CCN LAB

JOURNAL

By:

Vivek.R.Yalameli

2GI17EC164

R.No: 344

Div.: C

Sub: CCN Lab

Steps to be followed in Codeblocks Software

- Open Codeblocks software.
- In the main page tool bar create an empty file.
- Save the file with file name in respected folder of expesiment.
- Type the C-code in the file and save the file
- In tool bar select build project option.
- Then run the code if execution window pops up then note & verify output. else if any error then correct the code.

Experiment - 1: Implement of Bit stuffing

Theory: The data is generally in the form of fram-es. In the OSI Model, these frames are made by data link layer. The frames are made to avoid or detect the errors dusing data transfer more efficiently. If any errors occur in long string of data without frames the entire has to replace making it less efficient. If the data has fram-es then only that particular frames with error has to be replaced, making it more fast and efficient. These frames have a structure which helps the receiver easy to understand start & end of the frame. Every frame has a header, data and tail. The frame header usually is a string of specific bits that later continues into data. The issue that might arise, is if the bits in header occur in the data bits.

In bit stuffing, the header should be a sequence of 6 1s and for every 5 consecuti-ve 1s in the data it is made to add 1 '0' bit to da-ta to make it different from header. The stuffed 0 bit is taken out at the receiver and data is safely received.

* Algorithm:

step 1: Create an array to hold the data bits.

step 2: Read the size and value of bits from user.

step 3: In a loop, check if the bit is equal to 1, if it is then check next 5 bits if they are 1 then in stuffed sequence add '0' bit. keep adding all the data bits in stuffed seq.

step 4: continue step 3 till end of data bits.

step 5: After stuffing, print the sequence using a loop

* Results: The bit stuffing C code takes data input & gives bit stuffed output.

input 1 : 01111111 output : 011111[0]11

input 2 : 0000011111 output : 00000111111[0]

CALCULATIONS:

i) Size = 8

→ Data : 01111111

After bit stuffing = 0111111[0]11

↑ stuffed bit

ii) Size = 10

→ Data = 0000011111

After bit stuffing = 00000111110

↑ stuffed bit.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    int a[20],b[30],i,j,k,count,n;
    printf("Enter frame size (Example: 8):");
    scanf("%d",&n);
    printf("Enter the frame in the form of 0 and 1 :");
    for(i=0; i<n; i++)
        scanf("%d",&a[i]);
    i=0;
    count=1;
    j=0;
    while(i<n)
    {
        if(a[i]==1)
        {
            b[j]=a[i];
            for(k=i+1; a[k]==1 && k<n && count<5; k+
            {
                j++;
                b[j]=a[k];
                count++;
                if(count==5)
                {
                    j++;
                    b[j]=0;
                }
                i=k;
            }
        }
        else
        {
            b[j]=a[i];
        }
        i++;
        j++;
    }
    printf("After Bit Stuffing :");
    for(i=0; i<j; i++)
        printf("%d",b[i]);
    getch();
}
```


"D:\e\Bank Account\expt1.exe"
Enter frame size (Example: 8):12
Enter the frame in the form of 0 and 1 :1 1 1 1 1 1 1 1 1 1 1 1
After Bit Stuffing :111110111111
Frame with flags: 01111110111111011111101111110

Kavita

Page No. :
Subject :
Date :

Experiment 2: Implementation of Byte Stuffing

Theory: The data in an OSI model when sent to data link layer from network layer is converted into frames before transmitting it to next layer. The long string of data is converted into smaller parts called frames. To identify the begining and end of frame a flag, a header and a trailer are used in between which the data is sent. If any of the sequence from flag or header or trailer are used in between which the data leads to misreading of data and cases error. To avoid this bit stuffing and byte stuffing methods are used. As in bit stuffing a non-information bit is used to avoid this error. In byte stuffing, a byte is insested if the bits similar to flag occuss in data. This is called as escape chasactes.

This escape chasacter is insested in between the flag bits in data. This tells the receiver that following bits have to be considered as data. If the bits in escape chasactes is insested befose it. This tells that receiver that the next escape chasactes is acthally data.

Page No. :
Subject :
Date :

Algorithm:

step 1: start

step 2: Declare two arrays with size taken from users.

step 3: In a loop print all characters in frames.

step 4: In a loop go through every characters and if it's 's' then add 's' in resulting frame that is for escape character. if it is 'f' then add 's' for 'f' being frame. if the characters is other than 's' or 'f' then it is unaffected.

step 5: print the final result with byte stuffed

step 6: stop

Results: The sequence is well received by code and the escape characters are added to

input = a b s c f o/p: a b s s c s f

input: a b s f o/p: a b s s s f.

Calculations

1) Frame size = 5

frame = a b s c f

output = a b [s] s c [s] f

↑ escape characters

2) Frame size = 4

frame = a b s f

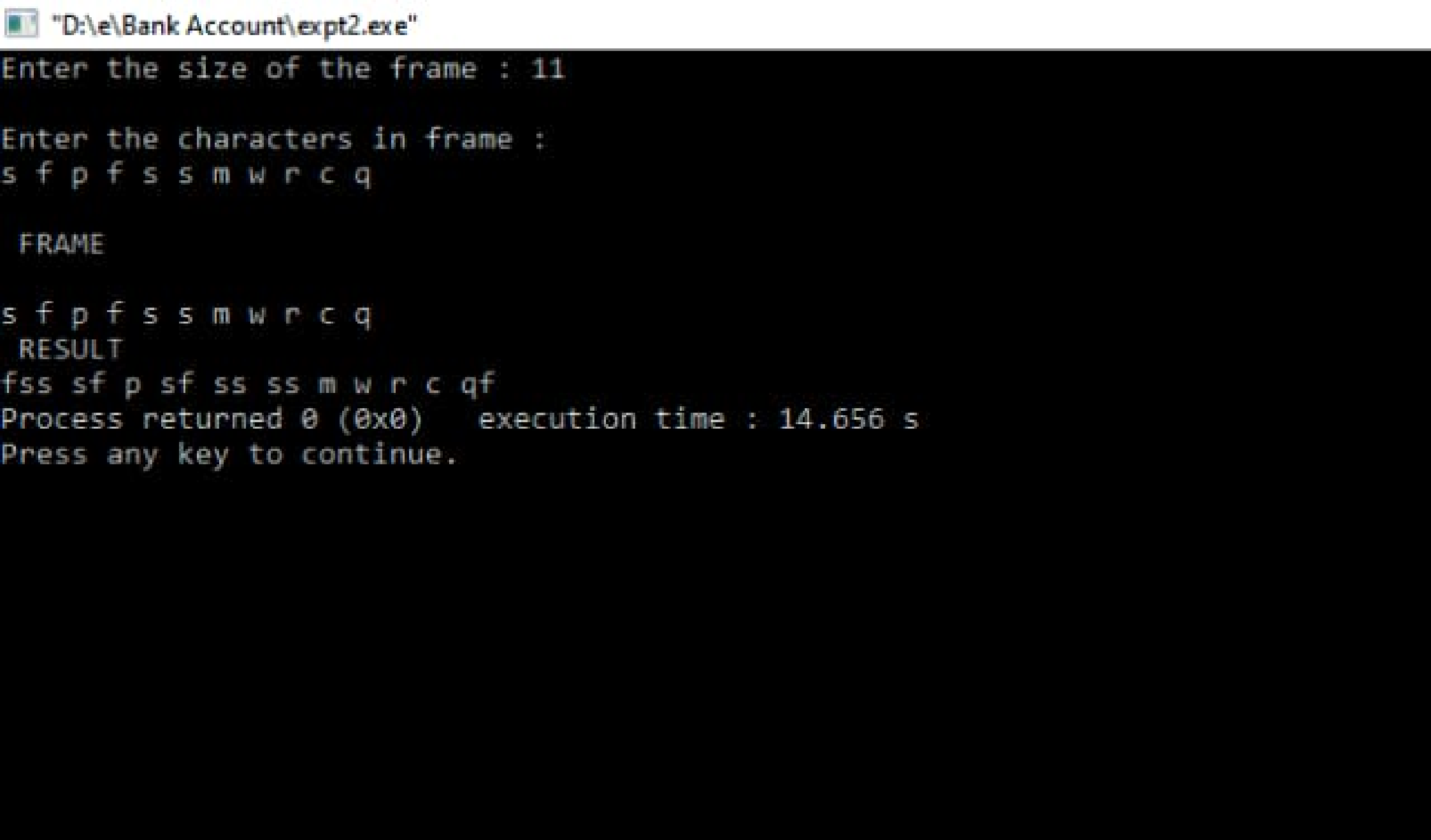
output = a b [s] s [s] f

↑ escape characters

```
#include<stdio.h>
#include<string.h>
main(){
        char a[20],b[20];
        int i,n,j;
        char f,s;
        printf("Enter the size of the frame : ");
        scanf("%d",&n);
        n=n*2;
        printf("\nEnter the characters in frame : \n");
        for(i=0;i<n;i++)
                scanf("%c",&a[i]);
        printf("\n FRAME \n ");
        for(i=0;i<n;i++)
                printf("%c",a[i]);
        j=0;
        for(i=0;i<n;i++)
        {
                if(a[i]=='f')
                {
                        b[j]='s';
                        j++;
                        b[j]=a[i];

                }
                else if(a[i]=='s')
                {
                        b[j]='s';
                        j++;
                        b[j]=a[i];
                }
                else
                        b[j]=a[i];

                j++;
        }
        printf("\n RESULT \n");
       // printf("f");
        for(i=0;i<j;i++)
        {
printf("\t");
                printf("%c",b[i]);
        }
        //printf("\nf");
}
```


"D:\e\Bank Account\expt2.exe"
Enter the size of the frame : 11
Enter the characters in frame :
s f p f s s m w r c q
FRAME
s f p f s s m w r c q
RESULT
fss sf p sf ss ss m w r c qf
Process returned 0 (0x0) execution time : 14.656 s
Press any key to continue.

Kavita

Page No. :
Subject :
Date :

Experiment 3: Implementation of Cycle Redundancy Check Algorithm

Theory: The data is sent from transmitter and to the receiver in the form of packets called frames but to check if the data sent is actually what was transmitted there is a checksum. In this code, we obtain checksum by complementing the sum of two binary sequences. The checksums can be made by various methods. The check sum is sent to receiver and this checksum has the components that helps the receiver to verify the received signal & check sum fail to match then receiver sends a message to transmitter as the message has an error. In the blogger picture the checksum is used to compare two files in a system. which will be compared and if they are same further actions will be taken, the conditions where checksum fails are in a corrupted file, in a network connection loss or interruption by a third party.

Page No. :
Subject :
Date :

Algosithm:

step1: stast

step2: Take values from 2 assays from usess

step3: In a loop, stasting from end till begining and for each binasy nor in both assays along with cassy we have total 8 possible combinations.

step4: Aftes forming the sum & make ~~en~~ in a comp -lement the sum & make checksum assay.

step5: Print sum & checksum/complement.

step6: Stop.

Result: The sum assay is successfully psinted & the checksum is also shown.

input1 = 1011111 sum = 01111111

input2 = 0100000 checksum = 10000000

input1 = 11111 sum = 111110

input2 = 11111 checksum = 000001

Calculations:

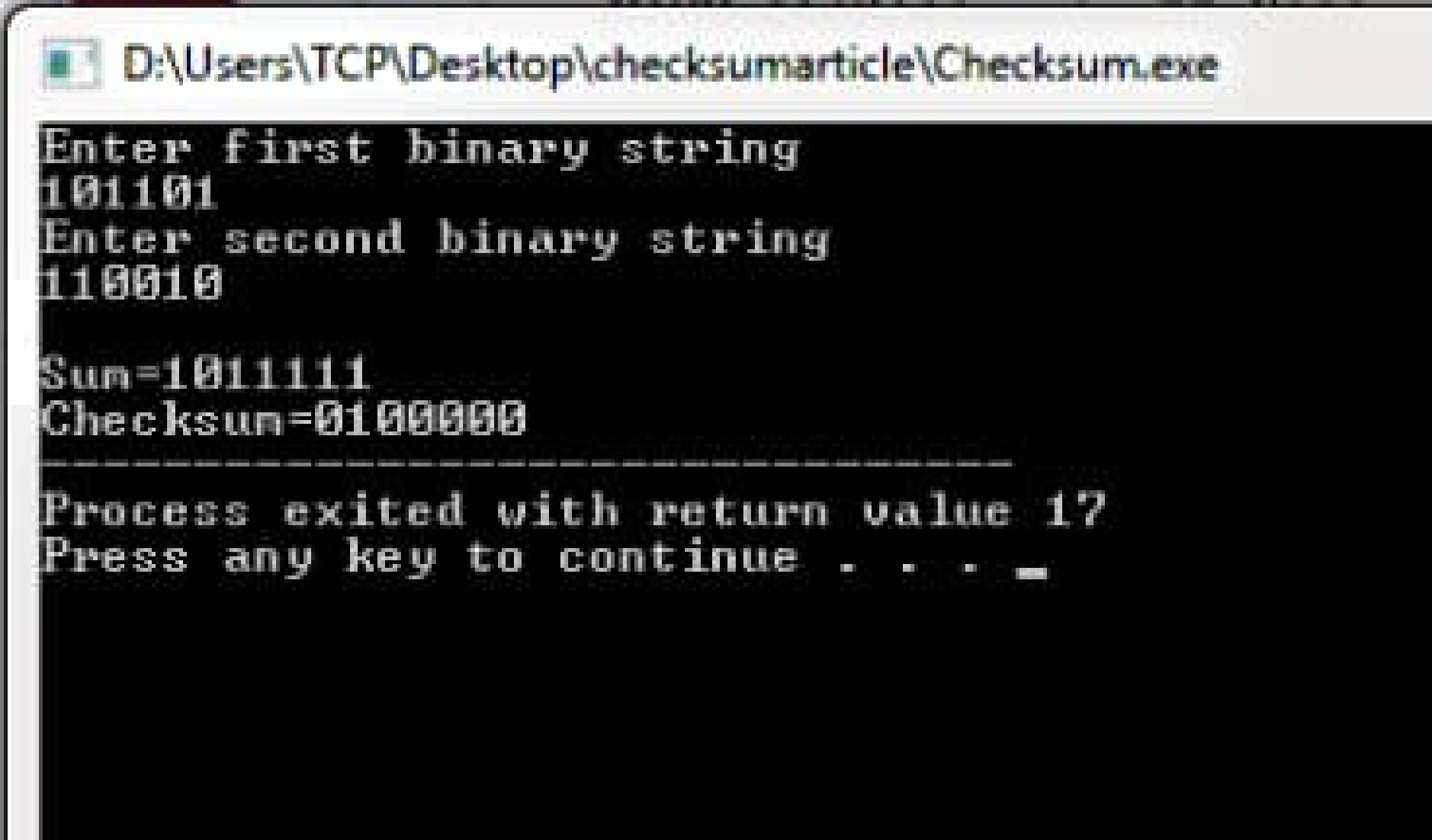
1) array 1 = 10111111
   array 2 = 01000000
   sum = 01111111
   checksum = 10000000

2) array 1 = 11111
   array 2 = 11111
   sum = 11110
   checksum = 00001

```
#include<stdio.h>
#include<string.h>
main(){
        char a[20],b[20];
        int i,n,j;
        char f,s;
        printf("Enter the size of the frame : ");
        scanf("%d",&n);
        n=n*2;
        printf("\nEnter the characters in frame : \n");
        for(i=0;i<n;i++)
                scanf("%c",&a[i]);
        printf("\n FRAME \n ");
        for(i=0;i<n;i++)
                printf("%c",a[i]);
        j=0;
        for(i=0;i<n;i++)
        {
                if(a[i]=='f')
                {
                        b[j]='s';
                        j++;
                        b[j]=a[i];

                }
                else if(a[i]=='s')
                {
                        b[j]='s';
                        j++;
                        b[j]=a[i];
                }
                else
                        b[j]=a[i];

                j++;
        }
        printf("\n RESULT \n");
       // printf("f");
        for(i=0;i<j;i++)
        {
printf("\t");
                printf("%c",b[i]);
        }
        //printf("\nf");
}
```


D:\Users\TCP\Desktop\checksumarticle\Checksum.exe
Enter first binary string
101101
Enter second binary string
110010
Sum=1011111
Checksun=0100000
Process exited with return value 17
Press any key to continue . . .

Experiment 4 Implementation of Shortest Path Algorithm

Theory: Dijkstra's Algorithm is an algorithm to find shortest path between nodes of graph. The Original algorithm fixes a node as source and finds distance from that node to all other nodes. A widely used application of this algorithm is for routing.

This algorithm uses labels that are positive integers or real values which are used which are ordered. It is possible to generalise labels that are partially ordered provided the subsequent labels are monotonically non decreasing. This generalisation is generic form of this algorithm. The application of this algorithm are least cost paths are calculated for instance to establish tracks of electricity lines or pipelines. The algorithm has also been used to calculate optical long distance footpaths in Ethiopia & constrast from them with solution for situation on ground This method follows the greedy algorithm which finds the shortest path after iterating entire set of distances & finds shortest

Page No. :
Subject :
Date :

Algorithm:

1) Create cost matrix c[][] from adjacency matrix adj[][]. c[i][j] is infinity.
2) Array visited[] is initialised to zero.

```
for (i=0; i<n; i++)
    visited[i]=0;
```

3) If the vertex 0 is the source vertex then visited [0] is masked as 1.
4) Create the distance matrix, by storing the cost of vertices from vertex no. 0 to n-1.

```
for (i=1; i<n; i++)
    distance[i]=cost[0][i];
```

Initially, distance of source vertex is taken as 0 i.e. distance[0]=0;

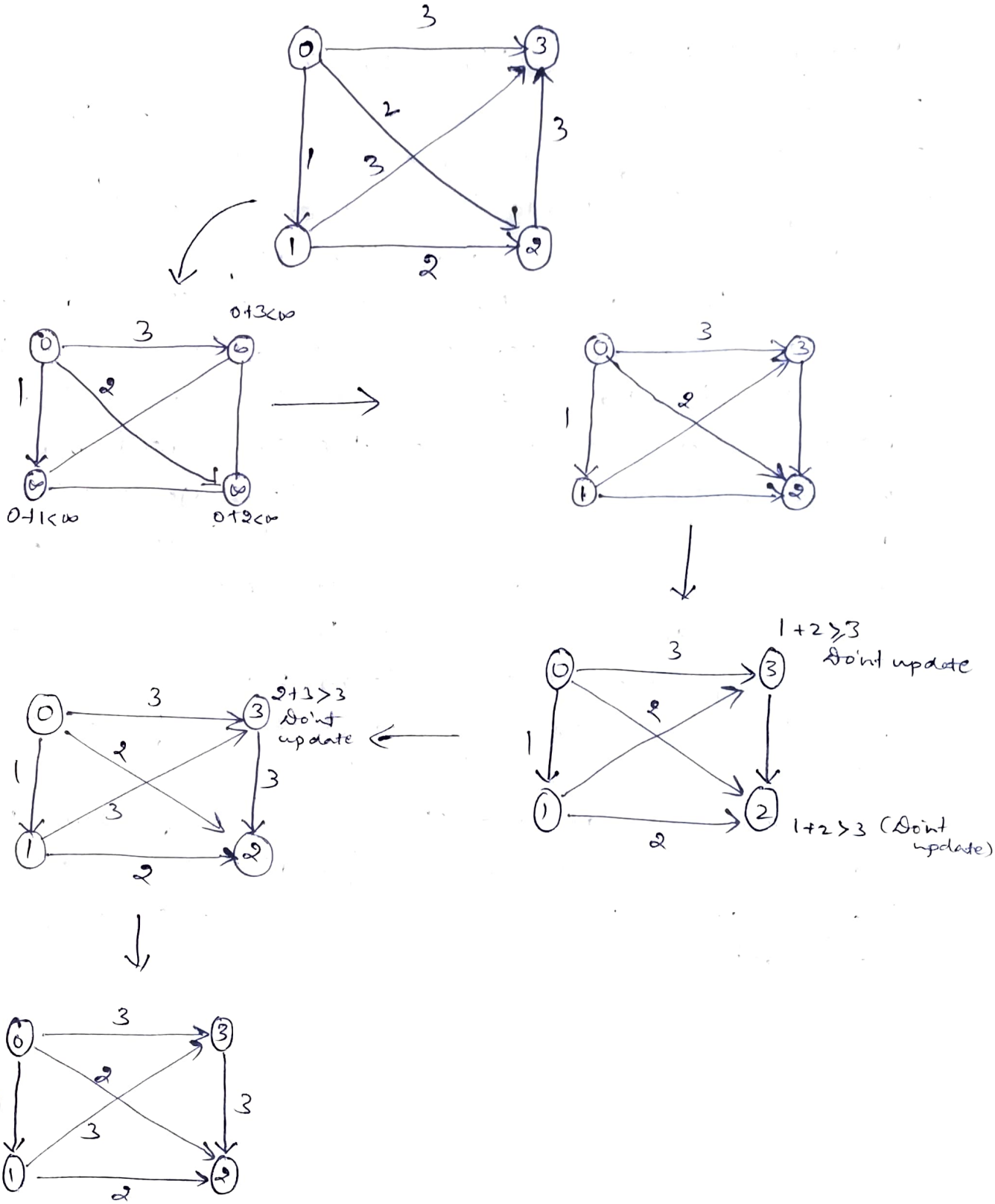
5) for (i=1; i<n; i++)
choose a vertex w, such that distance[w] is minimum & visited[w] is 0. mark visited[w] as 1. Recalculate the shortest distance of remaining vertices from source.

```
if (visited[v]==0) {distance[v] = min(distance[v] ; distance[w] + cost[w][v]).
```

Result:

For given no of nodes and distance matrix mentioned the code given relevant shortest distance.

Manual Calculations:



```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
	int G[MAX][MAX],i,j,n,u;
	printf("Enter no. of vertices:");
	scanf("%d",&n);
	printf("\nEnter the adjacency matrix:\n");

	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			scanf("%d",&G[i][j]);

	printf("\nEnter the starting node:");
	scanf("%d",&u);
	dijkstra(G,n,u);

	return 0;
}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{

	int cost[MAX][MAX],distance[MAX],pred[MAX];
	int visited[MAX],count,mindistance,nextnode,i,j;

	//pred[] stores the predecessor of each node
	//count gives the number of nodes seen so far
	//create the cost matrix
	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			if(G[i][j]==0)
				cost[i][j]=INFINITY;
			else
				cost[i][j]=G[i][j];

	//initialize pred[],distance[] and visited[]
	for(i=0;i<n;i++)
	{
		distance[i]=cost[startnode][i];
		pred[i]=startnode;
		visited[i]=0;
	}

	distance[startnode]=0;
	visited[startnode]=1;
	count=1;
	while(count<n-1)
```

```
{
    mindistance=INFINITY;

    //nextnode gives the node at minimum distance
    for(i=0;i<n;i++)
        if(distance[i]<mindistance&&!visited[i])
        {
            mindistance=distance[i];
            nextnode=i;
        }

        //check if a better path exists through nextnode
        visited[nextnode]=1;
        for(i=0;i<n;i++)
            if(!visited[i])
                if(mindistance+cost[nextnode][i]<distance[i])
                {
                    distance[i]=mindistance+cost[nextnode][i];
                    pred[i]=nextnode;
                }
    count++;
}

//print the path and distance of each node
for(i=0;i<n;i++)
    if(i!=startnode)
    {
        printf("\nDistance of node%d=%d",i,distance[i]);
        printf("\nPath=%d",i);

        j=i;
        do
        {
            j=pred[j];
            printf("<-%d",j);
        }while(j!=startnode);
}
```

"D:\e\Bank Account\expt4.exe"

```
Enter no. of vertices:5

Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

Enter the starting node:0

Distance of node1=10
Path=1<-0
Distance of node2=50
Path=2<-3<-0
Distance of node3=30
Path=3<-0
Distance of node4=60
Path=4<-2<-3<-0
Process returned 0 (0x0)   execution time : 46.464 s
Press any key to continue.
```

Kavita

Page No. :
Subject :
Date :

Experiment 5: Implementation of Distance Vector Routing Algorithm:

Theory: In distance vector routing, each router maintains a routing table indexed by containing one entry for that destination and an estimate of time or distance to that destination. The metric used weight to no of hops, time delay in msec. Total no of packets queued along the length of path or something similar. The router is assumed to know the 'distance' to each of its neighbour. If the metric queue length the routes simply examines each queue. If metric is delay, the router can measure it directly with special ECHO packets that the receiver just time stamps and sends back as fast as possible. The distance vector routing protocols include.

- Routing information protocol include
- Xerox networking system's XNS.
- Novell's IPX RIP
- Cisco's Internet Gateway routing protocol
- DEC's DNA phase IV
- Apple talk's routing table maintenance Protocol.

Kavita

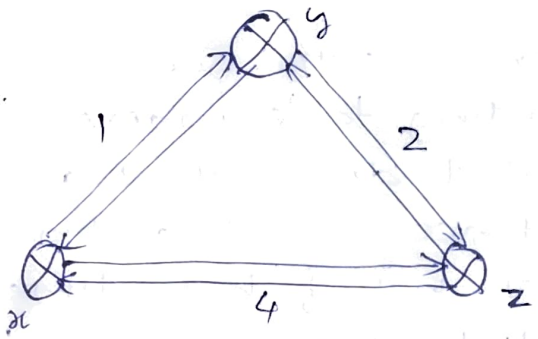
Page No. :
Subject :
Date :

Algorithms:

1) Start
2) By convection, the distance of node to itself is assigned to zero & when a node is unreachable the distance is accepted as 999.
3) Accept the input distance matrix from user dm[i][j] that represent the distance between each node in network.
4) Store the distance between nodes in a suitable variable.
5) Calculate minimum distance between two nodes by iterating. If distance between two nodes is larger than calculated alternate available path replace existing distance.
6) Print the shortest path calculated.
7) Stop

Result: The no of routers and distance matrix. gives the routing table as shown.

Manual Calculations:



Routing table

	x	y	z
x	0	1	3
y	1	0	2
z	3	2	0

```
    int from[20];
} route[10];

int main()
{
    int dm[20][20], no;

    cout << "Enter no of nodes." << endl;
    cin >> no;
    cout << "Enter the distance matrix:" << endl;
    for (int i = 0; i < no; i++) {
        for (int j = 0; j < no; j++) {
            cin >> dm[i][j];
            /*  Set distance from i to i as 0 */
            dm[i][i] = 0;
            route[i].dist[j] = dm[i][j];
            route[i].from[j] = j;
        }
    }

    int flag;
    do {
        flag = 0;
        for (int i = 0; i < no; i++) {
            for (int j = 0; j < no; j++) {
                for (int k = 0; k < no; k++) {
                    if ((route[i].dist[j]) > (route[i].dist[k] + route[k].dist[j])) {
                        route[i].dist[j] = route[i].dist[k] + route[k].dist[j];
                        route[i].from[j] = k;
                        flag = 1;
                    }
                }
            }
        }
    } while (flag);

    for (int i = 0; i < no; i++) {
        cout << "Router info for router: " << i + 1 << endl;
        cout << "Dest\tNext Hop\tDist" << endl;
        for (int j = 0; j < no; j++)
            printf("%d\t%d\t\t%d\n", j+1, route[i].from[j]+1, route[i].dist[j]);
    }
    return 0;}
```

"D:\e\Bank Account\expt5.exe"

```
Enter no of nodes.
3
Enter the distance matrix:
0 1 3
1 0 2
3 2 0
Router info for router: 1
Dest    Next Hop        Dist
1       1               0
2       2               1
3       3               3
Router info for router: 2
Dest    Next Hop        Dist
1       1               1
2       2               0
3       3               2
Router info for router: 3
Dest    Next Hop        Dist
1       1               3
2       2               2
3       3               0

Process returned 0 (0x0)   execution time : 63.766 s
Press any key to continue.
```

Kavita

Page No. :
Subject :
Date :

Part-B (Simulation)

Introduction:

Network Simulation is an important tool is developing, testing and evaluating Networks without the target hardware. It is possible to simulate network of any scale or bandwidth/delay which is impossible in real world. Any Software can be run on each of node. Extracting Information can be done by simply passing the generated trace files.

Simulations are useful if results are accurate, most simulations use a simplified FSM to model complex TCP protocol processing. NCTUNS is a high-fidelity network simulator and evaluator. By using a novel kernel re-entering simulation methodology, it provides many unique advantages over traditional network simulations and emulators

Setting up the Environment

* Setup the environment Variables i.e. setup the NCTUNS environment variables.
* Startup the dispatcher on terminal 1.
* Start up the cordinator on terminal 2.
* Startup the NCTUNS client on terminal 3.

Drawing a Network Topology

- NCTUNS is in a "Draw Topology" mode by default.
- In this mode we can change existing simulation topology.
- Move the cursor to the toolbar.
- Left Click the routes icon on toolbar.

- Now, if you left-Click anywhere on the workspace, the router is placed.
- Click the host icon to add required no of hosts.
- To add links left Click on link icon and drag them between any two nodes.
- Save the network topology.
- Take snapshot of it.
- when user switches to next mode the network topology cannot be changed.

Editing Nodes Properties:

1) A Network has many parameters to be set Eg. BW, max queue size, traffic generators.
2) In edit topology mode, the GUI automatically, find subnets in a network and assigns IP and MAC addresses to layer 3 network interfaces.
3) We have to select the node & edit the properties.
4) After Editing the node, should be changed from "Editing properties" to "Run Simulator".

Running the Simulations:

1) Entering this mode indicates that no other changes can be made.
2) In this mode many simulation files that collectively describes the simulation case will be exported. These simulation files will be transfered to simulation server. These files are stored in the main file_name.sim directory.

Kavita

Page No. :
Subject :
Date :

Playing Back Packet Animation Traces

After the simulation, the simulation server sends the simulation results to GUI after receiving it, it will store it in the results directory. It will then automatically switch to playback mode.

1) The file includes an packet animation trace file & all performances log files that the user specifies to generate outputting these performances of log file can be specified. In addition to this, application programs can generate their own data.
2) The Packet animation player plays the file & performance monitor plots the performance curve of log files.
3) The user can play it back using "Play-back" mode. The GUI will reload the results.

commands: Pause, continue, stop, Abort, Reconnect, Disconnect.

Kavita

Page No. :
Subject :
Date :

Experiment 6: Three Node Point to Point Network

Aim: To simulate a 3 node point to point network with a duplex link between them. Set the queue size and vary the bandwidth & find the no of packet dropped

Theory: A point to point communication refers to a communication connection bet 2 nodes. It refers to a direct link bet 2 devices opposed to other network topologies. To have a duplex link bet devices mean that devices can communicate with each other in both directions. There are 2 types, half and full duplex. A Queue in computer networking is collection of data packets. It consists of packets that are bound to be routed over network lined up in a sequential way. Packet drops occurs due to network congestion i.e. when content arrives at a rate which is greater the transmission rate of routes. There are 2 main policies: Drop tail and Random Early Discard. A host is a device connected to a network. A hub is used to connect them. A hub is a passive device whereas a switch is an active device, used to connect bet in a network.

Kavita

Page No. :
Subject :
Date :

Configuration:

1) Double-Click on Host 1 and Select "Add" Button and type in command,

stg-u 1024 100 1.0.1.2.

2) Double Click on Host 2 and Select "Add" button type in command,

rtg-u -w log1

3) Click on 'Node editor' Button on Host window and Select MAC tab, Select "Log statistics" & check "No of drop packet" & "No of collisions".

4) To get Queue size click on "Node Editor" select "FIFO" & Change queue size.

5) Change bandwidth of link 2 by double clicking on it.

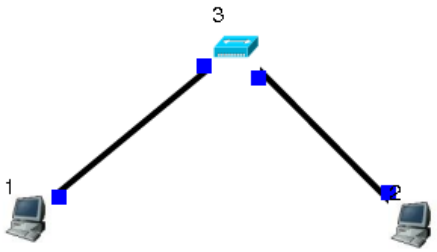
6) Removes the hub and replace it with switch

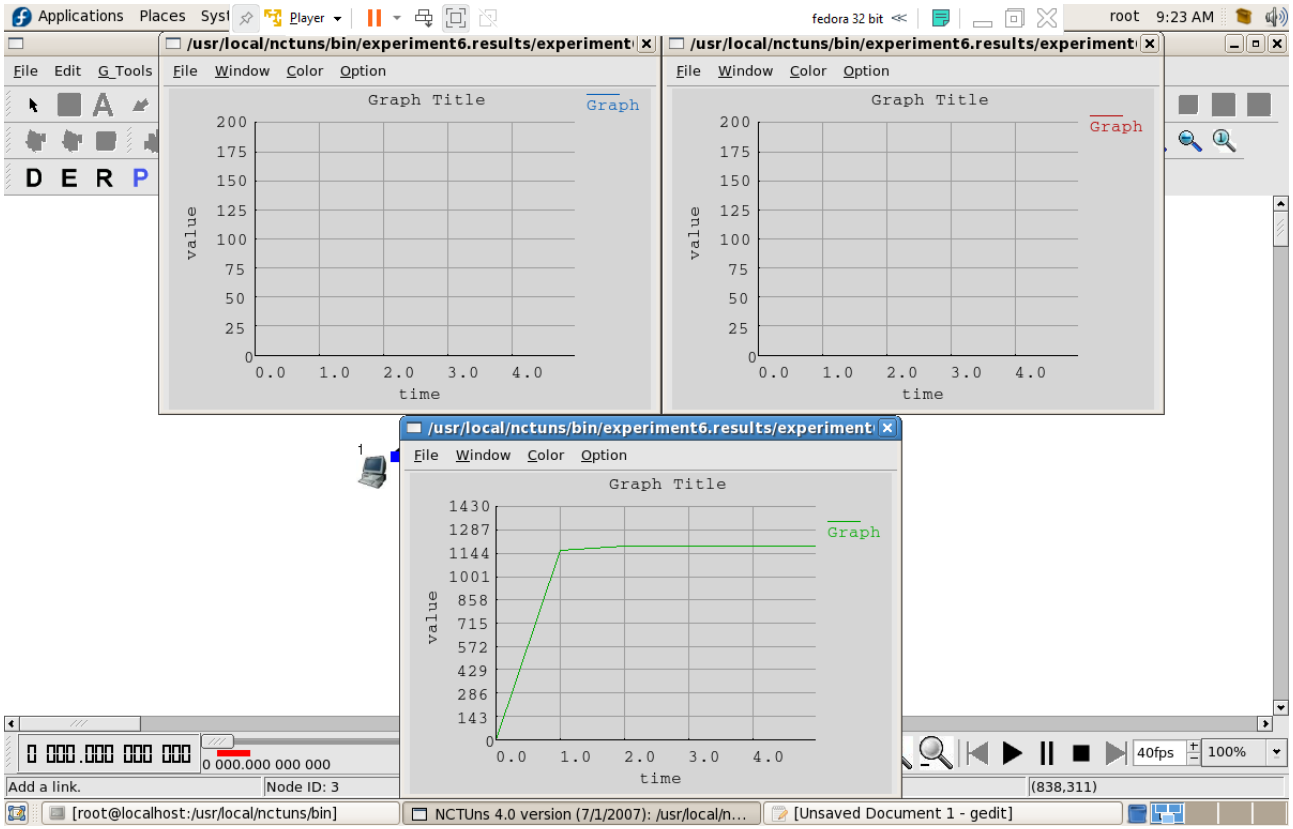
Analysis:

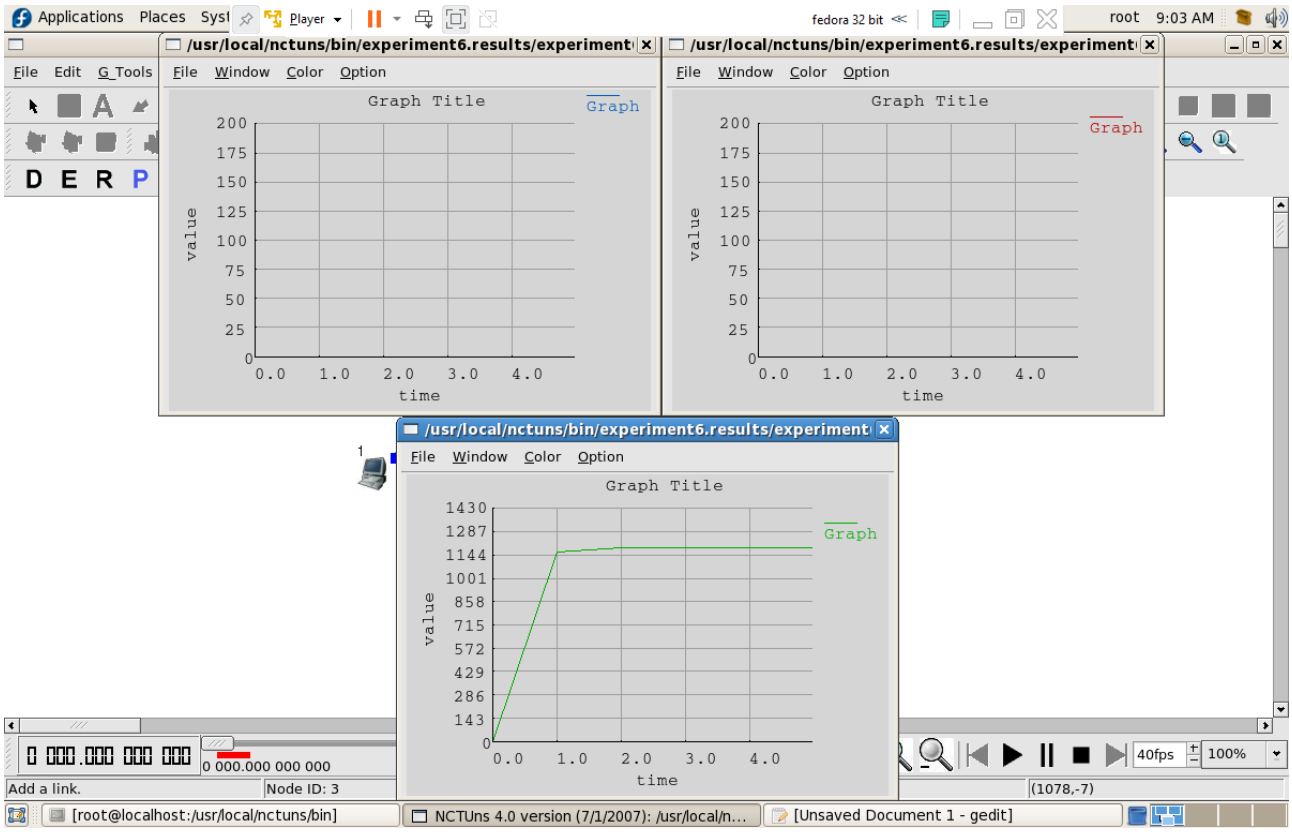
We observe that for Host1 collision and drop is 0. as there is sufficient B.W. Hence, throughput is high. On receiver end, we observe same graph as that of sender as only 100 packets are transferred to the line. So no packets are lost.

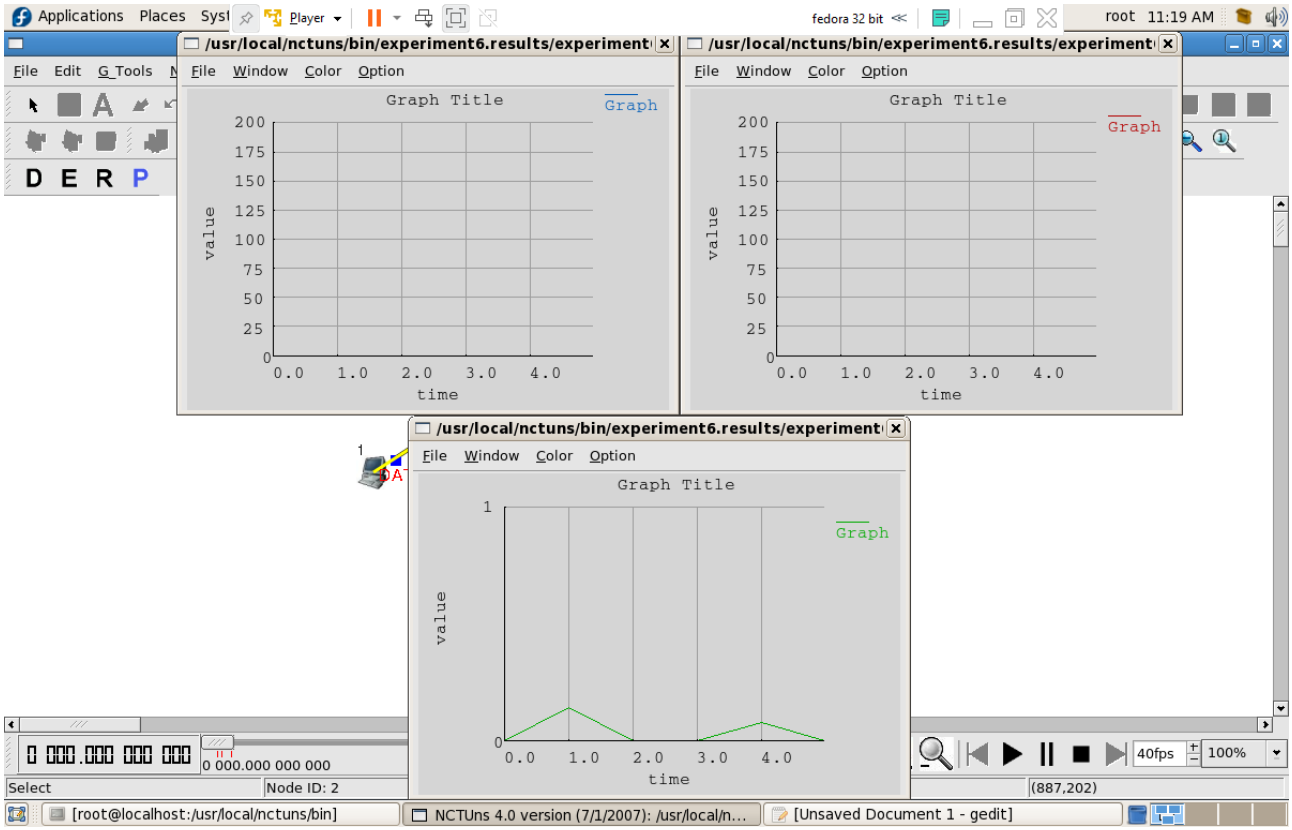
Where, we change B.W from 10 → 5 Mbps, we observe high packet collision drop as there isn't enough BW for 100 packets in second hop.

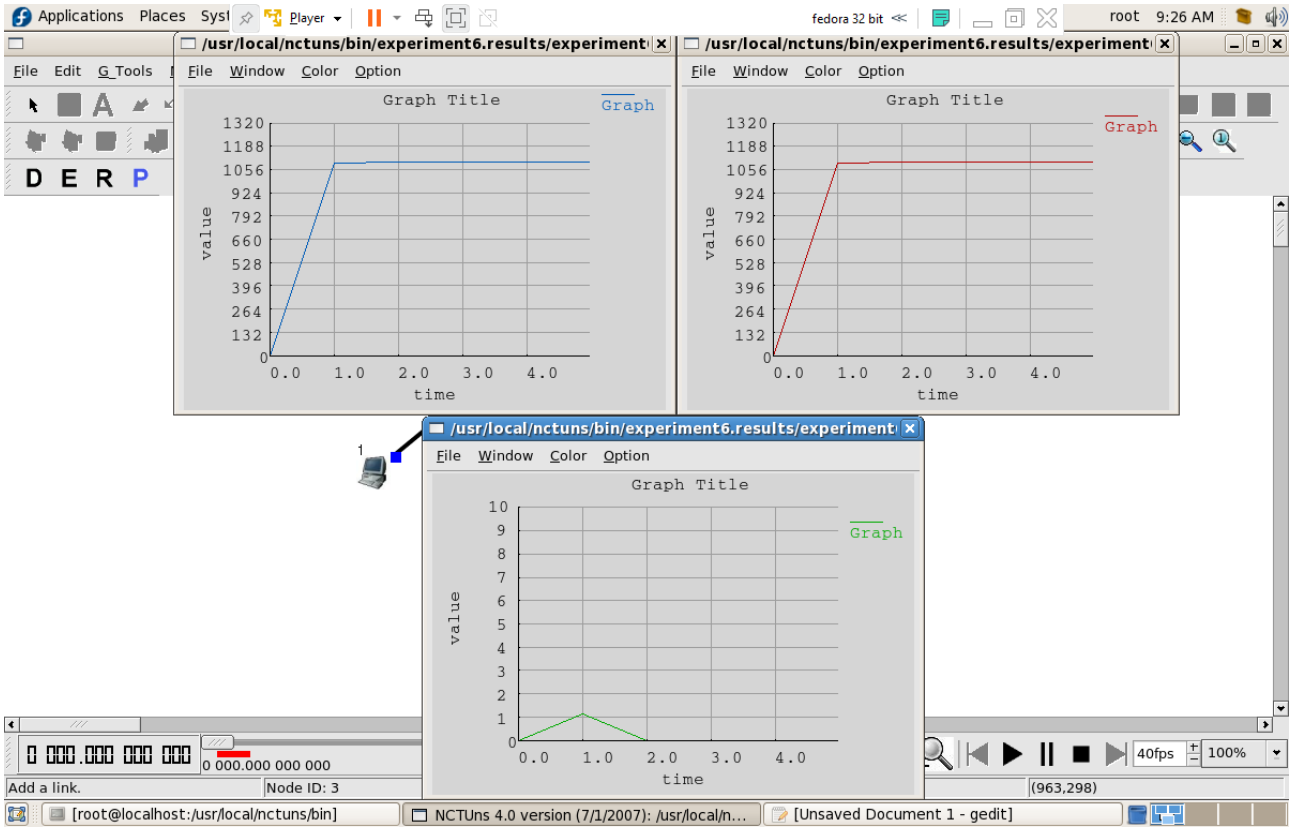
If we change hub with a switch the reduced B.W doesn't make any difference as switch is an active network device.

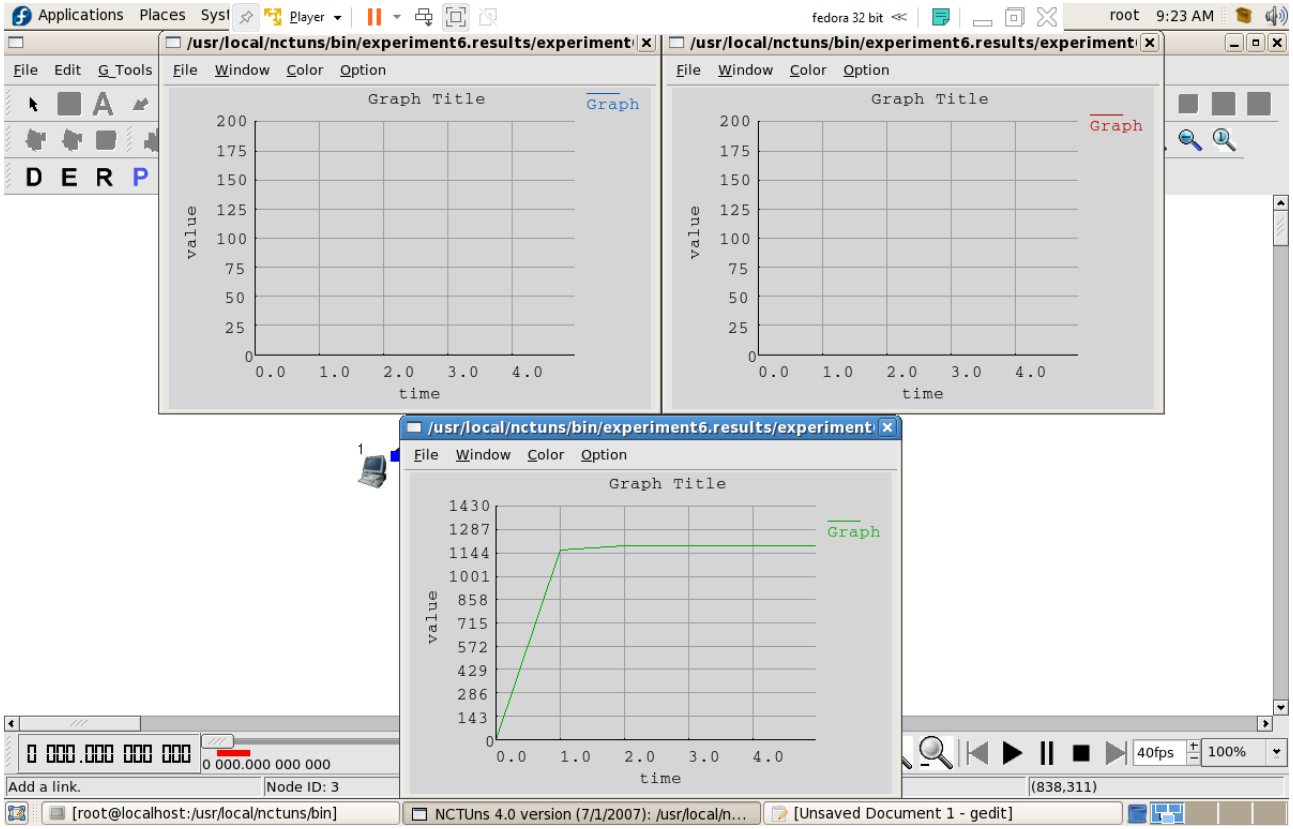












Kavita

Page No. :
Subject :
Date :

Experiment No.7: Four Node Point to Point Network

Aim: Simulate a 4 node point to point network and connect the link as follows. Apply a TCP agent bet n0 to n2 and apply a UDP agent bet n1 & n3. Apply relevant applications over TCP & UDP agents changing the parameters and determine no of packets sent by d agents.

Theory: A network protocol defines how to establish and maintain a network conversation through which applications can exchange data. TCP works wide the which defines how computers send packets of data to each other. It is connection oriented protocol that break application data to packets, sends packets, accepts them, controls data flow and provides error free transmission by collecting dropped packets. It is used by http, https, FTP, SMTP & Telnet.

This is Because there is no overhead for opening a connection or maintaining it, its used for broadcasting and multi-cast. It has basic error-check mechanisms, no sequencing of data used by DNS, DHCP & VoIP.

Kavita

Page No. :
Subject :
Date :

Configuration:

1) Double Click on 'HOST 1' Select 'Add' Button type in,

stcp -p 21-1 1024 1.0.1.3

2) Click "Node editor" on 'Host 1' select "MAC" tab & Select "Log Statistics" and check "Output thsoughp -ut".

3) Click on "Host 2" Select "Add" Button & type 1 in

stg -u 1024 100 1.0.1.3

4) Select "Log statistics" from 'MAC' tab and Chec -k Output thsoughput.

5) For the receiver end, click on 'Host 3' and "Add' Button type in,

stcp -p 21-1 1024.

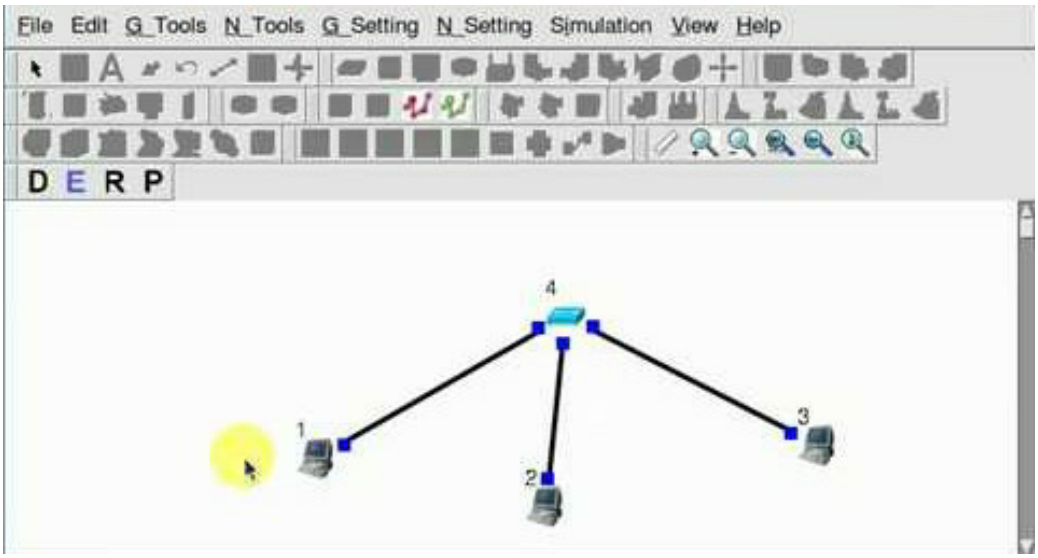
6) In "Host 4" click "Add" Button type in

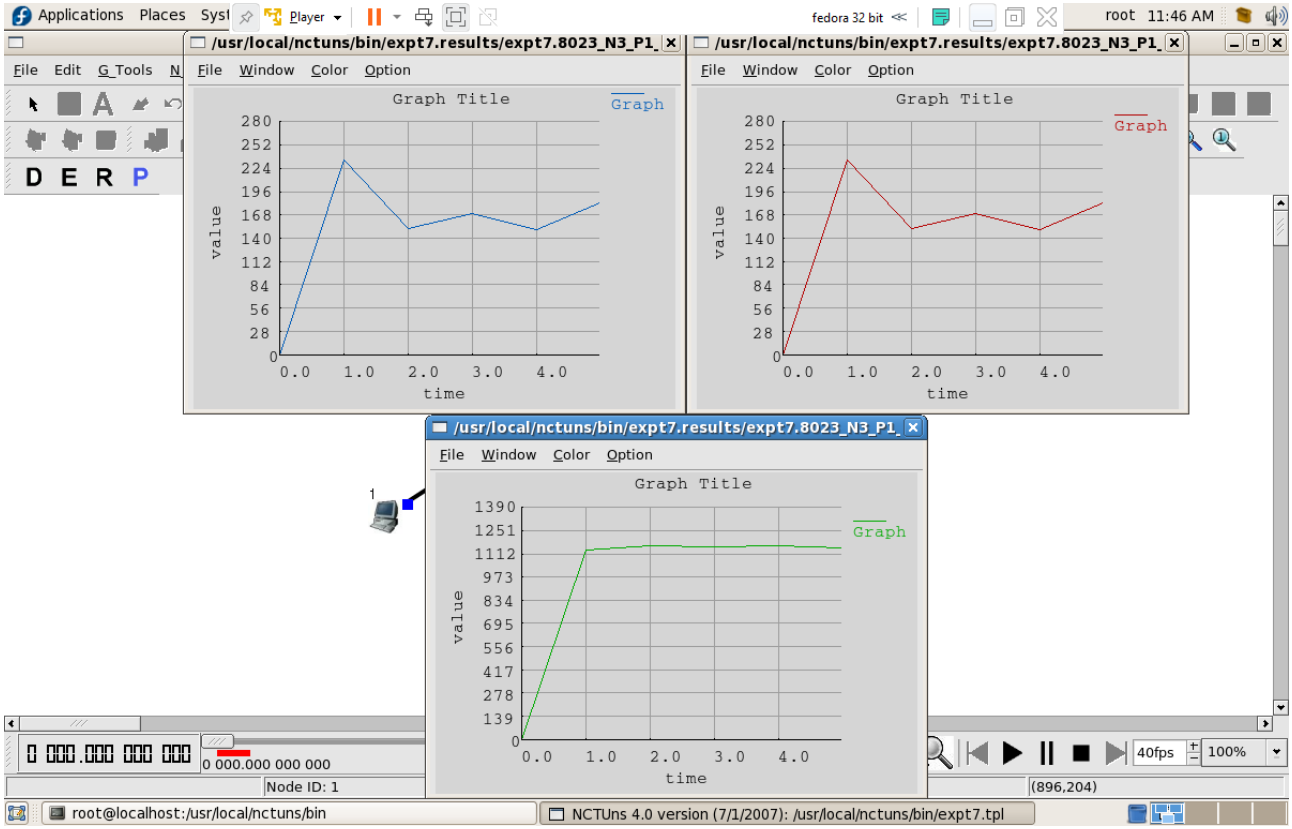
stg -u -w log1

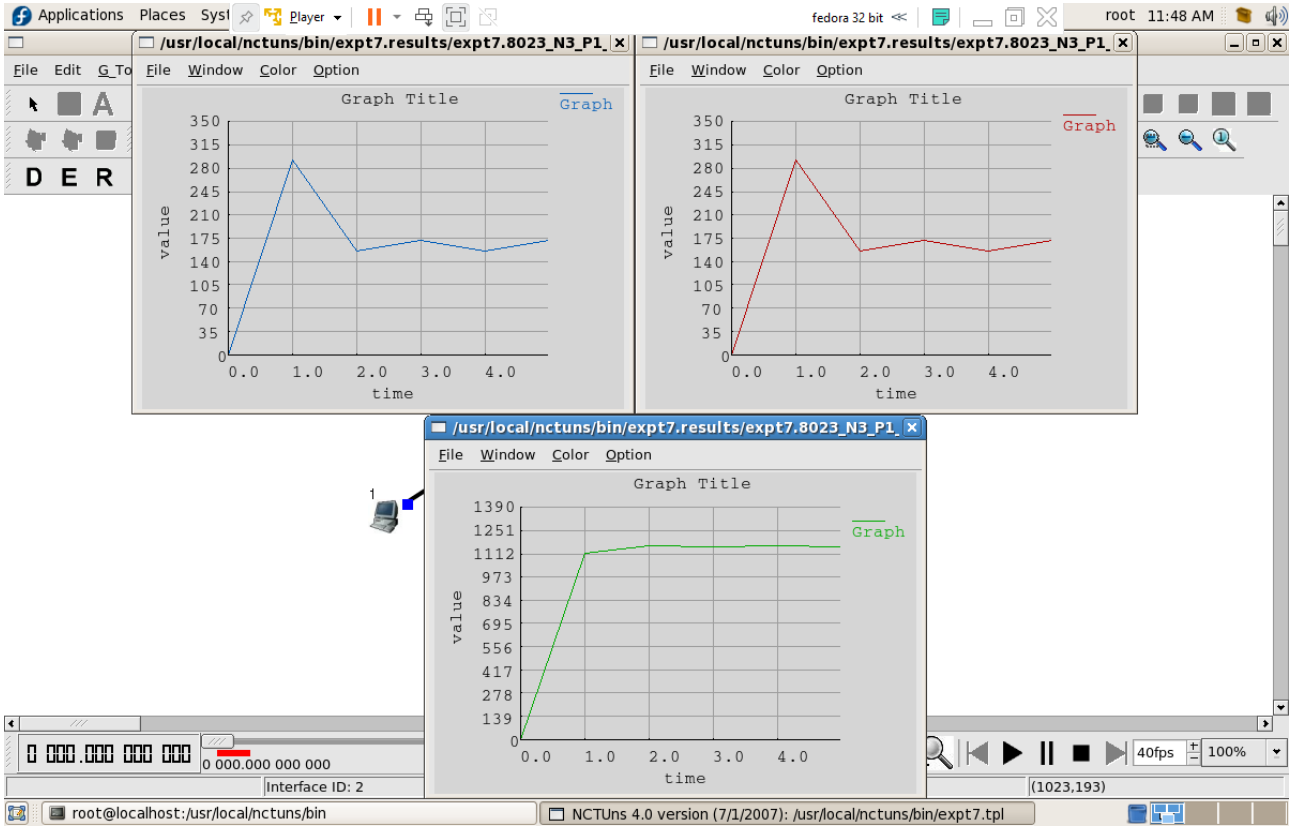
7) From "Log statistics" checkbox "input throughput" and "Output throughput".

Analysis:

For a queue size of 25, we set some collisions and drop with throughput greater than 1000. If we increase the queue size to 50 for all hosts then we see decrease in drop & collision with slight increase in throughput. If we reduce the B.W. throughput decreases significantly.







Kavita

Page No. :
Subject :
Date :

Experiment No. 8: Ethernet

Aim: Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare throughput.

Theory: A LAN is a group of computers and peripheral devices that share a common communication line or wireless link to server them a distinct geographical area. Ethernet and wi-fi are primary ways to enable LAN connection.

Ethernet is the traditional technology for connecting devices in wired LAN. It is fast, reliable, server backward compatible, noise resistance and has a relatively low cost. It is intended for small distances.

IEEE Specifies that Ethernet includes both physical and data link layer. For transmission it defines units i.e. packet and frame. The frame includes not just the payload of data being transmitted but also the MAC Address VLAN tagging and Quality of Service information. Error correction info to detect errors. Throughput is rate of successful message delivery over a communication channel.

Kavita

Page No. :
Subject :
Date :

Configurations

1) Double click on "Host 1" Select "Add" Button and type in

stcp -p 21 -l 1024 1.0.1.4

2) Repeat this for 'Host 2' & 'Host 3'.

Step -p 21 -l 1024 1.0.1.5 at Host-2

stcp -p 21 -l 1024 1.0.1.6 at Host-3

3) Double click on 'Host 4' Select 'Add' Button. and type in

stcp -p 21 -l 1024

4) Click on 'Node Editor' Select 'MAC' tab then Select 'Log Statistics' check on 'Output throughput'

5) Repeat step 3 & 4 for 'Host 5' and 'Host 6'.

stcp -p 21 -l 1024 at Host 5

stcp -p 21 -l 1024 at Host 6.

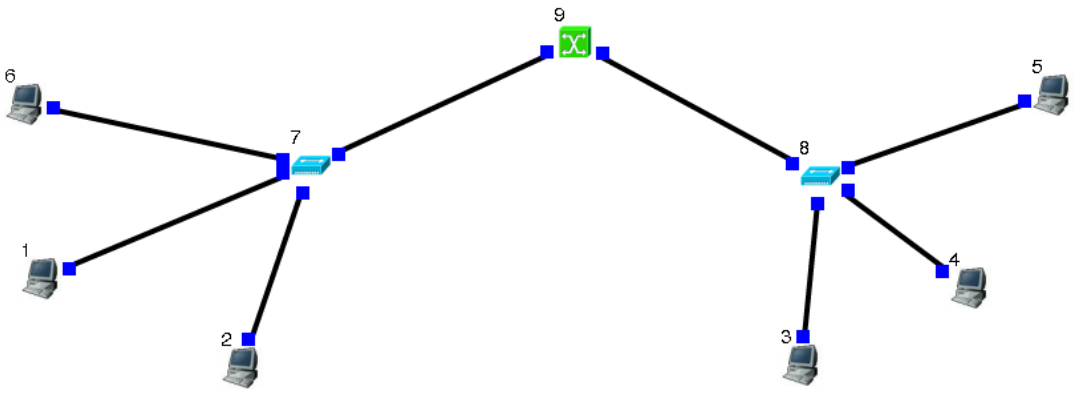
6) For 'Host 5' click on 'Node Editor' select 'Physical' tab change bit rate.

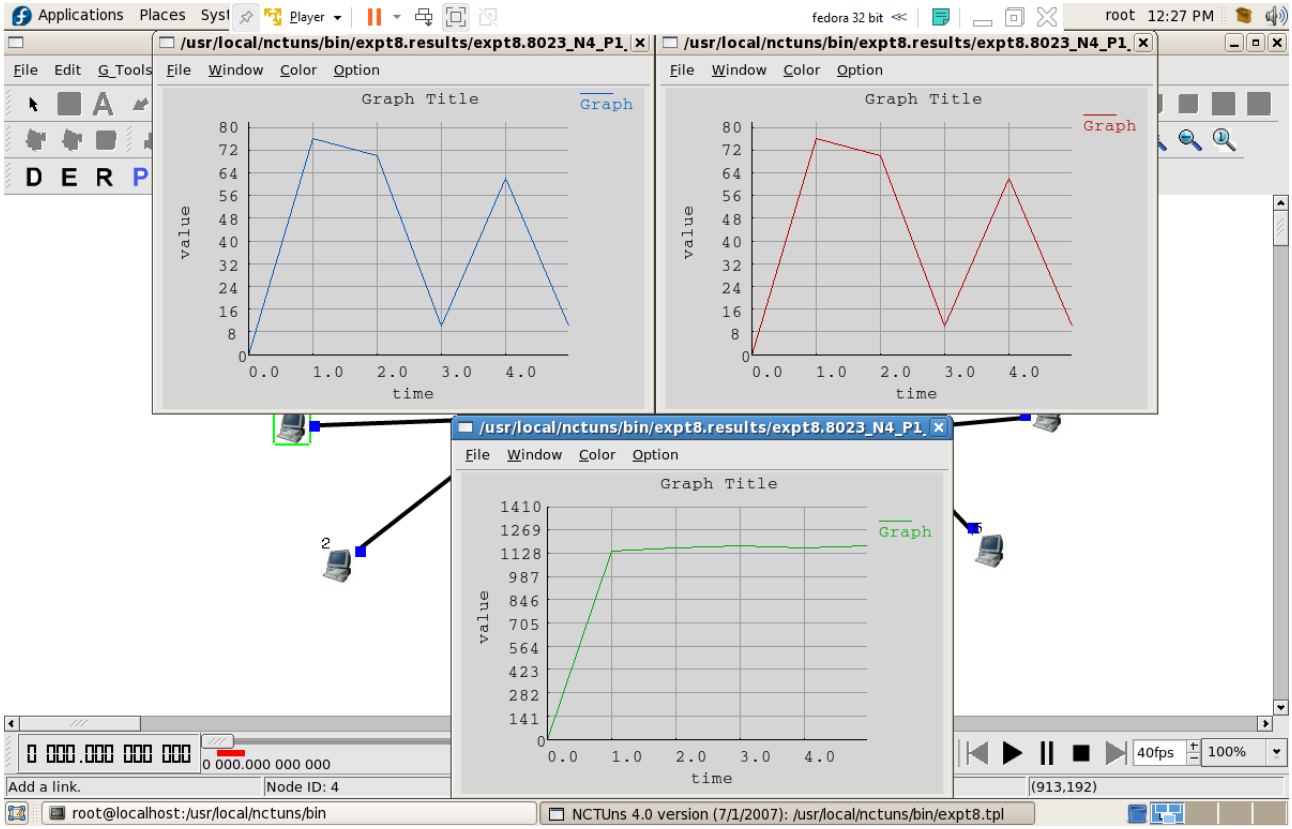
7) For 'Host 6' Click on 'Node Editor' Select 'Physical tab' and change B.W while doing previous change.

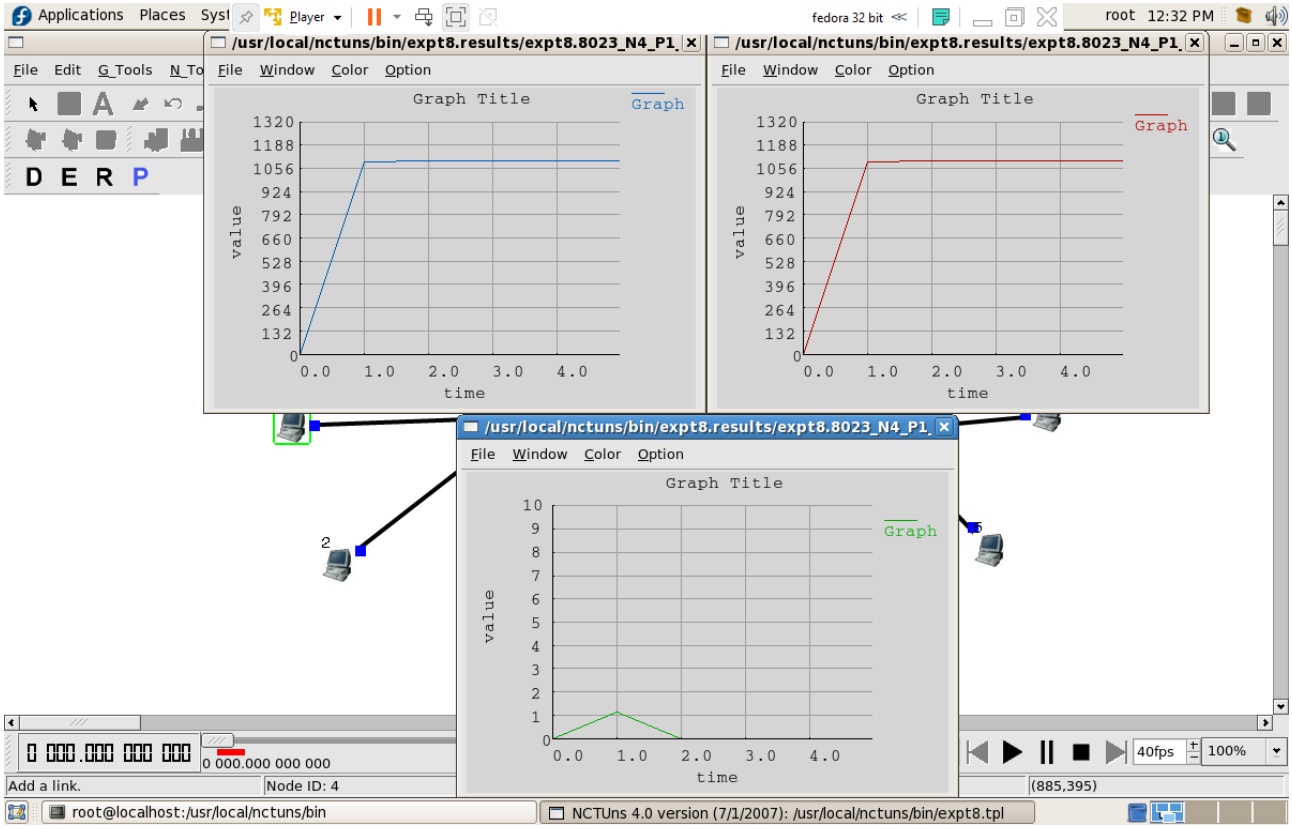
Analysis:

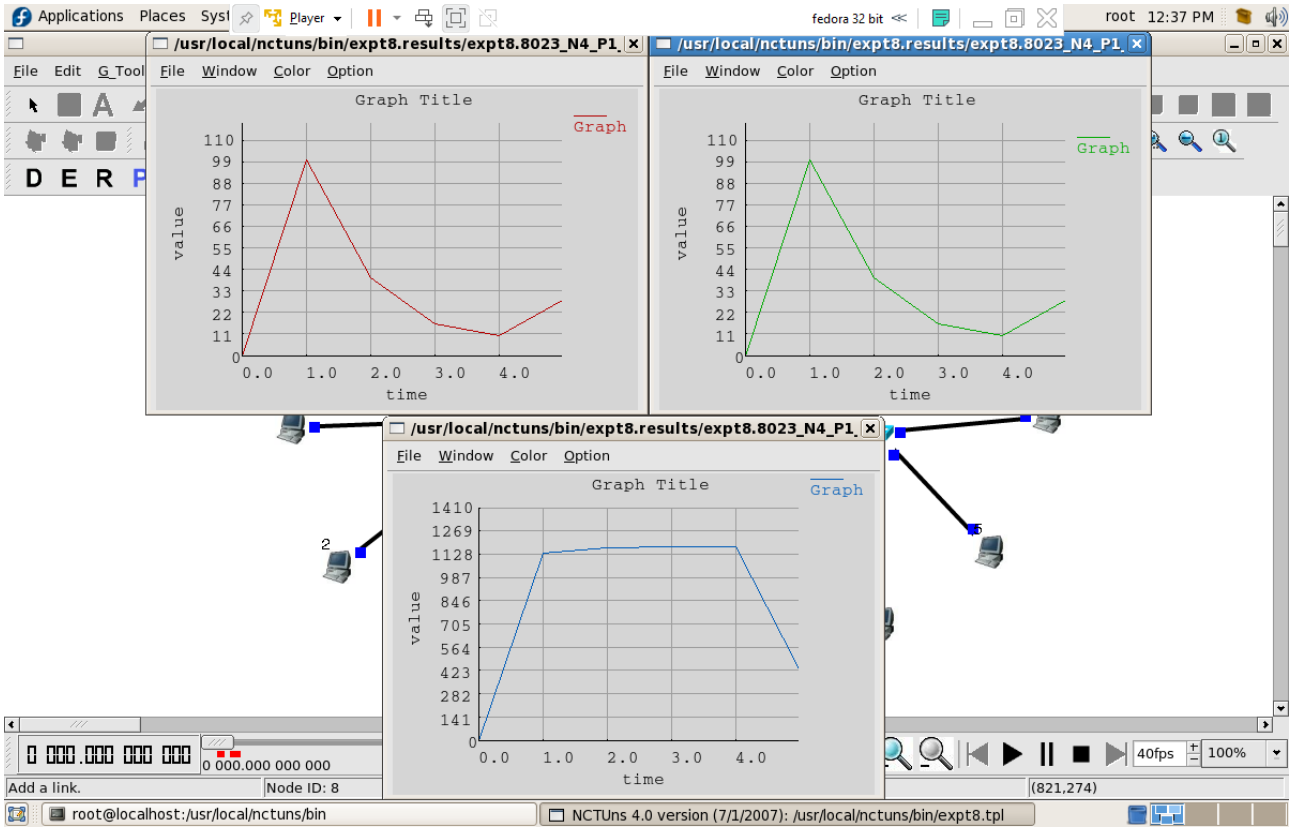
For this network we have set Host 1 as sender and Host 4 as receiver, we can set in the graph that very few packets are dropped due to collision. If BW is reduced from 10→5 there is a huge increase in drop and Collision & throughput decreases significantly.

Random Error occurs due to non-linear effect in channel. The factor B.E.R (Bit error rate) is used to simulate Random Errors.









Kavita

Page No
Subject
Date

Experiment 9: STAR Topology

Aim: Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the no. of packets dropped due to congestion, collision and PDR.

Theory: A ring network is a network topology in which each node connects to exactly two other nodes, forming a single continuous pathway for networks. A star topology is an implementation of a spoke-hub distribution paradigm in computer networks.

In a star network, every host is connected to a central hub. In its simplest form one central hub acts as a conduit to transmit messages. It is one of the most common topologies. Here, we use star topology to measure the send ping messages that measure round trip time for messages from originating host to destination devices to find packets dropped.

Kavita

Page No. :
Subject :
Date :

Configuration:

1) Double click on 'Host 1' click on 'Node editor' Select 'interface' tab and determine IP address of selected host. type command,

stg -u 1024 150 1.0.1.4
↓ ↓ ↓ ↓ ↓
sender UDP packet size No of packets IP ~~Destination~~.

2) Repeat this for all senders nodes (host)

3) Double click on 'Host 4' i.e receives node click on 'Add', type in

stg -u -s log1

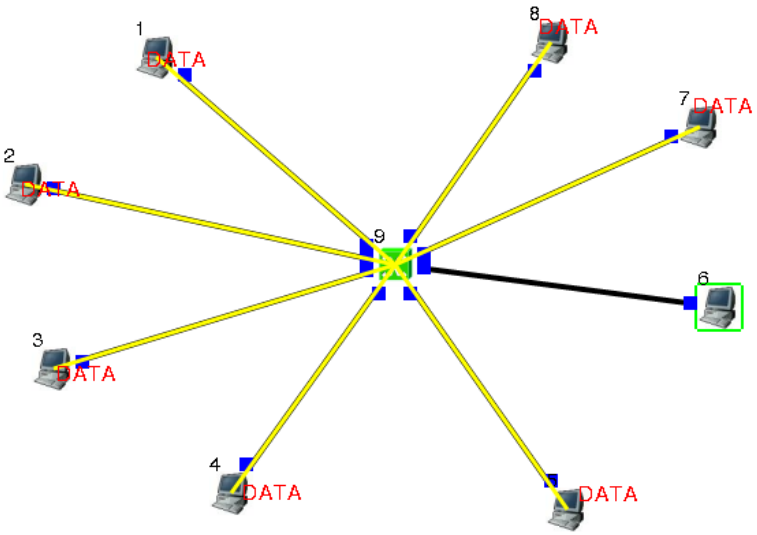
4) Set stop time as 20 in all experiments.

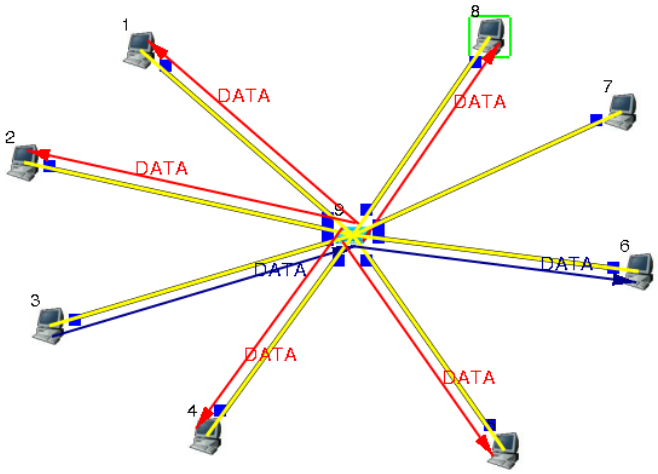
5) Click on 'Node editor' Select 'MAC tab', Select 'Log Statistics' check 'No of collisions', 'No of drop packets', 'Throughput'

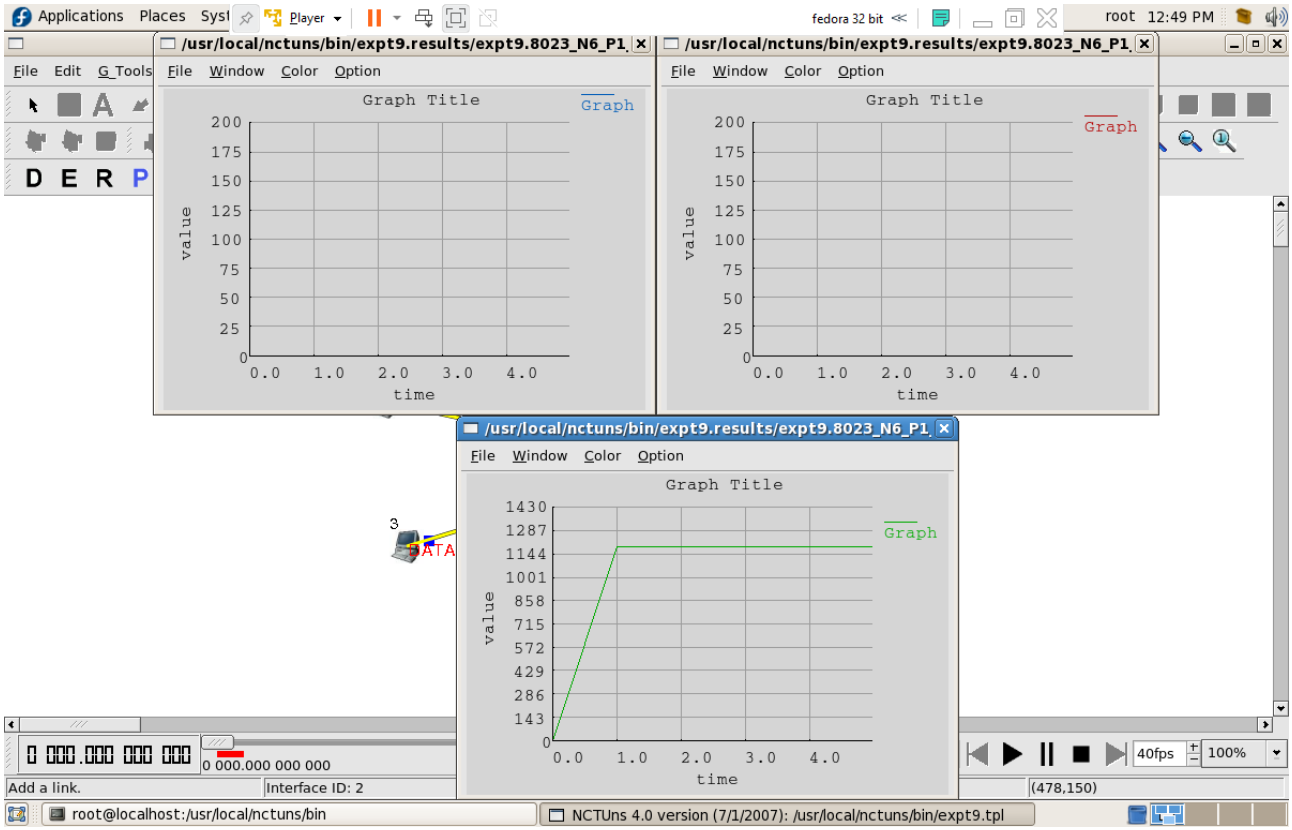
6) Repeat 1-5 using switch

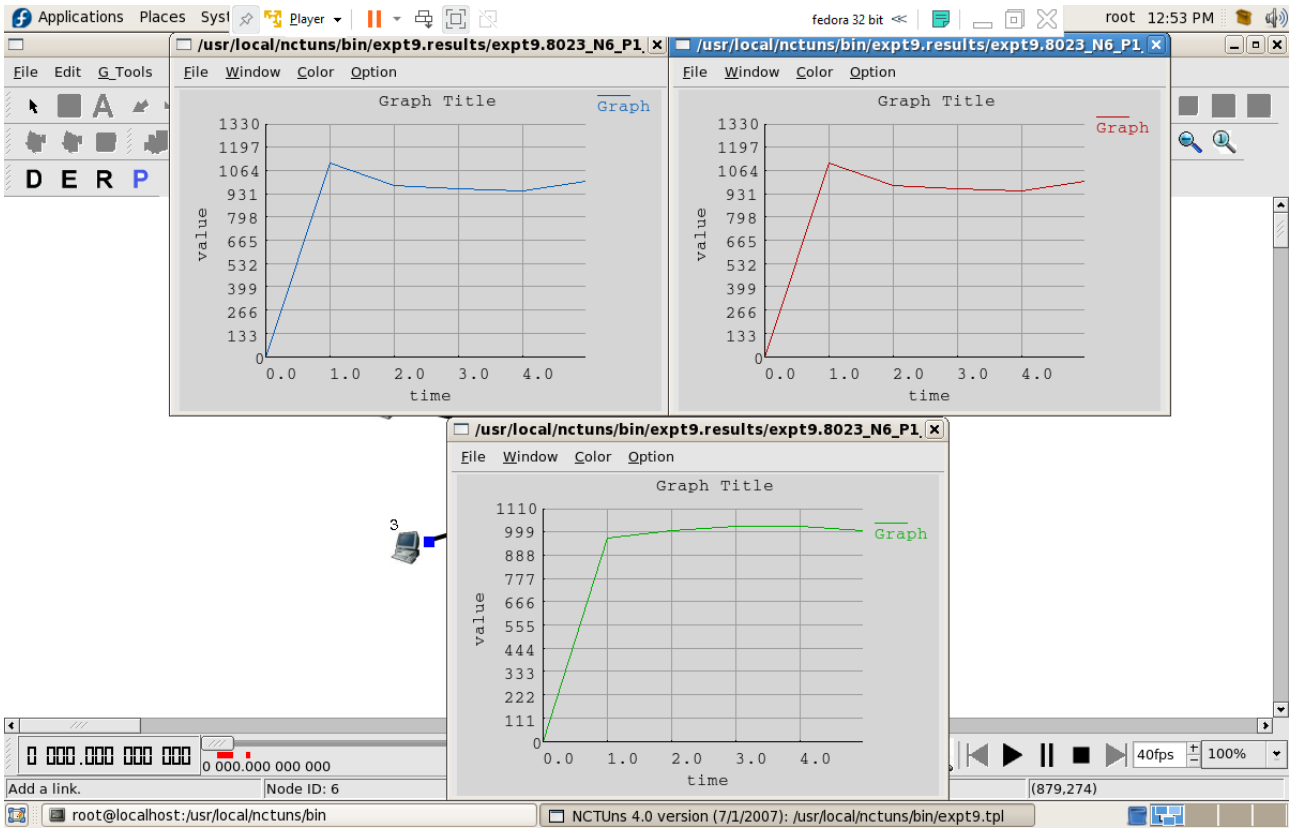
Analysis:

This simulation clearly shows us advantages of using a switch over a hub in high traffic networks. For a network with 6 end devices connected through a hub, the collision rate is high as hub is passive, non-intelligent device & works on half duplex, whereas switch provides no collision or drop as it is used full duplex and is intelligent.









Kavita

Page No.
Subject
Date

Experiment-10: Wireless Communication

Aim: Simulate simple BSS and with transmitting nodes in wireless LAN by simulating and determine the performance wrt transmission of packets.

Theory: 802.11 is an evolving family of specifications for wireless local area networks. All 802.11 specifications use ethernet protocols and carrier sense multiple access with collision avoidance for path sharing these are several standards like 802.11a, 802.11b, 802.11g, 802.11n etc they are used as ad-hoc networks as well. It provides 1-2mbps data rate in 2.4 GHz band and uses either FHSS or DSSS modulation. Some version use upto 5.9 GHz band. There are 2 MAC sub-layers:

1) Distributed Coordination function
2) Point coordination function.

In this exp. we set up a basic service set using WLAN workstation with preset mobility as it operates under LOS.

Kavita

Page No. :
Subject :
Date :

Configuration & Steps

* Drawing Topology

1) Select "Access point" from toolbar and place it on workspace. use 2 access points.

2) Select "Mobile node" & place 2 close to "Access pt 1" and 2 near "Access point 2".

3) Select 'Routes' and place it in workspace. connect "Access point 1" and "Access point 2" to 'Routes1'.

4) Select 'Host' connect to 'Routes1'.

5) Click on "Create a moving path" icon to draw a moving path across "Mobile Node 1" and 'Node 2' also for 3 & 4. Left Click and Drag to create paths. Right Click to terminate path.

6) To create Subnet select "wireless subnet' icon, select "Mobile Node1", '2' and "Access point1" and clicking right click will create a subnet.

7) Repeat this for "Mobile Node 3 & 4" and "Access point 2".

8) Save file.

8) Stop time :- 20 mobility :- 10m/s.

EDIT PROPERTY :

1) Select "Mobile Node 1" Click on "application", Select "Add" and type command

ttcp -t -u -s -p 8001 1.0.1.1

2) Select Show transmission range.

3) Click on "Node editor" Select "802.11e". Click on "Log Statistics" check "throughput".

4) Repeat Steps 2-3 for all mobile nodes.

ttcp -t -u -s -p 8002 1.0.1.1 at M.N.2

ttcp -t -u -s -p 8003 1.0.1.1 at M.N.3

ttcp -t -u -s -p 8004 1.0.1.1 at M.N.4

5) Select "Host 1" Click on 'Add' Button type in

ttcp -r -u -s p 8001

also add,

ttcp -r -u -s p 8002

ttcp -r -u -s p 8003

ttcp -r -u -s p 8004

6) Click on 'Node Editor' select 'MAC' Click on "Log Statistics' Check 'throughput"

7) Select 'Routes 1" Click on "Node editor". Select "MAC" of "Access point 1". Check on throughput (in). Repeat this for "A.P. 2", for "Host 1".

Select "MAC" Select "Log Statistics", Check throughput (out).

8) Stop time :- 20 mobility :- 10m/s.

Analysis:

For wireless communication we have used mo-bile nodes with a given mobility i.e 10m/s These mobile node connected to Access points which connect to routes. To reach the receiver these are 3 hops in total. The first set of outputs represent the outgoing throughput and second set of outputs represent the routes's incoming throughput.