	Study of UART and RS232-C Standard	1
2.	Programming Serial Port for Byte Transfer	7
3.	Programming Serial Port File Transfer	8
4.	Serial Port File Transfer with Character stuffing and de-stuffing Protocol	10
5.	Serial Port File Transfer with Stop & Wait Protocol	13
6.	IP Addressing and Network Planning	17
7.	Study of Network Commands	25
8.	Understanding Packet capturing using Wireshark	30
9.	Study of TCP and IP Header using Wireshark	33
10.	Understanding execution of PING Commands using Wireshark	35
11.	Understanding execution of TRACERT Command using Wireshark	38
12.	Performance analysis of TCP Protocol using ns-2	40

PART – 2 DATASHEET

PART – 3 SESSIONAL QUESTION PAPERS

P A R T I

LAB MANUAL

EXPERIMENT – 1

STUDY OF UART AND RS232-C STANDARD

OBJECTIVES:

(i) To understand RS232-C standard and UART functions.
(ii) To study RS232-C cabling scheme.
(iii) To test RS232-C cable using Windows inbuilt utility.

In most cases, any device you connect to the serial port will need the serial transmission converted back to parallel so that it can be used. This can be done using a UART. On software side, there are many more registers that you have to attend to than on a Standard Parallel Port. (SPP)

Advantages of Serial Data Transfer Over Parallel Data Transfer

- Serial Cables can be longer than Parallel cables. The serial port transmits a '1' as -3 to -25 volts and a '0' as +3 to +25 volts where as a parallel port transmits a '0' as 0v and a '1' as 5v. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not as much of a problem for serial cables then they are for parallel.
- You don't need as many wires then parallel transmission. If your device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be cheaper that running 19 or 25 core cable. However you must take into account the cost of the interfacing at each end.
- Many electronic diaries and palmtop computers have infra red capabilities build in. Serial transmission is used where one bit is sent at a time. IrDA-1 (The first infra red specifications) was capable of 115.2k baud and was interfaced into a UART. The pulse length however was cut down to 3/16th of an RS 232 bit length to conserve power considering these devices are mainly used on diaries, laptops and palmtops.
- Serial Communication reduces the pin count of Microcontrollers. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins if you use an 8 bit Parallel method (You may also require a Strobe).

Hardware Properties

Devices which use serial cables for their communication are split into two categories. These are DCE (Data Communications Equipment) and DTE (Data Terminal Equipment.) Data Communications Equipments are devices such as your modem, TA adapter, plotter etc while Data Terminal Equipment is your Computer or Terminal
The electrical specifications of the serial port are contained in the RS232C standard. It states many parameters such as

Table1.1 RS232 standard parameters

1.	A "Space" (logic 0) will be between +3 and +25 Volts.
2.	A "Mark" (Logic 1) will be between -3 and -25 Volts
3.	The region between +3 and -3 volts is undefined.
4.	An open circuit voltage should never exceed 25 volts. (In Reference to GND)
5.	A short circuit current should not exceed 500mA. The driver should be able to handle this without damage. (Take note of this one!)

The RS 232 C standard specifies a maximum baud rate of 20,000 BPS. Serial Ports come in two "sizes". There are the D-Type 25 pin connector and the D-Type 9 pin connector both of which are male on the back of the PC, thus you will require a female connector on your device. Below is a table of pin connections for the 9 pin and 25 pin D-Type connectors.

Serial Pin outs (D25 and D9 Connectors)

Table1.1 D9 & D25 Pin Connector Pin Configuration

D-Type-25 Pin No.	D-Type-9 Pin No.	Abbreviation	Full Name
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

Pin Functions

Table1.3 Pin Function of D9 Pin Connector

Abbreviation	Full Name	Function
TD	Transmit Data	Serial Data Output (TXD)
RD	Receive Data	Serial Data Input (RXD)
CTS	Clear to Send	This line indicates that the Modem is ready to exchange data.
DCD	Data Carrier Detect	When the modem detects a "Carrier" from the modem at the other end of the phone line, this Line becomes active.
DSR	Data Set Ready	This tells the UART that the modem is ready to establish a

		link.
DTR	Data Terminal Ready	This is the opposite of DSR. This tells the Modem that the UART is ready to link.
RTS	Request To Send	This line informs the Modem that the UART is ready to exchange data.
RI	Ring Indicator	Goes active when modem detects a ringing signal from the PSTN.

Null Modems

A Null Modem is used to connect two DTE's together without using intermediate DCEs..

9D to 25D Conversion

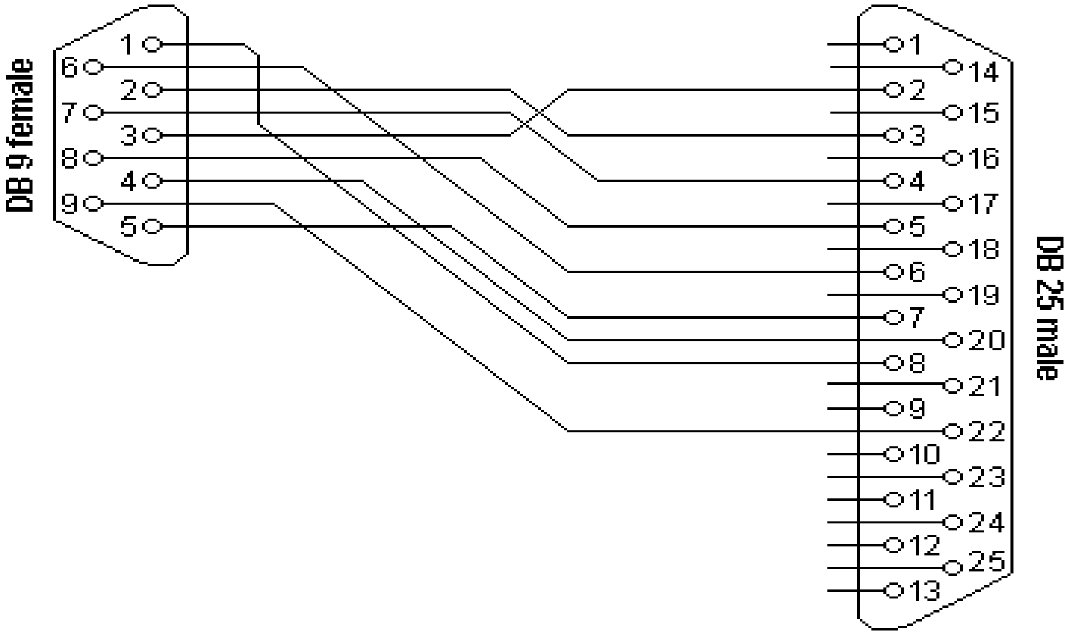


Fig.1.1 Null Modem Configuration

The UART (8250 and Compatibles)

UART stands for Universal Asynchronous Receiver / Transmitter. UART 8250 is the device that controls the serial port. Most cards will have the UART's integrated into other chips which may also control your parallel port, games port, floppy or hard disk drives and are typically surface mount devices. The 8250 series, which includes the 16450, 16550, 16650, & 16750 UARTS are the most commonly found type in your PC.

All the UARTs pins are TTL compatible. That includes TD, RD, RI, DCD, DSR, CTS, DTR and RTS which all interface into your serial plug, typically a D-type connector. Therefore RS 232 Level Converters are used.

The UART requires a Clock to run. If you look at your serial card a common crystal found is either a 1.8432 MHZ or an 18.432 MHZ Crystal. This clock will be used for the Programmable Baud Rate Generator which directly interfaces into the transmit timing circuits but not directly into the receiver timing circuits. For this an external connection mast be made from pin 15 (Baud Out) to pin 9 (Receiver clock in.) Note that the clock signal will be at Baud rate * 16.

Registers

RBR, THR, IER, IIR, FCR, LCR, MCR, LSR, MSR, SCR, DLL, DLM
The communication between the processor and the UART is completely controlled by twelve registers. These registers can be read or written to check and change the behavior of the communication device. Each register is eight bits wide. On a PC compatible, the registers are accessible in the I/O port map.

RBR: Receiver buffer register (RO)

The receiver buffer register contains the byte received if no FIFO is used, or the oldest unread byte with Fife's. If FIFO buffering is used, each new read action of the register will yield the next byte, until no more bytes are present. Bit 0 in the line status register can be used to check if all received bytes have been read. This bit will change to zero if no more bytes are present.

THR: Transmitter holding register (WO)

The transmitter holding register is used to buffer outgoing characters. If no FIFO buffering is used, only one character can be stored. Otherwise the amount of characters depends on the type of UART. Bit 5 in the line status register can be used to check if new information must be written to the transmitter holding register. The value 1 indicates that the register is empty. If FIFO buffering is used, more than one character can be written to the transmitter holding register when the bit signals an empty state. There is no indication of the amount of bytes currently present in the transmitter FIFO.

The transmitter holding register is not used to transfer the data directly. The byte is first transferred to a shift register where the information is broken in single bits which are sent one by one.

LCR: Line control registers (R/W)

The line control register is used at initialization to set the communication parameters. Parity and number of data bits can be changed for example. The register also controls the accessibility of the DLL and DLM registers. Because they are only accessed at initialization when no communication occurs this register swapping has no influence on performance.

Table1.4 LCR: Line Control Registers

Bit	Value			Detail
0,1	**Bit 1**	**Bit 0**		Data Word Length
	0	0		5 Bits
	0	1		6 Bits
	1	0		7 Bits
	1	1		8 Bits
2	0			1 stop bit
	1			1.5 stop bits(5 bits word) 2 stop bits (6,7 or 8 bit word)
3,4,5	**Bit 5**	**Bit 4**	**Bit 3**	
	X	X	0	No parity
	0	0	1	Odd Parity
	0	1	1	Even Parity
	1	0	1	High Parity(stick)
	1	1	1	Low Parity(stick)
6	0			Break signal disabled
	1			Break signal enabled
7	0			DLAB : RBR, THR and IER accessible
	1			DLAB : DLL and DLM accessible

Common settings are:
8 data bits, one stop bit, no parity
7 data bits, one stop bit, even parity

LSR : Line status register (RO)

The line status register shows the current state of communication. Errors are reflected in this register. The state of the receiver and transmitter buffers is also available.

Table1.5 LSR : Line status register

Bit	Comment
0	Data available
1	Overrun error
2	Parity error
3	Framing error
4	Break signal received

5	THR is empty
6	THR is empty, and line is idle
7	Erroneous data in FIFO

Bit 5 and 6 both show the state of the transmitting cycle. The difference is, that bit 5 turns high as soon as the transmitter holding register is empty whereas bit 6 indicates that also the shift register which outputs the bits on the line is empty

DLL and DLM : Divisor latch registers (R/W)

The frequency (1.8432 MHz) is divided by 16 to generate the time base for communication. Because of this division, the maximum allowed communication speed is 115200 bps. Modern UARTS like the 16550 are capable of handling higher input frequencies up to 24 MHz which makes it possible to communicate with a maximum speed of 1.5 Mbps. This 115200 bps communication speed is not suitable for all applications. To change the communication speed, the frequency can be further decreased by dividing it by a programmable value. For very slow communications, this value can go beyond 255. Therefore, the divisor is stored in two separate bytes, the DLL and DLM which contain the least, and most significant byte.

It is necessary that both the transmitting and receiving UART use the same time base. Default values have been defined which are commonly used. The table shows the most common values with the appropriate settings of the divisor latch bytes.

Table1.6 DLL and DLM : Divisor latch registers

Speed(bps)	**Divisor**	**DLL**	**DLM**
50	2304	0x00	0x09
300	384	0x80	0x01
1200	96	0x60	0x00
2400	48	0x30	0x00
4800	24	0x18	0x00
9600	12	0x0C	0x00
19200	6	0x06	0x00
38400	3	0x03	0x00
57600	2	0x02	0x00
115200	1	0x01	0x00

CONLUSION:

EXPERIMENT – 2

PROGRAMMING SERIAL PORT FOR BYTE TRANSFER

OBJECTIVE:

(i) To verify serial port communication between two PCs by writing simple C code.(Byte transfer)

SAMPLE PROGRAM (TRANSMITTER) :

```
#include<unistd.h>
#include <stdio.h>
#include <sys/io.h>
int main()
{
        char data='B';
        ioperm(0x3f8,7,1);
        outb(0x83,0x3fb);
        outb(0x0c,0x3f8);
        outb(0x00,0x3f9);
        outb(0x03,0x3fb);
        while(((inb(0x3fd))&0x60)!=0x60);
        outb(data,0x3f8);
        printf("data transmitted=%c",data);
}
```

SAMPLE PROGRAM (RECEIVER) :

```
#include<unistd.h>
#include <stdio.h>
#include <sys/io.h>
int main( ){
        char data;
        ioperm(0x3f8,7,1);
        outb(0x83,0x3fb);
        outb(0x0c,0x3f8);
        outb(0x00,0x3f9);
        outb(0x03,0x3fb);
        while(((inb(0x3fd)) & 0x01)!=0x01);
        data=inb(0x3f8);
        printf("data received=%c",data);
}
```

CONLUSION:

EXPERIMENT – 3

PROGRAMMING SERIAL PORT FILE TRANSFER

OBJECTIVES:

(i) Implement simple file transfer utility

(ii) Implementation of file transfer with Framing techniques.

SAMPLE PROGRAM (TRANSMITTER):

```
#include<unistd.h>
#include<stdio.h>
#include<sys/io.h>
int main()
{
        FILE *f1;
        char c;
        ioperm(0x3f8,7,1);
        outb(0x83,0x3fb);
        outb(0x0c,0x3f8);
        outb(0x00,0x3f9);
        outb(0x03,0x3fb);
        f1=fopen("tx.txt","r");
        do
        {
                while((inb(0x3fd) & 0x60)!=0x60);
                c=getc(f1);
                outb(c,0x3f8);
                printf("%c",c);
        }while(c!=EOF);
        fclose(f1);
        printf("\nthe file transmitted\n");
}
```

SAMPLE PROGRAM (RECEIVER) :

```
#include<stdio.h>
#include<sys/io.h>
#include<unistd.h>
int main()
{
        FILE *f1;
        char ch;
        ioperm(0x3f8,7,1);
        outb(0x83,0x3fb);
        outb(0x0c,0x3f8);
        outb(0x00,0x3f9);
        outb(0x03,0x3fb);
        f1=fopen("Rx.txt","w");
        do
        {
                while(((inb(0x3fd)) & 0x01)!=0x01);
                ch=inb(0x3f8);
                putc(ch,f1);
                printf("%c",ch);
        }while(ch!=EOF);
        fclose(f1);
        printf("\nTHE FILE RECEIVED\n");
}
```

MODIFICATION :

(1) Modify the above program to use faming concept**.**

CONLUSION:

EXPERIMENT - 4

SERIAL PORT FILE TRANSFER WITH CHARACTER STUFFING AND DE-STUFFING PROTOCOL

OBJECTIVE:

(i) Implement character stuffing & de-stuffing concept

SAMPLE PROGRAM (TRANSMITTER):

```
#include<unistd.h>
#include<stdio.h>
#include<sys/io.h>
#define DLE  0x10
#define STX  0x02
#define ETX  0x03
int main()
{
        FILE *f1;
        char a[20];
        int i=2,flag=0,j;
        ioperm(0x3f8,7,1);
        outb(0x83,0x3fb);
        outb(0x0c,0x3f8);
        outb(0x00,0x3f9);
        outb(0x03,0x3fb);
        f1=fopen("Tx.txt","r");

        do
        {
         a[0]=DLE;
         a[1]=STX;
         i=2;
         j=0;
         while((j<8) && (flag==0))
         {
                a[i]=getc(f1);
                if (a[i]==EOF)
                   flag=1;
                if (a[i]==DLE)
                {
                   i++;
                   a[i]=DLE;
                }
          i++;
          j++;
          }
         a[i]=DLE;
         i++;
         a[i]=ETX;
         j=0;

        do
        {
                while((inb(0x3fd) & 0x60)!=0x60);
                outb(a[j],0x3f8);
                printf("%c",a[j]);
                j++;
        }while(j<(i+1));
        }while(flag!=1);
        printf("\nthe file transmitted\n\n");
}
```

SAMPLE PROGRAM (RECEIVER):

```
#include<unistd.h>
#include<stdio.h>
#include<sys/io.h>
#define DLE  0x10
#define STX  0x02
#define ETX  0x03
int main()
{
        FILE *f1;
        char a[20];
        int i,j,flag=0;
        ioperm(0x3f8,7,1);
        outb(0x83,0x3fb);
        outb(0x0c,0x3f8);
        outb(0x00,0x3f9);
        outb(0x03,0x3fb);
        f1=fopen("hp6.txt","w");
        do
        {  i=0;
           do
            {
                    while(((inb(0x3fd)) & 0x01)!=0x01);
```

```
                        a[i]=inb(0x3f8);
                        i++;
                }while(a[i-1]!=ETX);
                for(j=0;j<i;j++)
                {
                   printf("%c",a[j]);
                }
                 j=0;
                 while(j<i)
                 {
                   if(a[j]==EOF)
                   flag=1;

                    if((a[j]!=DLE) && (a[j]!=STX) && (a[j]!=ETX))
                      { putc(a[j],f1);
                            j++;
                      }
                    else if((a[j]==DLE) && (a[j+1]==DLE))
                      {putc(a[j],f1);
                       j+=2;
                      }
                    else
                      j++;
                 }
            }while(flag==0);
            fclose(f1);
            printf("\nTHE FILE RECEIVED\n");
            }
}
```

CONLUSION:

EXPERIMENT – 5

SERIAL PORT FILE TRANSFER WITH STOP & WAIT PROTOCOL

OBJECTIVES:

(i) To implement simple stop & wait protocol
(ii) To understand the role of timer in the protocol

SAMPLE PROGRAM (TRANSMITTER):

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
void baud( )                        /* Baud rate setting */
{
                outb(0x83,0x3fb);
                outb(0x0c,0x3f8);
                outb(0x00,0x3f9);
                outb(0x03,0x3fb);
}
void frm_nw_layer( )                /* Go get something to send */
{
         int i;
         fp=fopen("number.txt",'r');
         for (i=0;i<8;i++)
         {
          data[i]=fgetc(fp);
          if (data[i]= =EOF)
          {
           flag=1;
           break;
         }
         }
         fclose(fp);
}
void frame_generate( )              /* To generate a frame */
{
         frame[0]=seq;                      /* Put Seq. No. into frame */
         for(i=0;i<8;i++)
         {
          frame[i+1]=data[i];       /* Put data into frame */
         }
```

```
            frame[i+1]='0';                         /* Put ‘0’ as a checksum into frame */
            if(seq=='1')                                      /* Alter the seq. */
             seq='0';
            else
             seq='1';
}
void to_phy_layer( )                               /* Transmit the frame */
{
           int i;
           status=0;
           do
           {
            for(i=0;i<10;i++)
            {
                      while((inb(0x3fd)&(0x60))!=0x60);
                      outportb(0x3f8,frame[i]);
            }
            delay(5000);                                      /* Wait for given delay */
            while((inb(0x3fd)&(0x01))!=0x10);
                       ack=inb(0x3f8);
                      if(ack= =seq)
                      {
                       status=1;
                       break;
                      }
          }
           while(!status= =1 && ack= =seq);
 }
          char data[8],frame[10];
          char seq='0',ack;
          FILE *fp;
          int status,flag=0,i;

void main( )
{
           baud( );                                 /* Baud rate Setting */
           frm_nw_layer( );                         /* Go get something to send */
           frame_generate( );                       /* Get data from network layer and bound
                                                                   it into frame */
           to_phy_layer( );                         /* Transmit frame through any physical
                                                                             medium */
}
```

SAMPLE PROGRAM (RECEIVER):

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
int flag=0,R=0,i;
char ack,frame[10],data[8];
FILE *fp;
void baud( )                                //Baud rate setting
{
        outb(0x83,0x3fb);
        outb(0x0c,0x3f8);
        outb(0x00,0x3f9);
        outb(0x03,0x3fb);
}
void fr_phy_layer()                         //To get the frame from physical layer.
{
        int DF;
        do
        {
                DF=0;                        //Set the Duplicate flag as 0 initially.
                for(i=1;i<10;i++)
                {
                        if ((inb(0x3fd)&(0x01))==0x01)
                        frame[i]=inb(0x3f8);
                }
                if(frame[9]=='0')
                {
                        if(frame[0]=='0')
                        ack='1';
                        if(frame[0]=='1')
                        ack='0';
                        if ((inb(0x3fd)&(0x60))==0x60)
                        outb(ack,0x3f8);
}
                if(frame[0]!=R)
{
                DF=1;
                }
        }while(DF!=0);

        if(R==0)
        R=1;
        else
```

```
            R=0;
            for(i=0;i<8;i++)
            {
                        data[i]=frame[i+1];
            }
}
void to_nw_lyr()                              //Pass the data to Network Layer.
{
            for(i=0;i<8;i++)
            {
                  if(data[i]!=EOF)
                  {
                  fputc(data[i],fp);
                  }
                  else
                  {
                   flag=1;
                   break;
                  }
            }
}
void main()
{
            fp=fopen("Rx.txt","w");
            do
            {
                  fr_phy_layer();                   //Call the function from physical layer.
                  to_nw_lyr();                      //Call the function to network layer.
            }while(flag!=1);                  //Continue untill the flag bit become 1.
            fclose(fp);                             //Close the file.
}
```

MODIFICATIONS:

(1) Demonstrate the retransmission of data at sender side using wrong Acknowledgement
(2) Include the timer at sender (After the timeout retransmission will take place)

CONLUSION:

EXPERIMENT – 6

IP ADDRESSING AND NETWORK PLANNING

OBJECTIVES:

(a) (i) To implement a program for IP class identification, IP address conversion (BCD to Binary)

(iii) To implement a program for finding Network ID (with or without subnet mask) and number of hosts.

SAMPLE PROGRAM (BINARY TO DOTTED DECIMAL FORM):

```
#include<stdio.h>
int main(){
        int i=0;
        long bin[4];
        int dec[4];
        int convert(long);
        printf("Enter IP address in binary notation ( separate each byte by space :\n ");
        for(i=0;i<4;i++)
        {
                scanf("%ld",&bin[i]);
                dec[i]=convert(bin[i]);
        }
        printf("IP address in dotted decimal form :\n");
        printf("%d.%d.%d.%d",dec[0],dec[1],dec[2],dec[3]);
}
int convert(long data)
{
        int power2(int , int);
        int sum = 0,i;
        for(i=0;data>0;i++)
        {
                sum+=(data%10)*(power2(2,i));
                data = data/10;
        }
        return sum;
}
int power2(int m , int n)
{
        int ans = 1,i;
        for(i=1;i<=n;i++)
                ans *= m;
        return ans;
}
```

SAMPLE PROGRAM (DOTTED DECIMAL TO BINARY FORM):

```
#include<stdio.h>
int main()
{
        int i=0;
        long bin[4];
        int dec[4];
        long convert(int);
        clrscr();
        printf("Enter IP address in dotted decimal notation ( separate each byte by . :\n ");
        scanf("%d.%d.%d.%d",&dec[0],&dec[1],&dec[2],&dec[3]);
        for(i=0;i<4;i++)
                bin[i]=convert(dec[i]);
        printf("IP address in pure binary notation :\n");
        printf("%08ld %08ld %08ld %08ld",bin[0],bin[1],bin[2],bin[3]);
}
long convert(int data)
{
        long power2(long , long);
        long sum = 0,i;
        for(i=0;data>0;i++)
        {
                sum+=(data%2)*(power2(10,i));
                data = data/2;
        }
        return sum;
}
long power2(long m , long n)
{
        long ans = 1,i;
        for(i=1;i<=n;i++)
                ans *= m;
        return ans;
}
```

MODIFICATIONS:

(1) Modify the above programs to find network id and host id from entered IP address (either in binary or dotted decimal form)
(2) Modify the above programs to find class as well as subnet mask of entered IP address.

CONLUSION:

(b) (i) To study network devices and the specifications of each for existing network plan
(ii) To study existing network plan
(iii) To draw a new network plan as per the new constraints

Hub

A network hub or repeater hub is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and thus making them act as a single network segment. Hubs work at the physical layer (layer 1) of the OSI model. The device is thus a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

Hubs also often come with a BNC and/or AUI connector to allow connection to legacy 10BASE2 or 10BASE5 network segments. The availability of low-priced network switches has largely rendered hubs obsolete but they are still seen in older installations and more specialized applications.

A network hub is a fairly unsophisticated broadcast device. Hubs do not manage any of the traffic that comes through them, and any packet entering any port is broadcast out on all other ports. Since every packet is being sent out through all other ports, packet collisions result—which greatly impedes the smooth flow of traffic.

Types of Hubs

1. Passive (A hub which does not need an external power source, because it does not regenerate the signal and therefore falls as part of the cable, with respect to maximum cable lengths)
2. Active (A hub which regenerates the signal and therefore needs an external power supply)
3. Intelligent (A hub which provides error detection (e.g. excessive collisions) and also does what an active hub does)

Passive hubs do not amplify the electrical signal of incoming packets before broadcasting them out to the network. Active hubs, on the other hand, do perform this amplification, as does a different type of dedicated network device called a repeater. Another, not so common, name for the term concentrator is referring to a passive hub and the term multiport repeater is referred to an active hub.

Intelligent hubs add extra features to an active hub that are of particular importance to businesses. An intelligent hub typically is stackable (built in such a way that multiple units can be placed one on top of the other to conserve space). It also typically includes remote management capabilities via Simple Network Management Protocol (SNMP) and virtual LAN (VLAN) support.

Uses

For inserting a protocol analyzer into a network connection, a hub is an alternative to a network tap or port mirroring.

Some computer clusters require each member computer to receive all of the traffic going to the cluster A hub will do this naturally; using a switch requires special configuration.

When a switch is accessible for end users to make connections, for example, in a conference room, an inexperienced or careless user (or saboteur) can bring down the network by connecting two ports together, causing a loop. This can be prevented by using a hub, where a loop will break other users on the hub, but not the rest of the network. (It can also be prevented by buying switches that can detect and deal with loops, for example by implementing the Spanning Tree Protocol.)

A hub with a 10BASE2 port can be used to connect devices that only support 10BASE2 to a modern network. The same goes for linking in an old thicknet network segment using an AUI port on a hub (individual devices that were intended for thicknet can be linked to modern Ethernet by using an AUI-10BASE-T transceiver).

Switch

In networks, a device that filters and forwards packets between LAN segments. Switches operate at the data link layer (layer 2) and sometimes the network layer (layer 3) of the OSI Reference Model and therefore support any packet protocol. LANs that use switches to join segments are called *switched LANs* or, in the case of Ethernet networks, *switched Ethernet LANs.*

Layer-1 Hubs Versus Higher-Layer Switches

A network hub, or repeater, is a fairly unsophisticated network device. Hubs do not manage any of the traffic that comes through them. Any packet entering a port is broadcast out or "repeated" on every other port, except for the port of entry. Since every packet is repeated on every other port, packet collisions result, which slows down the network.

There are specialized applications where a hub can be useful, such as copying traffic to multiple network sensors. High end switches have a feature which does the same thing called port mirroring. There is no longer any significant price difference between a hub and a low-end switch.

Layer 2

A network bridge, operating at the Media Access Control (MAC) sublayer of the data link layer, may interconnect a small number of devices in a home or office. This is a trivial case of bridging, in which the bridge learns the MAC address of each connected device. Single bridges also can provide extremely high performance in specialized applications such as storage area networks.

Once a bridge learns the topology through a spanning tree protocol, it forwards data link layer frames using a layer 2 forwarding method. There are four forwarding methods a bridge can use, of which the second through fourth method were performance-increasing methods when used on "switch" products with the same input and output port speeds:

1. Store and forward: The switch buffers and, typically, performs a checksum on each frame before forwarding it on.
2. Cut through: The switch reads only up to the frame's hardware address before starting to forward it. There is no error checking with this method.
3. Fragment free: A method that attempts to retain the benefits of both "store and forward" and "cut through". Fragment free checks the first 64 bytes of the frame, where addressing information is stored. According to Ethernet specifications, collisions should be detected during the first 64 bytes of the frame, so frames that are in error because of a collision will not be forwarded. This way the frame will always reach its intended destination. Error checking of the actual data in the packet is left for the end device in Layer 3 or Layer 4 (OSI), typically a router.
4. Adaptive switching: A method of automatically switching between the other three modes.

Cut-through switches have to fall back to store and forward if the outgoing port is busy at the time the packet arrives. While there are specialized applications, such as storage area networks, where the input and output interfaces are the same speed, this is rarely the case in general LAN applications. In LANs, a switch used for end user access typically concentrates lower speed (e.g., 10/100 Mbit/s) into a higher speed (at least 1 Gbit/s). Alternatively, a switch that provides access to server ports usually connects to them at a much higher speed than is used by end user devices.

Layer 3

Within the confines of the Ethernet physical layer, a layer 3 switch can perform some or all of the functions normally performed by a router. A true router is able to forward traffic from one type of network connection (e.g., T1, DSL) to another (e.g., Ethernet, WiFi).

The most common layer-3 capability is awareness of IP multicast. With this awareness, a layer-3 switch can increase efficiency by delivering the traffic of a multicast group only to ports where the attached device has signaled that it wants to listen to that group. If a switch is not aware of multicasting and broadcasting, frames are also forwarded on all ports of each broadcast domain, but in the case of IP multicast this causes inefficient use of bandwidth. To work around this problem some switches implement IGMP snooping.

Layer 4

While the exact meaning of the term Layer-4 switch is vendor-dependent, it almost always starts with a capability for network address translation, but then adds some type of load distribution based on TCP sessions. The device may include a stateful firewall, a VPN concentrator, or be an IPSec security gateway.

Layer 7

Layer 7 switches may distribute loads based on URL or by some installation-specific technique to recognize application-level transactions. A Layer-7 switch may include a web cache and participate in a content delivery network.

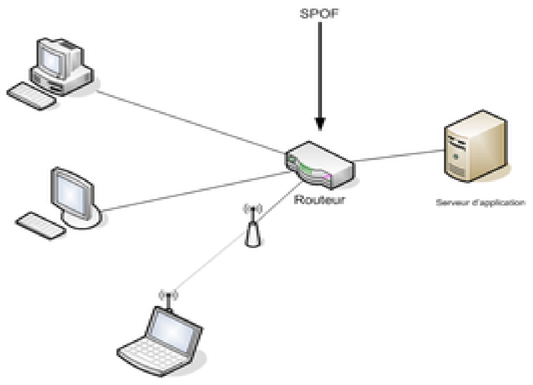


Fig.6.1 Router

Router

A device that forwards data packets along networks. A router is connected to at least two networks, commonly two LANs or WANs or a LAN and its ISP’s network. Routers are located at gateways, the places where two or more networks connect.

Routers use headers and forwarding tables to determine the best path for forwarding the packets, and they use protocols such as ICMP to communicate with each other and configure the best route between any two hosts. Very little filtering of data is done through routers.

Types of routers

Routers may provide connectivity inside enterprises, between enterprises and the Internet, and inside Internet Service Providers (ISPs). The largest routers (for example the Cisco CRS-1 or Juniper T1600) interconnect ISPs, are used inside ISPs, or may be used in very large enterprise networks. The smallest routers provide connectivity for small and home offices. Routers for Internet connectivity and internal use

Routers intended for ISP and major enterprise connectivity will almost invariably exchange routing information with the Border Gateway Protocol (BGP). RFC 4098 defines several types of BGP-speaking routers:

- Edge Router: Placed at the edge of an ISP network, it speaks external BGP (eBGP) to a BGP speaker in another provider or large enterprise Autonomous System(AS).
- Subscriber Edge Router: Located at the edge of the subscriber's network, it speaks eBGP to its provider's AS(s). It belongs to an end user (enterprise) organization.

- Inter-provider Border Router: Interconnecting ISPs, this is a BGP speaking router that maintains BGP sessions with other BGP speaking routers in other providers' ASes.
- Core router: A router that resides within the middle or backbone of the LAN network rather than at its periphery.

Within an ISP: Internal to the provider's AS, such a router speaks internal BGP (iBGP) to that provider's edge routers, other intra-provider core routers, or the provider's inter-provider border routers.

"Internet backbone:" The Internet does not have a clearly identifiable backbone, as did its predecessors. See default-free zone (DFZ). Nevertheless, it is the major ISPs' routers that make up what many would consider the core. These ISPs operate all four types of the BGP-speaking routers described here. In ISP usage, a "core" router is internal to an ISP, and used to interconnect its edge and border routers. Core routers may also have specialized functions in virtual private networks based on a combination of BGP and Multi-Protocol Label Switching (MPLS). Routers are also used for port forwarding for private servers.

RJ45

The RJ45 is a modular connector used to terminate twisted pair and multiconductor flat cable. It is not in common use, but RJ45 is used as the name for the 8P8C modular connector, which is commonly used to terminate Ethernet cable.

Originally, there was only the true telephone RJ45. It is one of the many registered jacks, like RJ11, a standard from which it gets the "RJ" in its name. As a registered jack, true telephone RJ45 specifies both the physical connector and wiring pattern. The true telephone RJ45 uses a special, keyed 8P2C modular connector, with Pins 5 and 4 wired for tip and ring of a single telephone line and Pins 7 and 8 connected to a programming resistor. It is meant to be used with a high speed modem, and is obsolete today.

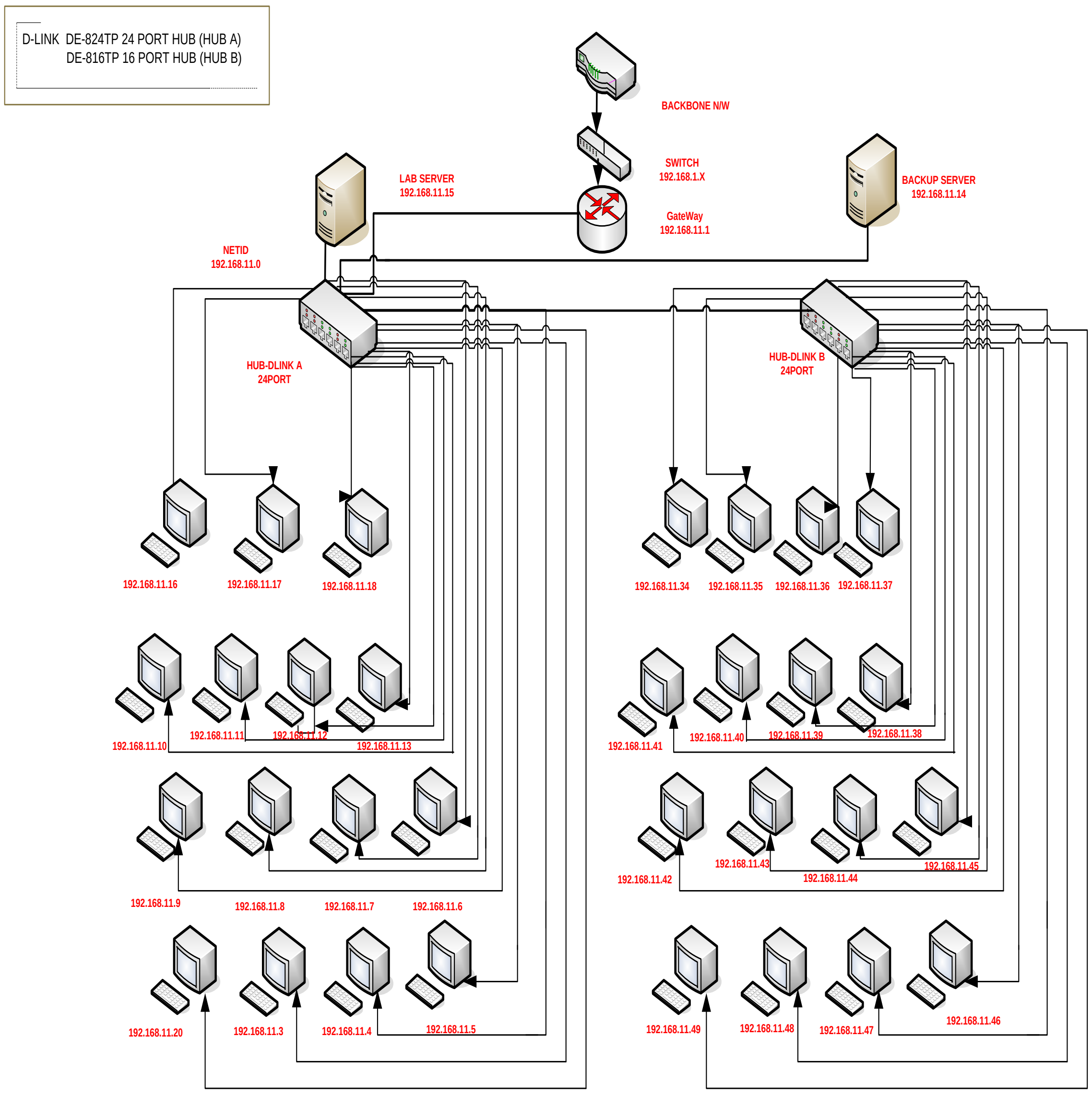


FIG.6.2 Network Diagram of lab 9

In this diagram BACKBONE SERVER connected to GATEWAY SERVER through SWITCH 192.168.1.X. this GATEWAY further connected with HUB A. HUB A is internally connected with HUB B and all the computers of IP address 192.168.11.2 to 192.168.11.18. and HUB B is connected with all computers with IP address 192.168.11.34 to 192.168.11.46.

CONCLUSION:

EXPERIMENT – 7

STUDY OF NETWORK COMMANDS

OBJECTIVES:

(i) To test different network utilities like ping, tracert, arp, ipconfig
(ii) To study and use different options for these utilities

Ping

If any system (host or router) want to communicate with the other system (host or route) then it is necessary to check the communication is possible or not? For this, first of all we have to check for destination system is reachable or not. Due to hardware failure or any other reason it is possible the system may on network but not reachable. How can we detect that the destination system is reachable or not?

PING command is useful for checking the reach ability of the system.

PROCEDURE:

(1) First go to command prompt
(2) For help and information about this command type ping /?
(3) Type ping IP address of system

Example(Different Ping Options):

1. [root@ec20 root]# ping 192.168.51.1

```
PING 192.168.51.1 (192.168.51.1) 56(84) bytes of data.
64 bytes from 192.168.51.1: icmp_seq=1 ttl=253 time=0.444 ms
64 bytes from 192.168.51.1: icmp_seq=2 ttl=253 time=0.406 ms
64 bytes from 192.168.51.1: icmp_seq=3 ttl=253 time=0.421 ms
64 bytes from 192.168.51.1: icmp_seq=4 ttl=253 time=0.402 ms
^C
--- 192.168.51.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.402/0.418/0.444/0.021 ms
```

2. [student@local ~]$ ping -t 1 192.168.51.1

```
PING 192.168.51.1 (192.168.51.1) 56(84) bytes of data.
From 192.168.11.1 icmp_seq=1 Time to live exceeded
From 192.168.11.1 icmp_seq=2 Time to live exceeded
From 192.168.11.1 icmp_seq=3 Time to live exceeded
^C
--- 192.168.51.1 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2400ms
```

3.
```
[student@local ~]$ ping -t 2 192.168.51.1

PING 192.168.51.1 (192.168.51.1) 56(84) bytes of data.
From 192.168.11.1 icmp_seq=1 Time to live exceeded
From 192.168.11.1 icmp_seq=2 Time to live exceeded
From 192.168.11.1 icmp_seq=3 Time to live exceeded
^C
--- 192.168.51.1 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2400ms
```

4.
```
[student@local ~]$ ping -t 3 192.168.51.1

PING 192.168.51.1 (192.168.51.1) 56(84) bytes of data.
64 bytes from 192.168.51.1: icmp_seq=1 ttl=253 time=0.444 ms
64 bytes from 192.168.51.1: icmp_seq=2 ttl=253 time=0.406 ms
64 bytes from 192.168.51.1: icmp_seq=3 ttl=253 time=0.421 ms
64 bytes from 192.168.51.1: icmp_seq=4 ttl=253 time=0.402 ms
^C
--- 192.168.51.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.402/0.418/0.444/0.021 ms
```

5.
```
[student@local ~]$ ping -c 2 192.168.51.1

PING 192.168.51.1 (192.168.51.1) 56(84) bytes of data.
64 bytes from 192.168.51.1: icmp_seq=1 ttl=253 time=0.436 ms
64 bytes from 192.168.51.1: icmp_seq=2 ttl=253 time=0.408 ms

--- 192.168.51.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.408/0.422/0.436/0.014 ms
```

6.
```
[student@local ~]$ ping -I eth0 192.168.51.1
Warning: cannot bind to specified iface, falling back: Operation not permitted
ping: unknown iface eth0
```

7.
```
[student@local ~]$ ping -I eth1 192.168.51.1
PING 192.168.51.1 (192.168.51.1) from 192.168.11.14 eth1: 56(84) bytes of data.
64 bytes from 192.168.51.1: icmp_seq=1 ttl=253 time=0.428 ms
```

```
64 bytes from 192.168.51.1: icmp_seq=2 ttl=253 time=0.395 ms
64 bytes from 192.168.51.1: icmp_seq=3 ttl=253 time=0.404 ms
64 bytes from 192.168.51.1: icmp_seq=4 ttl=253 time=0.402 ms
64 bytes from 192.168.51.1: icmp_seq=5 ttl=253 time=0.407 ms
^C
--- 192.168.51.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4315ms
rtt min/avg/max/mdev = 0.395/0.407/0.428/0.016 ms
```

8.

```
[student@local ~]$  ping -r 2 192.168.51.1
Pinging 192.168.51.1 with 32 bytes of data:
Reply from 192.168.51.1: bytes=32 time<1ms TTL=255
   Route: 192.168.51.1 ->
          192.168.51.1
Reply from 192.168.51.1: bytes=32 time<1ms TTL=255
   Route: 192.168.51.1 ->
          192.168.51.1
Reply from 192.168.51.1: bytes=32 time<1ms TTL=255
   Route: 192.168.51.1 ->
          192.168.51.1
Reply from 192.168.51.1: bytes=32 time<1ms TTL=255
   Route: 192.168.51.1 ->
          192.168.51.1
Ping statistics for 192.168.51.1:
   Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
   Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Tracert / Tracepath

When one system (host or router) send the packet of data to another system then there be two possibilities, Packet directly reach to destination system or it pass through one or more routers. TRACERT command is useful to trace the route through which packet passes.

Procedure:

(1) First go to command prompt
(2) For help and information about this command type tracert /?
(3) Type tracert <u>IP address or name of the destination.</u>

Example:

1.

```
[root@ec20 root]# tracert 192.168.1.100
tracert 192.168.1.100
```

```
Tracing route to 192.168.1.100 over a maximum of 30 hops

  1    <1 ms    <1 ms    <1 ms  192.168.51.1
  2    <1 ms    <1 ms    <1 ms  192.168.1.100

Trace complete.
```

ipconfig

The "ifconfig" command allows the operating system to setup network interfaces and allow the user to view information about the configured network interfaces. In absence of arguments, it displays the status of currently active interface. If a single interface argument is given it displays the status of the given interface only.

PROCEDURE:
(1) First go to command prompt
(2) For help and information about this command type ipconfig /?
(3) Type ipconfig.

Example:

1.

```
[root@ec20 root]# ifconfig
eth0     Link encap:Ethernet  HWaddr 00:80:AD:83:2B:1C
         inet addr:192.168.11.20  Bcast:192.168.11.255  Mask:255.255.255.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:10 errors:0 dropped:0 overruns:0 frame:0
         TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:100
         RX bytes:600 (600.0 b)  TX bytes:0 (0.0 b)
         Interrupt:11 Base address:0xc000

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         UP LOOPBACK RUNNING  MTU:16436  Metric:1
         RX packets:10 errors:0 dropped:0 overruns:0 frame:0
         TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:700 (700.0 b)  TX bytes:700 (700.0 b)
```

ARP

The Address Resolution Protocol (or ARP) is used to map the Internet Protocol address of a machine to its Ethernet address. It simply uses a table which lists each host, its IP address and its Ethernet address. You can view the the ARP table on your machine via the arp -a command

When a datagram is received addressed to an IP address on your network the router uses this table to find the Ethernet address of the specified host and forwards the datagram appropriately.

Procedure

(1) First go to command prompt
(2) For help and information about this command type tracert /?
(3) Type arp -a

Example(with different arp options):

1.
```
[root@local student]# cat /proc/net/arp or arp -e
IP address      HW type    Flags      HW address        Mask     Device
192.168.11.3     0x1        0x2       00:19:db:68:d4:9c     *       eth1
192.168.11.4     0x1        0x2       00:19:db:68:d4:8a     *       eth1
192.168.11.1     0x1        0x2       00:50:af:0a:5b:80     *       eth1
```

2.
```
[root@local student]# arp -d 192.168.11.3
```

3.
```
[root@local student]# cat /proc/net/arp or arp -e
IP address      HW type    Flags      HW address       Mask     Device
192.168.11.3     0x1        0x0       00:19:db:68:d4:9c     *       eth1
192.168.11.4     0x1        0x2       00:19:db:68:d4:8a     *       eth1
192.168.11.1     0x1        0x2       00:50:af:0a:5b:80     *       eth1
```

CONCLUSION:

EXPERIMENT – 8

UNDERSTANDING PACKET CAPTURING USING WIRESHARK

OBJECTIVE:

(i) Understanding of packet capturing tool

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

PROCEDURE:

(1) Open Wireshark

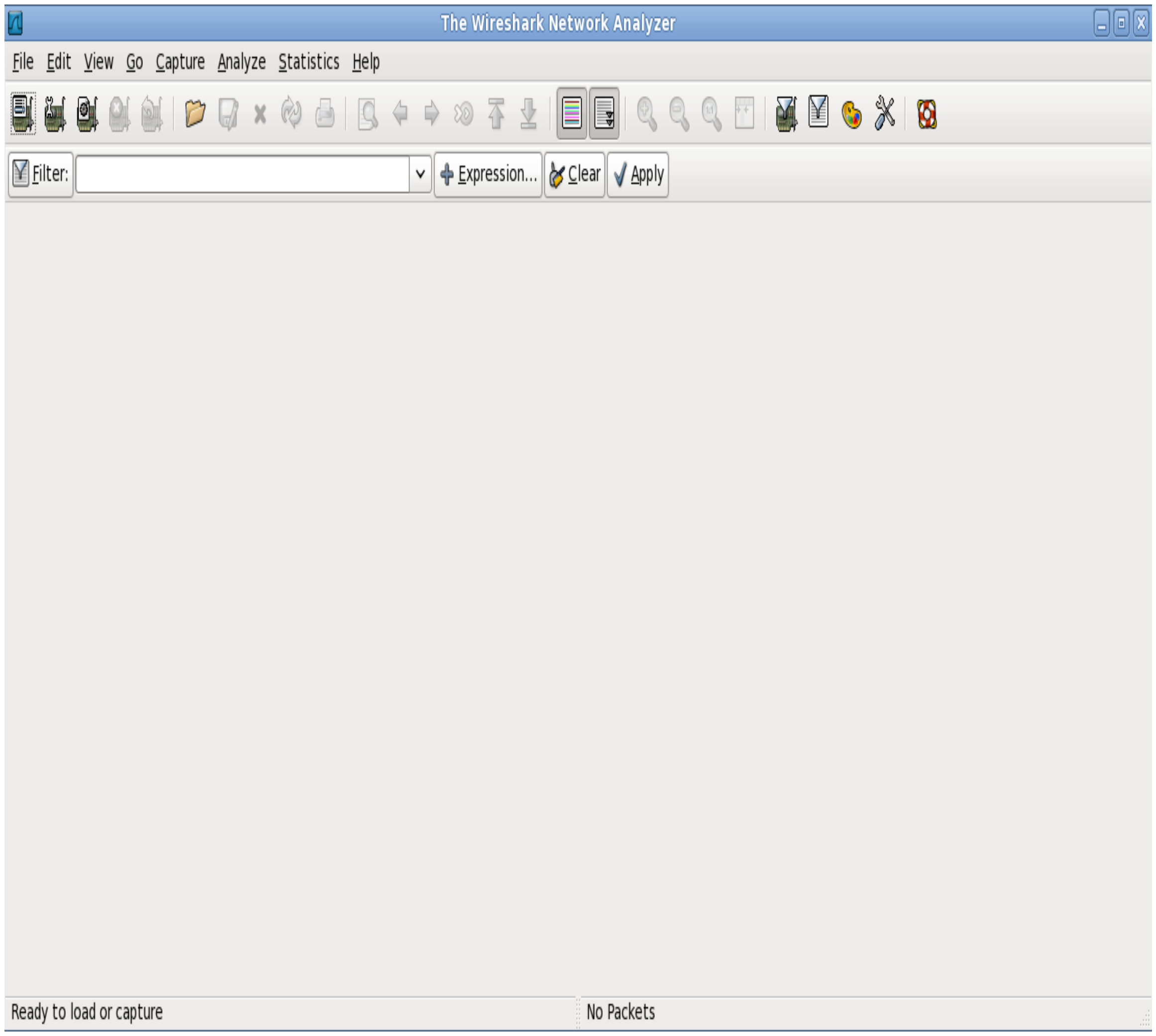

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply
Ready to load or capture
No Packets

Fig.8.1 Main Window of Wireshrk

(2) Goto Capture option

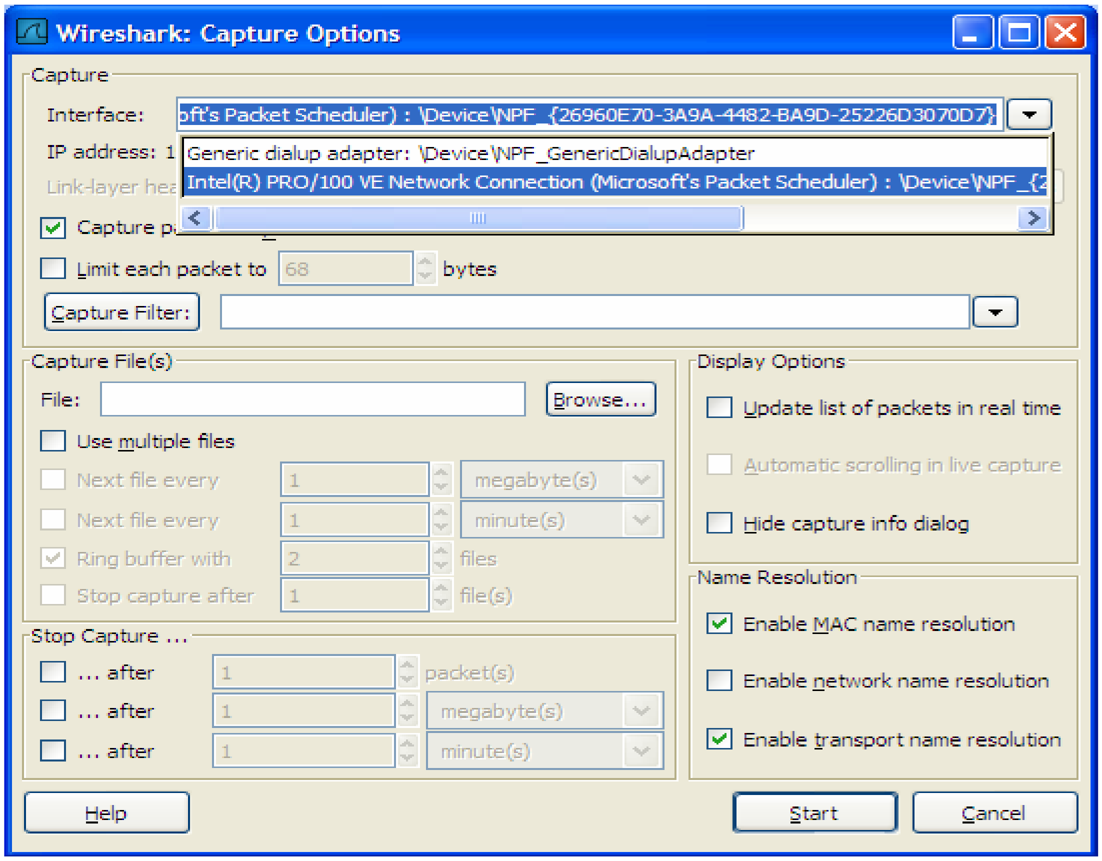
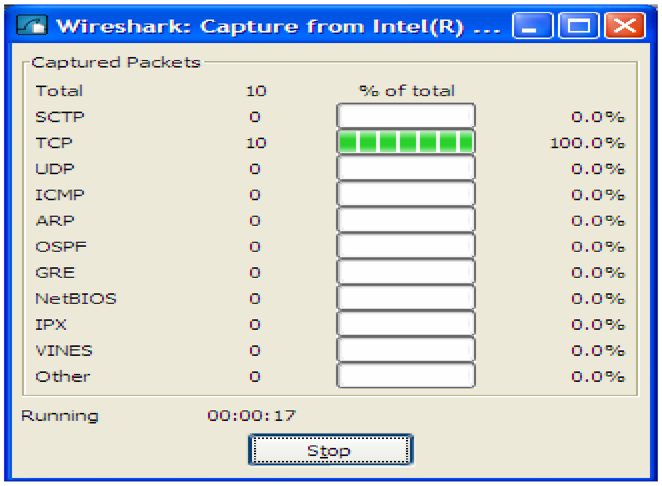

Wireshark: Capture Options
Capture
Interface: oft's Packet Scheduler) : \Device\NPF_{26960E70-3A9A-4482-BA9D-25226D3070D7}
IP address: 1
Generic dialup adapter: \Device\NPF_GenericDialupAdapter
Intel(R) PRO/100 VE Network Connection (Microsoft's Packet Scheduler) : \Device\NPF_{2
Link-layer hea
✓ Capture p
Limit each packet to 68 bytes
Capture Filter:
Capture File(s)
File:
Browse...
Use multiple files
Next file every 1 megabyte(s)
Next file every 1 minute(s)
✓ Ring buffer with 2 files
Stop capture after 1 file(s)
Stop Capture ...
... after 1 packet(s)
... after 1 megabyte(s)
... after 1 minute(s)
Display Options
Update list of packets in real time
Automatic scrolling in live capture
Hide capture info dialog
Name Resolution
✓ Enable MAC name resolution
Enable network name resolution
✓ Enable transport name resolution
Help
Start
Cancel

Fig.8.2 Capture Option

3. Select the Interface and Start capture



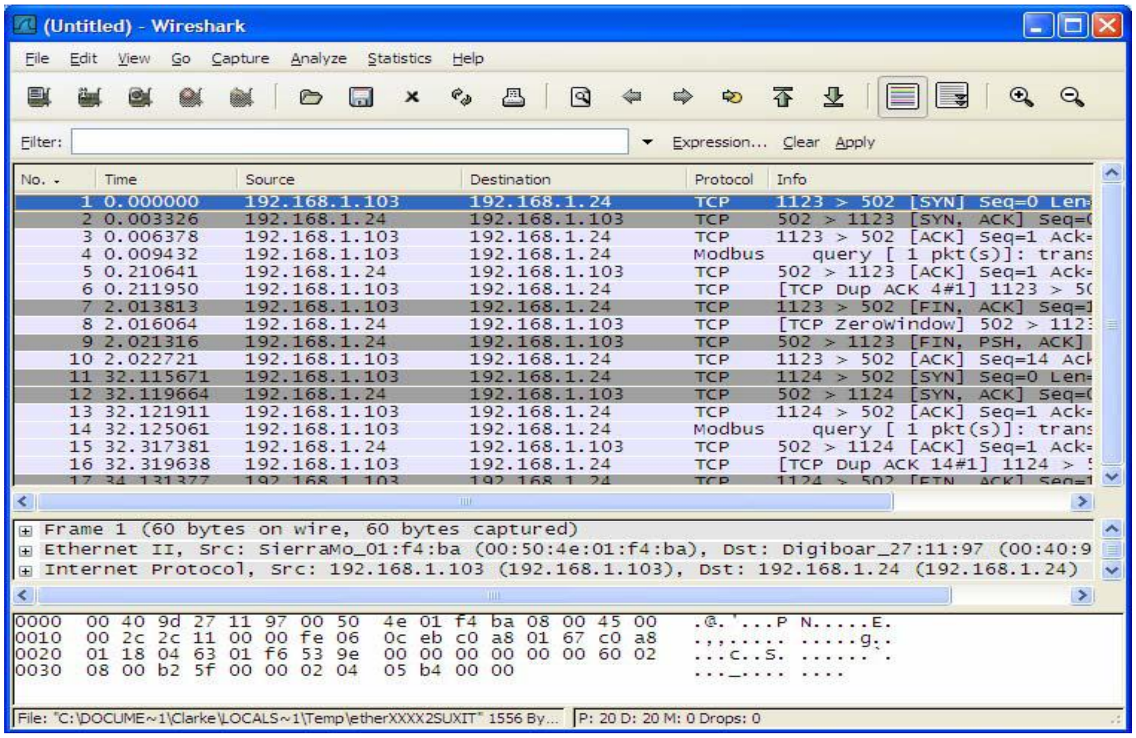
		% of total	
Total	10		
SCTP	0		0.0%
TCP	10	🟩	100.0%
UDP	0		0.0%
ICMP	0		0.0%
ARP	0		0.0%
OSPF	0		0.0%
GRE	0		0.0%
NetBIOS	0		0.0%
IPX	0		0.0%
VINES	0		0.0%
Other	0		0.0%

Running 00:00:17

Fig.8.3 Start Capturing

Stop capturing after some time.

4. It will display the captured packet and details of those packets


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -  Time        Source          Destination     Protocol  Info
     1 0.000000    192.168.1.103   192.168.1.24    TCP       1123 > 502 [SYN] Seq=0 Len=
     2 0.003326    192.168.1.24    192.168.1.103   TCP       502 > 1123 [SYN, ACK] Seq=(
     3 0.006378    192.168.1.103   192.168.1.24    TCP       1123 > 502 [ACK] Seq=1 Ack=
     4 0.009432    192.168.1.103   192.168.1.24    Modbus      query [ 1 pkt(s)]: trans
     5 0.210641    192.168.1.24    192.168.1.103   TCP       502 > 1123 [ACK] Seq=1 Ack=
     6 0.211950    192.168.1.103   192.168.1.24    TCP       [TCP Dup ACK 4#1] 1123 > 50
     7 2.013813    192.168.1.103   192.168.1.24    TCP       1123 > 502 [FIN, ACK] Seq=1
     8 2.016064    192.168.1.24    192.168.1.103   TCP       [TCP ZeroWindow] 502 > 112
     9 2.021316    192.168.1.24    192.168.1.103   TCP       502 > 1123 [FIN, PSH, ACK]
    10 2.022721    192.168.1.103   192.168.1.24    TCP       1123 > 502 [ACK] Seq=14 Ack
    11 32.115671   192.168.1.103   192.168.1.24    TCP       1124 > 502 [SYN] Seq=0 Len=
    12 32.119664   192.168.1.24    192.168.1.103   TCP       502 > 1124 [SYN, ACK] Seq=(
    13 32.121911   192.168.1.103   192.168.1.24    TCP       1124 > 502 [ACK] Seq=1 Ack=
    14 32.125061   192.168.1.103   192.168.1.24    Modbus      query [ 1 pkt(s)]: trans
    15 32.317381   192.168.1.24    192.168.1.103   TCP       502 > 1124 [ACK] Seq=1 Ack=
    16 32.319638   192.168.1.103   192.168.1.24    TCP       [TCP Dup ACK 14#1] 1124 > 5
    17 34.131377   192.168.1.103   192.168.1.24    TCP       1124 > 502 [FIN, ACK] Seq=1
```

```
⊞ Frame 1 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: SierraMo_01:f4:ba (00:50:4e:01:f4:ba), Dst: Digiboar_27:11:97 (00:40:9
⊞ Internet Protocol, Src: 192.168.1.103 (192.168.1.103), Dst: 192.168.1.24 (192.168.1.24)
```

```
0000  00 40 9d 27 11 97 00 50  4e 01 f4 ba 08 00 45 00   .@.'...P N.....E.
0010  00 2c 2c 11 00 00 fe 06  0c eb c0 a8 01 67 c0 a8   .,,..... .....g..
0020  01 18 04 63 01 f6 53 9e  00 00 00 00 00 00 60 02   ...c..S. ......`.
0030  08 00 b2 5f 00 00 02 04  05 b4 00 00               ..._.... ....
```

```
File: "C:\DOCUME~1\Clarke\LOCALS~1\Temp\etherXXXX2SUXIT" 1556 By...   P: 20 D: 20 M: 0 Drops: 0
```

Fig.8.4 Packet Captured

5. It will display the captured packet and details of those packets.

CONCLUSION:

EXPERIMENT – 9

STUDY OF TCP AND IP HEADER USING WIRESHARK

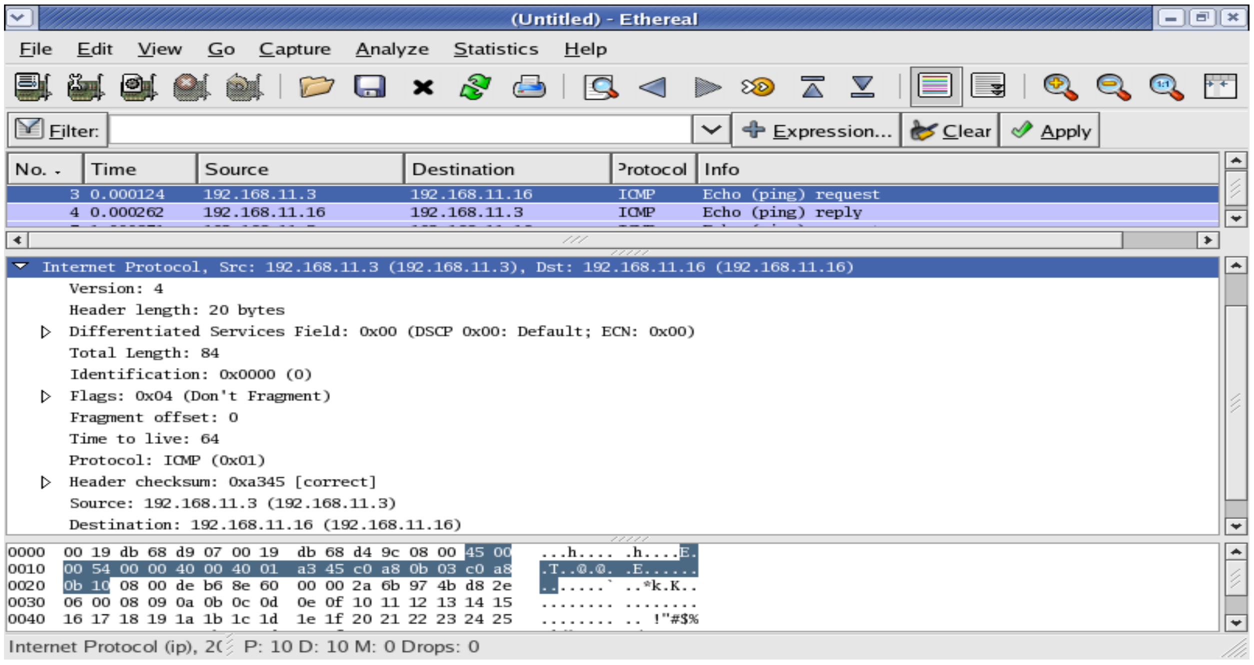
OBJECTIVES:

(i) Analysis of captured packets
(ii) Access the TCP and IP header using Wireshark

PROCEDURE :

(1) Prepare *Wireshark* for a packet capture.
(2) Open a Command Prompt window. (In Linux , from menu the run command and type konsole).
(3) Type “ping/ftp <IP address or website>”; Do NOT press ENTER.
(4) Start *Wireshark* packet capture.
(5) Press ENTER in Command Prompt window. You should obtain a series of replies in command prompt.
(6) Stop packet capture when Command Prompt returns.
(7) Save the contents in the Command Prompt window using a screen capture.

Protocol Analysis
IP Header
Command : ping 192.168.11.16

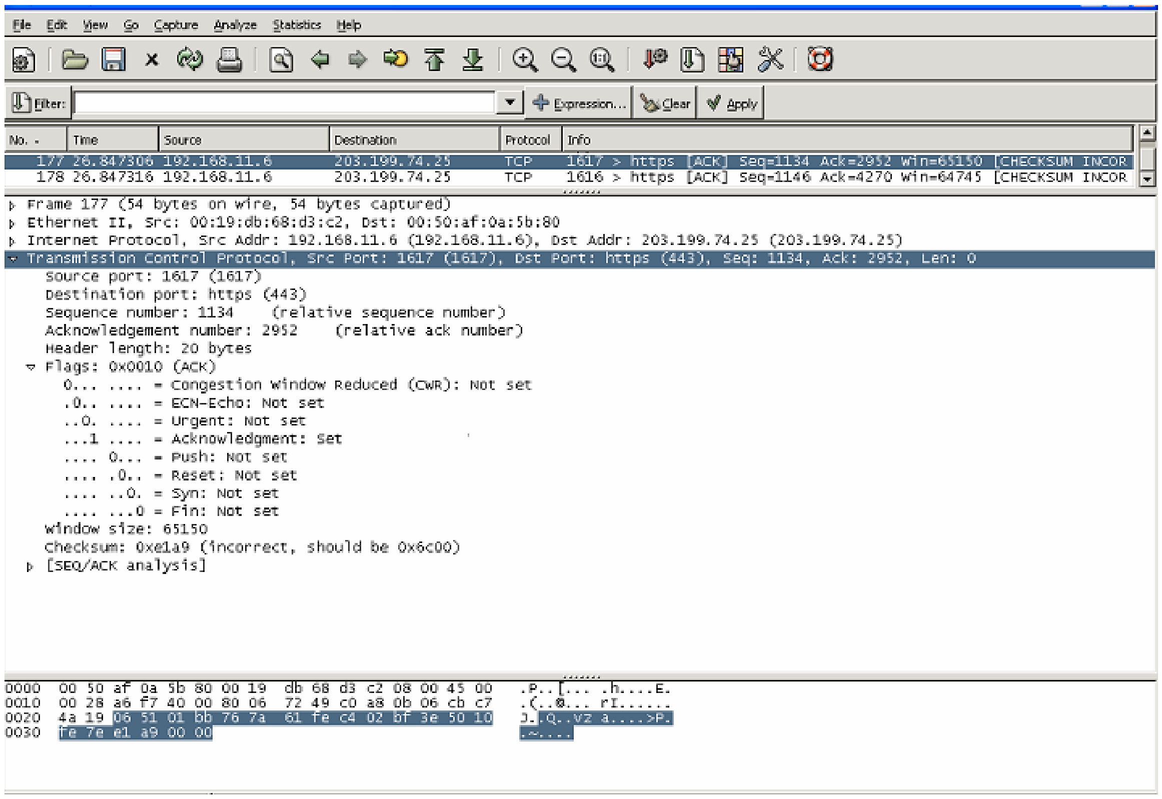


```
No. - Time      Source         Destination    Protocol Info
    3 0.000124  192.168.11.3   192.168.11.16  ICMP     Echo (ping) request
    4 0.000262  192.168.11.16  192.168.11.3   ICMP     Echo (ping) reply

▽ Internet Protocol, Src: 192.168.11.3 (192.168.11.3), Dst: 192.168.11.16 (192.168.11.16)
      Version: 4
      Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
      Total Length: 84
      Identification: 0x0000 (0)
  ▷ Flags: 0x04 (Don't Fragment)
      Fragment offset: 0
      Time to live: 64
      Protocol: ICMP (0x01)
  ▷ Header checksum: 0xa345 [correct]
      Source: 192.168.11.3 (192.168.11.3)
      Destination: 192.168.11.16 (192.168.11.16)

0000  00 19 db 68 d9 07 00 19  db 68 d4 9c 08 00 45 00   ...h.... .h....E.
0010  00 54 00 00 40 00 40 01  a3 45 c0 a8 0b 03 c0 a8   .T..@.@. .E......
0020  0b 10 08 00 de b6 8e 60  00 00 2a 6b 97 4b d8 2e   .......` ..*k.K..
0030  06 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%

Internet Protocol (ip), 2(   P: 10 D: 10 M: 0 Drops: 0
```

Fig.9.1 IP Packet Format Using Wireshrk

TCP Header

Command : ftp 192.168.11.6



File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -  Time       Source           Destination      Protocol  Info
  177 26.847306 192.168.11.6     203.199.74.25     TCP      1617 > https [ACK] Seq=1134 Ack=2952 Win=65150 [CHECKSUM INCOR
  178 26.847316 192.168.11.6     203.199.74.25     TCP      1616 > https [ACK] Seq=1146 Ack=4270 Win=64745 [CHECKSUM INCOR
```

```
▷ Frame 177 (54 bytes on wire, 54 bytes captured)
▷ Ethernet II, Src: 00:19:db:68:d3:c2, Dst: 00:50:af:0a:5b:80
▷ Internet Protocol, Src Addr: 192.168.11.6 (192.168.11.6), Dst Addr: 203.199.74.25 (203.199.74.25)
▽ Transmission Control Protocol, Src Port: 1617 (1617), Dst Port: https (443), Seq: 1134, Ack: 2952, Len: 0
    Source port: 1617 (1617)
    Destination port: https (443)
    Sequence number: 1134    (relative sequence number)
    Acknowledgement number: 2952    (relative ack number)
    Header length: 20 bytes
  ▽ Flags: 0x0010 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 65150
    Checksum: 0xe1a9 (incorrect, should be 0x6c00)
  ▷ [SEQ/ACK analysis]
```

```
0000   00 50 af 0a 5b 80 00 19  db 68 d3 c2 08 00 45 00   .P..[... .h....E.
0010   00 28 a6 f7 40 00 80 06  72 49 c0 a8 0b 06 cb c7   .(..@... rI......
0020   4a 19 06 51 01 bb 76 7a  61 fe c4 02 bf 3e 50 10   J..Q..vz a....>P.
0030   fe 7e e1 a9 00 00                                  .~....
```

Fig.9.2   TCP Packet Format Using Ehtereal

CONCLUSION :

EXPERIMENT - 10

UNDERSTANDING EXECUTION OF PING COMMANDS USING WIRESHARK

OBJECTIVE:

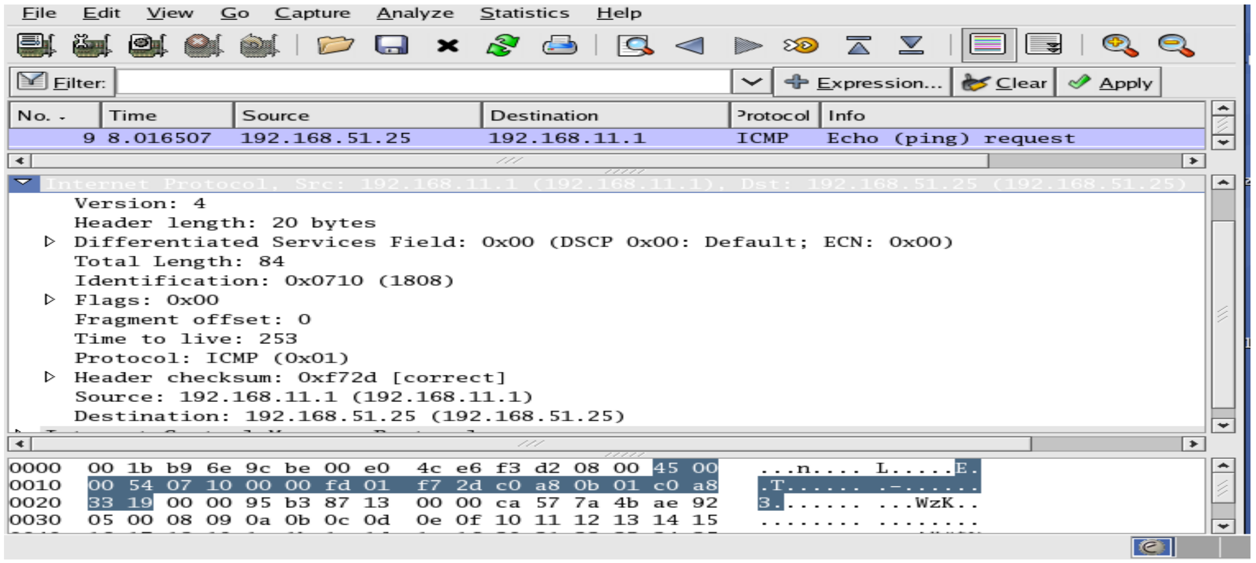
(i) Investigate an applications of the Internet Control Message Protocol (ICMP)

PROCEDURE:

(1) Prepare *Wireshark* for a packet capture.
(2) Open a Command Prompt window. (In Linux , from menu the run command and type konsole).
(3) Type “ping <IP address or website>”; Do NOT press ENTER.
(4) Start *Wireshark* packet capture.
(5) Press ENTER in Command Prompt window. You should obtain a series of replies in command prompt.
(6) Stop packet capture when Command Prompt returns.
(7) Save the contents in the Command Prompt window using a screen capture.

Protocol Analysis

1. ICMP Echo Request and Reply
   Command : ping 192.168.11.1

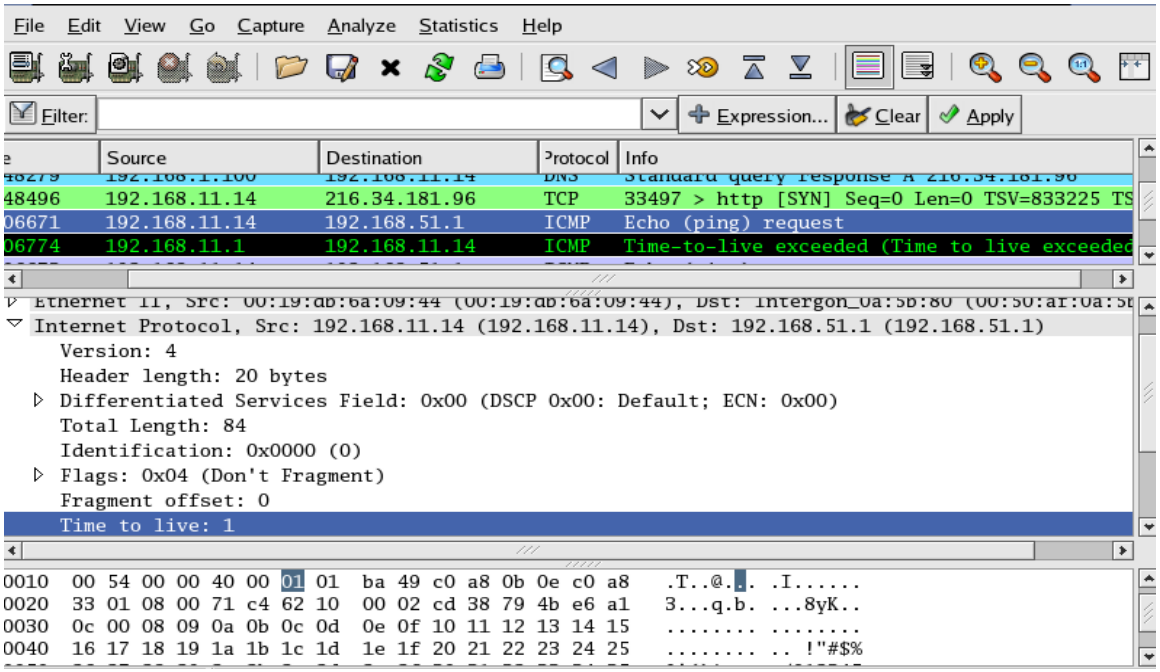

File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -  Time        Source           Destination     Protocol  Info
    9  8.016507    192.168.51.25    192.168.11.1    ICMP      Echo (ping) request

Internet Protocol, Src: 192.168.11.1 (192.168.11.1), Dst: 192.168.51.25 (192.168.51.25)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 84
    Identification: 0x0710 (1808)
  ▷ Flags: 0x00
    Fragment offset: 0
    Time to live: 253
    Protocol: ICMP (0x01)
  ▷ Header checksum: 0xf72d [correct]
    Source: 192.168.11.1 (192.168.11.1)
    Destination: 192.168.51.25 (192.168.51.25)

0000  00 1b b9 6e 9c be 00 e0  4c e6 f3 d2 08 00 45 00   ...n.... L.....E.
0010  00 54 07 10 00 00 fd 01  f7 2d c0 a8 0b 01 c0 a8   .T...... .-......
0020  33 19 00 00 95 b3 87 13  00 00 ca 57 7a 4b ae 92   3....... ...WzK..
0030  05 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
```

Fig.10.1 Request and Reply

2. IP Time-to-Live

Command : ping -t 1 192.168.51.1



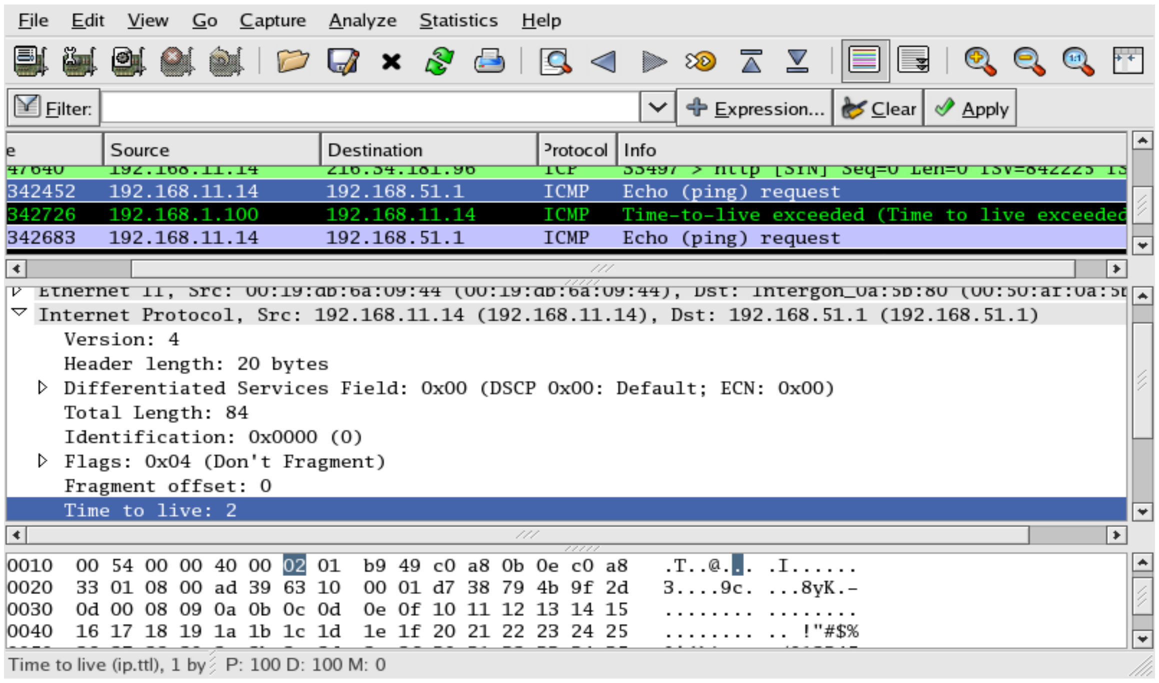
```
e        Source          Destination     Protocol Info
48279    192.168.1.100   192.168.11.14   DNS      Standard query response A 216.34.181.96
48496    192.168.11.14   216.34.181.96   TCP      33497 > http [SYN] Seq=0 Len=0 TSV=833225 TS
06671    192.168.11.14   192.168.51.1    ICMP     Echo (ping) request
06774    192.168.11.1    192.168.11.14   ICMP     Time-to-live exceeded (Time to live exceeded
```

```
▷ Ethernet II, Src: 00:19:db:6a:09:44 (00:19:db:6a:09:44), Dst: Intergon_0a:5b:80 (00:50:af:0a:5b
▽ Internet Protocol, Src: 192.168.11.14 (192.168.11.14), Dst: 192.168.51.1 (192.168.51.1)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 84
    Identification: 0x0000 (0)
  ▷ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 1
```

```
0010  00 54 00 00 40 00 01 01  ba 49 c0 a8 0b 0e c0 a8   .T..@... .I......
0020  33 01 08 00 71 c4 62 10  00 02 cd 38 79 4b e6 a1   3...q.b. ...8yK..
0030  0c 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
```

Fig.10.2 IP Time to Live Request

Command : ping -t 2 192.168.51.1



```
e        Source          Destination     Protocol Info
47640    192.168.11.14   216.34.181.96   TCP      33497 > http [SYN] Seq=0 Len=0 TSV=842225 TS
342452   192.168.11.14   192.168.51.1    ICMP     Echo (ping) request
342726   192.168.1.100   192.168.11.14   ICMP     Time-to-live exceeded (Time to live exceeded
342683   192.168.11.14   192.168.51.1    ICMP     Echo (ping) request
```

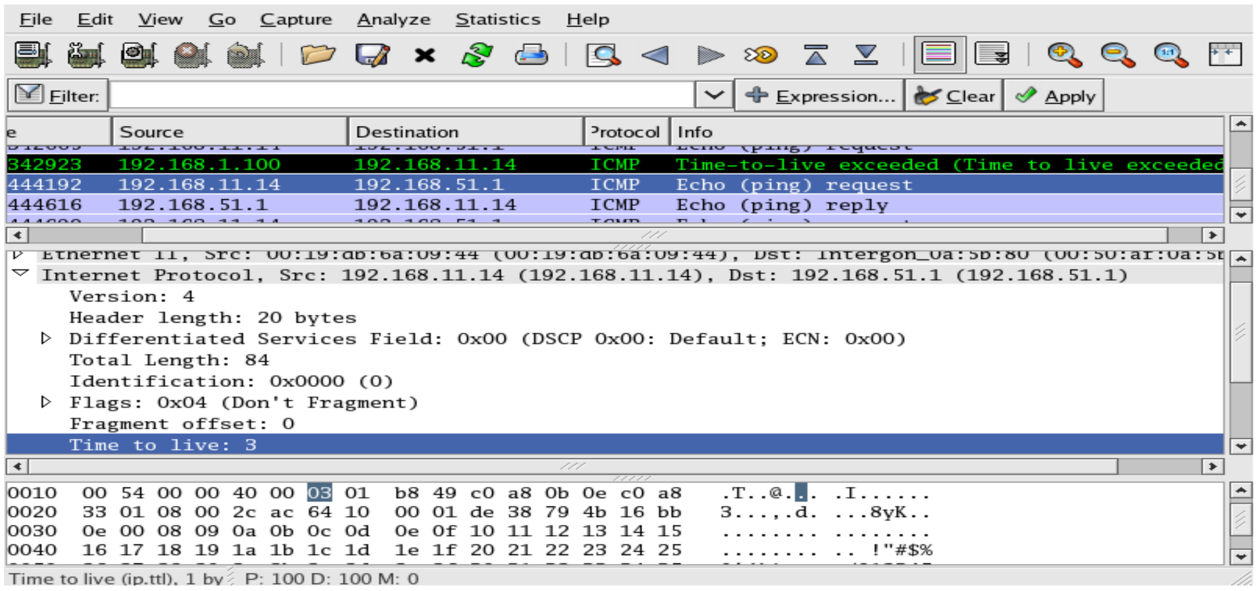
```
▷ Ethernet II, Src: 00:19:db:6a:09:44 (00:19:db:6a:09:44), Dst: Intergon_0a:5b:80 (00:50:af:0a:5b
▽ Internet Protocol, Src: 192.168.11.14 (192.168.11.14), Dst: 192.168.51.1 (192.168.51.1)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 84
    Identification: 0x0000 (0)
  ▷ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 2
```

```
0010  00 54 00 00 40 00 02 01  b9 49 c0 a8 0b 0e c0 a8   .T..@... .I......
0020  33 01 08 00 ad 39 63 10  00 01 d7 38 79 4b 9f 2d   3....9c. ...8yK.-
0030  0d 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
```

```
Time to live (ip.ttl), 1 by   P: 100 D: 100 M: 0
```

Fig.10.3 IP Time to Live Reply

Command : ping -t 3 192.168.51.1

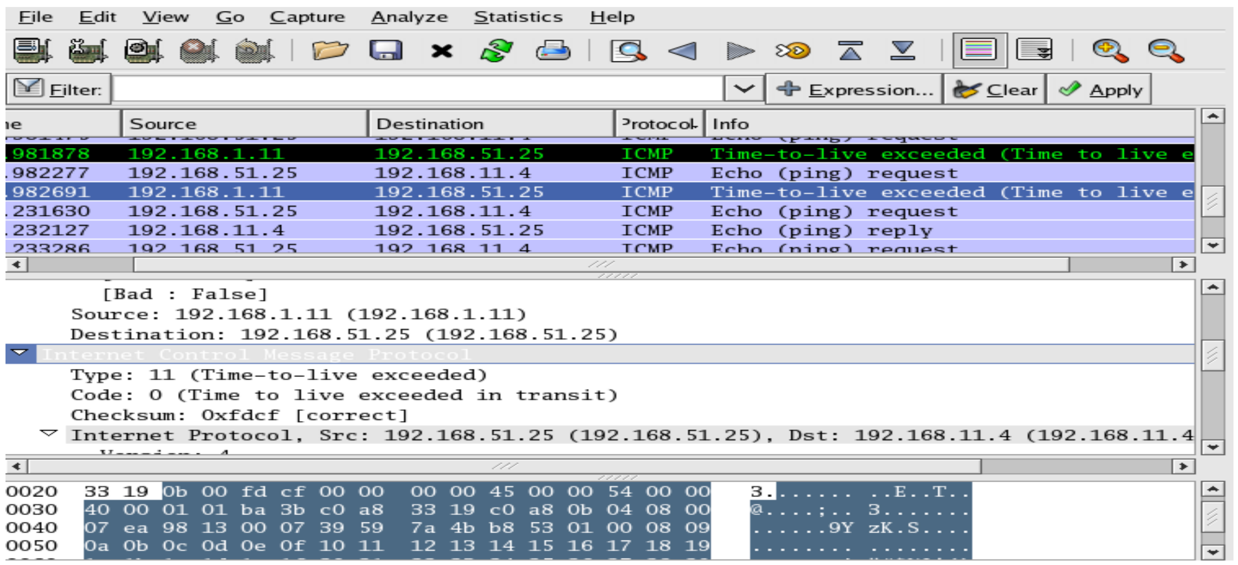

File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
e        Source           Destination      Protocol  Info
342923   192.168.1.100    192.168.11.14    ICMP      Time-to-live exceeded (Time to live exceeded
444192   192.168.11.14    192.168.51.1     ICMP      Echo (ping) request
444616   192.168.51.1     192.168.11.14    ICMP      Echo (ping) reply

▽ Internet Protocol, Src: 192.168.11.14 (192.168.11.14), Dst: 192.168.51.1 (192.168.51.1)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 84
    Identification: 0x0000 (0)
  ▷ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 3

0010  00 54 00 00 40 00 03 01  b8 49 c0 a8 0b 0e c0 a8   .T..@... .I......
0020  33 01 08 00 2c ac 64 10  00 01 de 38 79 4b 16 bb   3...,.d. ...8yK..
0030  0e 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
Time to live (ip.ttl), 1 by   P: 100 D: 100 M: 0
```

Fig.10.4 IP Time to Live = 3

3. Time Exceeded Message

Command : ping -t 1 192.168.11.4


File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
ne       Source           Destination      Protocol  Info
.981878  192.168.1.11     192.168.51.25    ICMP      Time-to-live exceeded (Time to live e
.982277  192.168.51.25    192.168.11.4     ICMP      Echo (ping) request
.982691  192.168.1.11     192.168.51.25    ICMP      Time-to-live exceeded (Time to live e
.231630  192.168.51.25    192.168.11.4     ICMP      Echo (ping) request
.232127  192.168.11.4     192.168.51.25    ICMP      Echo (ping) reply

        [Bad : False]
      Source: 192.168.1.11 (192.168.1.11)
      Destination: 192.168.51.25 (192.168.51.25)
▽ Internet Control Message Protocol
      Type: 11 (Time-to-live exceeded)
      Code: 0 (Time to live exceeded in transit)
      Checksum: 0xfdcf [correct]
    ▽ Internet Protocol, Src: 192.168.51.25 (192.168.51.25), Dst: 192.168.11.4 (192.168.11.4

0020  33 19 0b 00 fd cf 00 00  00 00 45 00 00 54 00 00   3....... ..E..T..
0030  40 00 01 01 ba 3b c0 a8  33 19 c0 a8 0b 04 08 00   @....;.. 3.......
0040  07 ea 98 13 00 07 39 59  7a 4b b8 53 01 00 08 09   ......9Y zK.S....
0050  0a 0b 0c 0d 0e 0f 10 11  12 13 14 15 16 17 18 19   ........ ........
```

Fig.10.5   Time Exceeded Message

CONCLUSION :

EXPERIMENT – 11

UNDERSTANDING EXECUTION OF TRACERT COMMAND USING WIRESHARK

OBJECTIVE:

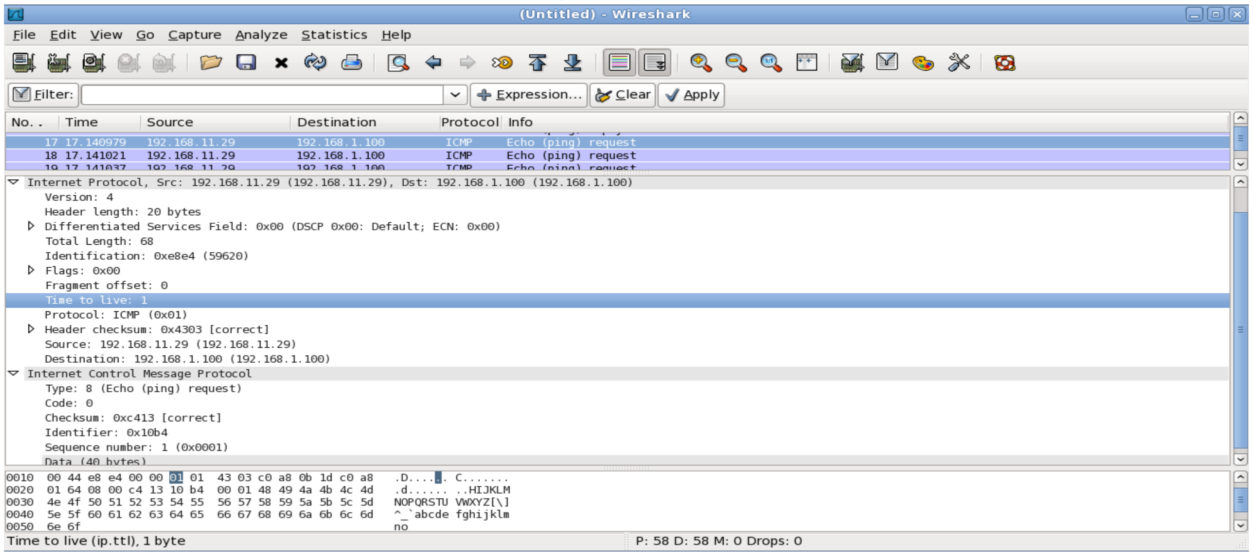
(i) Investigate internal working of TRACERT as an applications of the Internet Control Message Protocol (ICMP)

PROCEDURE:

(1) Prepare *Wireshark* for a packet capture.
(2) Open a Command Prompt window. (In Linux , from menu the run command and type konsole).
(3) Type “tracert <IP address or website>”; Do NOT press ENTER.
(4) Start *Wireshark* packet capture.
(5) Press ENTER in Command Prompt window. You should obtain a series of replies in command prompt.
(6) Stop packet capture when Command Prompt returns.
(7) Save the contents in the Command Prompt window using a screen capture.

PROTOCOL ANALYSIS:

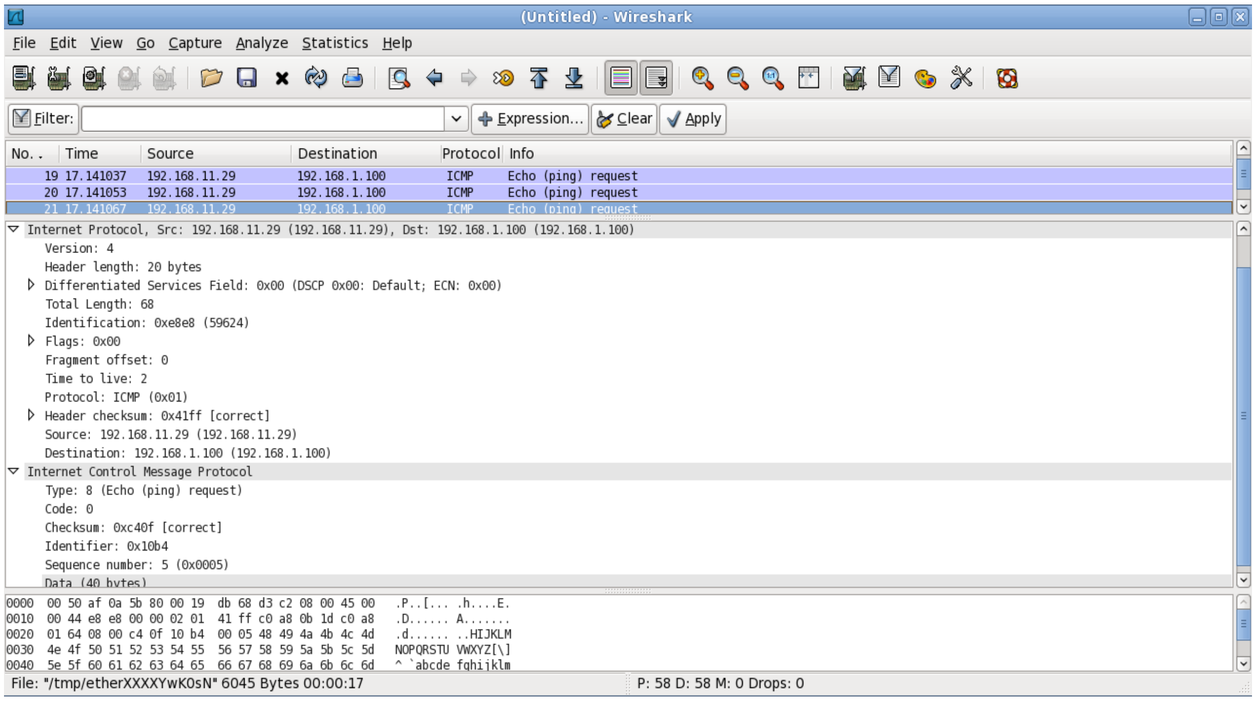
1. ICMP Echo Request and Reply
Command : tracert 192.168.1.100 (ttl = 1)



```
No. .  Time       Source          Destination     Protocol  Info
   17  17.140979  192.168.11.29   192.168.1.100   ICMP      Echo (ping) request
   18  17.141021  192.168.11.29   192.168.1.100   ICMP      Echo (ping) request
   19  17.141037  192.168.11.29   192.168.1.100   ICMP      Echo (ping) request

Internet Protocol, Src: 192.168.11.29 (192.168.11.29), Dst: 192.168.1.100 (192.168.1.100)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 68
    Identification: 0xe8e4 (59620)
  ▷ Flags: 0x00
    Fragment offset: 0
    Time to live: 1
    Protocol: ICMP (0x01)
  ▷ Header checksum: 0x4303 [correct]
    Source: 192.168.11.29 (192.168.11.29)
    Destination: 192.168.1.100 (192.168.1.100)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xc413 [correct]
    Identifier: 0x10b4
    Sequence number: 1 (0x0001)
    Data (40 bytes)

0010  00 44 e8 e4 00 00 01 01  43 03 c0 a8 0b 1d c0 a8   .D...... C.......
0020  01 64 08 00 c4 13 10 b4  00 01 48 49 4a 4b 4c 4d   .d...... ..HIJKLM
0030  4e 4f 50 51 52 53 54 55  56 57 58 59 5a 5b 5c 5d   NOPQRSTU VWXYZ[\]
0040  5e 5f 60 61 62 63 64 65  66 67 68 69 6a 6b 6c 6d   ^_`abcde fghijklm
0050  6e 6f                                              no
Time to live (ip.ttl), 1 byte                    P: 58 D: 58 M: 0 Drops: 0
```

Fig.11.1 ICMP Request and Reply

ttl=2



```
No. .  Time        Source          Destination     Protocol  Info
    19 17.141037   192.168.11.29   192.168.1.100   ICMP      Echo (ping) request
    20 17.141053   192.168.11.29   192.168.1.100   ICMP      Echo (ping) request
    21 17.141067   192.168.11.29   192.168.1.100   ICMP      Echo (ping) request
```

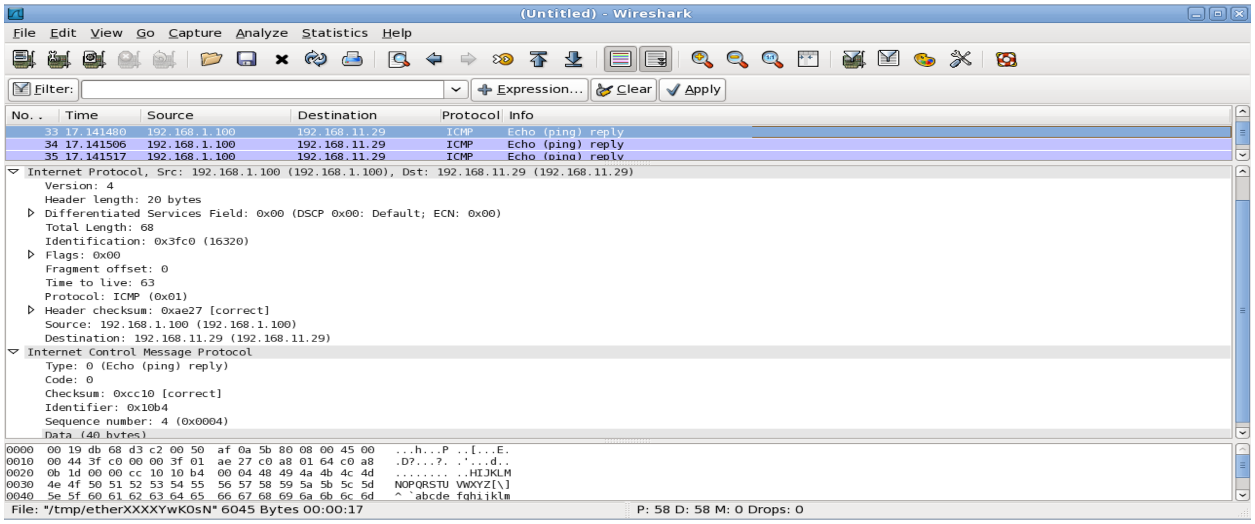
```
▽ Internet Protocol, Src: 192.168.11.29 (192.168.11.29), Dst: 192.168.1.100 (192.168.1.100)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 68
    Identification: 0xe8e8 (59624)
  ▷ Flags: 0x00
    Fragment offset: 0
    Time to live: 2
    Protocol: ICMP (0x01)
  ▷ Header checksum: 0x41ff [correct]
    Source: 192.168.11.29 (192.168.11.29)
    Destination: 192.168.1.100 (192.168.1.100)
▽ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xc40f [correct]
    Identifier: 0x10b4
    Sequence number: 5 (0x0005)
    Data (40 bytes)
```

```
0000  00 50 af 0a 5b 80 00 19  db 68 d3 c2 08 00 45 00   .P..[... .h....E.
0010  00 44 e8 e8 00 00 02 01  41 ff c0 a8 0b 1d c0 a8   .D...... A.......
0020  01 64 08 00 c4 0f 10 b4  00 05 48 49 4a 4b 4c 4d   .d...... ..HIJKLM
0030  4e 4f 50 51 52 53 54 55  56 57 58 59 5a 5b 5c 5d   NOPQRSTU VWXYZ[\]
0040  5e 5f 60 61 62 63 64 65  66 67 68 69 6a 6b 6c 6d   ^_`abcde fghijklm
```

File: "/tmp/etherXXXXYwKOsN" 6045 Bytes 00:00:17   P: 58 D: 58 M: 0 Drops: 0

Fig.11.2 ttl =2

Reply Packet:



```
No. .  Time        Source          Destination     Protocol  Info
    33 17.141480   192.168.1.100   192.168.11.29   ICMP      Echo (ping) reply
    34 17.141506   192.168.1.100   192.168.11.29   ICMP      Echo (ping) reply
    35 17.141517   192.168.1.100   192.168.11.29   ICMP      Echo (ping) reply
```

```
▽ Internet Protocol, Src: 192.168.1.100 (192.168.1.100), Dst: 192.168.11.29 (192.168.11.29)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 68
    Identification: 0x3fc0 (16320)
  ▷ Flags: 0x00
    Fragment offset: 0
    Time to live: 63
    Protocol: ICMP (0x01)
  ▷ Header checksum: 0xae27 [correct]
    Source: 192.168.1.100 (192.168.1.100)
    Destination: 192.168.11.29 (192.168.11.29)
▽ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xcc10 [correct]
    Identifier: 0x10b4
    Sequence number: 4 (0x0004)
    Data (40 bytes)
```

```
0000  00 19 db 68 d3 c2 00 50  af 0a 5b 80 08 00 45 00   ...h...P ..[...E.
0010  00 44 3f c0 00 00 3f 01  ae 27 c0 a8 01 64 c0 a8   .D?...?. .'...d..
0020  0b 1d 00 00 cc 10 10 b4  00 04 48 49 4a 4b 4c 4d   ........ ..HIJKLM
0030  4e 4f 50 51 52 53 54 55  56 57 58 59 5a 5b 5c 5d   NOPQRSTU VWXYZ[\]
0040  5e 5f 60 61 62 63 64 65  66 67 68 69 6a 6b 6c 6d   ^_`abcde fghijklm
```

File: "/tmp/etherXXXXYwKOsN" 6045 Bytes 00:00:17   P: 58 D: 58 M: 0 Drops: 0

Fig.11.3 ICMP Reply

CONCLUSION:

EXPERIMENT – 12

PERFORMANCE ANALYSIS OF TCP PROTOCOL USING NS-2

OBJECTIVE:

(i) Introduction of Network simulator ns-2
(ii) Understanding of network simulator ns-2
(iii) Impact of link error rate on TCP performance (throughput, cwnd).
(iv) Plot the graph for TCP cwnd **.**

THEORY:

ns-2 is a network simulator for simulating computer networks and protocols. It was chosen because it is free, open source and is used commonly by the research community for evaluating work in computer networks. ns-2 support simulation of routing protocols over both wired as well as wireless networks, and also supports links from Ethernet to satellite links. It also includes various models to represent the characteristics and behavioral aspects of different network devices and components. A user has complete access over the entire software setup and can attempt any kind of modifications.

ns-2 is an event driven network simulator, written in c++ language with an OTcl interpreter as a front end. The simulator supports a class hierarchy in c++ and a similar class hierarchy within the OTcl interpreter. The two hierarchies are closely related to each other, from the user's perspective. There is one to one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy.

ns-2 uses two languages because simulator has two different kind of things to do. On one hand detailed simulations of protocols require a system programming language, which can efficiently manipulate packet headers, and implement algorithms that run over large data sets. For this task run time speed is important and turn-around time is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In this case, interaction time (change the model and re-run) is more important. That's why, ns satisfy both of this needs with two languages, c++ and OTcl.

C++ is fast to run but slower to change as large amount of program need to change, making it more suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly and interactively, making it ideal for simulation configuration.

SIMULATION WITH NS :

ns is invoked with scripts which defines the simulation. The scripting language creates the simulation objects with appropriate links between them. Often analytical or evaluation experiments involve slight variation in configurations and parameters. OTcl is easy to change and therefore it is ideal to use for simulation configuration. Network Simulator includes The

Network Animator, which is helping Network Simulator to support a graphical environment. The Network Animator uses the parameters from the script and writes it into .namfile and then according to that it will create a topology identical to the script. So we can see graphically how the packets are forwarded and traffic is handled in the network, so this can give the more realistic implementation and observation.xgraph inbuilt utility of ns can be used to plot graphs. Based on individual requirements and preferences, plots of various network parameters can be generated.

An overview of how to run simulation in ns is shown in Fig-12.1

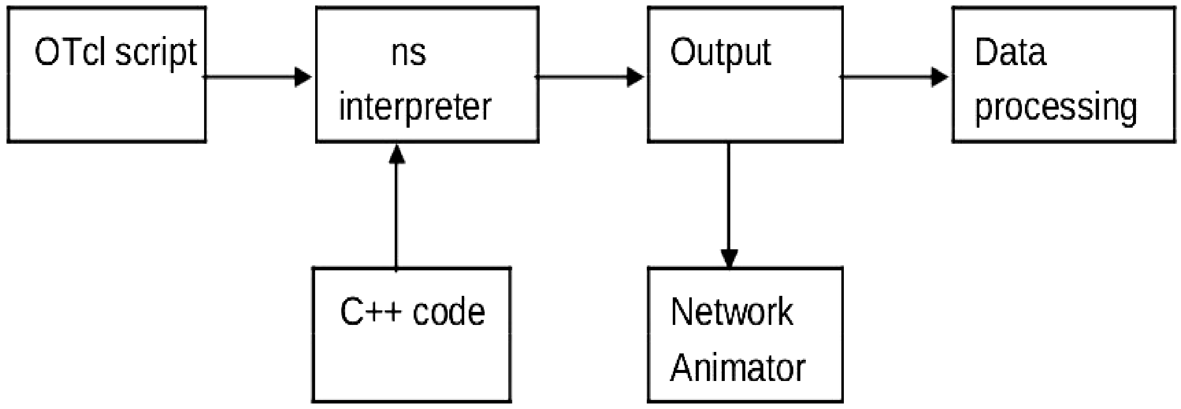


Fig. 12.1 Overview of simulation

Network Simulator uses tcl script for simulation. Syntax of this script can be understood by referring the manual of ns. We can understand Network Simulator’s simulation process and tcl script format and The Network Animator graphical representation with the help of the following script. Script is showing simulation of simple TCP Tahoe connection. Its topology is shown below.

INSTALLATION PROCEDURE:

1. Copy ns-allinone-2.30.tar.gz file to /root
2. Type tar xvzf ns-allinone-2.30.tar.gz and press enter
3. Directory of the name ns-allinone-2.30 will be created
4. After entering in this directory, type ./install and press enter
5. After the completion of the installation set the library paths in .bash_profile file in /root
6. After the completion of the installation goto ns-2.30 directory which is in the ns-allinone-2.30 directory
7. In this directory goto ns-2.30/tcl directory
8. In this directory goto ex directory. .tcl files will be available there.
9. To run .tcl files type ns filename.tcl
10. To view source code type vi filename.tcl

TCL Script:

```
set ns [new Simulator]
$ns color 0 blue
$ns color 1 red
$ns color 2 green

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

set f [open out1.tr w]
ns trace-all $f
set nf [open out1.nam w]
$ns namtrace-all $nf
$ns trace-all $f
set tcptrace [open tracelink w]

$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n4 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail

$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n4 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5
$ns queue-limit $n2 $n3 50

# ErrorModel
set em [new ErrorModel]
$em unit pkt
$em set rate_ 0.00
# Introducing PER on n0-r0 link
$em ranvar [new RandomVariable/Uniform]
$em drop-target [new Agent/Null]
$ns lossmodel $em $n2 $n3
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
set sink1 [new Agent/TCPSink]
$ns attach-agent $n0 $tcp1
```

```
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 0.1 "$ftp1 start"

set tcp [new Agent/TCP]
$tcp set class_ 2
$tcp set packetSize_ 500 # Packet size
$ns add-agent-trace $tcp trname $tcptrace
$ns monitor-agent-trace $tcp
$tcp tracevar rtt\_
$tcp tracevar cwnd\_
$tcp tracevar ssthresh\_
$tcp tracevar nrexmitpack\_
$tcp tracevar dupacks\_
$tcp tracevar ack\_
$tcp tracevar seqno\_
$tcp tracevar srtt\_
$tcp tracevar rttvar\_
$tcp tracevar backoff\_
$tcp tracevar maxseq_
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $tcp
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.2 "$ftp start"
$ns at 0.0001 "$tcp set ssthresh_ 50"

#$ns at 1.35 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

$ns at 20.0 "finish"
proc finish {} {
global ns f nf tcptrace
$ns flush-trace
close $f
close $nf
close $tcptrace
puts "running nam..."
exec nam out1.nam &
exec awk {
{
```

```
if ($6 == "cwnd_")
{
print $1,$7
}
}
} tracelink > cwnd

exec xgraph cwnd
}
$ns run
```

PROCEDURE:

1. Form a topology of nodes s0,no and r0 with link between so and r0 with data rate 100mbps with 1ms delay and between no and ro with 40 mbps and 20ms delay.
2. Vary values of delay and observe the changes in network utilisation and srtt .
3. Now, vary values of ssthresh and note down the duration of slow start mode in TCP.
4. change error rate and note how it affects network utilisation and cwnd.
5. Now, queue size is fixed and cwnd is varied.observe network utilisation.
6. Use these formulas

   Throughput= No. of bits/total time

   Utilization=throughput/bottleneck BW

   where, bottleneck BW is the max data rate defined in a particular link.

OUTPUT:

The output of TCL script are :

1) Tracefile

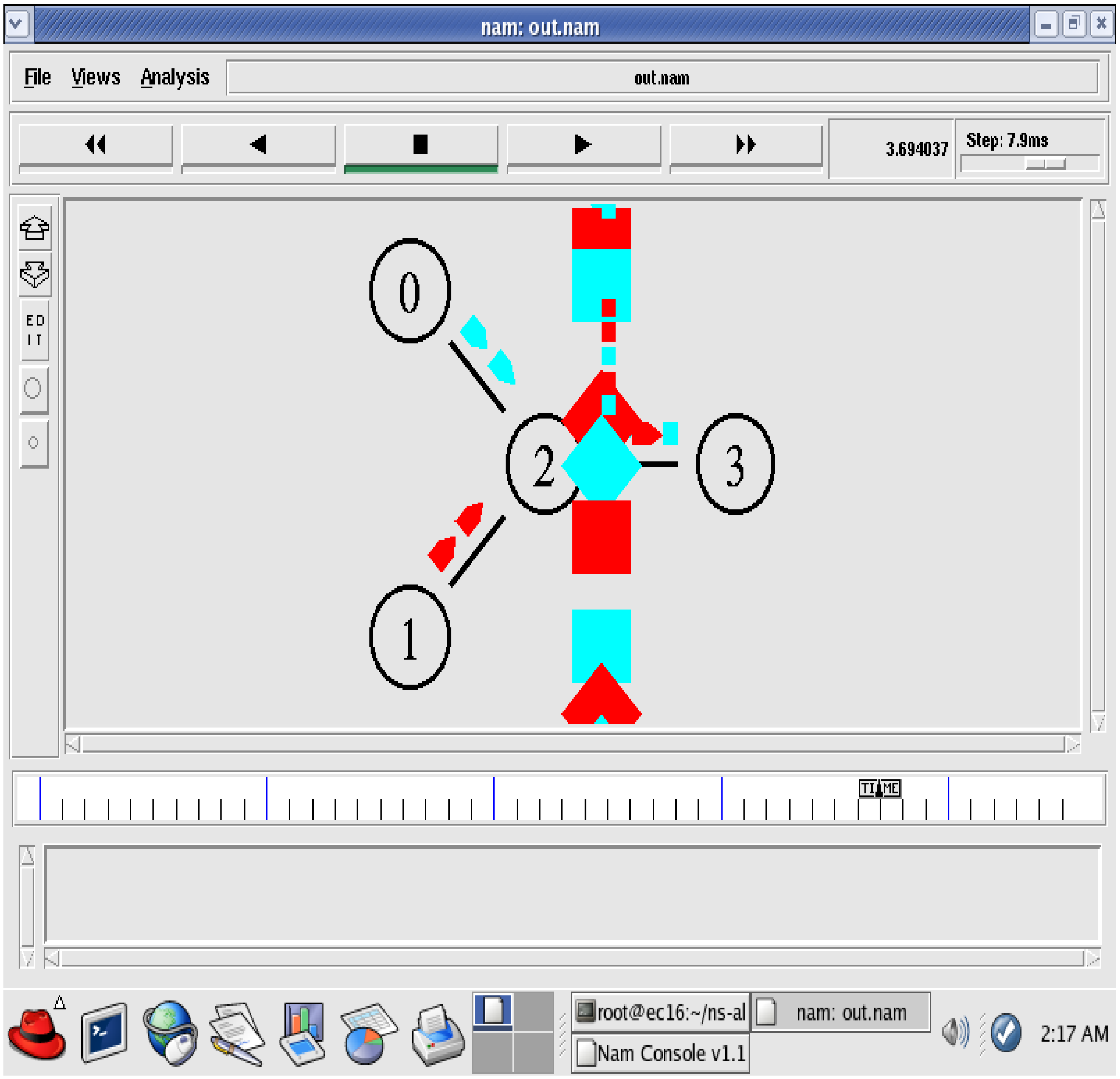
2) Name file

1) TraceFile :

```
+ 0.1 0 2 tcp 40 ------- 2 0.0 3.0 0 0
- 0.1 0 2 tcp 40 ------- 2 0.0 3.0 0 0
r 0.102064 0 2 tcp 40 ------- 2 0.0 3.0 0 0
+ 0.102064 2 3 tcp 40 ------- 2 0.0 3.0 0 0
- 0.102064 2 3 tcp 40 ------- 2 0.0 3.0 0 0
r 0.112277 2 3 tcp 40 ------- 2 0.0 3.0 0 0
+ 0.112277 3 2 ack 40 ------- 2 3.0 0.0 0 1
- 0.112277 3 2 ack 40 ------- 2 3.0 0.0 0 1
r 0.122491 3 2 ack 40 ------- 2 3.0 0.0 0 1
+ 0.122491 2 0 ack 40 ------- 2 3.0 0.0 0 1
- 0.122491 2 0 ack 40 ------- 2 3.0 0.0 0 1
r 0.124555 2 0 ack 40 ------- 2 3.0 0.0 0 1
+ 0.124555 0 2 tcp 1040 ------- 2 0.0 3.0 1 2
- 0.124555 0 2 tcp 1040 ------- 2 0.0 3.0 1 2
+ 0.124555 0 2 tcp 1040 ------- 2 0.0 3.0 2 3
- 0.126219 0 2 tcp 1040 ------- 2 0.0 3.0 2 3
```

```
r 0.128219 0 2 tcp 1040 ------- 2 0.0 3.0 1 2
+ 0.128219 2 3 tcp 1040 ------- 2 0.0 3.0 1 2
- 0.128219 2 3 tcp 1040 ------- 2 0.0 3.0 1 2
r 0.129883 0 2 tcp 1040 ------- 2 0.0 3.0 2 3
+ 0.129883 2 3 tcp 1040 ------- 2 0.0 3.0 2 3
- 0.133765 2 3 tcp 1040 ------- 2 0.0 3.0 2 3
r 0.143765 2 3 tcp 1040 ------- 2 0.0 3.0 1 2
```

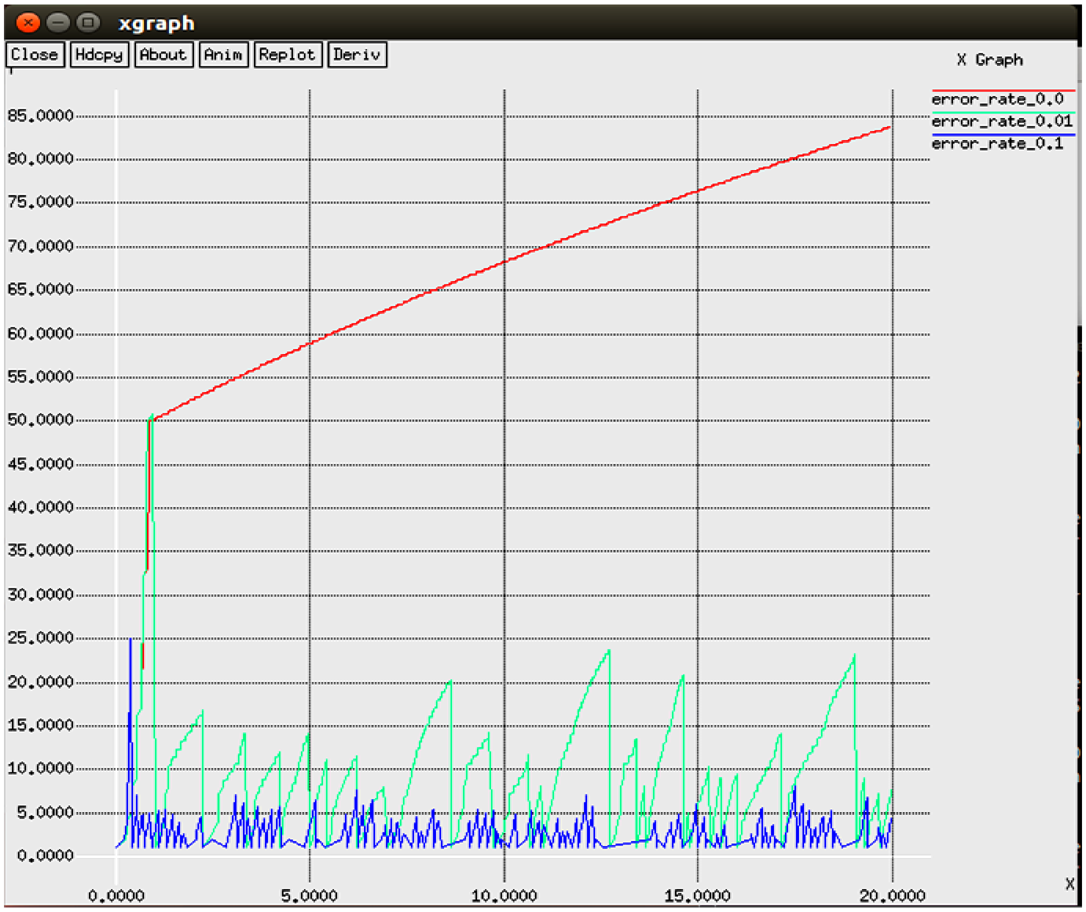
2) Nam File:



OBSERVATION TABLE:

Error Rate	Throughput (bps)
0.00	
0.01	
0.1	

Result:



CONCLUSION:

PART II

DATASHEET

NETGEAR® PROSAFE®

ProSafe® 24-port 10/100 Mbps Fast Ethernet Switch **JFS524**

Data Sheet

Affordable Network Capacity

Need a quantum leap in office network capability? NETGEAR delivers a cost-effective solution – the JFS524. This unmanaged 10/100 Mbps Fast Ethernet switch provides expanded connectivity for small office networks so users no longer have to compete with each other for bandwidth. All of the essential features – automatic speed and full/half duplex sensing, 200 Mbps throughput per port, Auto Uplink™ – are included, but at an economical price that a value-conscious business owner can't ignore.

Frugal

Twenty-four ports of speed and capacity at a modest price, with no compromising on quality – that's what you can expect from NETGEAR's hardworking JFS524.

Skillful

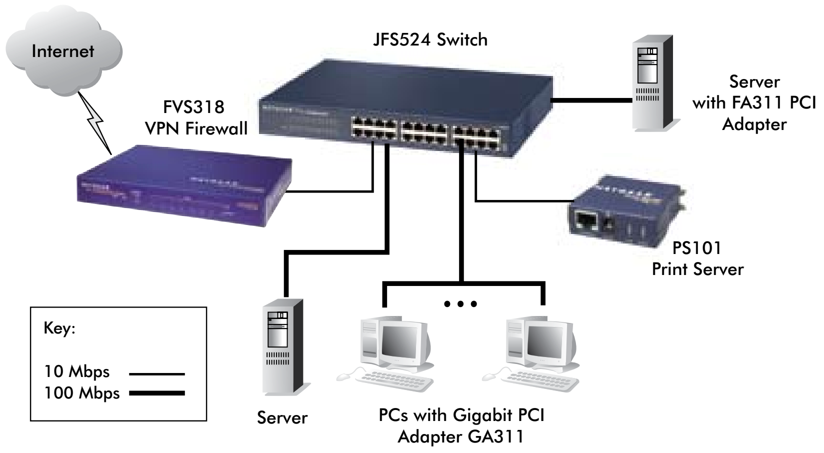
Switched 10/100 Mbps ports provide private bandwidth for PCs, servers, or hubs, and support 200 Mbps of throughput. Each port delivers 200 Mbps of network speed in full-duplex mode.

Alert

Ever-vigilant, the JFS524 begins to operate as soon as it's powered on – with no configuration required. All 24 ports deliver automatic speed and full/half duplex sensing, plus Auto Uplink™, which adjusts for straight-through or crossover cables to make the right link.

Obliging

This rugged metal unit is conveniently compact for an uncluttered desktop or easy rack mounting. And it lets you easily intermix 10 and 100 Mbps devices within your unmanaged network.



24/7 TECHNICAL SUPPORT*

1-888-NETGEAR (638-4327)
Email: info@NETGEAR.com

NETGEAR PROSAFE LIFETIME WARRANTY*

Technical Specifications

- **Network Ports**
  - 24 10 or 100 Mbps RJ-45 ports
- **Performance (64 byte packets)**
  - Store-and-forward packet switching
  - Forward rate (10 Mbps) 14,800 packet per sec.
  - Forward rate (100 Mbps) 148,000 packet per sec.
  - Latency (10 to 100 Mbps) 20 usec max
  - MAC addresses 4,000
  - Queue buffer KBytes 320
  - Mean time between failures (MTBF): 110,000 hours (~13 years)
- **Physical Specifications**
  - Dimensions (h x w x d): 43 x 330 x 202 mm (1.7 x 13 x 8 in)
  - Weight: 2.0 kg (4.4 lb)
- **AC Power**
  - 15W maximum
  - Auto-sensing 100 to 240V, 50/60 Hz
  - Localized plug for North America, Japan, UK, Europe, Australia
- **Standards Compliance**
  - IEEE 802.3i 10BASE-T Ethernet
  - IEEE 802.3u 100BASE-TX Fast Ethernet
  - Compatible with all major network software
- **Status LEDs**
  - Unit: Power
  - Per network port: 10 or 100 Mbps operations, Rx/Tx and collision, link, half/full duplex
- **Electromagnetic Emissions**
  - CE mark, commercial
  - FCC Part 15 Class
  - EN 55022 (CISPR 22), Class A
  - VCCI Class A
  - C-Tick
  - MIC
- **Environmental Specifications**
  - Operating temperature: 0° to 40° C
  - Operating humidity: 80% max. relative humidity, non-condensing
- **Safety Agency Approvals**
  - UL listed (UL 1950)/CUL

Warranty

- NETGEAR Lifetime Warranty†

ProSupport™ Service Packs Available

- **OnCall 24x7, Category 1**
  - PMB0331-100 (US)
  - PMB0331 (non-US)
- **XPressHW, Category 1**
  - PRR0331

Package Contents

- ProSafe 24-Port 10/100 Mbps Fast Ethernet Switch (JFS524)
- Power cord
- Rack-mount kit
- User guide
- Warranty card
- Support card

Ordering Information

- North America: JFS524NA
- Europe: JFS524-100EUS
- Asia: JFS524AU
- Japan: JFS524JP

NETGEAR®

350 E. Plumeria Drive
San Jose, CA 95134-1911 USA
1-888-NETGEAR (638-4327)
E-mail: info@NETGEAR.com
www.NETGEAR.com

*Free basic installation support provided for 90 days from date of purchase. Advanced product features and configurations are not included in free basic installation support; optional premium support available.

†Lifetime warranty for product purchased after 05/01/2007. For product purchased before 05/01/2007, warranty is 2 years.

PART III

SESSIONAL QUESTION PAPER

DHARMSINH DESAI UNIVERSITY, NADIAD
FACULTY OF TECHNOLOGY
FIRST SESSIONAL EXAMINATION
SUBJECT: (EC702) DATA & COMPUTER COMMUNICATIONS

Examination	**: B. Tech. (EC) - Semester -VII**	**Seat No.**	**:**
Date	**: 31/07/2019**	**Day**	**: Wednesday**
Time	**: 1.45 pm - 3.00 pm**	**Max. Marks**	**: 36**

INSTRUCTIONS:
1. Figures to the right indicate maximum marks for that question.
2. The symbols used carry their usual meanings.
3. Assume suitable data, if required & mention them clearly.
4. Draw neat sketches wherever necessary.

Q.1 Do as directed. **[6]**

(A) Choose the most appropriate alternate (s). (2)

(i) Which of the following is FALSE for slotted ALOHA ? T_{slot} represents the slot interval and T_{frame} represents the frame period.
(a) $T_{slot} > T_{frame}$ (b) $T_{slot} < T_{frame}$ (c) None of the Above (d) Can't say

(ii) In pure ALOHA, the vulnerable time is ______ the frame transmission time.
(a) two times (b) three times (c) the same as (d) none of the above

(B) List-out service primitive essentially used for connection oriented services. (1.5)

(C) “Collision Detection is an Analog Process”. Justify. (1)

(D) “The static channel access techniques are not suitable in Computer Networks”. Justify. (1.5)

(E) In the case of *go-back-n* protocol with MAXSEQ = 7, the sender needs _______ buffers. (1)
(a) 0 (b) 1 (c) 7 (d) 8

(F) In order to ensure reliable communication, data link layer implementations need to use at least (1)
(a) a negative ACK (b) a positive ACK (c) (a) and (b) (d) none of above

(G) “Piggybacking uses the bandwidth efficiently.” Justify. (2)

(H) The physical layer sends data as raw bits and the data link layer uses group of bits called ‘frames’. Why this creation of frames required? (2)

Q.2 Attempt ANY TWO **[12]**

(A) Explain the concept of Peer to Peer Communication giving neat sketch of Network Architecture. (6)

(B) What is 'Vulnerable Period' ? How it is related with the efficiency of Multiple Access Protocol ? Explain the same for various ALOHA techniques. (6)

(C) Explain ANY THREE design issues with reference to Computer Communication in general. Also mention concerned OSI layer and the mechanism adopted for mitigating the same. (6)

Q.3 (A) (6)
i) What is the range of sequence numbers used in stop and wait protocol? Justify your answer.
ii) What causes generation of duplicate frames? How such frames are handled at receiver side?
iii) *r* represents a variable of the type frame. Data link layer receiver obtains frame from the lower layer using *from_physicallayer(&r)*. Write corresponding function call for handing over the packet to network layer.
iv) Inspite of having error free channel, why acks are used in stop and wait protocol for noisy environment?

(B) Discuss the role of retransmission timer. (3)

(C) State the various outcome of *wait for event(&event); in go-back-n protocol allows the data flow in both direction.* (3)

OR

Q.3 (A) Write short note on framing methods (any two). (6)

(B) The following Stop and Wait Protocol allows unidirectional data flow over a noisy channel with constant round trip time. (6)

```
void sender3(void) {
        seq nr next frame to send;
        frame s;
        packet buffer;
        event type event;
        next frame to send = 1;
        while (true) {
                s.info = buffer;
                s.seq = next frame to send;
                to physical layer(&s);
                wait for event(&event);
                start timer( );
                        if (event == frame arrival) {
                                from physical layer(&s);
                                stop timer( );
                                from network layer(&buffer);
                                inc(next frame to send);
                        }
        } }
```

Point out the missing or incorrectly placed statements and discuss their effects in the above version of the protocol.

DHARMSINH DESAI UNIVERSITY, NADIAD
FACULTY OF TECHNOLOGY
B.TECH. SEMESTER VII [EC]
SUBJECT: (EC702) DATA & COMPUTER COMMUNICATION

Examination	**: Second Sessional**	**Seat No.**	**: ________________**
Date	**: 05-09-2019**	**Day**	**: Thursday**
Time	**: 1.45 pm – 3.00 pm**	**Max. Marks**	**: 36**

INSTRUCTIONS:
1. Figures to the right indicate maximum marks for that question.
2. The symbols used carry their usual meanings.
3. Assume suitable data, if required & mention them clearly.
4. Draw neat sketches wherever necessary.

Q.1 (A) Do as directed. **[06]**

(1) State true or false with reason(s).: "In a network, the size of all packets required to be sent is fairly short then using *datagram subnet* is not beneficial." **(2)**

(2) If data link sender is allowed to use 4 bit sequence number, what should be the number of buffers required at receiver (a) with go back n (b) with selective repeat protocol? **(2)**

(3) What is congestion? What are the causes of congestion in network? **(2)**

(B) Select the most appropriate option. **[06]**

(1) Which of the followings is NOT TRUE for token bus network ? **(1)**
(a) station can not transmit multiple frames within Toke holding Time.
(b) station can have one frame whose period is higher than Token holding time.
(c) station may not transmit a single frame, successfully within a Token Holding Time.
(d) station may not be able to join the logical ring despite of having physical connectivity.

(2) For an IEEE 802.3 frame with 50 bytes of payload, the size of padding field is **(1)**
(a) 46 bytes (b) 0 bytes (c) 14 bytes (d) 4 Bytes

(3) 10base2 cabling standard uses _______ connector. **(1)**
(a) Vampire Tap (b) BNC (c) RJ45 (d) Opto-coupler

(4) Matched Load Impedance of physical medium in IEEE 802.4 is **(1)**
(a) 50 Ohm (b) 75 ohm (c) 100 ohm (d) 25 ohm

(5) With increase in BDP, the transmission efficiency of Ethernet ____________. **(1)**
(Assume all other parameters remain unchanged.)
(a) increases (b) decreases (c) remains unchanged (d) can't say

(6) For a CSMA/CD network running at 1 Gbps over a 1 km long cable with no repeaters and 200,000 km/sec signal speed, the minimum frame length requirement is **(1)**
(a) 64 Bytes (b) 2048 Bytes (c) 1024 Bytes (d) 512 Bytes

Q.2 Attempt *Any Two* from the following questions. **[12]**

(A) Answer the following question with respected to data link layer protocols, Where MAX_SEQ=7? **(2)**
(i) A go back n sender's sequence number for next frame to send and acknowledgement expected is 3 and 6 respectively. What will be response of sender, when it receives the acknowledgement number is 1? **(2)** **(2)**
(ii) What is the sender and receiver window size in selective repeated protocol?
(iii) What is the main benefit of using negative acknowledgement?

(B) What do you mean by traffic shaping? Explain the algorithms used at network layer for traffic shaping. **(6)**

(C) (i) What are the disadvantages of distance vector routing algorithm? How it is resolved? **[(3)**
(ii) Discus the techniques used to learn about neighbors and calculation of cost in link state routing. **(3)**

Q.3 (A) Explain various persistent techniques of CSMA ? Compare them in terms of transmission efficiency and delay. In which situation, CSMA fails in providing significant improvement over ALOHA ? **(6)**

(B) Discuss token passing mechanism in IEEE 802.4. List out atleast 3 requirements for maintaining successor address by each IEEE 802.4 node. **(6)**

OR

Q.3 (A) Explain the IEEE 802.4 mechanism to deal with the problems caused due to failed station(s). Failed station can be a Token Holder or Non-Token Holder. **(6)**

(B) Draw IEEE 802.3 frame structure. Briefly explain the significance of the following fields. **(6)**
(1) Preamble (2) Sender Address (3) Checksum (d) Length

DHARMSINH DESAI UNIVERSITY, NADIAD
FACULTY OF TECHNOLOGY
B.TECH. (EC) SEMESTER VII [EC702 – DATA & COMPUTER COMMUNICATION]

Examination	**: Third Sessional**	**Seat No.**	**: ___________**
Date	**: 10/10/2019**	**Day**	**: Thursday**
Time	**: 01:45 pm to 03:00 pm**	**Max. Marks**	**: 36**

INSTRUCTIONS:
1. Figures to the right indicate maximum marks for that question.
2. The symbols used carry their usual meanings.
3. Assume suitable data, if required & mention them clearly.
4. Draw neat sketches wherever necessary.

Q.1 **[A] Do as directed.** **[3]**
1) What is the minimum and maximum sizer of TCP header?
2) What are the option fields covered in TCP Header?
3) Which combination of flags of TCP header will be used while setting up a connection?

[B] State True/False with reason(s). **[3]**
1) Congestion feeds itself.
2) Providing more buffer memory at router always helps in avoiding congestion.
3) When TCP client application attempts to connect to a server, only destination IP is enough.

[C] Select the most appropriate option. **[2]**
1) Which of the following IPv4 header fields needs to be modified by router before forwarding IP packet?
(a) Sender IP Address (b) Checksum (c) TOS bits (d) None of the above
2) Which of the following is invalid for an IPv4 packet ?
(a) Header Length = 16 bytes (b) Header Length = 30 bytes (c) TTL = 0 (d) All of the above

[D] Answer in brief **[4]**
1) Which of the following representations of IPv4 addresses are invalid ? Give reasons.
(a) 192.168.011.26 (b) 192.168.00001011.26 (c) 192.168.11.265 (d) 192.168. .26
2) One of the IP address from Class C network is 192.168.11.26. Find **(a)** network address **(b)** network broadcast address **(c)** another IP address from the same network **(d)** loopback address.

Q.2 **Attempt Any TWO.** **[12]**
1) Answer the following with reference to TCP. [6]
   I) What is the impact of followings on congestion window?
      a) Acknowledgement arrival during slow start
      b) Retransmission timeout
      c) Three duplicate ACKs received.
   II) What is the function of persistence timer and keep alive timer?
2) What is silly window syndrome? Explain Nagle's Algorithm and Clark's solution to optimize the transmission performance at TCP. [6]
3) Explain the client as well as server TCP state transition from Established state to Closed state. [6]

Q-3 **Answer the following questions.** **[12]**
1) (a) Explain Classful IP Addressing Scheme and also mention limitations of the scheme. [6]
   (b) List out Private and Public IP addresses from each class.
2) (a) Why IP Fragmentation is required? [6]
   (b) Which IP fragmentation technique is deployed over Internet?
   (c) Compare various IP framentation techniques in terms of end-to-end delay, fragmentation overhead and reliability.

OR **[12]**

Q-3 **Answer the following questions.**
1) Explain IP tunnelling in brief. Which is the most essential network element required for IP tunnelling? [3]
2) Explain the significance of following IPv4 header fields. [3]
   (a) Total Length (b) Sender IP Address (c) Protocol
3) (a) What is the responsibility of ICMP? [6]
   (b) How ICMP services are utilized for network debugging purpose – explain this with reference to execution of PING command.