

**College Logo**

**Department of Electronics and Communication Engineering  
Computer Networks and Protocols  
( 21EC63)**

### **Course Outcomes (As per Bloom's revised taxonomy)**

After going through this course the student will be able to:

1. Acquire the knowledge of network architecture, topologies and security issues.
2. Design a network for given configuration by assigning IP addresses.
3. Analyze various aspects involved in network control and traffic management.
4. Analyze the performance of various scheduling algorithms.

**College logo**

**Department of Electronics and Communication  
Engineering**

**Laboratory Certificate**

This is to certify that Mr. / Ms \_\_\_\_\_  
\_\_\_\_\_ has satisfactorily completed the course of  
Experiments in Practical \_\_\_\_\_ prescribed  
by the Department during the year \_\_\_\_\_

Name of the Candidate: \_\_\_\_\_

USN No.: \_\_\_\_\_ Semester: \_\_\_\_\_

Signature of the staff in-charge

Head of the Department

Date:    /    / 2024

**RV COLLEGE OF ENGINEERING, BENGALURU – 560059**  
*(Autonomous Institution affiliated to VTU, Belagavi)*  
**Department of Electronics and Communication Engineering**  
**Computer Networks and Protocols (21EC63)**  
**SCHEME OF CONDUCT AND EVALUATION**

CLASS: VI SEMESTER  
YEAR: 2024  
SEE: 3 Hrs

CIE MARKS: (Max.): 50  
SEE MARKS: (Max): 50

<b>Expt. No.</b>	<b>Title</b>	<b>Course Outcome</b>	<b>Duration in Hrs</b>	<b>Max. Marks</b>	<b>Marks obtained</b>
1	a) Implement Bit stuffing Algorithm b) Character stuffing algorithms and c) Cyclic Redundancy Check codes for error detection using C programs.	CO2	2hrs	10	
2	Implement Encryption and Decryption algorithms using C program	CO1	2hrs	10	
3	Implement following Minimum Spanning Tree algorithms using C program i) Kruskal's Algorithm ii) Prim's Algorithms	CO1	2hrs	10	
4	Simulate & Analyze CSMA/CD and CSMA/CA Protocols using QualNet	CO2	2hrs	10	
5	Implement STOP and WAIT protocol using socket programming concept in C Program.	CO3	2hrs	10	
6	Test and verify Network configurations using Packet Tracer.	CO4	2hrs	10	
7	Configure Inter VLAN network using Packet Tracer	CO3	2hrs	10	
8	Implement RSA algorithm using C program	CO4	2hrs	10	
9	Configure and test a given network using Packet Tracer	CO4	2hrs	10	
10	Design an Ethernet network comprising of 25 nodes and calculate Packet delivery ratio given the packet size to be 1024 bytes, consider the following application layer protocols a ) FTP b) CBR	CO4	2hrs	10	
<b>Open ended Experiments</b>					
11	i) Simulate and Compare following Routing Protocols using Cisco packet tracer/Qualnet a) Open-Shortest Path First (OSPF) b) Routing Information Protocol (RIP)	CO4	2hrs		
<b>Grand Total</b>					
<b>Total</b>				<b>40</b>	
<b>Test</b>				<b>10</b>	
<b>Total Lab CIE Marks</b>				<b>50</b>	

# RV COLLEGE OF ENGINEERING, BENGALURU – 560059

(Autonomous Institution affiliated to VTU, Belagavi)

## Department of Electronics and Communication Engineering Computer Networks and Protocols (18EC62)

### *INDEX*

CLASS: VI SEMESTER

YEAR: 2023

SEE: 3 Hrs

CIE MARKS: (Max.): 50

SEE MARKS: (Max): 50

Exp No.	Title	Date	Page No.	Marks obtained	Staff Signature
1	d) Implement Bit stuffing Algorithm e) Character stuffing algorithms and f) Cyclic Redundancy Check codes for error detection using C programs.	/ /2024.			
2	Implement Encryption and Decryption algorithms using C program	/ /2024			
3	Implement following Minimum Spanning Tree algorithms using C program iii) Kruskal's Algorithm iv) Prim's Algorithms	/ /2024			
4	Simulate & Analyze CSMA/CD and CSMA/CA Protocols using QualNet	/ /2024			
5	Implement STOP and WAIT protocol using socket programming concept in C Program.	/ /2024			
6	Test and verify Network configurations using Packet Tracer.	/ /2024			
7	Configure Inter VLAN network using Packet Tracer	/ /2024			
8	Implement RSA algorithm using C program	/ /2024			
9	Configure and test a given network using Packet Tracer	/ /2024			
10	Design an Ethernet network comprising of 25 nodes and calculate Packet delivery ratio given the packet size to be 1024 bytes, consider the following application layer protocols a ) FTP b) CBR	/ /2024			
Open ended Experiments					

11	i) Simulate and Compare following Routing Protocols using QualNet a) Open-Shortest Path First (OSPF) b) Routing Information Protocol (RIP) .	/ /2024			
----	---	---------	--	--	--

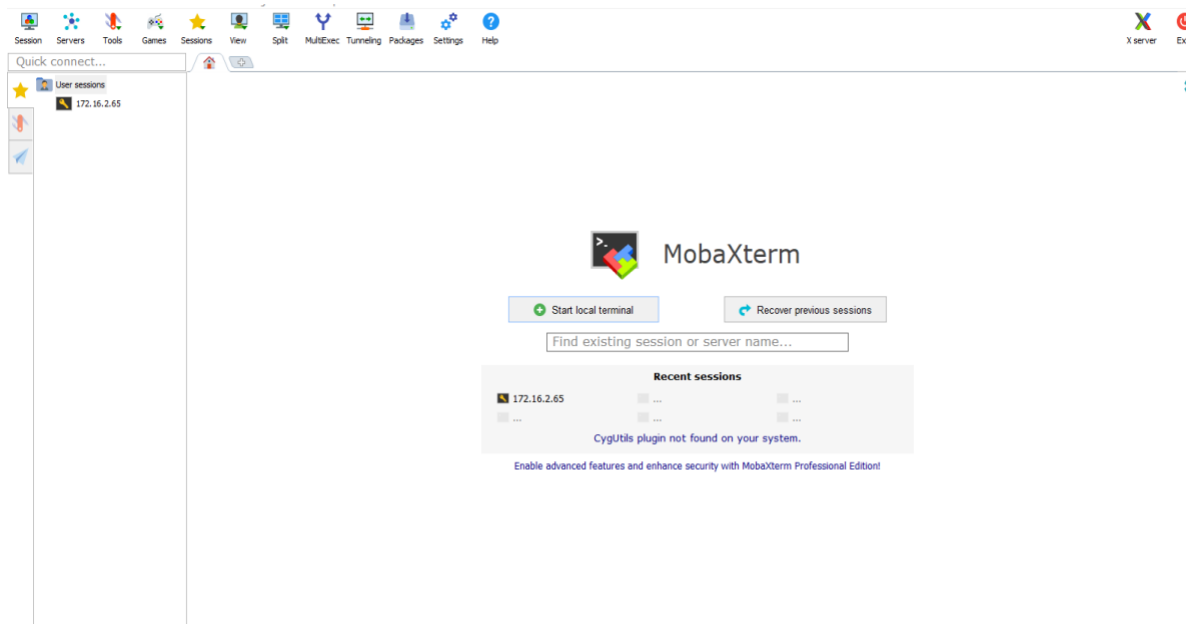
Sl. No	Criteria	Excellent	Good	Average	Max Score
<b>Data sheet</b>					
A	Problem statement	9-10	6-8	1-5	<b>10</b>
B	Design & specifications	9-10	6-8	1-5	<b>10</b>
C	Expected output	9-10	6-8	1-5	<b>10</b>
<b>Record</b>					
D	Simulation/ Conduction of the experiment	14-15	11-13	1-10	<b>15</b>
E	Analysis of the result.	14-15	11-13	1-10	<b>15</b>
<b>Viva</b>					<b>40</b>
<b>Total</b>					<b>100</b>
<b>Scale down to 10 marks</b>					



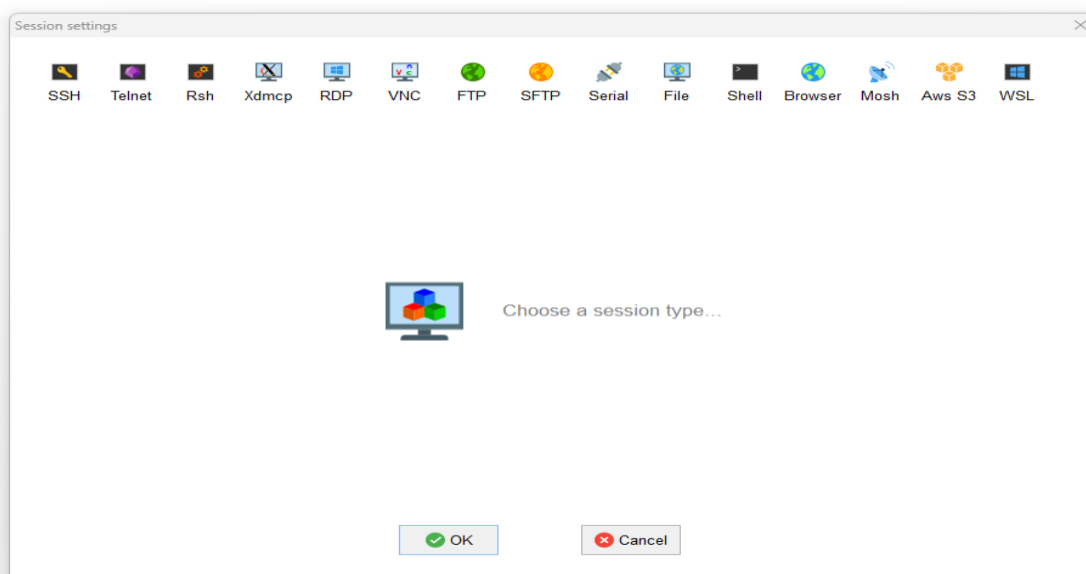
## LOGIN PROCEDURE

**Students are instructed to use Server Terminal to execute the experiments :  
Following is the Login Procedure to connect to the Remote Server from your local system Terminal :**

**Step 1 : Click on Mobaxterm on Desktop**

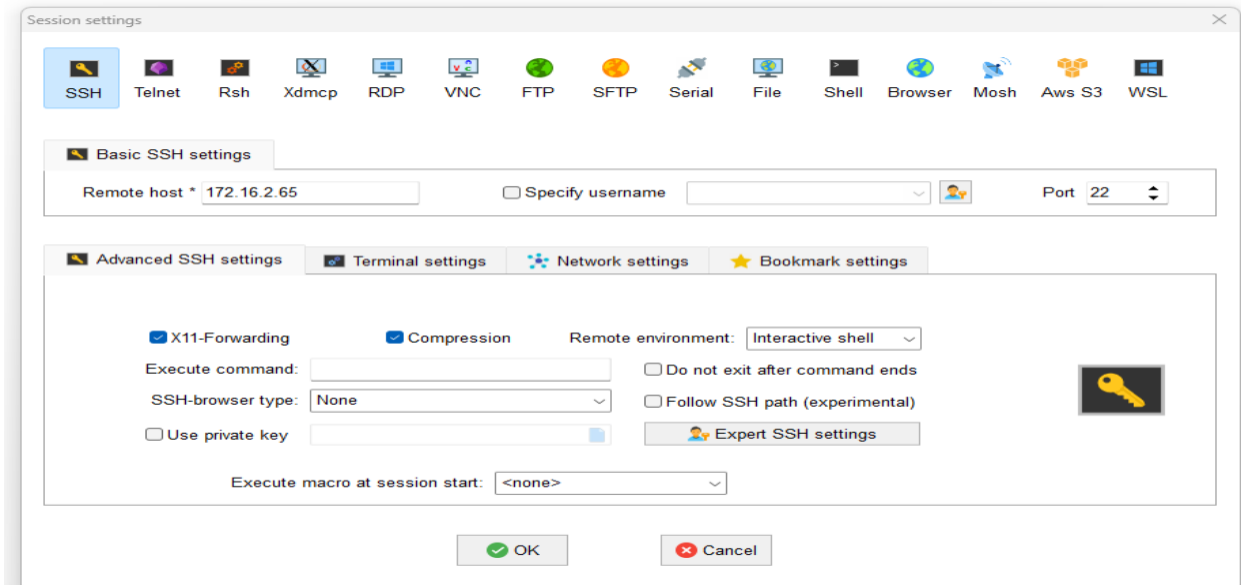


**Step 2: click on Session on top left side**

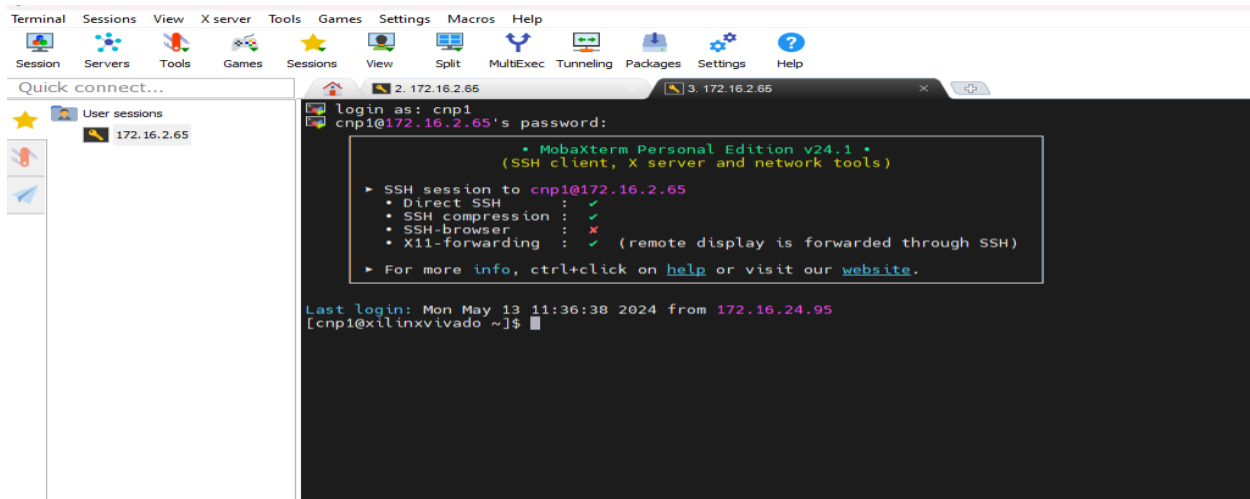


**Step 3: Click on SSH and type on Remote desktop as 172.16.2.65  
On SSH browser type select none then click ok. Click on NO to password**





Step 4: type login as cnp1 and press enter , provide password as rvcnp1 , then enter.



Step 5:

Type cd CNP

Type mkdir batch\_name

Type gedit exp1.c

Type the programme and save

For compilation again come to login and type gcc exp1.c

If there are no error type ./a.out

Enter input as per program requirement and observe expected output.

## EXPERIMENT 1

## BIT STUFFING AND CHARACTER STUFFING ALGORITHMS AND CYCLIC REDUNDANCY CHECK CODES FOR ERROR DETECTION

- Objective:** (a) To simulate bit stuffing and character stuffing algorithms for HDLC frame  
Using C program  
(b) To simulate Cyclic Redundancy Check codes for Error detection using C Program.

**Theory:**

- (a) The High Level Data Link Control HDLC protocol is a general purpose protocol which operates at the data link layer of the OSI reference model. The protocol uses the services of a physical layer, and provides either a best effort or reliable communications path between the transmitter and receiver (i.e. with acknowledged data transfer). The type of service provided depends upon the HDLC mode which is used.

Each piece of data is encapsulated in an HDLC frame by adding a trailer and a header. The header contains an HDLC address and an HDLC control field. The trailer is found at the end of the frame, and contains a Cyclic Redundancy Check (CRC) which detects any errors which may occur during transmission. The frames are separated by HDLC flag sequences which are transmitted between each frame and whenever there is no data to be transmitted.

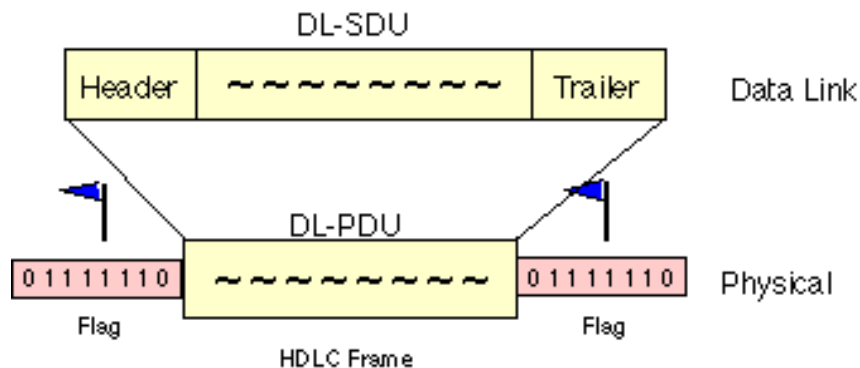


Fig: 1.1 HDLC Frame

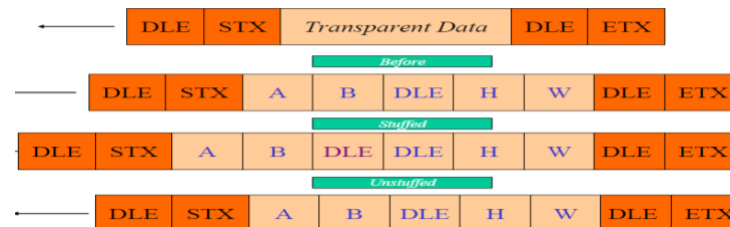
In data transmission and telecommunication, **bit-stuffing** is the insertion of non-information bits into data. Stuffed bits should not be confused with overhead bits.

Bit stuffing is used for various purposes, such as for bringing bit streams that do not necessarily have the same or rationally related bit rates up to a common rate, or to fill buffers or frames. The location of the stuffing bits is communicated to the receiving end of the data link, where these extra bits are removed to return the bit streams to their original bit rates or form. Bit stuffing may

be used to synchronize several channels before multiplexing or to rate-match two single channels to each other.

This is done to create additional signal transitions to ensure reliable reception or to escape special reserved code words such as frame sync sequences when the data happens to contain them.

Applications include Controller Area Network, HDLC, and Universal Serial Bus.



**Fig: 1.2 Byte Stuffing**

#### ALGORITHM:

##### Algorithm for Bit–Stuffing

1. Start
2. Initialize the array for transmitted stream with the special bit pattern 0111 1110 which indicates the beginning of the frame.
3. Get the bit stream to be transmitted in to the array.
4. Check for five consecutive ones and if they occur, stuff a bit 0
5. Display the data transmitted as it appears on the data line after appending 0111 1110 at the end
6. For de–stuffing, copy the transmitted data to another array after detecting the stuffed bits
7. Display the received bit stream
8. Stop

#### PROGRAM:

```
/*Program for BITSTUFFING and DESTUFFING*/

#include<stdio.h>
#include<string.h>
int main()
{
    char ch, array[50]="01111110", read_array[50];
    int count=0,i=8,j,k;
    printf("Enter data to be transmitted:");

do
    {
        scanf ("%c",&ch);
        if (ch=='\n')
```

```

        break;
    if(ch=='1')
        count++;
    else
        count=0;
    array[i++]=ch;
    if(count==5)
    {
        array[i++]='0';
        count=0;
    }
} while(ch!='\n');

strcat(array,"01111110"); /* FLAG pattern for START and END of HDLC
frame format*/
printf("\n Transmitted bit stream(After stuffing) at the transmitter
side is:%s", array);

/*DESTUFFING*/
j=strlen(array);
count=0;
k=0;
for(i=8;i<j-8;i++)
{
    if(array[i]=='1')
        count++;
    else
        count=0;
        read_array[k]=array[i];
        k++;
        if(count==5 && array[i+1]=='0')
        {
            i++;
            count=0;
        }
}
read_array[k]='\0';
printf("\n Destuffed data at the receiver is :");
for(i=0;i<k;i++)
    printf("%c",read_array[i]);
return 0;
}

```

#### ALGORITHM:

##### Algorithm for Character stuffing :

1. Start
2. Append DLE STX at the beginning of the string
3. Check the data if character is present; if character DLE is present in the string (example DOODLE) insert another DLE in the string (ex: DOODLEDLE)
4. Transmit DLE ETX at the end of the string
5. Display the string
6. Stop

### Algorithm for Character De-stuffing :

1. Start
2. Neglect initial DLE STX
3. If DLE is present in the text, ignore it; if another DLE follows, copy the same to output.
4. Neglect the trailing DLE ETX
5. Stop

/\*Program to implement character/Byte Stuffing and De-stuffing\*/

```
#include<stdio.h>
#include<string.h>
#define DLE 16
#define STX 2
#define ETX 3

int main()
{
    char ch;
    char arr[100]={DLE,STX};
    int i=2,j;
    printf("\n Enter the data stream(CTRL+B->STX,CTRL+C->ETX,CTRL+P->DLE):\n");

    do
    {
        scanf("%c", &ch);
        printf("char is ",ch);
        if(ch=='\n')
            break;

        if(ch==DLE)
        {
            arr[i++]=DLE;
            printf("DLE");
        }
        elseif(ch==2)
            printf("STX");
        elseif(ch==3)
            printf("ETX");
        else
            printf("%c",ch);
        arr[i++]=ch;
    }while(ch!='\n');

    arr[i++]=DLE;
    arr[i++]=ETX;

    printf("\n The stuffed stream is \n");
```

```

for (j=0; j<i; j++)
{
    if (arr[j]==DLE)
        printf("DLE");
    elseif (arr[j]==STX)
        printf("STX");
    elseif (arr[j]==ETX)
        printf("ETX");
    else
        printf("%c", arr[j]);
}

printf("\n The de-stuffed data is \n");
for (j=2; j<i-2; j++)
{
    if (arr[j]==DLE)
    {
        printf("DLE");
        j++;
    }
    elseif (arr[j]==STX)
        printf("STX");
    elseif (arr[j]==ETX)
        printf("ETX");
    else
        printf("%c", arr[j]);
}
return 0;
}

```

(b) A **cyclic redundancy check (CRC)** is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short *check value* attached, based on the remainder of a polynomial division of their contents; on retrieval the calculation is repeated, and corrective action can be taken against presumed data corruption if the check values do not match.

CRCs are so called because the *check* (data verification) value is a *redundancy* (it adds no information to the message) and the algorithm is based on *cyclic* codes. CRCs are popular because they are simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels. Because the check value has a fixed length, the function that generates it is occasionally used as a hash function.

A CRC-enabled device calculates a short, fixed-length binary sequence, known as the *check value* or improperly *the CRC*, for each block of data to be sent or stored and appends it to the data, forming a *codeword*. When a codeword is received or read, the device either compares its check value with one freshly calculated from the data block, or equivalently, performs a CRC on the whole codeword and compares the resulting check value with an expected *residue* constant. If the check values do not match, then the block contains a *data error*. The device may take corrective action, such as rereading the block or requesting that it be sent again. Otherwise, the data is assumed to be error-free (though, with some small probability, it may contain undetected errors; this is the fundamental nature of error-checking).

### ALGORITHM:

#### Algorithm for CRC :

- 1) Multiply Message polynomial  $M(x)$  by highest power of generator polynomial  $G(x)$
- 2) Divide the result by  $G(x)$ , remainder i.e.,  $R(x)$
- 3) Transmit  $M(x) + R(x)$ , at the receive i.e.,  $T(x) = G(x) \times M(x) + R(x)$
- 4) At the receiver end :  $T(x)$  divided by  $G(x)$ , should have remainder 0.

/\* Program for CRC calculation \*/

```
#include<stdio.h>
#include<conio.h>
#define DEGREE 16

int mod2add(int,int);
int getnext(int*,int);
int result[30];

void calc_crc(int length)
{
    int ccitt[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};
    int i=0,pos=0,newpos;

    while(pos<length-DEGREE)
    {
        for(i=pos;i<pos+DEGREE+1;++i)
            result[i]=mod2add(result[i],ccitt[i-pos]);
        newpos=getnext(result,pos);
        if(newpos>pos+1)
            pos=newpos-1;
        ++pos;
    }
}

int getnext(int array[],intpos)
{
    int i=pos;
    while(array[i]==0)
        ++i;
```

```

        return i;
    }

    int mod2add(int x, int y)
    {
        return (x==y?0:1);
    }

    int main()
    {
        int array[30], length, i=0;
        char ch;
        printf("Enter the data (Message) stream:");
        do{
            scanf("%c", &ch);
            if(ch=='\n')
                break;
            array[i++] = ch - '0';
        } while(ch != '\n');

        length = i;

        /*Appending DEGREE number of zeros to the message*/

        for(i=0; i<DEGREE; ++i)
            array[i+length] = 0;
        length += DEGREE;

        for(i=0; i<length; ++i)
            result[i] = array[i]; /*Copy into result Array*/
        calc_crc(length); /*CRC Calculation*/
        printf("\nThe transmitted frame is :");

        for(i=0; i<length-DEGREE; ++i)
            printf("%d", array[i]); /*Message*/
        for(i=length-DEGREE; i<length; ++i)
            printf("%d", result[i]); /*Remainder*/

        /*Decoding*/

        printf("\n Enter the stream for which CRC has to be checked:");

        i=0;
        do{
            scanf("%c", &ch);
            if(ch=='\n')
                break;
            array[i++] = ch - '0';
        } while(ch != '\n');

        length = i;

        for(i=0; i<length; i++)

```



```

        result[i]=array[i];
        calc_crc(length);
        printf("\n Calculated Checksum:");

    for(i=length-DEGREE;i<length;i++)
        printf("%d",result[i]);
    return 0;
}

```

Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			<u>        </u> 10
<b>Staff Signature with Date</b>			

## EXPERIMENT 2

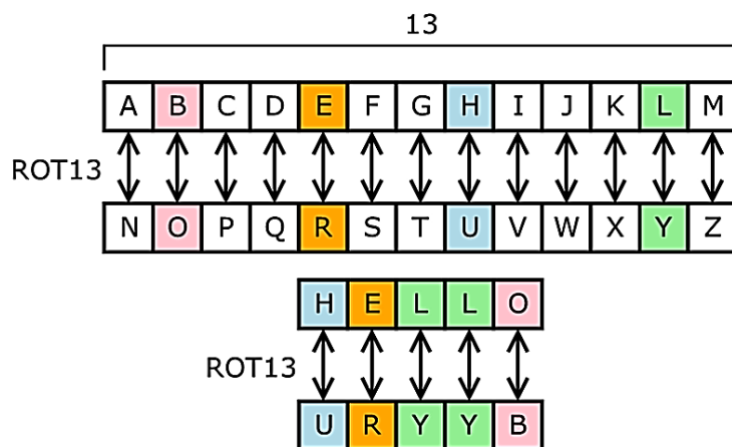
### ENCRYPTION AND DECRYPTION ALGORITHMS

**Objective:** To simulate Substitution method, Transposition Method Encryption algorithms using C Program.

#### Theory:

**Substitution Cipher:** In cryptography, a **substitution cipher** is a method of encryption by which units of plaintext are replaced with cipher text, according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a **simple substitution cipher**; a cipher that operates on larger groups of letters is termed **poly alphabetic**. A **mono alphabetic cipher** uses fixed substitution over the entire message, whereas a **polyalphabetic cipher** uses a number of substitutions at different positions in the message, where a unit from the plaintext is mapped to one of several possibilities in the cipher text and vice versa.



Here is a quick example of the encryption and decryption steps involved with the simple substitution cipher. The text we will encrypt is 'defend the east wall of the castle'.

Keys for the simple substitution cipher usually consist of 26 letters. An example key is:

plain alphabet : abcdefghijklmnopqrstuvwxyz  
cipher alphabet: phqgiumeaylnofdxjkrevstzwb

An example encryption using the above key:

plaintext : defend the east wall of the castle  
 ciphertext: giuifgceiiprctpnn du ceiqpreni

It is easy to see how each character in the plaintext is replaced with the corresponding letter in the cipher alphabet. Decryption is just as easy, by going from the cipher alphabet back to the plain alphabet.

### Algorithm:

Read the plain text and key value .

For i=1 to length (plain text)

- 1 Get the ascii value of ith character.
- 2 Add the key text to that ascii value.
- 3 If the plain text is alphabet then the cipher text should also be alphabet and vice - versa.
- 4 The alphabet series for this case is, a, b, c, d...x, y, z, a, b, c, d... A, B, C, D...X, Y, Z, A, B, C...
- 5 The user should add key value according to the process to be taken, in such a way that the series should continue as explained in the previous point.

**Transposition Cipher :** In [cryptography](#), a transposition cipher is a method of encryption by which the positions held by units of [plaintext](#) (which are commonly characters or groups of characters) are shifted according to a regular system, so that the [ciphertext](#) constitutes a [permutation](#) of the plaintext. That is, the order of the units is changed. Mathematically a [bijective](#) function is used on the characters' positions to encrypt and an [inverse function](#) to decrypt.

In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

In a regular columnar transposition cipher, any spare spaces are filled with nulls; in an irregular columnar transposition cipher, the spaces are left blank. Finally, the message is read off in columns, in the order specified by the keyword. For example, suppose we use the keyword ZEBRAS and the message WE ARE DISCOVERED. FLEE AT ONCE. In a regular columnar transposition, we write this into the grid as:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E Q K J E U
```

Providing five nulls (QKJEU) at the end. The ciphertext is then read off as:

EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE

In the irregular case, the columns are not completed by nulls:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E
```

This results in the following ciphertext:

EVLNA CDTES EAROF ODEEC WIREE

To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length. Then he can write the message out in columns again, then re-order the columns by reforming the key word.

Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.

#### ALGORITHM:

- 1 Read the plain text and key text.
- 2 Initialize the cipher text with a NULL value.
- 3 The plain text's length should be a multiple of key text's length.
- 4 If necessary, add extra characters (i.e. repeat the plain text characters one by one in cyclic order) at the end of the plain text to set its length as a multiple of key text's length.
- 5 Find the order (ascending order) of the key text. For example, if the key text is king then the ascending order is 3,2,4,1.
- 6 Find the position of '1'. Here the position is 4. So append the 4th character and 8th character if present, in the cipher text.
- 7 And the position of '2' is 2. So append the 2nd character and 6th character if present, in the cipher text.
- 8 Proceed till the length of the key text.

#### PROGRAM:

**/\*Encryption by Substitution Method\*/**

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>

void main()
{
    char seq[36]="qwertyuiopasdfghjklzxcvnm1234567890";
    char data[50];
    char encoded[50];
    int i,j,len,present=0;
    printf("\nEnter data:");
    gets(data);
    len=strlen(data);

    for(i=0;i<len;i++)
    {
        if(isupper(data[i]))
            encoded[i]=seq[data[i]]-'A';
        elseif(islower(data[i]))
            encoded[i]=toupper(seq[data[i]-'a']);
        elseif(isdigit(data[i]))
            encoded[i]=seq[data[i]-'0'+26];
        else
            encoded[i]=data[i];
    }

    encoded[len]='\0';

    printf("\n encoded string is:%s",encoded);

return 0;
}

```

#### /\*Decryption by Substitution Method\*/

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>

void main()
{
    char seq[36]="qwertyuiopasdfghjklzxcvbnm1234567890";
    char data[100];
    char decoded[100];
    int i,j,len,present=0;
    printf("\n Enter data:");
    gets(data);
    len=strlen(data);

    for(i=0;i<len;i++)
    {
        for(j=0;j<36 && !present;++j)

```

```

{
    if(seq[j]==tolower(data[i]))
    {
        if(isupper(data[i]))
            decoded[i]='a'+j;
        elseif(islower(data[i]))
            decoded[i]='A'+j;
        else decoded[i]='0'+(j-26);
            present=1;
    }
}

if(!present)
    decoded[i]=data[i];
else
    present=0;
}

decoded[len]='\0';
printf("\n Decoded string is:%s", decoded);

}

```

### **/\*Encryption by Transposition Method \*/**

```

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

void main()
{
    char data[100];
    char wrd[]="MEGABUCK";
    char cipher[20][8];
    int seq[8];
    int i,j,k,cnt;
    for(i=0;i<strlen(wrd);i++)
    {
        cnt=0;
        for(j=0;j<8;j++)
            if(wrd[i]>wrd[j])
                ++cnt;
        seq[i]=cnt;
    }
    printf("\n Enter data :");
    gets(data);
    cnt=strlen(data);

    for(i=0;i<cnt;i++)
        cipher[i/strlen(wrd)][i%strlen(wrd)]=data[i];

    if(i%strlen(wrd)!=0)
    {
        for(j=i%strlen(wrd);j<strlen(wrd);j++)

```

```

        {
            cipher[i/strlen(wrd)][j]='.';
            cnt++;
        }
    }

    printf("\n Encrypted data :\n");
    for(i=0;i<8;i++)

    {
        for(j=0;j<8;j++)
            if(seq[j]==i) break;
        for(k=0;k<cnt/8||k==0;k++)
            printf("%c", cipher[k][j]);

    }
}

```

### **/\*Decryption by using Transposition Method\*/**

```

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

int main()
{
    char data[100];
    char wrd[]="MEGABUCK";
    char cipher[20][8];
    int seq[8];
    int i,j,cnt,c;

    for(i=0;i<strlen(wrd);i++)
    {
        cnt=0;
        for(j=0;j<strlen(wrd);j++)
            if(wrd[i]>wrd[j])
                ++cnt;
        seq[i]=cnt;
    }
    printf("\nEnter data :");
    gets(data);

    cnt=strlen(data);
    if(cnt%strlen(wrd)!=0)

    printf("\nError: Invalid Input");

    else
    {
        for(i=0;i<8;i++)
        {
            for(c=0;c<8;c++)
                if(seq[c]==i)
                    break;
            for(j=0;j<cnt/strlen(wrd);j++)

```

```

        cipher[j][c]=data[i*(cnt/strlen(wrd))+j];
    }
    for(j=0;j<strlen(wrd);j++)
    {
        if(cipher[cnt/strlen(wrd)-1][j]=='.')
            cipher[cnt/strlen(wrd)][i*strlen(wrd)]==' ';
    }
    printf("Decrypted data:");
    for(i=0;i<cnt;i++)
        printf("%c",cipher[i/strlen(wrd)][i*strlen(wrd)]);
    }
    return 0;
}

```

Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			<div style="border-top: 1px solid black; width: 50px; margin: 0 auto;"></div> 10
<b>Staff Signature with Date</b>			

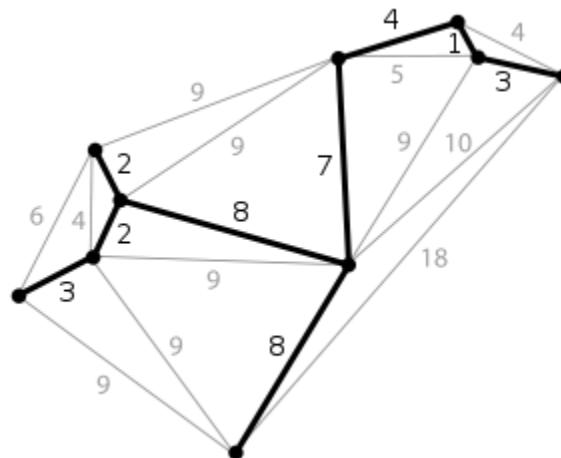
### EXPERIMENT 3 MINIMUM SPANNING TREE ALGORITHM



**Objective:** To simulate Minimum Spanning Tree algorithm using C Program.

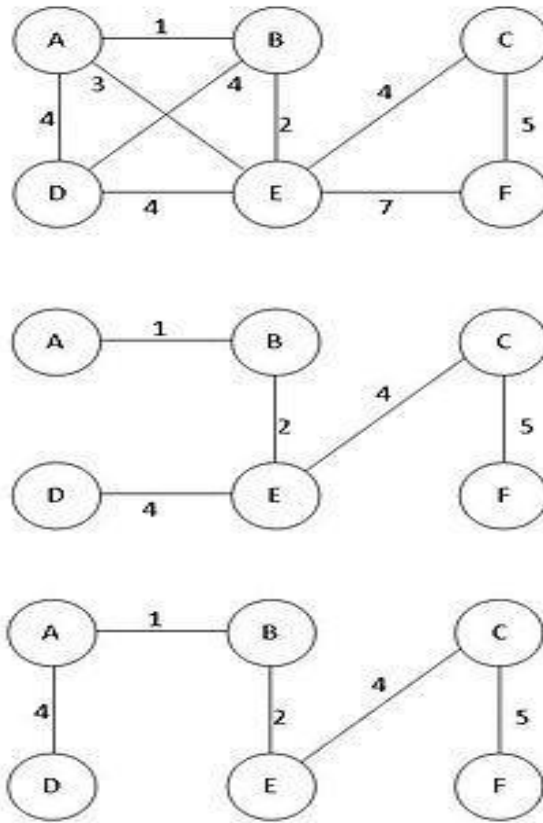
**Theory:** Given a connected, undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. We can also assign a *weight* to each edge, which is a number representing how unfavorable it is, and use this to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree. A **minimum spanning tree (MST)** or **minimum weight spanning tree** is then a spanning tree with weight less than or equal to the weight of every other spanning tree. More generally, any undirected graph (not necessarily connected) has a **minimum spanning forest**, which is a union of minimum spanning trees for its connected components.

One example would be a telecommunications company laying cable to a new neighborhood. If it is constrained to bury the cable only along certain paths, then there would be a graph representing which points are connected by those paths. Some of those paths might be more expensive, because they are longer, or require the cable to be buried deeper; these paths would be represented by edges with larger weights. A *spanning tree* for that graph would be a subset of those paths that has no cycles but still connects to every house. There might be several spanning trees possible. A *minimum spanning tree* would be one with the lowest total cost.



Kruskal's algorithm is a greedy algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component).

The following figure shows there may be more than one minimum spanning tree in a graph. In the figure, the two trees below the graph are two possibilities of minimum spanning tree of the given graph.



### Algorithm

- 1 Read the number of nodes
- 2 List all the edges of the graph in the ascending order of weight.
- 3 Pick the smallest edge from the list and check if the addition of the edge in the spanning tree doesn't form a cycle. Then add the edge in the spanning tree. If it forms a cycle, then discard that edge. Note: Initially the spanning tree is empty.
- 4 Repeat the above step until there are no edges in the list, or there are  $n-1$  edges in spanning tree where  $n$  is number of nodes.
- 5 If there are  $n-1$  edges in the spanning tree, the spanning tree for the graph is formed
- 6 If there are no edges in the list and there are less than  $n-1$  edges in the spanning tree, then no spanning tree can be formed for the graph.

**PROGRAM:**

```

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

struct node
{
    int set;
} node[100];

struct edge
{
    int first_node,second_node,selected,distance;
}e[100];
int edge_count=0;

void getdata(int index,int total)
{
    int i;
    for(i=index;i<total;i++)
    {
        if(i!=index)
        {
            printf("\nEnter distance between Vertex %c and
%c:",index+65,i+65);
            scanf("%d",&e[edge_count].distance);
            e[edge_count].first_node=index;
            e[edge_count].second_node=i;
            ++edge_count;
        }
    }
}

void init(int total)
{
    int i;
    for(i=0;i<total;i++)
        node[i].set=i;
    for(i=0;i<edge_count;i++)
        e[i].selected=-1;
}

void sort()
{
    int i,j;
    struct edge temp;

    for(i=0;i<edge_count-1;i++)
    for(j=0;j<edge_count-i-1;j++)
        if(e[j].distance>e[j+1].distance)
        {
            temp=e[j];
            e[j]=e[j+1];
            e[j+1]=temp ;
        }
}

```

```

}

void main()
{
    int i,total,j,k,m,n,edgeselectd=0,nodel,noder;

    printf("\nEnter the number of nodes:");
    scanf("%d",&total);
    for(i=0;i<total;i++)
        getdata(i,total);
        init(total);
        sort();

    printf("\nThe Sorted order of edges:");
    for(i=0;i<edge_count;i++)
        printf("\nedge: %d first node: %c second node: %c distance:
        %d",i,e[i].first_node+65,e[i].second_node+65,e[i].distance);
    i=0;
    do
    {
        e[i].selected=1;
        nodel=e[i].first_node;
        noder=e[i].second_node;
        if (node[nodel].set==node[noder].set)
            e[i].selected=-1;
    else
        {
            edgeselectd++;
            m=node[nodel].set;
            k=node[noder].set;
            for (n=0;n<total;n++)
            {
                if (node[n].set==k)
                    node[n].set=m;
            }
        }
    i++;
    }
    while(edgeselectd<(total-1));
    printf("\nMinimum Spanning Tree is:");
    for(i=0;i<edge_count;++i)
    {
        if(e[i].selected==1)
            printf("\nc<----->c Distance %d",e[i].first_node+65,
            e[i].second_node+65, e[i].distance);
    }
}

```

### Prim's Algorithm :

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.

### ALGORITHM :

1. Associate with each vertex  $v$  of the graph a number  $C[v]$  (the cheapest cost of a connection to  $v$ ) and an edge  $E[v]$  (the edge providing that cheapest connection). To initialize these values, set all values of  $C[v]$  to  $+\infty$  (or to any number larger than the maximum edge weight) and set each  $E[v]$  to a special flag value indicating that there is no edge connecting  $v$  to earlier vertices.
2. Initialize an empty forest  $F$  and a set  $Q$  of vertices that have not yet been included in  $F$  (initially, all vertices).
3. Repeat the following steps until  $Q$  is empty:
  - a. Find and remove a vertex  $v$  from  $Q$  having the minimum possible value of  $C[v]$
  - b. Add  $v$  to  $F$  and, if  $E[v]$  is not the special flag value, also add  $E[v]$  to  $F$
  - c. Loop over the edges  $vw$  connecting  $v$  to other vertices  $w$ . For each such edge, if  $w$  still belongs to  $Q$  and  $vw$  has smaller weight than  $C[w]$ , perform the following steps:
    - i. Set  $C[w]$  to the cost of edge  $vw$
    - ii. Set  $E[w]$  to point to edge  $vw$ .
4. Return  $F$

### PROGRAM :

```
#include<stdio.h>

#define infinity 999
int prime(int cost[10][10],int source,int n)
{
    int i,j,sum=0,visited[10],cmp[10],vertex[10];
    int min,u,v;
    for(i=1;i<=n;i++)
    {
        vertex[i]=source;
        visited[i]=0;
        cmp[i]=cost[source][i];
    }
    visited[source]=1;
```

```

    for(i=1;i<=n-1;i++)
    {
        min=infinity;
        for(j=1;j<=n;j++)
            if(!visited[j] && cmp[j]<min)
            {
                min=cmp[j];
                u=j;
            }

        visited[u]=1;
        sum=sum+cmp[u];
        printf("\n %d-> %d sum=%d",vertex[u],u,cmp[u]);
        for(v=1;v<=n;v++)
            if(!visited[v] && cost[u][v] < cmp[v])
            {
                cmp[v]=cost[u][v];
                vertex[v]=u;
            }
    }

return sum;

}

void main()
{
    int a[10][10],n,i,j,m,source;
    printf("\n Enter the number of vertices");
    scanf("%d",&n);
    printf("\n Enter the cost matrix:0 self loop& 999 no edge\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);

    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            if(a[i][j]!=a[j][i]||(a[i][i]!=0))
            {
                printf("\n Invalid entry \n cost matrix should be symmetrical &
the diagonal elements are zero");
            }
    printf("\n Entert the source:");
    scanf("%d",&source);

    m=prime(a,source,n);
    printf("\n\n total cost=%d",m);
}

```

Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			<u>        </u> 10
<b>Staff Signature with Date</b>			

## EXPERIMENT 4

### ANALYSIS AND COMPARISON OF CSMA/CD AND CSMA/CA USING QualNet

**Objective:** To analyze CSMA/CD and CSMA/CA protocols with respect to following parameters and compare them: (i) Number of transmitting nodes vs Collision count

(ii) Number of transmitting nodes vs Mean delay

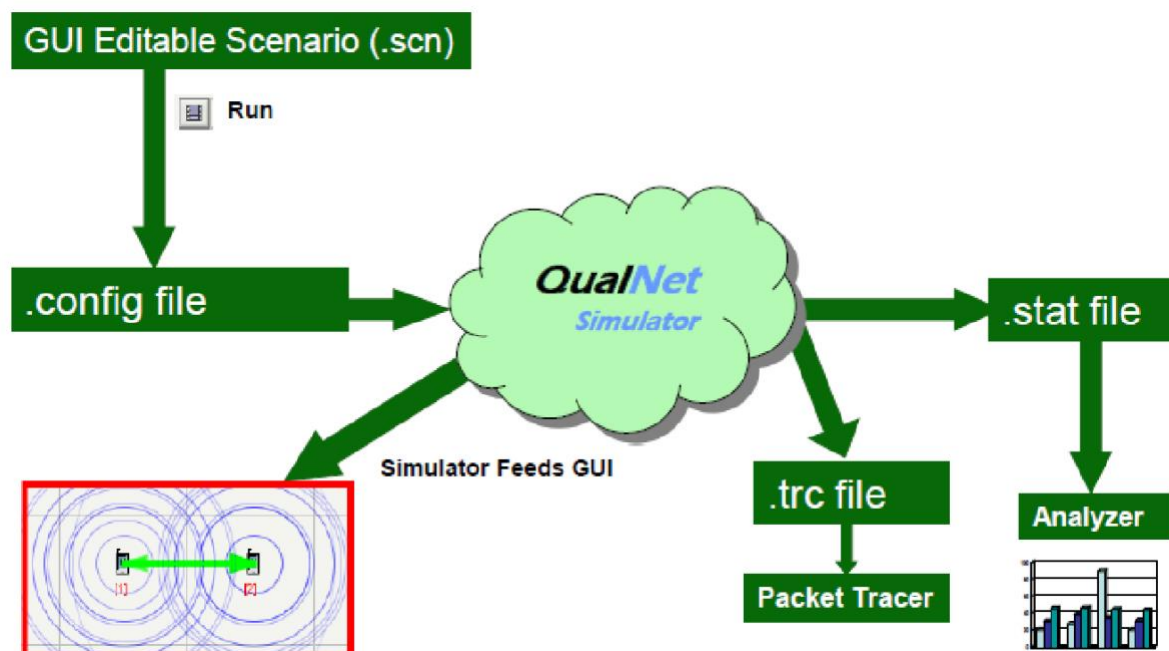
(iii) Number of retransmissions

(iv) Number of frames dropped

#### About QualNet :

The QualNet communications simulation platform (QualNet) is a planning, testing and training tool that "mimics" the behavior of a real communications network. Simulation is a cost-effective method for developing, deploying and managing network-centric systems throughout their entire lifecycle. Users can evaluate the basic behavior of a network, and test combinations of network features that are likely to work. QualNet provides a comprehensive environment for designing protocols, creating and animating network scenarios, and analyzing their performance.

#### QualNet Architecture:



#### Theory for CSMA/CD:



Ethernet is a LAN (Local area Network) protocol operating at the MAC (Medium Access Control) layer. Ethernet has been standardized as per IEEE 802.3. The underlying protocol in Ethernet is known as the CSMA / CD – Carrier Sense Multiple Access / Collision Detection. The working of the Ethernet protocol is as explained below,

- A node which has data to transmit senses the channel,
- If the channel is idle then the data is transmitted
- If the channel is busy then the station defers transmission until the channel is sensed to be idle and then data is immediately transmitted.
- If more than one node starts data transmission at the same time, the data collides. This collision is heard by the transmitting nodes which enter into contention phase.
- The contending nodes resolve contention using an algorithm called truncated binary exponential back off.

### **Medium Access Control (MAC) :**

Under the IEEE 802 series of standards, the data link layer of the OSI Reference Model is subdivided into two sublayers: Logical Link Control (LLC) and the Medium Access Control (MAC). The MAC sublayer is responsible for checking the channel and transmitting data if the channel is idle, checking for occurrence of a collision, and taking a series of predefined steps if a collision is detected. The MAC layer is an interface between user data and the physical placement and retrieval of data on the network.

This exercise focuses on one major function performed by the MAC sublayer -- transmission medium access management. The activities associated with this functionality are:

- Defer transmission if the medium is busy.
- Delay transmission for a specified interframe gap period.
- Present a serial bit stream to the physical layer for transmission.
- Halt transmission when a collision is detected.
- Transmit a jam signal to ensure that the news of a collision propagates throughout the network.
- Reschedule retransmission after a collision until successful, or until a specified retry limit is reached.
- **Carrier sense**  
CSMA/CD can be described as a listen-before-speaking multiple access method. Thus, the first function associated with transmission media access management is to find out whether any data is already being transmitted on the network and, if so, to defer transmission. During the listening process, each station attempts to sense the carrier signal on the bus, hence the name carrier sense (CS) is used for this access method.

Under Manchester encoding, a transition occurs in the middle of each bit period. The binary bit 1 is represented by a high-to-low transition, while the binary bit 0 is represented by a low-to-high voltage transition. Thus, an examination of the voltage on the medium of a base-band network enables a station to determine whether a carrier is present. If a carrier

signal is found, the station with data to transmit will continue to monitor the channel.

- **Waiting time**

If the medium is busy, the station waits until it goes idle; otherwise it transmits immediately. If two or more stations simultaneously begin transmitting on an idle cable, they will collide. All colliding stations then terminate their transmission, wait for a random time, and repeat the whole process all over again.

- **Collision detection**

A transmitting station keeps listening to the channel to check for collisions while it is sending. Since IEEE 802.3 Manchester encoded signals have a 1 volt average DC voltage level, a collision results at an average DC level of 2 volts. Thus, a transceiver or network interface card can detect collision by monitoring the voltage level of the Manchester line signal.

- **Jam sequence**

If a collision is detected during transmission, the transmitting station will cease transmitting of data and initiate transmitting of a jam pattern. The jam pattern consists of 32 ~ 48 bits. These bits can have any value other than the CRC value that corresponds to the partial frame transmitted before the jam. The transmission of the jam pattern ensures that the collision lasts long enough to be detected by all stations on the network.

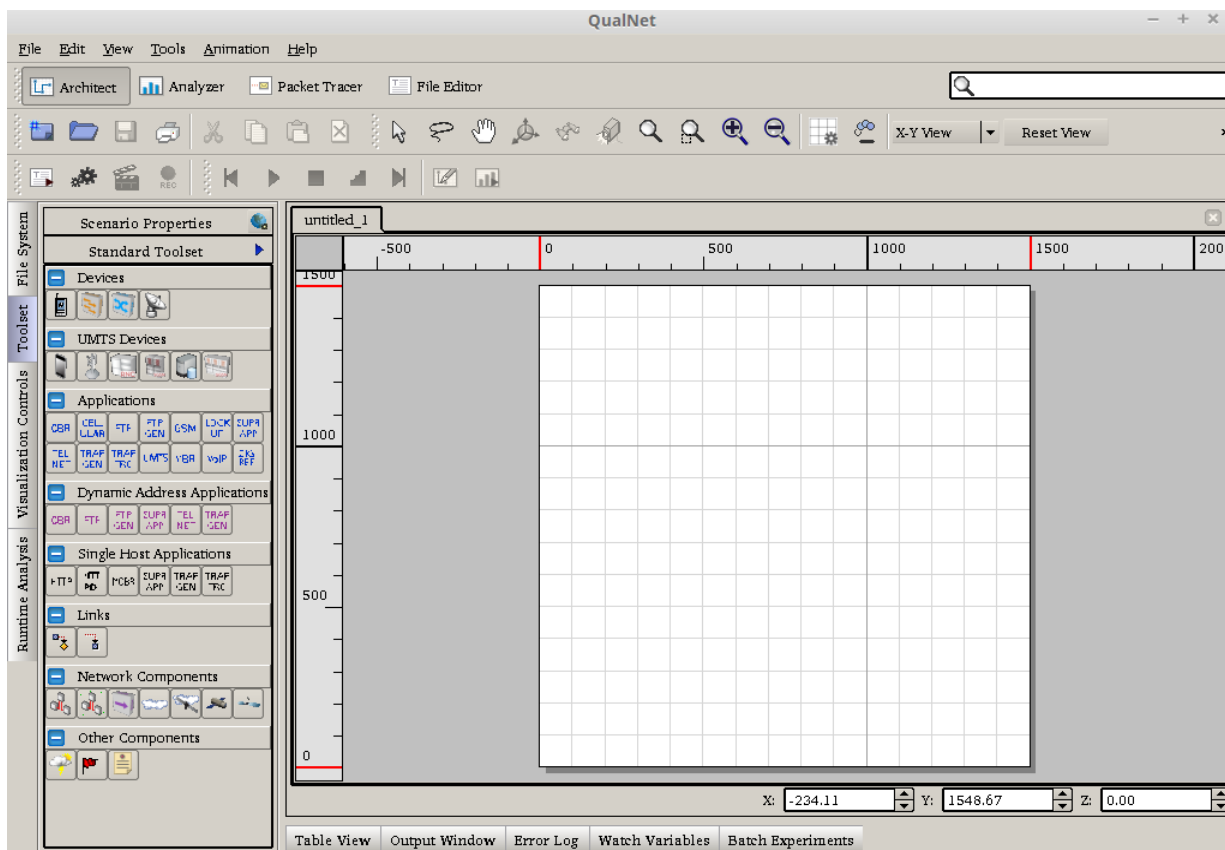
- **Collision backoff and retransmission**

Once a collision is detected, the transmitting station waits for a random number of time slots before attempting to retransmit. The term "time slot" represents the time required to transmit 512 bits on the network. The actual number of time slots the station waits is selected by a randomization process, formally known as "truncated binary exponential backoff". Under this randomization process, a randomly selected integer  $r$  defines the number of time slots the station must wait before listening again to determine whether the channel is clear. The number of time slots to delay before the  $n$ -th retransmission attempt is chosen as a uniformly distributed random integer  $r$  in the range between 0 and  $(2^k)-1$ , where  $k=\min(n,10)$ . After the backoff, the station begins to retransmit the frame when it senses the channel to be free, while listening for another collision. After a user-defined maximum number of attempts, the MAC entity assumes that some problem exists, gives up sending, and reports failure to LLC.

### Procedure for CSMA/CD:

To begin with the experiment, run QualNet-7.1-GUI Application.  
by typing following command in terminal

QualNetGui



**QualNet GUI**

1. Create a new scenario named 'CSMA/CD' under the directory  
/Scalable/qualnet/7.1/scenarios/user
2. Place 30 nodes automatically in the network.

From the **Tools** pull down menu, select **Place Nodes** (If it's gray/disabled, double click the canvas/**Scenario Designer**).

In Node Placement Dialog Box, input “30” as Number of Nodes, and click on OK.

**Node Placement**

Number of Nodes:

Device Type:

Placement Strategy:

Placement Model:

Seed:

☐ Use Altitudes from Terrain File

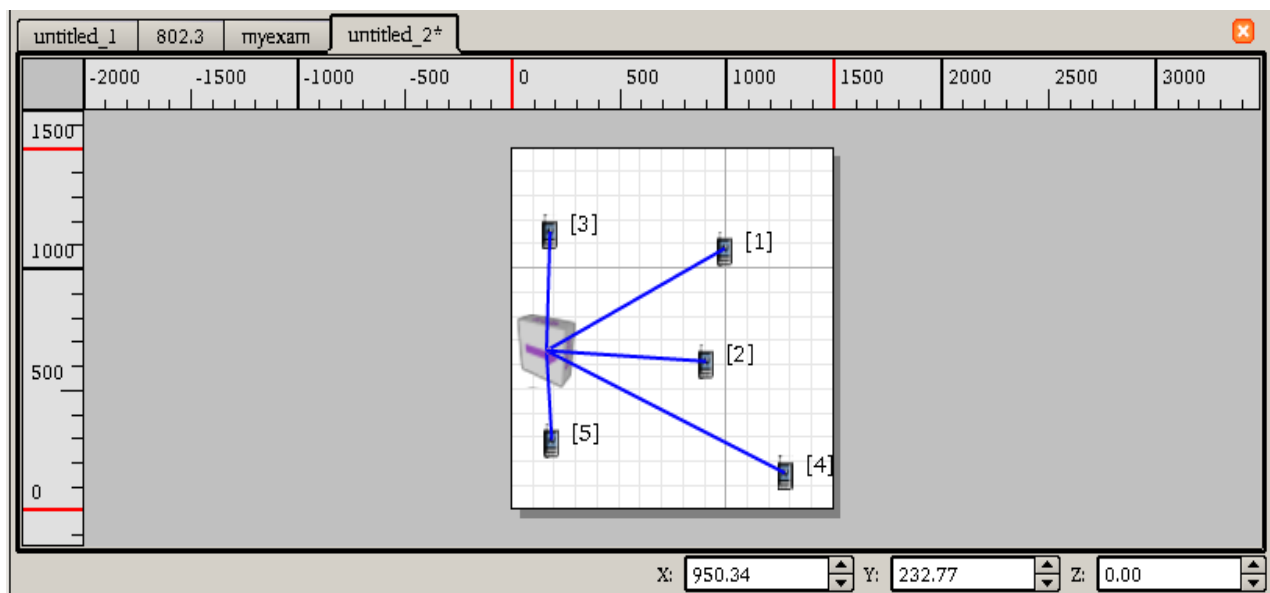
Placement Data:

Origin : X Axis  Y Axis

Dimensions (meters) : X Axis  Y Axis

Apply OK Cancel

3. Select All Nodes, from **Standard Toolset** → **Network Componets** → choose **HUB**, drag and drop the Hub in your Scenario. (Now all the nodes are connected to hub)



4. Now Select All nodes, right click → **Properties** → Default Device Properties Dialog Will appears,

From **Interface** Tab Double Click on Interface 0 →

- i) For **Physical Layer**, set Radio Type as None or Abstract

ii) For **MAC Layer**, set

MAC Protocol = Generic MAC

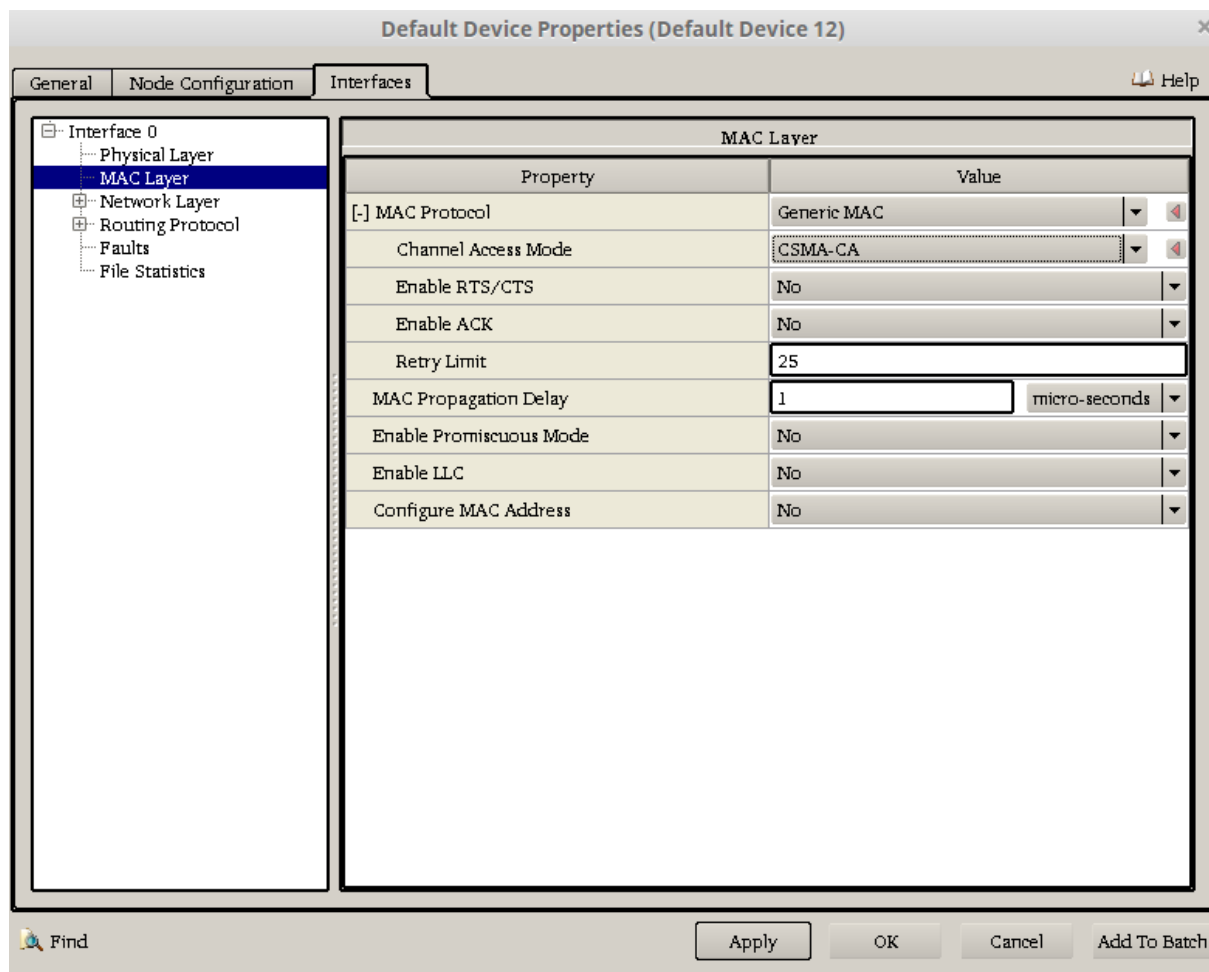
Channel Access Mode = CSMA

MAC Propagation Delay = 10 msec.

Enable LLC = NO

iii) For **Network Layer**, set Enable Explicit Congestion Notification = YES.

Then click on apply and OK.



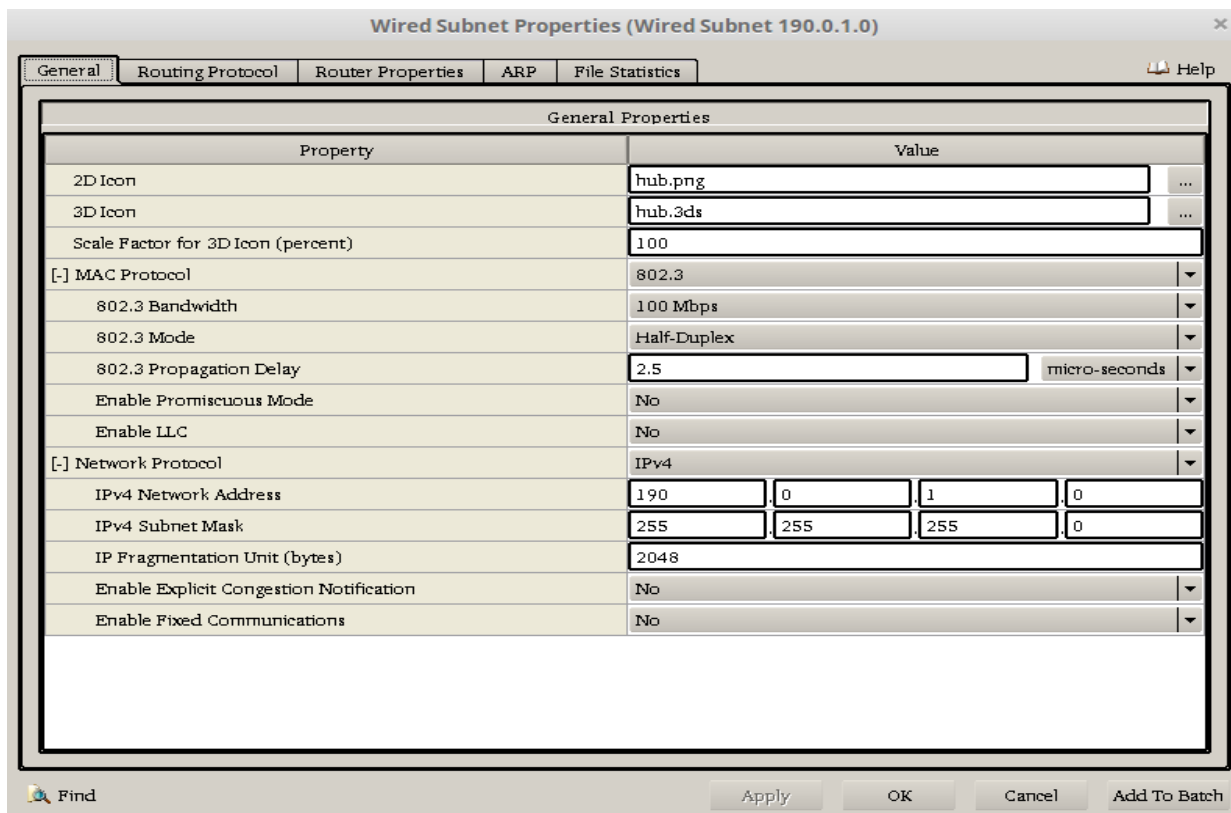
5. Select the hub, right click → Properties → in **Wired Subnet Properties**

In General Tab → Mac Protocol → set it as **802.3**

802.3 Bandwidth as 10 Mbps

802.3 mode as Half Duplex.

In Routing Protocol → Choose Routing Protocol as Bellman Ford. then click on apply and OK.



## 6. Add Applications

After setting up the network, the next step is creation of application layer connections between the nodes. At any time, there may be several connections active simultaneously in the network. Each connection will have a source and a destination node. The source node generates traffic and transmits to a destination node. Let us create some applications : -

- Traffic-Gen generates UDP traffic while FTP Generic generates TCP traffic.

From Standard Toolset → Applications → Choose any application type, drag & drop over desired source and destination nodes. Add n number of applications.

Example : node 3 to 29 – FTP

node 10 to 27 – TRAF GEN

The next step is to choose the parameter values of this connection:

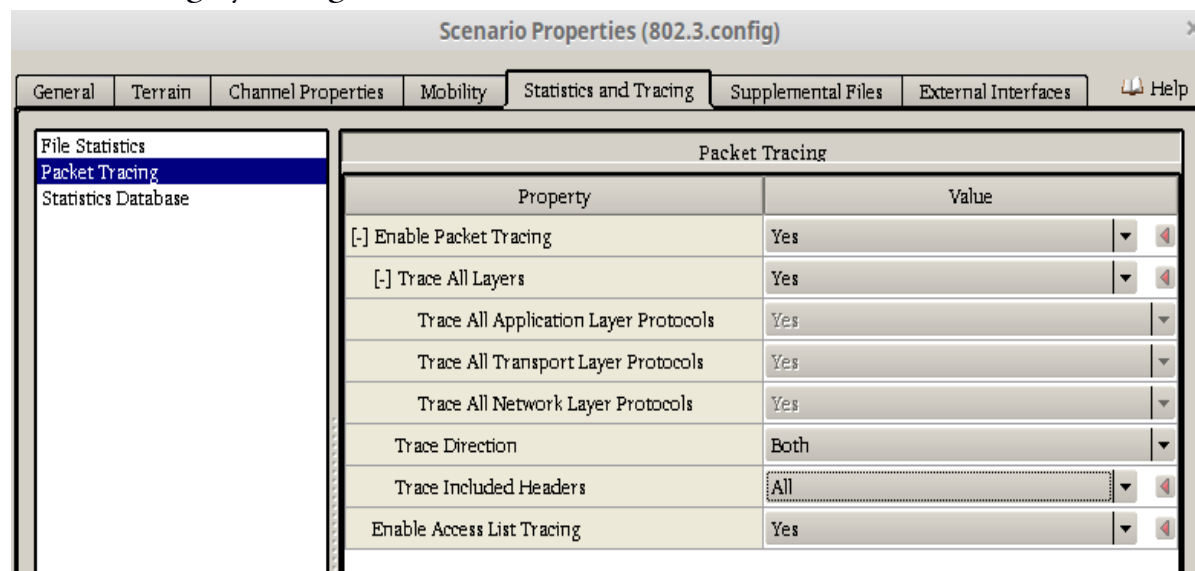
Let us create a Poisson process of data packets at the source and make the packet length exponentially distributed. To do this, click Table View on the bottom of the right window, and select the Application tab. You should see the connection Traffic-Gen 10->27 listed. Double click the connection.

**Packet Size :** packet lengths to be exponentially distributed with mean packet length 2048 bytes. The default size is deterministically set to 512 bytes. Change the Data Size from Deterministic to

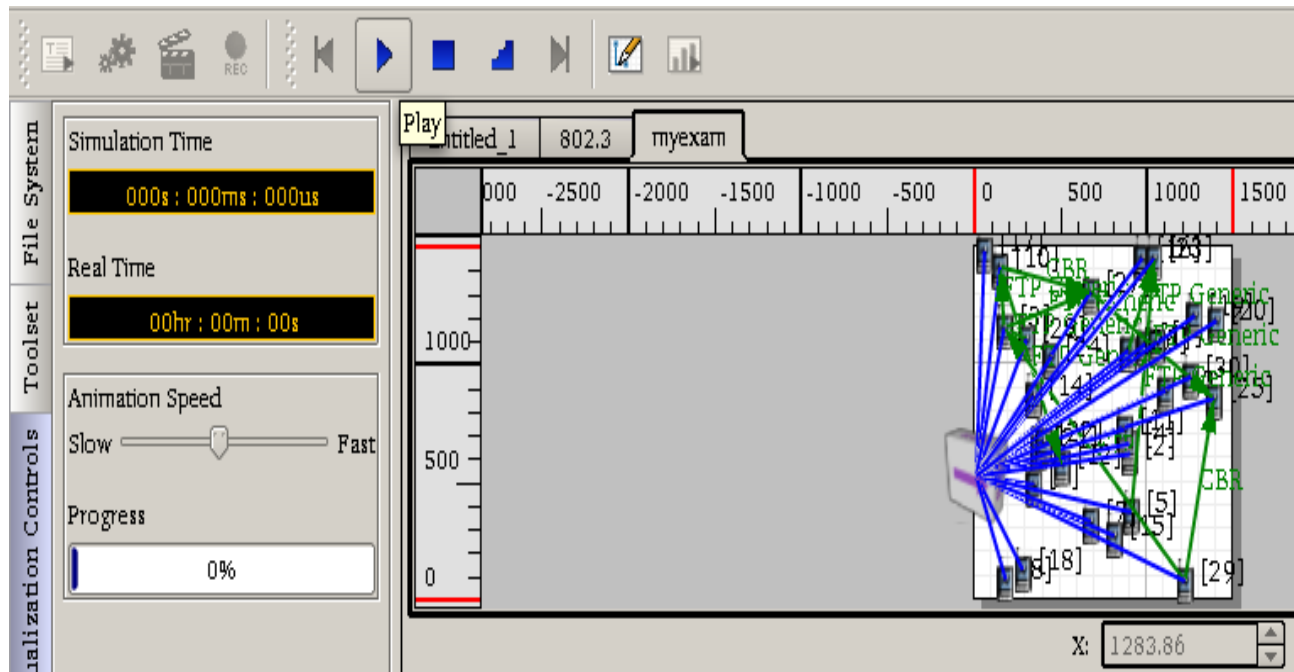
Exponential. Then change the Mean Size to 2048 bytes. To make the source generate traffic according to a Poisson process, click on Interval from the Traffic Type menu for this connection. By default traffic is generated deterministically every 1 second. Change the distribution of the interval between successive packets to Exponential and change the Mean Interval to 0.5 seconds (recall that the inter-arrival times for a Poisson process are exponentially distributed). As well, set the Start Time of the connection to Exponential with mean 0.5 seconds, and the Duration to Deterministic with fixed duration of 30 seconds.

#### 7. Enable **Packet Tracing** :

Goto **Scenario Properties** : From **Statistics & Tracing** → Choose **Packet tracing** → Enable Packet tracing by setting the value to YES.



#### 8. Click on **Run Scenario**



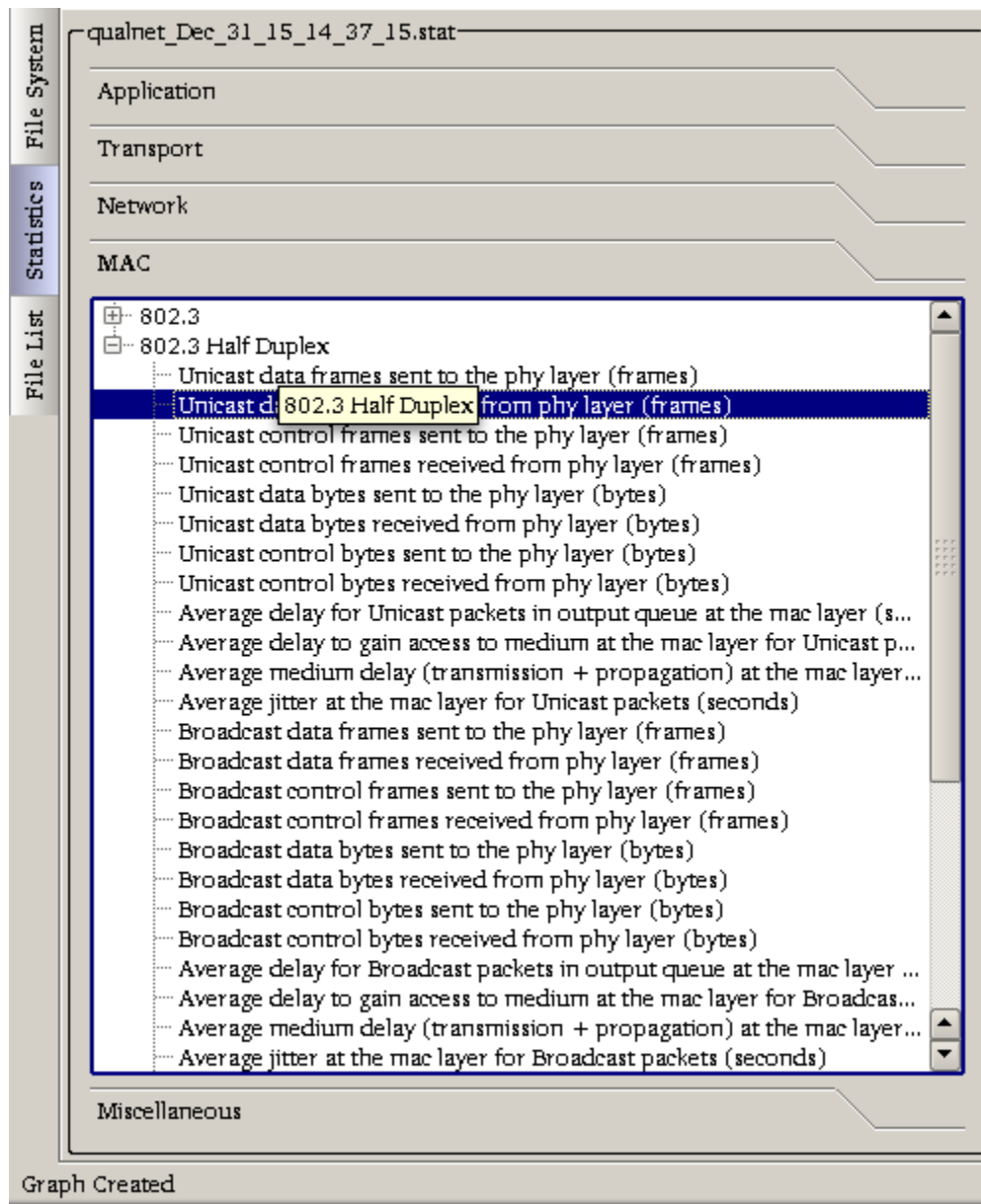
9. Click on **Play** Button.

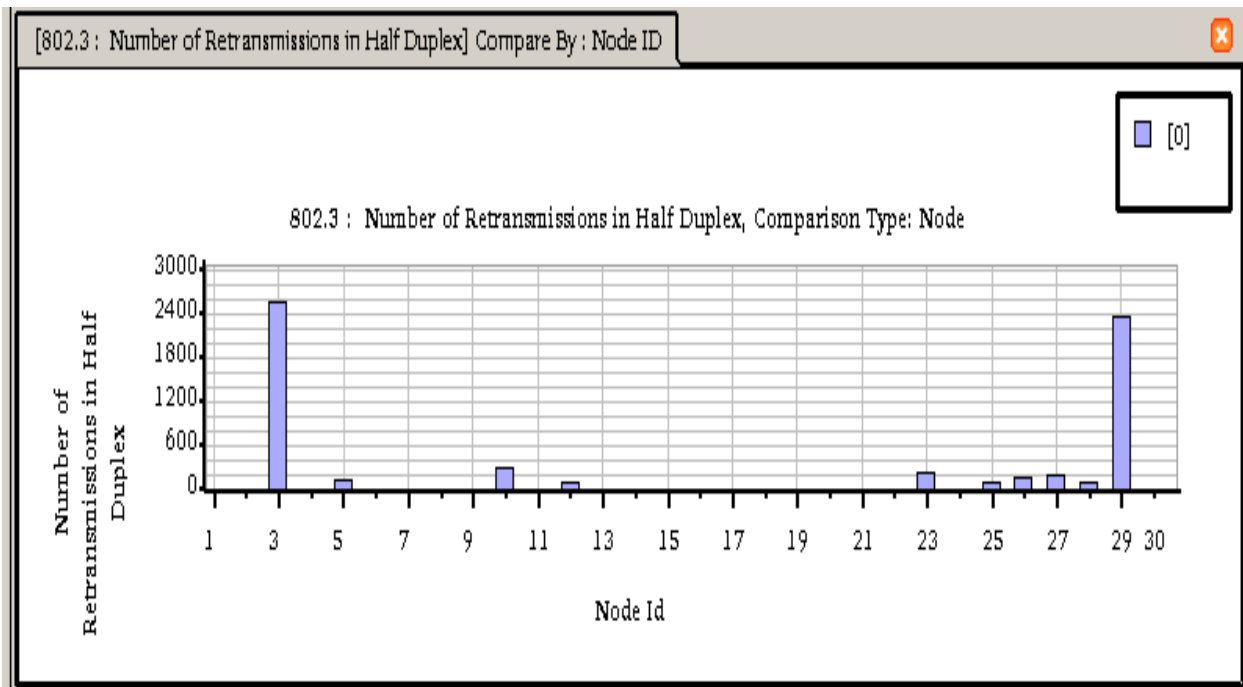
10. After completion of **Simulation**, click on **Analyze Statistics of Current Scenario**. (to switch to Analysis Window). Now you will see the performance metrics of your network scenario with respect to the each layers.

### Results :

11. Now Verify the statistics of your network Scenario – CSMA -CD with respect to MAC Layer.





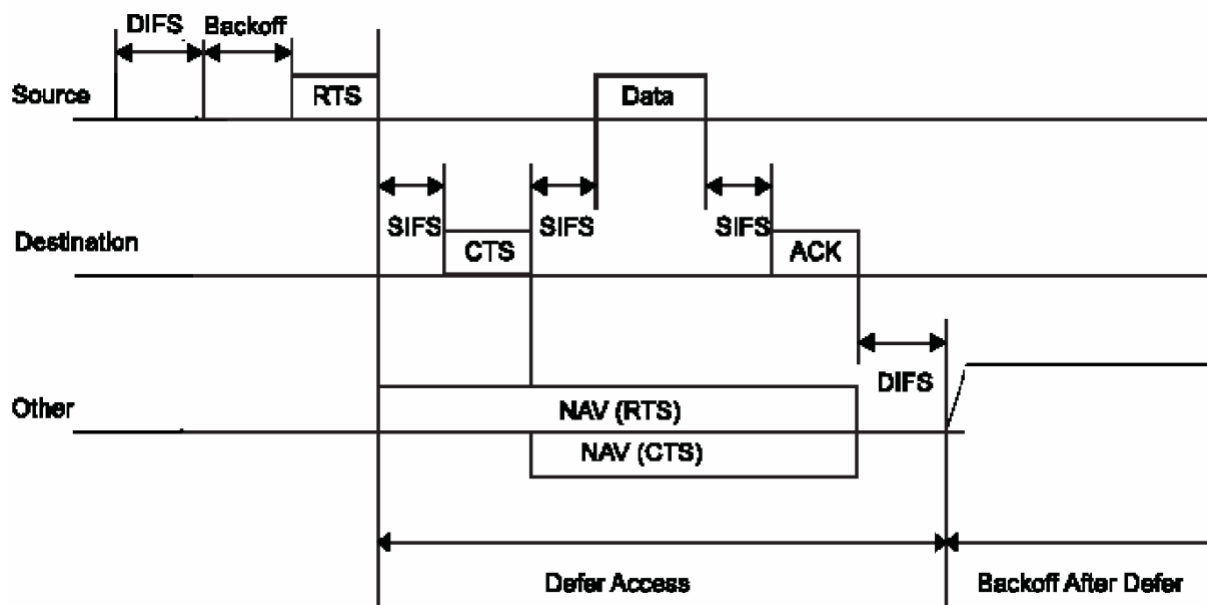


### Theory for CSMA/CA:

Wireless LAN is basically a LAN that transmits data over air, without any physical connection between devices. The transmission medium is a form of electromagnetic radiation. Wireless LAN is ratified by IEEE in the IEEE 802.11 standard.

The underlying algorithm used in Wireless LAN is known as the CSMA / CA – Carrier Sense Multiple Access / Collision Avoidance algorithm. The working of CSMA / CA algorithm is given below.

- The node which has data to transmit senses the medium. If the medium has been idle for longer than the DIFS (DCF Inter Frame Space), it finishes its back off interval & transmits Request To Send (RTS) signal immediately.
- The access point responds with Clear to Send (CTS) signal. Now the node has reserved the medium and transmits data.
- If the medium is busy, the node waits for the channel to become idle for the DIFS.
- If two nodes sense the medium at the same time & transmit RTS simultaneously, RTS collision occurs and the transmission is retried. Hence data collision is avoided.
- For each retransmission, contention window increases exponentially hence back off time is selected from larger contention window.

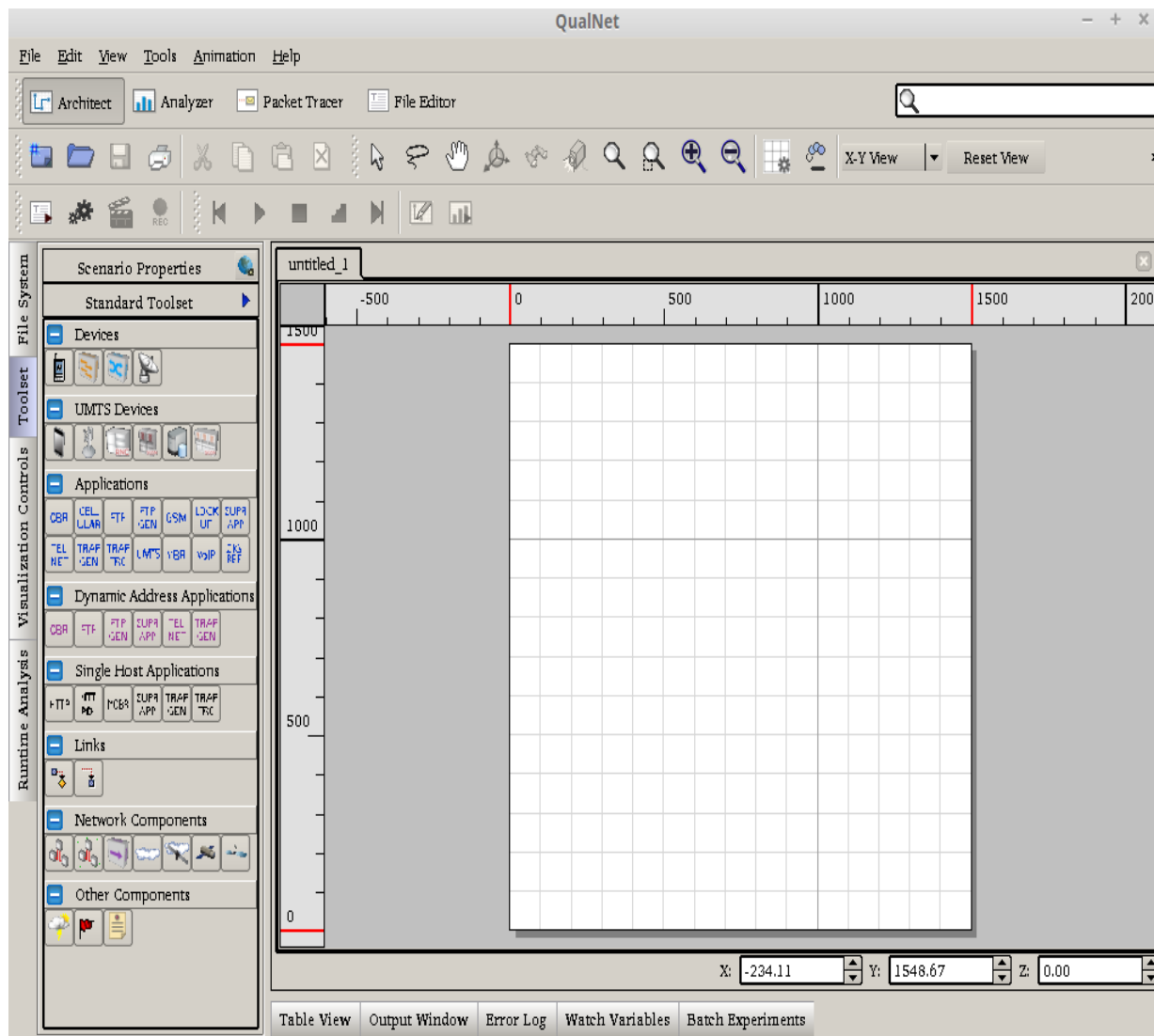


### Procedure for CSMA/CA

To begin with the experiment, run QualNet-7.1-GUI Application.

by typing following command in terminal

### QualNetGui

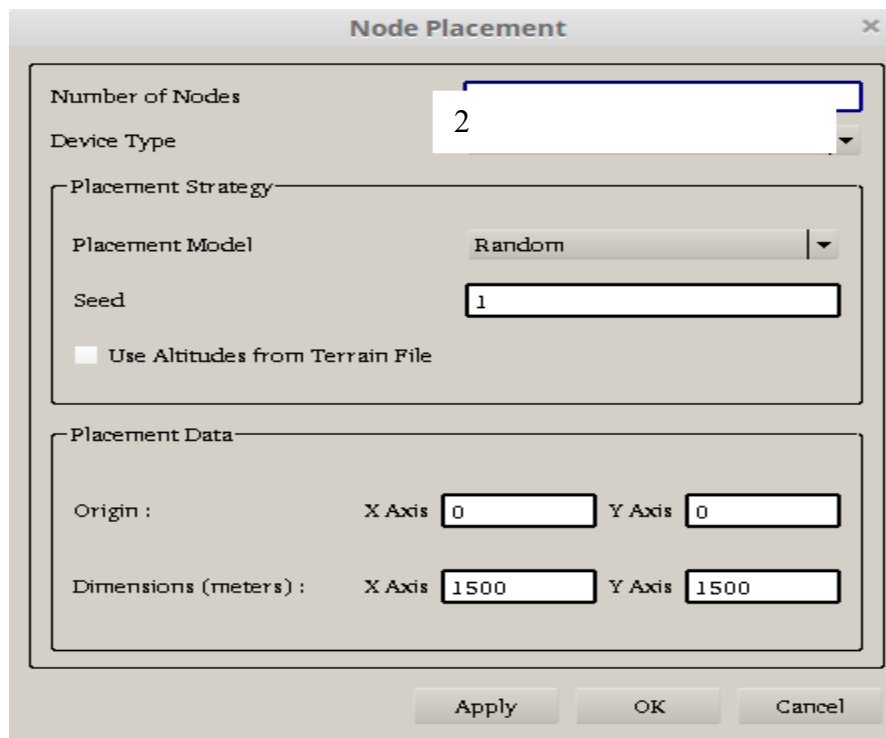


**QualNet GUI**

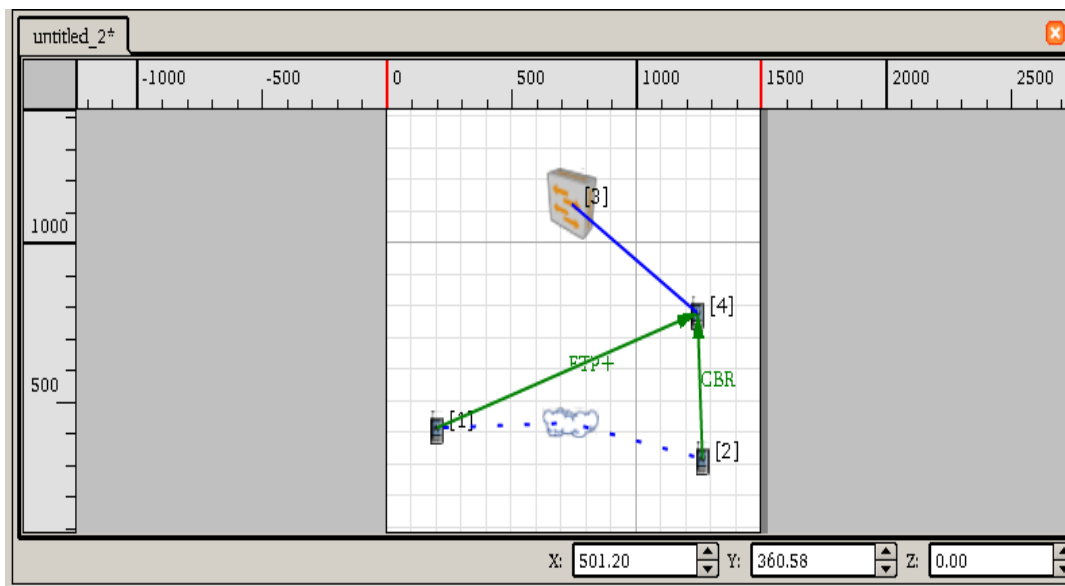
1. Create a new scenario named 'CSMA-CA' under the directory /Scalable/qualnet/7.1/scenarios/user
2. Place 2 nodes automatically in the network.

From the **Tools** pull down menu, select **Place Nodes** (If it's gray/disabled, double click the canvas/**ScenarioDesigner**).

In Node Placement Dialog Box, input “2” as Number of Nodes, and click on OK.



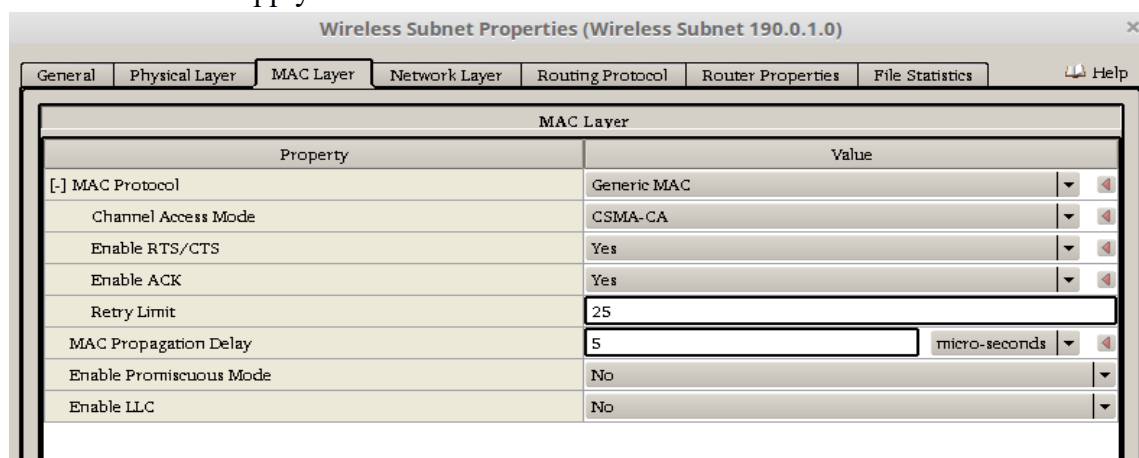
3. Select All Nodes, from **Standard Toolset** → **Network Components** → choose **Wireless Subnet**, drag and drop the wireless subnet in your Scenario. (Now all the nodes are connected to hub)



Add one switch and one external node connect using wire.

4. Now Select All nodes, right click → **Properties** → Default Device Properties Dialog Will appears,  
From **Interface** Tab Double Click on Interface 0 →

- i) For **Physical Layer**, set Radio Type as None or Abstract
- ii) For **MAC Layer**, set
- MAC Protocol = Generic MAC
  - Channel Access Mode = CSMA-CA
  - MAC Propagation Delay = 10 msec.
  - Enable LLC = YES
- iii) For **Network Layer**, set Enable Explicit Congestion Notification = YES.
- Then click on apply and OK.

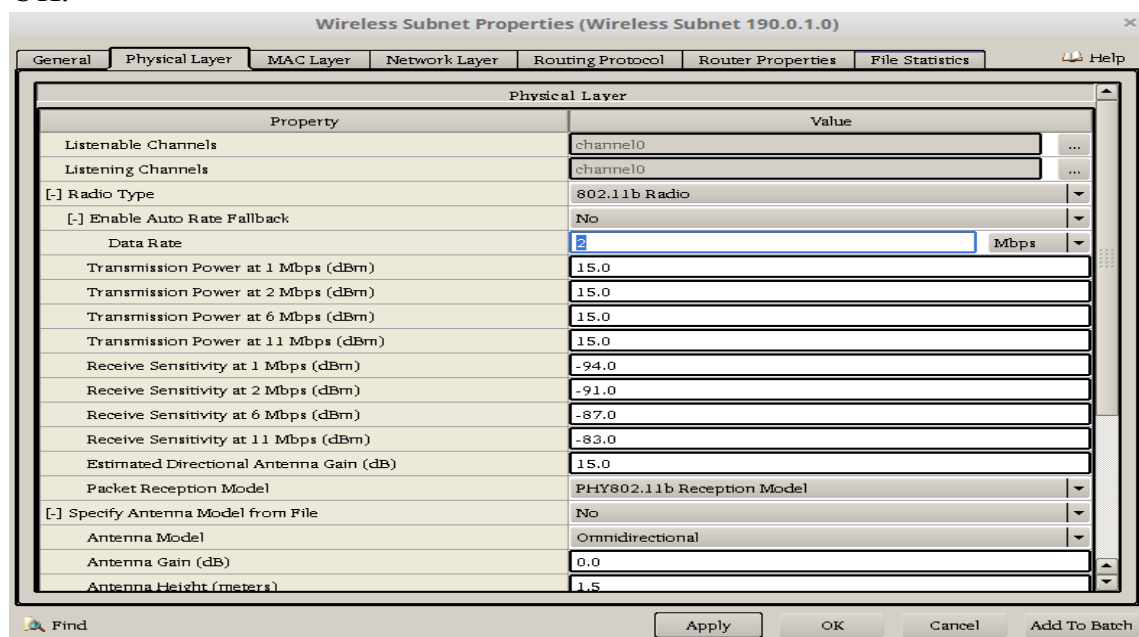


5. Select the hub, right click → Properties → in **Wired Subnet Properties**

In General Tab → Mac Protocol → set it as **802.11b**

Datarate as → **2 Mbps**

In Routing Protocol → Choose Routing Protocol as Bellman Ford. then click on apply and OK.



MAC protocol is 802.11 standard. This standard has two modes of operations known as Distributed Coordination Function (DCF) and Point Coordination Function (PCF). The DCF is more popular in practice and based on the random access mechanism. Briefly, DCF function as follows,

- DCF uses the CSMA protocol, thus a node, with a packet to transmit, first has to sense the channel. If the channel is busy the node backs off for a random amount of time.
- We note that the channel may be idle at the source but not at the destination node. If the source node senses the channel idle, it transmits a short request to send (RTS) packet. If the channel is also idle at the destination node and it receives the RTS packet then replies with clear to send (CTS) packet. After that the source transmits the data packet and destination node replies with an ACK packet if the packet is received error free which completes the transmission of the packet. If the source node does not receive the CTS packet or an ACK following the transmission of the data packet then it backs off for a random amount of time. A source node may retransmit an unsuccessful packet upto a transmission limit, when that limit is passed then the packet is discarded.

## 6. Add Applications

From **Standard Toolset** → **Applications** → Choose any application type, drag & drop over desired source and destination nodes. Add n number of applications.

Example : node 1 to 4 – FTP

node 2 to 4 - CBR

## 7. Enable Packet Tracing :

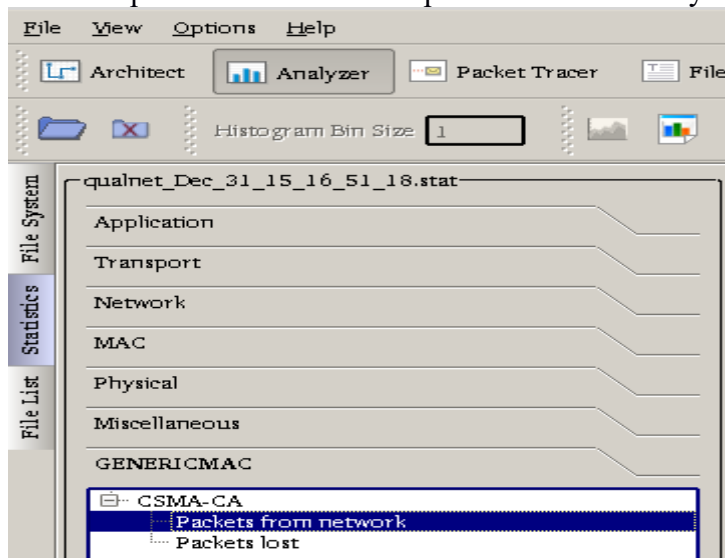
Goto **Scenario Properties** : From **Statistics & Tracing** → Choose **Packet tracing** → Enable Packet tracing by setting the value to YES.



## 8. Click on Run Scenario

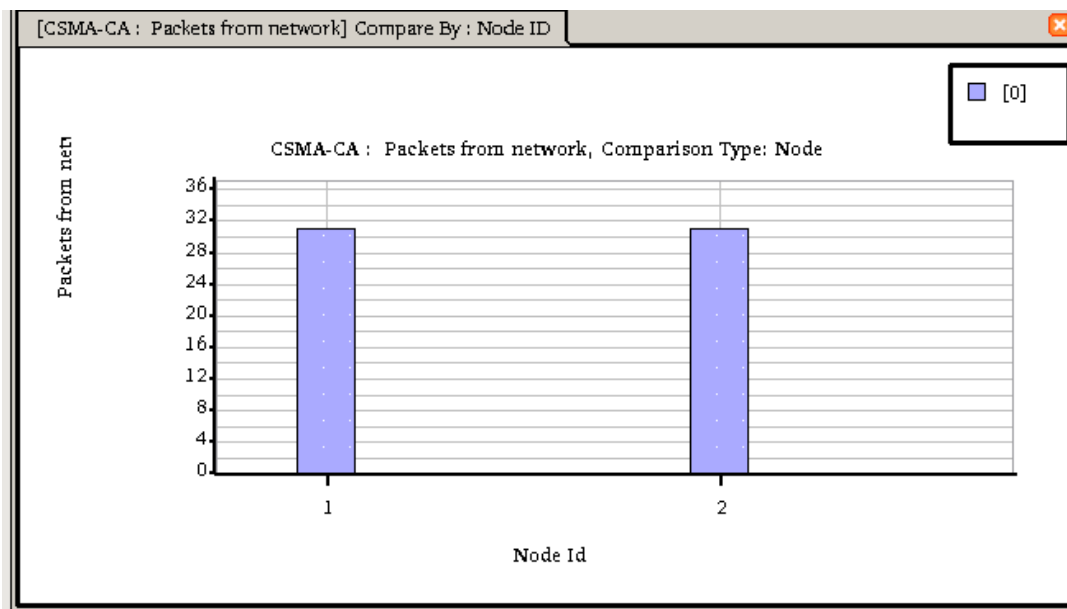
9. Click on **Play** Button.

10. After completion of **Simulation**, click on **Analyze Statistics of Current Scenario**. (to switch to Analysis Window). Now you will see the performance metrics of your network scenario with respect to the CSMA-CA protocol and other layers.



## Results :

11. Now Verify the statistics of your network Scenario – CSMA -CA with respect to MAC Layer.





Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			<u>        </u> 10
<b>Staff Signature with Date</b>			

## EXPERIMENT 5

### IMPLEMENT STOP AND WAIT PROTOCOL USING SOCKET PROGRAMMING CONCEPT IN C PROGRAM.

**Objective:** Implement stop and wait protocol using socket programming in C.

#### Socket Programming :

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server. When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

#### Client-server model :

1. The client-server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.
2. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system.
3. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function.
4. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client-server model are Email, network printing, and the World Wide Web.

#### Socket :

1. Socket is a method for communication between a client program & a server program in a network. A socket is defined as “the endpoint in a connection”.
2. The two values that identify each endpoint, an IP address and a port number, are called a socket.

#### Types of Sockets :

There are 3 types of sockets:

1. Stream Socket  
Designed to be used with a connection-oriented protocol such as TCP.
2. Datagram Socket  
Designed to be used with a connectionless protocol such as UDP.

### 3. Raw Socket

Provide access to ICMP to move secret facilities of an existing protocol.

#### Port Number :

- ✓ Port is an application-specific or process-specific software construct serving as a communications endpoint in a computer's host operating system.
- ✓ A port is associated with an IP address of the host, as well as the type of protocol used for communication.
- ✓ The port numbers are divided into three categories
  - i) **Well known ports:** 0 through 1023 .These port numbers are controlled and assigned by the IANA. When possible same port number is assigned to a given service of both TCP and UDP.
  - ii) **The Registered Ports:** 1024 through 49151. These are not controlled by the IANA, (Internet Assigned Numbers Authority) but the IANA registers and lists the uses of these ports as a convenience to the community.
  - iii) **Dynamic private ports:** 49152 through 65535. These are what we call as ephemeral ports. It is also called Short lived transport protocol port that is created by OS when a program requests any available user ports.

#### Stop-and-Wait Protocol :

- ✓ Stop-and-Wait Protocol means that the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame.
- ✓ If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use.
- ✓ Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service.
- ✓ To prevent the receiver from becoming over-whelmed with frames, we need to tell the sender to slow down.

#### ALGORITHM :

1. Start the program.
2. Create the socket by specifying the address and establishes the connection
3. Send and receive information.
4. The sender sends one frame, stops until it receives confirmation from the receiver and then sends the next frame.
5. Stop the program.

// Program : Server Side

```
// Server.c
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

void error(const char *msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
    int sockfd, newsockfd, portno;
    socklen_t clilen;
    char buffer[256];
    struct sockaddr_in serv_addr, cli_addr;
    int n;
    if (argc < 2)
    {
        fprintf(stderr, "You have't provided port Number, please
enter port number\n");
        exit(1);
    }
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("Server : error at port opening");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("Server : Error at binding");
    listen(sockfd, 5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd,
        (struct sockaddr *) &cli_addr,
        &clilen);
    if (newsockfd < 0)
        error("Server : Error while accepting");
    bzero(buffer, 256);
    n = read(newsockfd, buffer, 255);
    if (n < 0) error("Server : ERROR reading from socket");
    printf("MY message is : %s\n", buffer);
}
```

```

        n = write(newsockfd,"I Have Recieved your message",30);
        if (n < 0) error("Server : Error while writing to server");
        close(newsockfd);
        close(sockfd);
        return 0;
    }

// Program : Client Side
// Client.c

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

void error(const char *msg)
{
    perror(msg);
    exit(0);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[256];
    if (argc < 3) {
        fprintf(stderr,"usage %s Enter your hostname & port number \n",
argv[0]);
        exit(0);
    }
    portno = atoi(argv[2]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("Client : Error While opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"Client : Error, host not found\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
        (char *)&serv_addr.sin_addr.s_addr,
        server->h_length);
    serv_addr.sin_port = htons(portno);

```

```

    if (connect(sockfd, (struct sockaddr *)
&serv_addr, sizeof(serv_addr)) < 0)
        error("Client : Error while connecting to server");
    printf("Please enter your message: ");
    bzero(buffer,256);
    fgets(buffer,255,stdin);
    n = write(sockfd,buffer,strlen(buffer));
    if (n < 0)
        error("Client : Error while writing to socket");
    bzero(buffer,256);
    n = read(sockfd,buffer,255);
    if (n < 0)
        error(" Client : Error while reading from socket");
    printf("%s\n",buffer);
    close(sockfd);
    return 0;
}

```

Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
<b>A</b>	<b>Problem statement</b>	<b>10</b>	
<b>B</b>	<b>Design &amp; specifications</b>	<b>10</b>	
<b>C</b>	<b>Expected output</b>	<b>10</b>	
<b>Record</b>			
<b>D</b>	<b>Simulation/ Conduction of the experiment</b>	<b>15</b>	
<b>E</b>	<b>Analysis of the result.</b>	<b>15</b>	
	<b>Viva</b>	<b>40</b>	
	<b>Total</b>	<b>100</b>	
<b>Scale down to 10 marks</b>			<div style="border-bottom: 1px solid black; width: 50px; margin: 0 auto;"></div> <b>10</b>
<b>Staff Signature with Date</b>			

### EXPERIMENT 6

## TESTING AND VERIFICATION OF NETWORK CONFIGURATIONS USING PACKET TRACER.

**Objective:** To cable a network according to the given network topology and test and verify configurations using packet tracer by using ping commands.



Device	Interface	IP Address	Subnet Mask	Default Gateway
R1	Fa0/0	192.168.1.1	255.255.255.0	N/A
	S0/0/0	192.168.2.1		N/A
R2	Fa0/0	192.168.3.1	255.255.255.0	N/A
	S0/0/0	192.168.2.2	255.255.255.0	N/A
PC1	N/A		255.255.255.0	192.168.1.1
PC2	N/A	192.168.3.10	255.255.255.0	192.168.3.1

### Procedure

1. Complete the given table with reference to the topology diagram.
2. Perform basic configuration tasks on a router and end devices.
3. Create routing table entries for the routers.
4. Test and verify configurations using ping commands.

#### 1. To configure end devices (PC):

Click on end device- Go to global settings- Give gateway address  
Go to fast Ethernet- Give corresponding IP address and subnet mask-Close.

To Configure Routers(R): Verify whether the router is on.

Click on router- Go to CLI-Write the configuration command functions as follows:

1. **\*Router>Enable (en)**
2. **Router# Configure terminal (config t)**
3. **Router (config)# interface fa0/0**

**4. Router (config-if)#ip address 192.168.1.1 255.255.255.0**

**5. Router (config-if)#no shut**

**Link goes up.**

**Router (config-if)#exit**

Repeat the same procedure for other interfaces.

**Note:** If there is clock indication, then use the command

**Router (config-if)#clock rate 64000** after giving IP address and then give no shut down command.

2. To create routing table entries:

For the PC1 to communicate with PC2, a route should be established from PC1-R1-R2-PC2. Use the following command to create this route.

**Router (config-if)#ip route destination network address subnet mask next hop address**

For router R1 the command would be

**Router (config-if)#ip route 192.168.3.0 255.255.255.0 192.168.2.2**

For router R2 the command would be

**Router (config-if)#ip route 192.168.1.0 255.255.255.0 192.168.2.1**

3. To test and verify configurations using ping commands.

Click on simple PDU icon present in the right corner of the packet tracer window to get a packet.

Click the packet on respective source (PC1) and then on respective destination (PC2) and verify whether the ping is successful which appears at right bottom corner of the packet tracer window.

Alternatively,

Go to the router CLI window and give the following commands

**Router# ping 192.168.1.10** (IP address of the destination host)

If the ping is successful it is indicated by success rate 100% , if unsuccessful success rate will 0%.

\*Letters in bold need not be typed.

**Inference:**



Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			<hr/> 10
<b>Staff Signature with Date</b>			

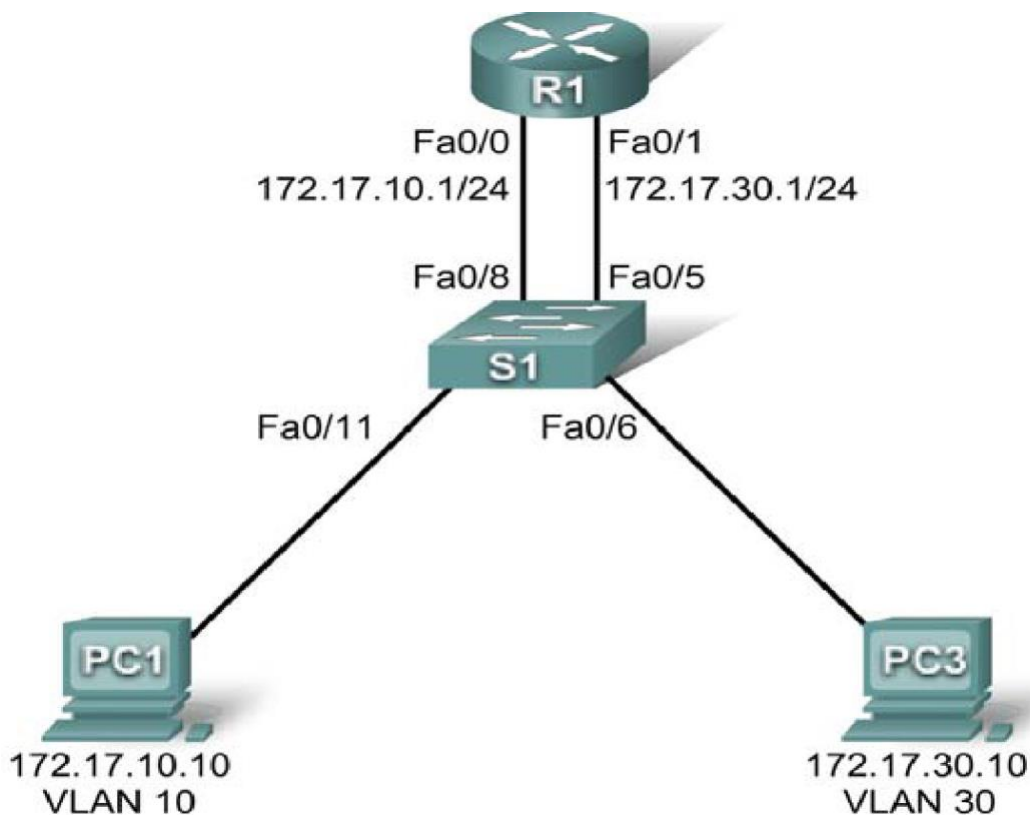
### EXPERIMENT 7

## CONFIGURATION OF INTER VLAN NETWORK

**Objective:** Configuring Traditional Inter-VLAN Routing and test and verify configurations using packet tracer

1

**Topology Diagram:**



**Addressing table:**

DEVICE	INTERFACE	IP ADDRESS	SUBNET MASK	DEFAULT GATEWAY
R1	Fa0/0	172.17.10.1	255.255.255.0	N/A
	Fa0/1	172.17.30.1	255.255.255.0	N/A
PC1	NIC	172.17.10.10	255.255.255.0	172.17.10.1
PC3	NIC	172.17.30.10	255.255.255.0	172.17.30.1

### Introduction

In this activity, you will configure traditional inter-VLAN routing simply by configuring two Fast Ethernet interfaces on a router. R1 has two connections to S1—one for each of the two VLANs. S1 and R1 already have basic configurations. The user EXEC password is **cisco**, and the privileged EXEC password is **class**. You will complete the configuration by adding VLANs to S1 and assigning VLANs to the correct ports. Then you will configure R1 with IP addressing. In traditional inter-VLAN routing, there are no additional, VLAN-related configurations needed on R1.

### Task 1: Test Connectivity without Inter-VLAN Routing

#### Step 1. Ping between PC1 and PC3.

Wait for switch convergence. The link lights on the switch connecting to PC1 and PC3 change from amber to green. When the link lights are green, ping between PC1 and PC3. Because the two PCs are on separate networks and the router is not configured, they cannot communicate with one another, so the ping fails.

#### Step 2. Switch to Simulation mode to monitor pings.

- Switch to **Simulation** mode by clicking the **Simulation** tab or pressing **Shift+S**
- Use the **Add Simple PDU** tool to ping between PC1 and PC3.
- Click **Capture/Forward** to see the steps the ping takes between PC1 and PC3.
- Notice how the ping cannot even cross the switch.

Your completion percentage should be 0%.

### Task 2: Add VLANs

#### Step 1. Create VLANs on S1.

Return to **Realtime** mode. Create two VLANs on S1, one for PC1 and one for PC3. PC1 belongs to VLAN10, and PC3 belongs to VLAN 30. To create the VLANs, issue the **vlan 10** and **vlan 30** commands in global configuration mode.

S1#**configure terminal**

S1(config)#**vlan 10**

S1(config-vlan)#**vlan 30**

To check whether the VLANs were created, issue the **show vlan brief** command from the privileged EXEC prompt.

S1#**show vlan brief**

VLAN Name Status Ports

-----

1 default active Fa0/1, Fa0/2, Fa0/3, Fa0/4  
 Fa0/5, Fa0/6, Fa0/7, Fa0/8  
 Fa0/9, Fa0/10, Fa0/11, Fa0/12

Fa0/13, Fa0/14, Fa0/15, Fa0/16  
 Fa0/17, Fa0/18, Fa0/19, Fa0/20  
 Fa0/21, Fa0/22, Fa0/23, Fa0/24  
 Gig1/1, Gig1/2

10 VLAN0010 active  
 30 VLAN0030 active  
 1002 fddi-default active  
 1003 token-ring-default active  
 1004 fddinet-default active  
 1005 trnet-default active

## Step 2. Assign the VLANs to ports.

Each port on the switch is assigned to a VLAN to allow for inter-VLAN communication. Assign the switch ports as follows:

- Assign the Fa0/5 and Fa0/6 interfaces to VLAN 30.
- Assign the Fa0/8 and Fa0/11 interfaces to VLAN 10.

To assign a VLAN to a port, enter the interface configuration. For Fa0/8, the command is **interface fa0/8**.

The **switchport access vlan 10** assigns VLAN 10 to that port. The **switchport mode access** command sets the port to access mode.

```
S1(config)#interface fa0/8
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 10
```

Repeat the above steps for Fa0/5, Fa0/6, and Fa0/11, assigning the correct VLANs to each interface.

## Step 3. Test connectivity between PC1 and PC3.

Now issue a ping between PC1 and PC3. The ping should still fail.

## Step 4. Check results.

Your completion percentage should be 45%. If not, click **Check Results** to see which required components are not yet completed.

## Task 3: Configure IP Addressing

Configure the Fa0/0 interface of R1 with the IP address 172.17.10.1 and subnet mask 255.255.255.0.

Configure the Fa0/1 interface with the IP address 172.17.30.1 and subnet mask 255.255.255.0. Issue the **no shutdown** command on both interfaces to bring them up.

```
R1(config)#interface fa0/0
R1(config-if)#ip address 172.17.10.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#interface fa0/1
R1(config-if)#ip address 172.17.30.1 255.255.255.0
R1(config-if)#no shutdown
```

### Step 2. Check results.

Your completion percentage should be 100%. If not, click **Check Results** to see which required components are not yet completed.

### Task 4: Test Connectivity Again

#### Step 1. Ping between PC1 and PC3.

Wait for STP to converge. Then ping from PC1 to PC3. The ping should succeed.

#### Step 2. Switch to simulation mode to monitor pings.

- Switch to simulation mode by clicking the **Simulation** tab or pressing **Shift+S**.
- Click **Capture/Forward** to see the steps the ping takes between PC1 and PC3.
- Watch as the ping goes from PC1 through S1, then to R1, then back to S1, and finally to the PC3.

### Inference:

Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			<div></div> <div>10</div>
<b>Staff Signature with Date</b>			

## EXPERIMENT 8

### IMPLEMENT RSA ALGORITHM USING C

**Objective:** To write a C program to simulate RSA algorithm

Algorithm :

1. Choose prime numbers  $p$  and  $q$ .
2. Multiply the above two primes to find  $n$ , ie  $n=pxq$
3. Calculate another number  $\Phi=(p-1) \times (q-1)$ .
4. Choose a random integer  $e$ . And then calculate  $d$  so that  $d \times e \equiv 1 \pmod{\Phi}$
5. Calculate the cipher text, using  $e$  and  $n$  as follows:

$$C = \text{Plaintext}^e \pmod{n}$$

6. When receives the cipher text , use private key  $d$  to decrypt the message:

$$P = C^d \pmod{n}$$

*Implement the above algorithm in C program.*

Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			<hr/> 10
<b>Staff Signature with Date</b>			

### EXPERIMENT 9



## **CONFIGURATION AND TESTING OF GIVEN NETWORK PACKET TRACER**

## **USING**

**Objective:** To configure and test a given network using Packet Tracer.

Network

Configuration table:

Command Line Interface code

Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			<hr/> 10
<b>Staff Signature with Date</b>			

### EXPERIMENT 10

**Design an Ethernet network comprising of 25 nodes and calculate Packet delivery ratio given the packet size to be 1024 bytes, consider the following application layer protocols**

**a ) FTP b) CBR**

**Objectives:** Analysis of Ethernet base network

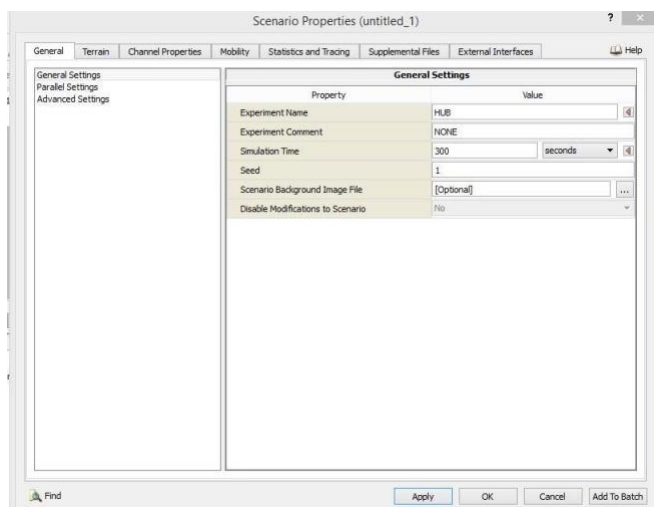
Analysis of UDP and TCP Traffic

Steps to create the experiment:

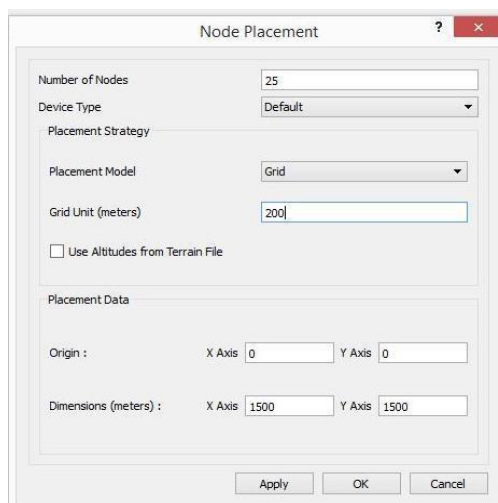
- Open the QualNet QUI and Place a 25 nodes in a Grid manner:

Step1: Go to scenario property set the simulation time 300sec and terrain size as per user defined standard, default size 1500\*1500 sq. m.

Step2: To place a nodes at a time on a canvas, go to tool -> node placement enter the details filled in fig 1:

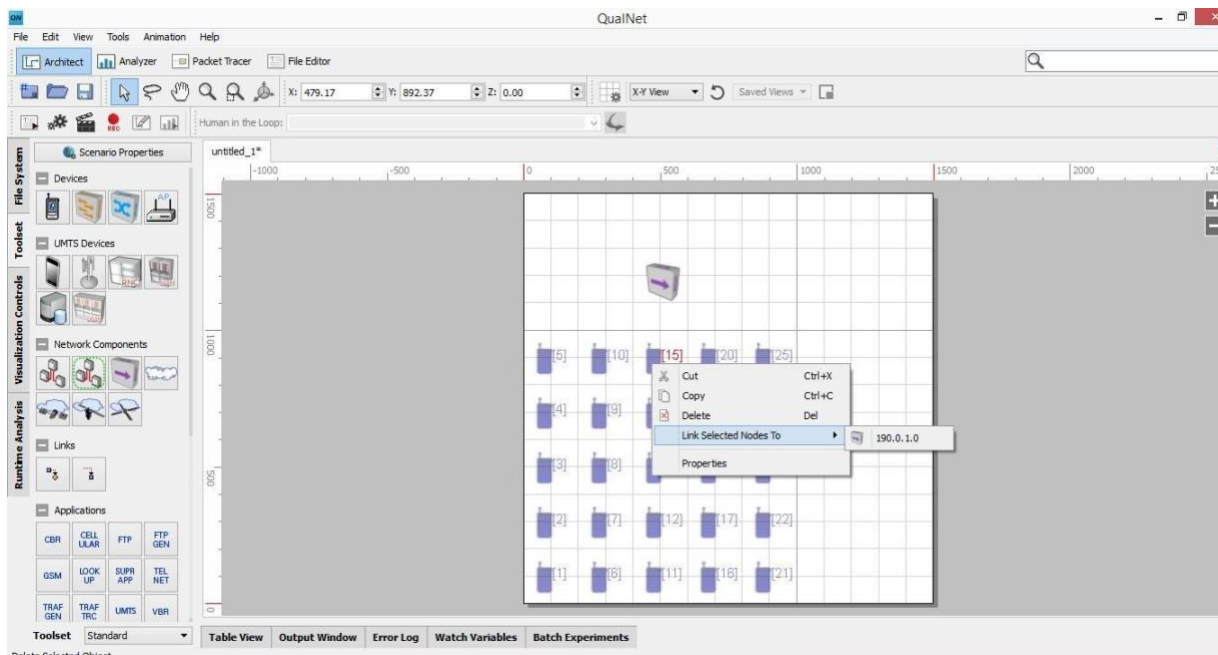


Scenario Property



Node Placement Property

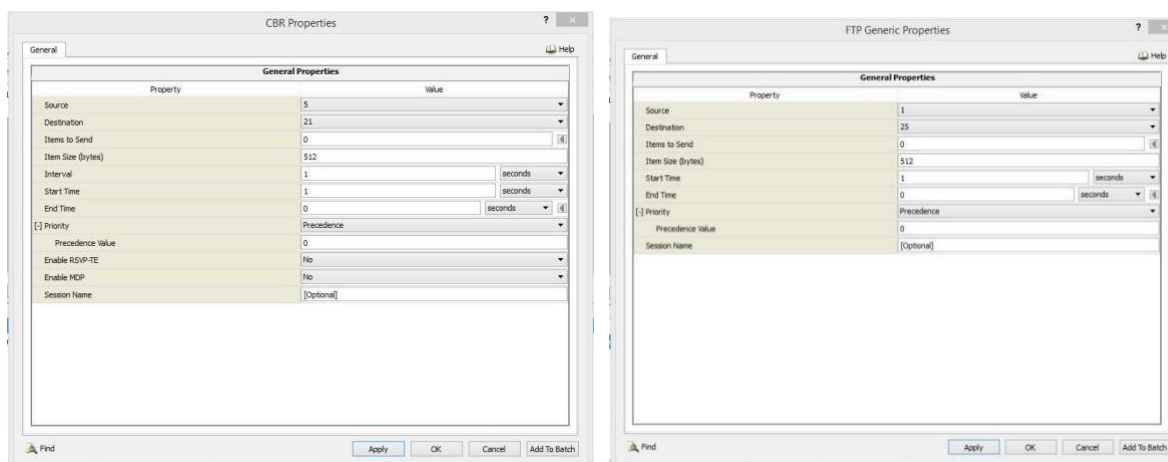
Step 3: Select the hub under network component and place on a canvas to connect the all the node to the hub, select the all node right click on the mouse go to link to the hub as shown in below figure: 1.2 Connect all the node to hub and Traffic has to assign to send the data between the nodes.







Step4: Select CBR and FTP-GEN under Application in toolset, assign the traffic between the nodes

From node 5 to node 21 assign a CBR traffic it's a UDP based protocol and from node 1 to node 25 assign a FTP-GEN traffic it's a TCP based protocol.

To configure the CBR and Application parameter go to Table view -> Application tab here configure the CBR and FTP-GEN property as shown in fig

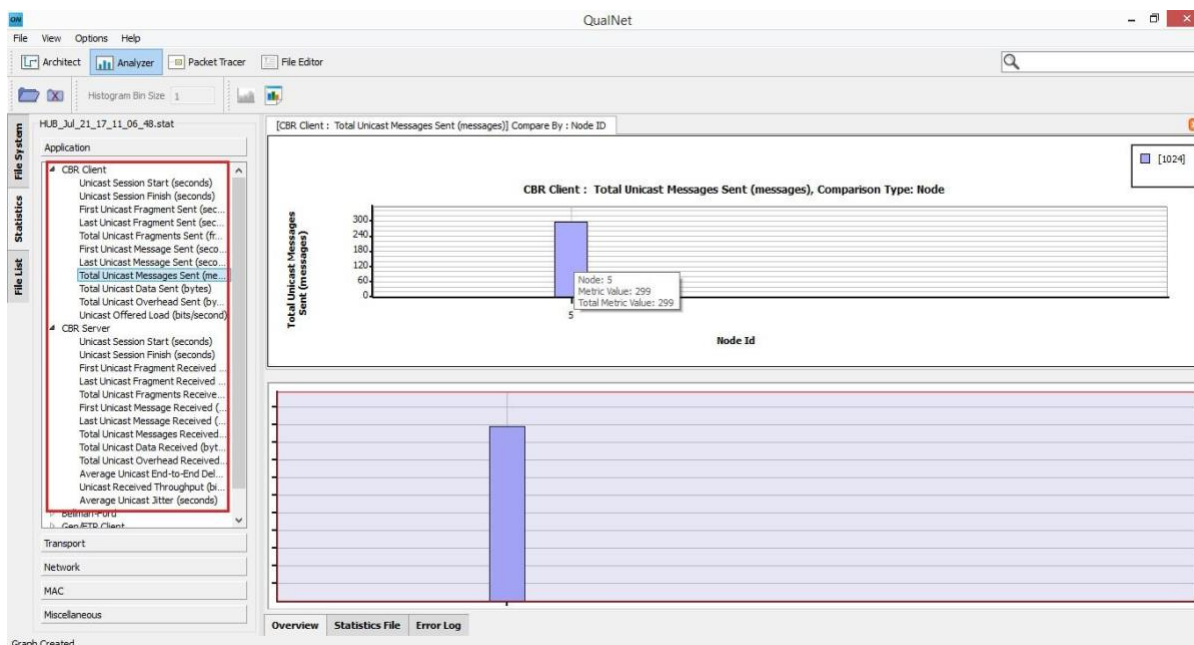


Step 5: Save  and Run  the scenario, play  the scenario to see the statistics of current scenario press the icon 

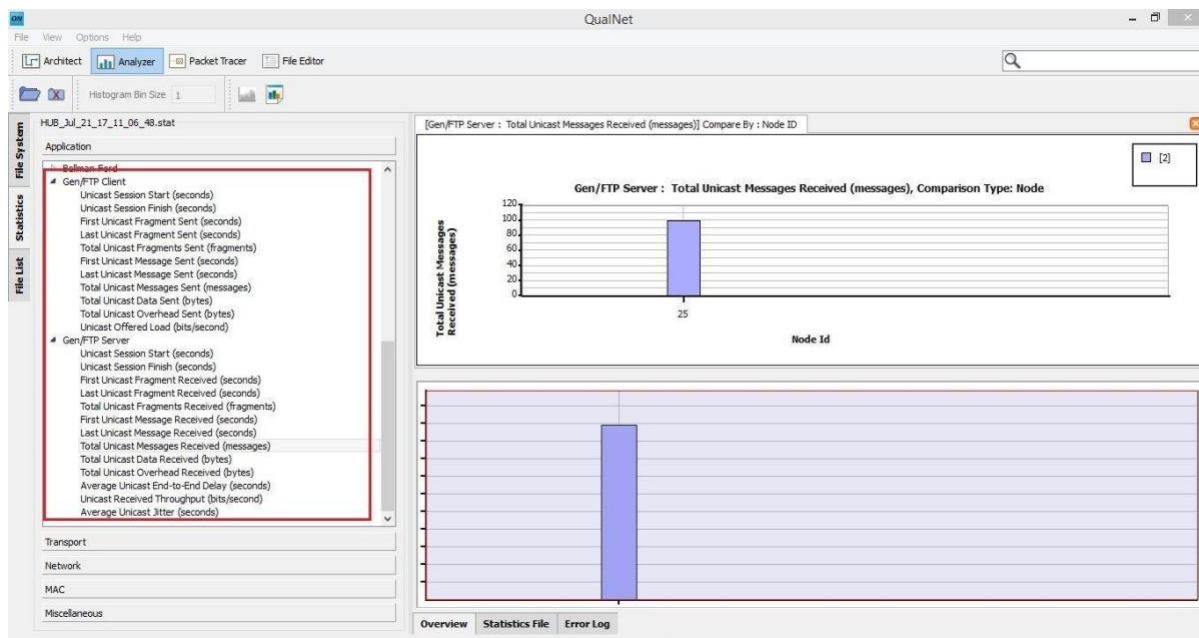
### Analysis:

Analyze the statistics based on layer wise under application layer CBR and FTP statistics is available and in Transport layer UDP and TCP statistics will be available and in Network layer will get the IP related statistics and In Mac layer 802.3 standard statistics will be available.

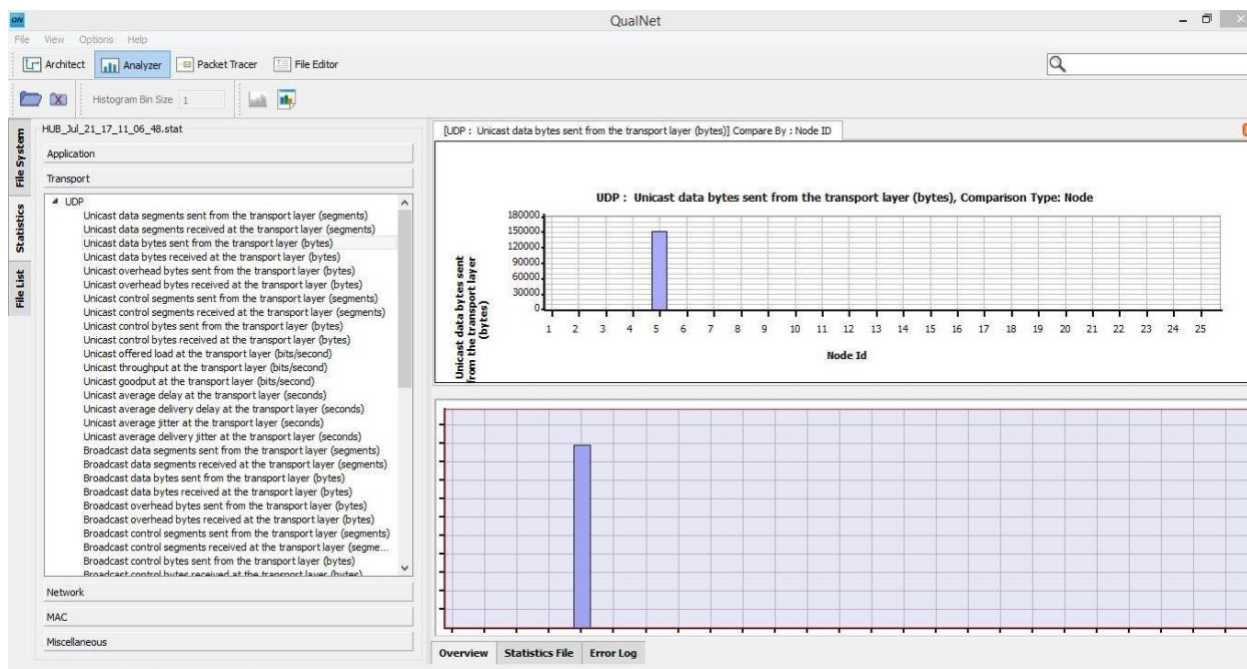
CBR Statistics in analyzer Tab:



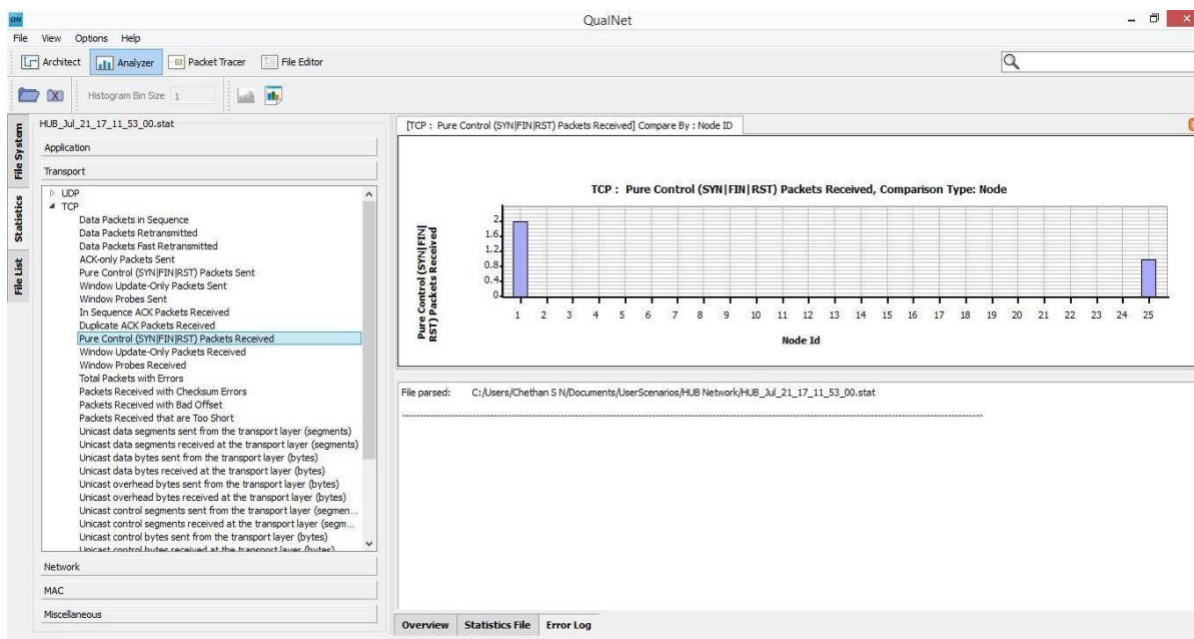
FTP Statistics in analyzer Tab:



UDP Statistics in analyzer Tab:



TCP Statistics in analyzer Tab:



Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
<b>A</b>	<b>Problem statement</b>	<b>10</b>	
<b>B</b>	<b>Design &amp; specifications</b>	<b>10</b>	
<b>C</b>	<b>Expected output</b>	<b>10</b>	
<b>Record</b>			
<b>D</b>	<b>Simulation/ Conduction of the experiment</b>	<b>15</b>	
<b>E</b>	<b>Analysis of the result.</b>	<b>15</b>	
	<b>Viva</b>	<b>40</b>	
	<b>Total</b>	<b>100</b>	
<b>Scale down to 10 marks</b>			<b>10</b>
<b>Staff Signature with Date</b>			

## EXPERIMENT 11

### SIMULATION AND COMPARISON OF ROUTING PROTOCOLS USING Cisco Packet tracer/QualNet

#### Objective:

- i) To Simulate and Compare following Routing Protocols using Cisco Packet tracer /QualNet
  - a)Open-Shortest Path First (OSPF)
  - b) Routing Information Protocol (RIP)
  - c) Interior Gateway Routing Protocol (IGRP)
- ii) Compile and Simulate Multicast Ad-hoc On-Demand Distance Vector (MAODV) Routing Protocol using

Cisco Packet tracer /QualNet.

Sl.No	Criteria	Max Marks	Marks obtained
<b>Data sheet</b>			
A	Problem statement	10	
B	Design & specifications	10	
C	Expected output	10	
<b>Record</b>			
D	Simulation/ Conduction of the experiment	15	
E	Analysis of the result.	15	
	Viva	40	
	Total	100	
<b>Scale down to 10 marks</b>			_____
			10
<b>Staff Signature with Date</b>			



