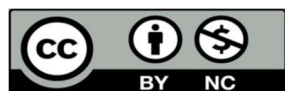


19 Dec 2019

Network Programming Lab Manual

SIT, Tumkur



Author:



AKSHAT AGARWAL
1SI16CS010

<https://github.com/git-akshat/NP-Lab>

NP-Lab

This repository contains programs implemented in Network Programming Lab in my 7th semester of SIT(VTU).

Part A

1. For the given network graph, write a program to implement Link state routing algorithm to build a routing table for the given node.
2. Write a program to divide the message into variable length frames and sort them and display the message at the receiving side.
3. Using TCP/IP sockets, write a client – server program, the client sends the file name and the server sends back the requested text file if present.
4. Using FIFOs as IPC channels, write a client – server program, the client sends the file name and the server sends back the requested text file if present.
5. Using UDP, write a client – server program, to exchange messages between client and the server.
6. Write a socket program to demonstrate IP multicasting which provides the capability for an application to send a single IP datagram that a group of hosts in a network can receive.
7. Write a program to implement sliding window protocol between two hosts.
8. Write a program for error detecting code using 16 bits CRC-CCITT (Consultative Committee for International Telephony and Telegraphy).

Part B : using NS2 simulator

1. Simulate a three nodes point – to – point network with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets dropped.
2. Simulate the different types of Internet traffic such as FTP and TELNET over a network and analyze the throughput.
3. Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare the throughput.
4. Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and determine the collision across different nodes.
5. Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
6. Simulate simple ESS with transmitting nodes in wire-less LAN and determine the performance with respect to transmission of packets.
7. Simulate simple ad-hoc network with transmitting nodes and determine the performance with respect to transmission of packets.

```

1  /* Author : Gangadhar, Akshat
2
3  1. For the given network graph, write a program to
4     * implement Link state routing algorithm
5     * build a routing table for the given node. */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <stdbool.h>
10
11 #define INFINITY 999
12 #define MAX 100
13
14 int cost[MAX][MAX]; // cost matrix
15 int distance[MAX]; // distance from source
16 int visited[MAX] = {0};
17 int parent[MAX];
18 int source;
19 int n; // number of nodes
20
21 void initialize()
22 {
23     int i;
24     visited[source] = 1;
25     parent[source] = source;
26
27     for(i=0; i<n; i++)
28     {
29         distance[i] = cost[source][i];
30         if( cost[source][i] != INFINITY )
31         {
32             parent[i] = source;
33         }
34     }
35 }
36
37 /* Get minimum distant node not already in network */
38 int GetMin()
39 {
40     int minIdx = -1;
41     int minDist = INFINITY;
42
43     int i;
44     for(i=0; i<n; i++)
45     {
46         if( !visited[i] && minDist >= distance[i] )
47         {
48             minIdx = i;
49             minDist = distance[i];
50         }
51     }
52     return minIdx;
53 }
54
55 /* update distance for adjacent nodes */
56 void updateTable(int node)
57 {
58     int i;
59     for(i=0; i<n; i++)
60     {

```

```

61     if( cost[node][i] != INFINITY && distance[i] > distance[node]+cost[node][i]
62 )
63     {
64         distance[i] = distance[node] + cost[node][i];
65         parent[i] = node;
66     }
67 }
68
69 void display()
70 {
71     int i;
72     int node;
73
74     printf("\nNode \t Distance from source \t Path \n");
75     for(i=0; i<n; i++)
76     {
77         printf("%d \t\t %d \t\t", i, distance[i]);
78
79         // node <- parent[node] <- parent[parent[node]] <- ... <- source
80         node = i;
81         printf("%d", node);
82         while( node != source)
83         {
84             printf(" <- %d", parent[node]);
85             node = parent[node];
86         }
87         printf("\n");
88     }
89 }
90
91 int main()
92 {
93     int i, j, node;
94
95     printf("Enter the number of nodes: ");
96     scanf("%d", &n);
97
98     printf("Enter the source node    : ");
99     scanf("%d", &source);
100
101     printf("\nEnter the cost matrix: \n");
102     for(i=0; i<n; i++)
103     {
104         for(j=0; j<n; j++)
105         {
106             scanf("%d", &cost[i][j]);
107         }
108     }
109
110     initialize();
111
112     for(i=0; i<n-1; i++) // for all remaining vertices(since source is already
visited)
113     {
114         node = GetMin();
115         visited[node] = 1;
116         updateTable(node);
117     }
118     display();
119     return 0;

```

```

120 }
121
122
123 /***** OUTPUT-1 *****/
124 Enter the number of nodes: 9
125 Enter the source node      : 3
126
127 Enter the cost matrix:
128 0   4   999  999  999  999  999  8   999
129 4   0   8   999  999  999  999 11  999
130 999 8   0   7   999  4   999 999  2
131 999 999 7   0   9   14  999 999  999
132 999 999 999 9   0  10  999 999  999
133 999 999 4   14  10  0   2   999 999
134 999 999 999 999 999 2   0   1   6
135 8   11  999  999  999 999 1   0   7
136 999 999 2   999  999 999 6   7   0
137
138 Node      Distance from source      Path
139 0          19                      0 <- 1 <- 2 <- 3
140 1          15                      1 <- 2 <- 3
141 2          7                       2 <- 3
142 3          0                       3
143 4          9                       4 <- 3
144 5          11                      5 <- 2 <- 3
145 6          13                      6 <- 5 <- 2 <- 3
146 7          14                      7 <- 6 <- 5 <- 2 <- 3
147 8          9                       8 <- 2 <- 3
148 *****/

```

```

1  /* Author : Gangadhar, Akshat
2
3  2. Write a program to divide the message into variable length frames and sort them
   and display the message at the receiving side. */
4
5  #include<stdio.h>
6  #include<stdlib.h>
7  #include<string.h>
8  #include<time.h>
9
10 #define MAX 100
11
12 typedef struct{
13     int id;
14     char data[MAX];
15 }frame;
16
17 // Fisher yates algorithm to shuffle the frame
18 void shuffleFrame(frame f[MAX], int n)
19 {
20     srand(time(NULL));
21
22     int i;
23     for(i=n; i>=0; i--)
24     {
25         int j = rand()%(i+1);
26
27         frame temp = f[j];
28         f[j] = f[i];
29         f[i] = temp;
30     }
31 }
32
33 // Insertion sort algorithm to sort frames based on id
34 void sortFrames(frame f[MAX], int n)
35 {
36     int i, j;
37
38     for(i=1; i<=n; i++)
39     {
40         frame t = f[i];
41         j = i-1;
42         while(j>=0 && f[j].id > t.id)
43         {
44             f[j+1] = f[j];
45             j=j-1;
46         }
47         f[j+1] = t;
48     }
49 }
50
51 int main()
52 {
53     frame f[MAX];
54     int n = -1;    // no of frames
55     int fsize;    // size of frame
56
57     char msg[MAX];
58     int m = 0; // message iterator
59     int i, j;

```

```

60
61 printf("Enter a message : ");
62 fgets(msg , MAX, stdin);
63 msg[strlen(msg)-1] = '\0'; // to remove '\n' from string
64
65 srand(time(NULL));
66 // Divide the message into frames
67 for(i=0 ; m < strlen(msg) ; i++)
68 {
69     f[i].id = i;
70
71     n++; // count number of frames
72     fsize = rand()%5+1; // variable Frame size in range [1,5]
73
74     for(j=0 ; j<fsize && m < strlen(msg); j++)
75     {
76         f[i].data[j] = msg[m++];
77     }
78 }
79
80 shuffleFrame(f, n);
81
82 printf("\nShuffled frames:");
83 printf("\nframe_id \t frame_data \n");
84 printf("-----\n");
85 for(i=0 ; i <= n; i++)
86 {
87     printf("%d \t\t %s \n", f[i].id, f[i].data);
88 }
89
90 sortFrames(f, n);
91
92 printf("\nSorted frames:");
93 printf("\nframe_id \t frame_data \n");
94 printf("-----\n");
95 for(i=0 ; i <= n; i++)
96 {
97     printf("%d \t\t %s \n", f[i].id, f[i].data);
98 }
99
100 printf("\nfinal message : ");
101 for(i=0; i<= n; i++)
102 {
103     printf("%s", f[i].data);
104 }
105
106 printf("\n");
107 }
108
109

```

```
110 /***** OUTPUT-1 *****/
111 Enter a message : hello beautiful world
112
113 Shuffled frames:
114 frame_id      frame_data
115 -----
116 6             ld
117 1             lo be
118 0             hel
119 2             auti
120 5             wor
121 3             ful
122 4
123
124 Sorted frames:
125 frame_id      frame_data
126 -----
127 0             hel
128 1             lo be
129 2             auti
130 3             ful
131 4
132 5             wor
133 6             ld
134
135 final message : hello beautiful world
136 *****/
```



```

1  /* Author : Akshat Agarwal
2
3  3. Using TCP/IP sockets, write a client - server program,
4     - the client sends the file name and
5     - the server sends back the requested text file if present. */
6
7  /* Server Program */
8
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <fcntl.h>
12 #include <arpa/inet.h>
13 #include <unistd.h>
14
15 int main()
16 {
17     int sersock, sock, fd, n, reuse = 1;
18     char buffer[1024], fname[50];
19
20     /* sockfd = socket(domain, type, protocol) */
21     sersock = socket(AF_INET, SOCK_STREAM, 0);
22
23     struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };
24
25     // Forcefully connecting to same port everytime
26     setsockopt(sersock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));
27
28     /* attaching socket to port */
29     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
30     printf("\nServer is Online");
31
32     listen(sersock, 5); // listen(int sockfd, int backlog)
33     sock = accept(sersock, NULL, NULL);
34
35     /* receive the filename */
36     recv(sock, fname, 50, 0);
37     printf("\nRequesting for file: %s\n", fname);
38
39     /* open the file in read-only mode */
40     fd = open(fname, O_RDONLY);
41     if (fd < 0)
42     {
43         send(sock, "\nFile not found\n", 15, 0); // strlen(\nFile not found)=15
44     }
45     else
46     {
47         while ((n = read(fd, buffer, sizeof(buffer))) > 0)
48         {
49             send(sock, buffer, n, 0);
50         }
51     }
52     printf("\nFile content sent\n");
53
54     close(fd);
55     return 0;
56 }

```

```

1  /* Author : Akshat Agarwal
2
3  3. Using TCP/IP sockets, write a client - server program,
4     - the client sends the file name and
5     - the server sends back the requested text file if present. */
6
7  /* Client Program */
8
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <fcntl.h>
12 #include <arpa/inet.h>
13 #include <unistd.h>
14
15 int main()
16 {
17     int sock, n;
18     char buffer[1024], fname[50];
19
20     sock = socket(AF_INET, SOCK_STREAM, 0);
21     struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };
22
23     /* keep trying to establish connection with server */
24     while(connect(sock, (struct sockaddr *) &addr, sizeof(addr))) ;
25     printf("\nClient is connected to Server");
26
27     /* send the filename to the server */
28     printf("\nEnter file name: ");
29     scanf("%s", fname);
30     send(sock, fname, sizeof(fname), 0);
31
32     printf("\nReceived file data\n");
33     printf("-----\n");
34
35     /* keep printing any data received from the server */
36     while ((n = recv(sock, buffer, sizeof(buffer), 0)) > 0)
37     {
38         buffer[n] = '\0' ;
39         printf("%s", buffer);
40     }
41
42     printf("-----\n");
43     return 0;
44 }

```

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ gcc ClientTCP.c -o client.out
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ ./client.out
```

Client is connected to Server
Enter file name: file.txt

Recieved file data

-
1. Go 5km North
 2. Turn to left and continue 3 km.
 3. turn to left and walk straight 2km.
 4. which direction you are pointing to?
-

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ █
```

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ gcc ServerTCP.c -o server.out
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ ./server.out
```

Server is Online
Requesting for file: file.txt

File content sent

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ █
```

```

1  /* Author : Akshat Agarwal
2
3  4. Using FIFOs as IPC, write a client - server program,
4     - the client sends the file name and
5     - the server sends back the requested text file if present. */
6
7  /* Server Program */
8
9  #include <stdio.h>
10 #include <unistd.h>
11 #include <fcntl.h> // used for file handling
12 #include <sys/stat.h> // used for mkfifo function
13 #include <sys/types.h> // mkfifo() has dependency on both types.h and stat.h
14
15 int main()
16 {
17     char fname[50], buffer[1025];
18     int req, res, n, file;
19
20     mkfifo("req.fifo", 0777);
21     mkfifo("res.fifo", 0777);
22
23     printf("Waiting for request...\n");
24     req = open("req.fifo", O_RDONLY);
25     res = open("res.fifo", O_WRONLY);
26
27     read(req, fname, sizeof(fname));
28     printf("Received request for %s\n", fname);
29
30     file = open(fname, O_RDONLY);
31     if (file < 0)
32     {
33         write(res, "File not found\n", 15);
34     }
35     else
36     {
37         while((n = read(file, buffer, sizeof(buffer))) > 0)
38         {
39             write(res, buffer, n);
40         }
41     }
42
43     close(req);
44     close(res);
45
46     unlink("req.fifo");
47     unlink("res.fifo");
48
49     return 0;
50 }

```

```

1  /* Author : Akshat Agarwal
2
3  4. Using FIFOs as IPC, write a client - server program,
4     - the client sends the file name and
5     - the server sends back the requested text file if present. */
6
7  /* Client Program */
8
9  #include <stdio.h>
10 #include <unistd.h>
11 #include <stdlib.h>
12 #include <fcntl.h>
13 #include <sys/stat.h>
14 #include <sys/types.h>
15
16 int main()
17 {
18     char fname[50], buffer[1025];
19     int req, res, n;
20
21     req = open("req.fifo", O_WRONLY);
22     res = open("res.fifo", O_RDONLY);
23
24     if(req < 0 || res < 0)
25     {
26         printf("Please Start the server first\n");
27         exit(-1);
28     }
29
30     printf("Enter filename to request : ");
31     scanf("%s", fname);
32
33     // write file name to request file
34     write(req, fname, sizeof(fname));
35
36     printf("Received response\n");
37     printf("-----\n");
38     while((n = read(res, buffer, sizeof(buffer)))>0)
39     {
40         printf("%s", buffer);
41     }
42     printf("-----\n");
43
44     close(req);
45     close(res);
46     return 0;
47 }
48

```

storm-breaker@stormbreaker: ~/Documents/NP Lab/A4 (FIFO)

File Edit View Search Terminal Help

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$ gcc ServerFIFO.c -o server.out
```

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$ ./server.out
```

Waiting for request...

Received request for file.txt

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$
```

storm-breaker@stormbreaker: ~/Documents/NP Lab/A4 (FIFO)

File Edit View Search Terminal Help

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$ gcc ClientFIFO.c -o client.out
```

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$ ./client.out
```

Enter filename to request : file.txt

Received response

-
1. Go 5km North
 2. Turn to left and continue 3 km.
 3. turn to left and walk straight 2km.
 4. which direction you are pointing to?
-

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$
```

```

1  /* Author : Akshat Agarwal
2
3  5. Using UDP, write a client-server program, to exchange messages between client and
   the server. */
4
5  /* Server Program */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <unistd.h>
10 #include <string.h>
11 #include <sys/types.h>
12 #include <sys/socket.h>
13 #include <arpa/inet.h>
14 #include <netinet/in.h>
15
16 #define MAX 1024
17
18 int main()
19 {
20     int sock;
21     int len, n, reuse = 1;
22     char buffer[MAX], msg[MAX];
23     struct sockaddr_in servaddr, cliaddr;
24
25     sock = socket(AF_INET, SOCK_DGRAM, 0);
26     // Forcefully connecting to same port everytime
27     setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));
28
29     // initialize servaddr and cliaddr to 0
30     memset(&servaddr, 0, sizeof(servaddr));
31     memset(&cliaddr, 0, sizeof(cliaddr));
32     len = sizeof(cliaddr);
33
34     // Filling server information
35     servaddr.sin_family = AF_INET; // IPv4
36     servaddr.sin_addr.s_addr = INADDR_ANY;
37     servaddr.sin_port = htons(1234);
38
39     // Bind the socket with the server address
40     if (bind(sock, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0 )
41     {
42         printf("Binding error\n"); exit(0);
43     }
44     printf("Waiting for message from client...\n");
45
46     while(1)
47     {
48         n = recvfrom(sock, (char *)buffer, sizeof(buffer), 0, ( struct sockaddr *)
&cliaddr, &len);
49         buffer[n] = '\0';
50         printf("Client : %s", buffer);
51
52         printf("Server : ");
53         fgets(msg, MAX, stdin);
54         sendto(sock, (const char *)msg, strlen(msg), 0, (const struct sockaddr *)
&cliaddr, len);
55     }
56     return 0;
57 }

```

```

1  /* Author : Akshat Agarwal
2
3  5. Using UDP, write a client - server program, to exchange messages between client
   and the server. */
4
5  /* Client Program */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <unistd.h>
10 #include <string.h>
11 #include <sys/types.h>
12 #include <sys/socket.h>
13 #include <arpa/inet.h>
14 #include <netinet/in.h>
15
16 #define MAX 1024
17
18 int main()
19 {
20     int n, len, sock;
21     char buffer[MAX], msg[MAX];
22     struct sockaddr_in servaddr;
23
24     // Creating socket file descriptor
25     sock = socket(AF_INET, SOCK_DGRAM, 0);
26
27     memset(&servaddr, 0, sizeof(servaddr));
28     // Filling server information
29     servaddr.sin_family = AF_INET;
30     servaddr.sin_port = htons(1234); // htons(port)
31     servaddr.sin_addr.s_addr = INADDR_ANY;
32
33     while( connect(sock, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0);
34     printf("\nConnection Established" ) ;
35
36     while(1)
37     {
38         printf("Client : ");
39         fgets(msg, MAX, stdin);
40         sendto(sock, (const char *)msg, strlen(msg), 0, (const struct sockaddr *)
&servaddr, sizeof(servaddr));
41
42         n = recvfrom(sock, (char *)buffer, sizeof(buffer), 0, NULL, NULL);
43         buffer[n] = '\0';
44         printf("Server : %s", buffer);
45     }
46     return 0;
47 }

```


storm-breaker@stormbreaker: ~/Documents/NP Lab/A5 (UDP)

File Edit View Search Terminal Help

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$ gcc ServerUDP.c -o server.out
```

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$ ./server.out
```

Waiting for message from client...

Client : Hi

Server : Hello

Client : Bye.

Server : ^C

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$
```

storm-breaker@stormbreaker: ~/Documents/NP Lab/A5 (UDP)

File Edit View Search Terminal Help

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$ gcc ClientUDP.c -o client.out
```

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$ ./client.out
```

Client : Hi

Server : Hello

Client : Bye.

^C

```
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$
```

```

1  /* Author : Akshat Agarwal
2
3  6. Write a socket program to demonstrate ip multicasting which provides the
   capability for an application to send IP datagram that a group of hosts in a network
   can receive. */
4
5  /**** server sends the message ****/
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <unistd.h>
10 #include <string.h>
11 #include <sys/types.h>
12 #include <sys/socket.h>
13 #include <arpa/inet.h>
14 #include <netinet/in.h>
15
16 int main ()
17 {
18     int sock;
19     char msg[1024];
20     struct sockaddr_in groupaddr;
21     struct in_addr localInterface;
22
23     sock = socket(AF_INET, SOCK_DGRAM, 0);
24
25     memset(&groupaddr, 0, sizeof(groupaddr));
26     groupaddr.sin_family = AF_INET;
27     groupaddr.sin_addr.s_addr = inet_addr("226.1.1.1");
28     groupaddr.sin_port = htons(1234);
29
30     localInterface.s_addr = inet_addr("127.0.0.1"); // or system ip address
31     setsockopt(sock, IPPROTO_IP, IP_MULTICAST_IF, (char *)&localInterface,
   sizeof(localInterface));
32
33     printf("Enter message : ");
34     fgets(msg, 1024, stdin);
35     msg[strlen(msg)-1] = '\0'; // to remove '\n' from string
36     sendto(sock, msg, sizeof(msg), 0, (struct sockaddr*)&groupaddr,
   sizeof(groupaddr));
37     printf("Message Sent.\n");
38
39     return 0;
40 }

```

```

1  /* Author : Akshat Agarwal
2
3  6. Write a socket program to demonstrate ip multicasting which provides the
   capability for an application to send IP datagram that a group of hosts in a network
   can receive. */
4
5  /**** clients recieves the message ****/
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <unistd.h>
10 #include <string.h>
11 #include <sys/types.h>
12 #include <sys/socket.h>
13 #include <arpa/inet.h>
14 #include <netinet/in.h>
15
16 int main()
17 {
18     int sock, reuse = 1;
19     char msg[1024];
20     struct sockaddr_in addr;
21     struct ip_mreq group;
22
23     sock = socket(AF_INET, SOCK_DGRAM, 0);
24
25     setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));
26
27     memset(&addr, 0, sizeof(addr));
28     addr.sin_family = AF_INET;
29     addr.sin_port = htons(1234);
30     addr.sin_addr.s_addr = INADDR_ANY; // to listen on all available interfaces.
31
32     if(bind(sock, (struct sockaddr*)&addr, sizeof(addr)) < 0)
33     {
34         printf("Binding failed");
35         close(sock);
36         exit(1);
37     }
38
39     group.imr_multiaddr.s_addr = inet_addr("226.1.1.1");
40     group.imr_interface.s_addr = inet_addr("127.0.0.1");
41     setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, (char *)&group, sizeof(group));
42     printf("Waiting for message from server.....");
43
44     read(sock, msg, sizeof(msg));
45     printf("\nThe message from multicast server is : %s \n", msg);
46
47     close(sock);
48     return 0;
49 }

```

```
akshat@pop-os: ~/Desktop/NP Lab/A6 (IP Multicast)
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ gcc ServerMC.c -o server.out
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ ./server.out
Enter message : Everyone meet me in 10 minutes.
Message Sent.
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$
```

```
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ gcc ClientMC.c -o client1.out
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ ./client1.out
Waiting for message from server...
The message from multicast server is : Everyone meet me in 10 minutes.
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$
```

```
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ gcc ClientMC.c -o client2.out
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ ./client2.out
Waiting for message from server...
The message from multicast server is : Everyone meet me in 10 minutes.
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$
```

```
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ gcc ClientMC.c -o client3.out
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ ./client3.out
Waiting for message from server...
The message from multicast server is : Everyone meet me in 10 minutes.
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$
```

```

1  /* Author : Akshat Agarwal
2
3  7.  to implement sliding window protocol, between two hosts(TCP Flow Control)
4      - Client sends the frame
5      - Server recieves the frame */
6
7  /* Server Program */
8
9  #include<stdio.h>
10 #include<stdlib.h>
11 #include<string.h>
12 #include<time.h>
13 #include<unistd.h>
14 #include<sys/socket.h>
15 #include<sys/types.h>
16 #include<arpa/inet.h>
17
18 #define MAX 20
19
20 int main()
21 {
22     int sersock, sock, reuse=1;
23     char frame[MAX];
24     char res[MAX]; // to store all bytes that are recieved successfully
25     int ack;
26
27     sersock = socket(AF_INET, SOCK_STREAM, 0);
28
29     struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };
30
31     // Forcefully connecting to same port everytime
32     setsockopt(sersock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));
33
34     bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
35     printf("\nServer is Online");
36
37     listen(sersock, 5);
38     sock = accept(sersock, NULL, NULL);
39
40     int k=0; // iterator for res[MAX]
41     srand(time(NULL));
42
43     while(1)
44     {
45         int recvsize = 5;
46         memset(frame, 0, MAX); // re-initialise frame buffer with 0
47
48         recv(sock, frame, recvsize, 0); // recv(socket, buffer, length, flag)
49
50         if(strlen(frame) < recvsize)
51         {
52             recvsize = strlen(frame);
53         }
54         // at end exit frame is recieved from client
55         if(strcmp(frame, "Exit") == 0) break;
56
57         int err_idx = rand()%8; // probability of byte to get corrupted = 50%
58         int j;
59
60         if(err_idx < recvsize)

```

```

61 {
62     for(j=0; j<err_idx ; j++)
63     {
64         res[k++] = frame[j];
65     }
66     frame[err_idx]='x';
67     printf("\n\nPacket received = %s", frame);
68     printf("\nError at byte = %d", err_idx+1);
69     printf("\nReceiving window: ");
70     printf("\n start seqno = %d", k-err_idx);
71 }
72 else
73 {
74     for(j=0; j<recvsize ; j++)
75     {
76         res[k++] = frame[j];
77     }
78     printf("\n\nPacket received = %s", frame);
79     printf("\nReceiving window: ");
80     printf("\n start seqno = %d", k-recvsize);
81 }
82 printf("\n end seqno = %d", k-1);
83 ack = k ;
84 printf("\nSending ack = %d", ack);
85 send(sock, &ack, sizeof(ack), 0) ;
86 }
87
88 res[k] = '\0';
89 printf("\n\nFinal string recieved at Destination = ");
90 fputs(res, stdout);
91
92 printf("\n\n");
93 close(sock); close(sersock);
94 }

```

```

1  /* Author : Akshat Agarwal
2
3  7. to implement sliding window protocol, between two hosts(TCP Flow Control)
4  - Client sends the frame
5  - Server recieves the frame */
6
7  /* Client Program */
8
9  #include<stdio.h>
10 #include<stdlib.h>
11 #include<string.h>
12 #include<unistd.h>
13 #include<sys/socket.h>
14 #include<sys/types.h>
15 #include<arpa/inet.h>
16
17 #define MAX 20
18
19 int main()
20 {
21     int sock, ack;
22     char msg[MAX], frame[MAX];
23
24     sock = socket(AF_INET, SOCK_STREAM, 0);
25     struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };
26
27     /* keep trying to establish connection with server */
28     while(connect(sock, (struct sockaddr *) &addr, sizeof(addr))) ;
29     printf("\nClient is connected to Server\n");
30
31     printf("\nEnter message to send : "); scanf("%s", msg);
32
33     int i = 0;
34     while(i<strlen(msg))
35     {
36         int sendsize = 5;
37         memset(frame, 0, MAX); // re-initialise frame buffer with 0
38
39         // strncpy(destination , source , length)
40         strncpy(frame, msg+i, sendsize); //copy msg to frame
41         if( sendsize > strlen(frame) )
42         {
43             sendsize = strlen(frame);
44         }
45         printf("\n\nSending packet = %s", frame);
46         printf("\nSending window: ");
47         printf("\n start seqno = %d", i);
48         printf("\n end seqno   = %d", i+sendsize-1);
49
50         send(sock, frame, strlen(frame), 0);
51         printf("\nData sent. Waiting for ack...");
52
53         recv(sock, &ack, sendsize, 0);
54         printf("\nreceived ack no = %d ",ack);
55
56         i = ack; // next data seq no = incoming ack no
57     }
58     send(sock, "Exit", strlen("Exit"), 0);
59     close(sock); printf("\n\n");
60 }

```

```
storm-breaker@stormbreaker: ~/Desktop
File Edit View Search Terminal Help
storm-breaker@stormbreaker:~/Desktop$ gcc ClientSW.c -o client.out
storm-breaker@stormbreaker:~/Desktop$ ./client.out

Client is connected to Server

Enter message to send : helloworld

Sending packet = hello
Sending window:
  start seqno = 0
  end seqno   = 4
Data sent. Waiting for ack...
received ack no = 2

Sending packet = llowo
Sending window:
  start seqno = 2
  end seqno   = 6
Data sent. Waiting for ack...
received ack no = 3

Sending packet = lowor
Sending window:
  start seqno = 3
  end seqno   = 7
Data sent. Waiting for ack...
received ack no = 8

Sending packet = ld
Sending window:
  start seqno = 8
  end seqno   = 9
Data sent. Waiting for ack...
received ack no = 10
```

```
storm-breaker@stormbreaker: ~/Desktop
File Edit View Search Terminal Help
storm-breaker@stormbreaker:~/Desktop$ gcc ServerSW.c -o server.out
storm-breaker@stormbreaker:~/Desktop$ ./server.out

Server is Online

Packet received = hexlo
Error at byte   = 3
Receiving window:
  start seqno = 0
  end seqno   = 2
Sending ack = 2

Packet received = lxowo
Error at byte   = 2
Receiving window:
  start seqno = 2
  end seqno   = 3
Sending ack = 3

Packet received = lowor
Receiving window:
  start seqno = 3
  end seqno   = 7
Sending ack = 8

Packet received = ld
Receiving window:
  start seqno = 8
  end seqno   = 9
Sending ack = 10

Final string recieved at Destination = helloworld

storm-breaker@stormbreaker:~/Desktop$
```



```

1 /* Author : Akshat Agarwal
2
3 8. Write a program for Error Detection using CRC-CCITT(16 bits). */
4
5 # include <stdio.h>
6 # include <string.h>
7 # include <stdlib.h>
8
9 # define MAX 30
10
11 /* crc(dividend , divisor, remainder) */
12 void crc(char *data, char *gen, char *rem)
13 {
14     int i, j, k=0;
15     char out[MAX]; // xored val after each step
16
17     strcpy(out, data);
18
19     /* Perform XOR on the msg */
20     for(i=0; i<strlen(data)-strlen(gen)+1; i++)
21     {
22         if(out[i] == '1')
23         {
24             out[i] = '0' ;
25             for(j=1; j<strlen(gen); j++)
26             {
27                 out[i+j] = (out[i+j] == gen[j]) ? '0' : '1';
28             }
29         }
30     }
31
32     // size of output = strlen(gen)-1 = 16 bits
33     int idx = strlen(out)-strlen(gen)+1; // last 16 bits of out array
34     for(i=0; i<strlen(gen)-1; i++)
35     {
36         rem[i] = out[idx+i]; // last 16 bits of out array
37     }
38 }
39
40 int main()
41 {
42     int i, j;
43     char dword[MAX]; // dataword
44     char augWord[MAX]; // augmented dataword
45     char cword[MAX]; // codeword
46     char rem[MAX]; // remainder from crc
47     char recv[MAX]; // received message
48     char gen[MAX] = "10001000000100001\0";
49
50     printf("\nCRC-16 Generator : x^16 + x^12 + x^5 + 1 ");
51     printf("\nBinary Form      : %s", gen);
52
53     printf("\n\nEnter Dataword   : ");
54     scanf("%s", dword);
55
56     strcpy(augWord, dword);
57     for(i=0; i<strlen(gen)-1; i++)
58     {
59         strcat(augWord, "0");
60     }

```

```

61     printf("\nAugmented dataword is   : %s",augWord);
62
63     crc(augWord, gen, rem);
64
65     strcpy(cword, dword);
66     strcat(cword, rem);
67     printf("\n\nFinal data transmitted   : %s", cword);
68
69     printf("\n\nEnter the data received : ");
70     scanf("%s", recv);
71     if(strlen(recv) < strlen(cword))
72     {
73         printf("\n Invalid input \n");
74         exit(0);
75     }
76
77     crc(recv, gen, rem);
78
79     printf("\nSyndrome = %s ", rem);
80     for(i=0; i<strlen(rem); i++)
81     {
82         if(rem[i] == '1')
83         {
84             printf("\nError occured !!! Corrupted data received. \n");
85             exit(0);
86         }
87     }
88     printf("\nNo Error. Data received successfully.\n");
89 }
90
91 /***** Output -1 *****/
92 CRC-16 Generator : x^16 + x^12 + x^5 + 1
93 Binary Form      : 10001000000100001
94
95 Enter Dataword   : 11110001
96
97 Augmented dataword is   : 111100010000000000000000
98
99 Final data transmitted   : 111100011111111100111110
100
101 Enter the data received : 111100011111111100111110
102
103 Syndrome = 0000000000000000
104 No Error. Data received successfully.
105 *****/
106
107 /***** Output -2 *****/
108 CRC-16 Generator : x^16 + x^12 + x^5 + 1
109 Binary Form      : 10001000000100001
110
111 Enter Dataword   : 10101011
112
113 Augmented dataword is   : 101010110000000000000000
114
115 Final data transmitted   : 101010110000010010000001
116
117 Enter the data received : 101010110000000000000000
118
119 Syndrome = 0000010010000001
120 Error occured !!! Corrupted data received.
121 *****/

```

For Network simulation using NS2

1. Install NS2

```
sudo apt install ns2
```

2. Install NAM (Network animator)

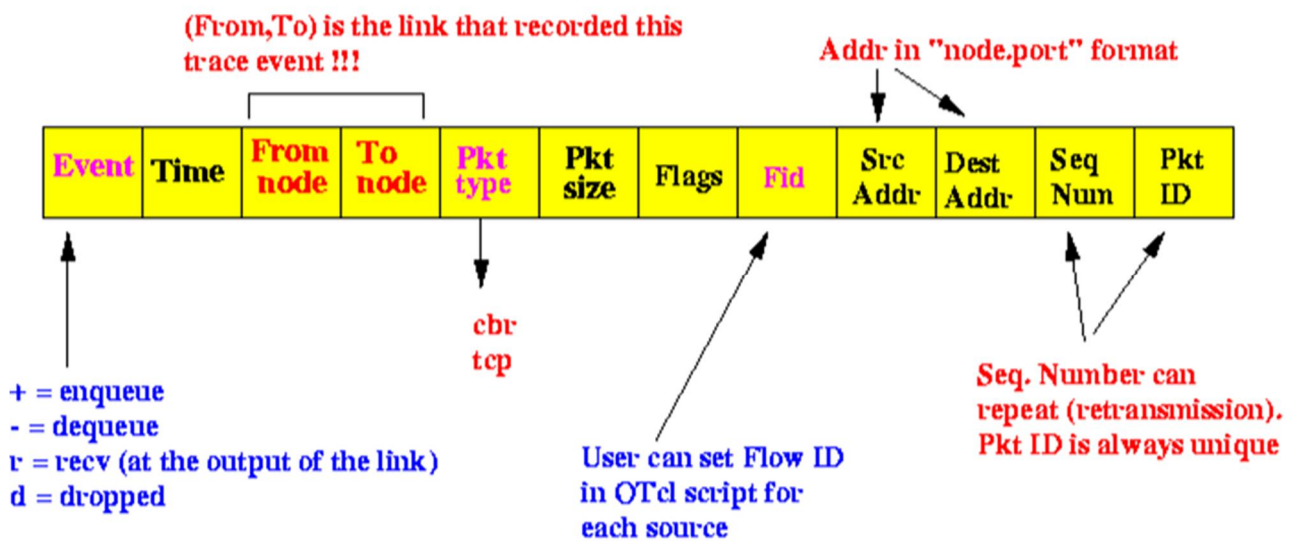
```
sudo apt install nam
```

3. Install tcl (Tool command language)

```
sudo apt install tcl
```

Structure of trace file

The trace(.tr) file contains 12 fields as shown in below figure:



```

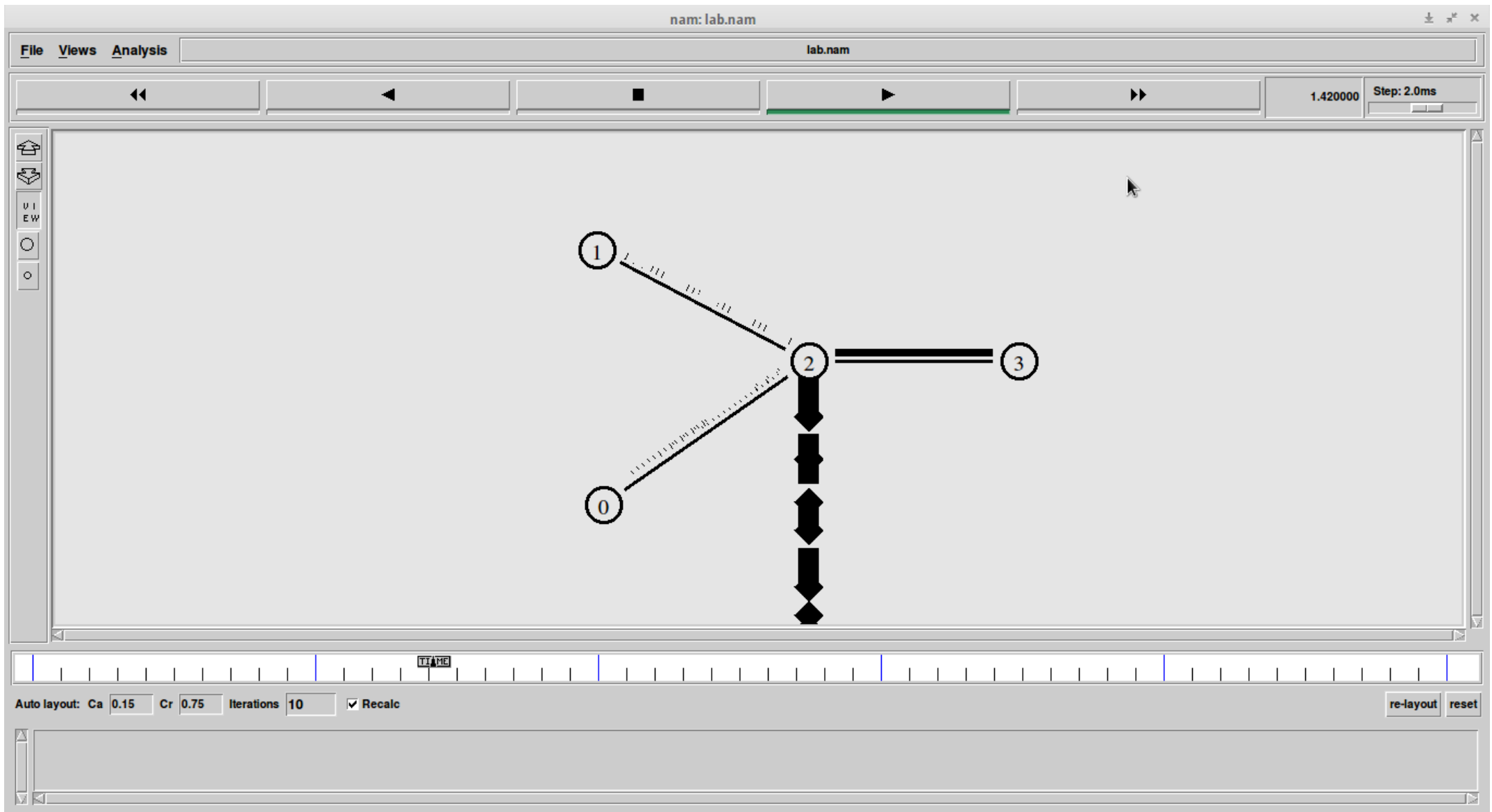
1 # Author : Akshat Agarwal
2
3 # B1. Simulate a three nodes point - to - point network with duplex links between
  them.Set the queue size and vary the bandwidth and find the number of packets
  dropped.
4
5
6 # create a new simulator
7 set ns [new Simulator]
8
9 # open trace and NAM trace file in write mode
10 set tf [open out.tr w]
11 $ns trace-all $tf
12 set nf [open out.nam w]
13 $ns namtrace-all $nf
14
15 ##### Decide a topology #####
16 #
17 #   [udp][cbr]
18 #   [0]-----
19 #           |           [null]
20 #           [2]-----[3]
21 #           |
22 #   [1]-----
23 # [udp][cbr]
24 #
25 #####
26
27 # create 4 nodes
28 set n0 [$ns node]
29 set n1 [$ns node]
30 set n2 [$ns node]
31 set n3 [$ns node]
32
33 # create duplex links between nodes
34 $ns duplex-link $n0 $n2 10Mb 300ms DropTail
35 $ns duplex-link $n1 $n2 10Mb 300ms DropTail
36 $ns duplex-link $n2 $n3 1Mb 300ms DropTail
37
38 # set up queue size
39 $ns queue-limit $n0 $n2 10
40 $ns queue-limit $n1 $n2 10
41 $ns queue-limit $n2 $n3 10
42
43 # setup udp connection for transport layer
44 set udp0 [new Agent/UDP]
45 set udp1 [new Agent/UDP]
46 set null3 [new Agent/Null]
47
48 $ns attach-agent $n0 $udp0
49 $ns attach-agent $n1 $udp1
50 $ns attach-agent $n3 $null3
51
52 # setup cbr(constant bit rate) over udp for application layer
53 set cbr0 [new Application/Traffic/CBR]
54 set cbr1 [new Application/Traffic/CBR]
55 $cbr0 attach-agent $udp0
56 $cbr1 attach-agent $udp1
57
58 # connect source to destination

```

```

59 $ns connect $udp0 $null3
60 $ns connect $udp1 $null3
61
62 # set bandwidth (vary values for different output)
63 $cbr0 set packetSize_ 500Mb
64 $cbr1 set packetSize_ 500Mb
65 $cbr0 set interval_ 0.005
66 $cbr1 set interval_ 0.005
67
68 # define a finish procedure
69 proc finish {} {
70     global ns nf tf
71     $ns flush-trace
72     exec nam out.nam &
73     close $tf
74     close $nf
75
76     set count 0
77     set tf [open out.tr r]
78     while {[gets $tf line] != -1} {
79         # d is event in the trace file which denotes dropped packets
80         if { [string match "d*" $line] } {
81             set count [expr $count + 1]
82         }
83     }
84     puts "Number of packets dropped: $count"
85     exit 0
86 }
87
88 # schedule events
89 $ns at 0.01 "$cbr0 start"
90 $ns at 0.01 "$cbr1 start"
91 $ns at 5.0 "finish"
92 $ns run
93
94 ##### output #####
95
96 # Number of packets dropped: 700
97
98 #####

```



```

1 # Author : Akshat Agarwal
2
3 # 2. Simulate the different types of Internet traffic such as FTP and TELNET over a
  network and analyze the throughput.
4
5
6 # create a new simulator
7 set ns [new Simulator]
8
9 # open trace and NAM trace file in write mode
10 set tf [open out.tr w]
11 $ns trace-all $tf
12 set nf [open out.nam w]
13 $ns namtrace-all $nf
14
15 ##### Decide a topology #####
16 #
17 #   [ftp]
18 #   [tcp]
19 #   [0]-----
20 #           |           [sink0]
21 #           [2]-----[3]
22 #           |           [sink1]
23 #   [1]-----
24 #   [tcp]
25 #   [telnet]
26 #
27 #####
28
29 # create 4 nodes
30 set n0 [$ns node]
31 set n1 [$ns node]
32 set n2 [$ns node]
33 set n3 [$ns node]
34
35 # create duplex links between nodes
36 $ns duplex-link $n0 $n2 2Mb 1ms DropTail
37 $ns duplex-link $n1 $n2 2Mb 1ms DropTail
38 $ns duplex-link $n2 $n3 2Mb 1ms DropTail
39
40 # set n0 and n1 as tcp source
41 set tcp0 [new Agent/TCP]
42 set tcp1 [new Agent/TCP]
43 $ns attach-agent $n0 $tcp0
44 $ns attach-agent $n1 $tcp1
45
46 # set n3 as tcp destination for n0 and n1
47 set TCPS0 [new Agent/TCPSink]
48 set TCPS1 [new Agent/TCPSink]
49 $ns attach-agent $n3 $TCPS0
50 $ns attach-agent $n3 $TCPS1
51
52 # set ftp over tcp0
53 set ftp0 [new Application/FTP]
54 $ftp0 attach-agent $tcp0
55
56 #set telnet over tcp1
57 set tell [new Application/Telnet]
58 $tell attach-agent $tcp1
59

```

```

60 $tel1 set packetSize_ 500Mb
61 $tel1 set interval_ 0.001
62
63 # connect source to destination
64 $ns connect $tcp0 $TCPS0
65 $ns connect $tcp1 $TCPS1
66
67 proc finish { } {
68     global ns nf tf
69     $ns flush-trace
70     exec nam out.nam &
71     close $tf
72     close $nf
73
74     # because time difference between start and finish is 2
75     set time 2
76     set fCount 0
77     set tCount 0
78     set tf [open out.tr r]
79     while {[gets $tf line] != -1} {
80         if { [string match "*tcp*0.0*3.0*" $line] } {
81             set fCount [expr $fCount + 1]
82         }
83         if { [string match "*tcp*1.0*3.1*" $line] } {
84             set tCount [expr $tCount + 1]
85         }
86     }
87     puts "Throughput of FTP: [expr $fCount/$time]"
88     puts "Throughput of TELNET: [expr $tCount/$time]"
89     exit 0
90 }
91
92 # schedule events
93 $ns at 0.01 "$ftp0 start"
94 $ns at 0.01 "$tel1 start"
95 $ns at 2.01 "finish"
96 $ns run
97
98 ##### output #####
99
100 # No of FTP packets: 767
101 # No of TELNET packets: 750
102
103 #####

```


Activities Toplevel

Fri Oct 18 1:39:30 PM

100% 1 Mb/s

56%

nam: out.nam

out.nam

0.075898 Step: 1.0us

```
graph LR; 0((0)) --- 2((2)); 1((1)) --- 2((2)); 2((2)) --- 3[3];
```

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

re-layout reset

```

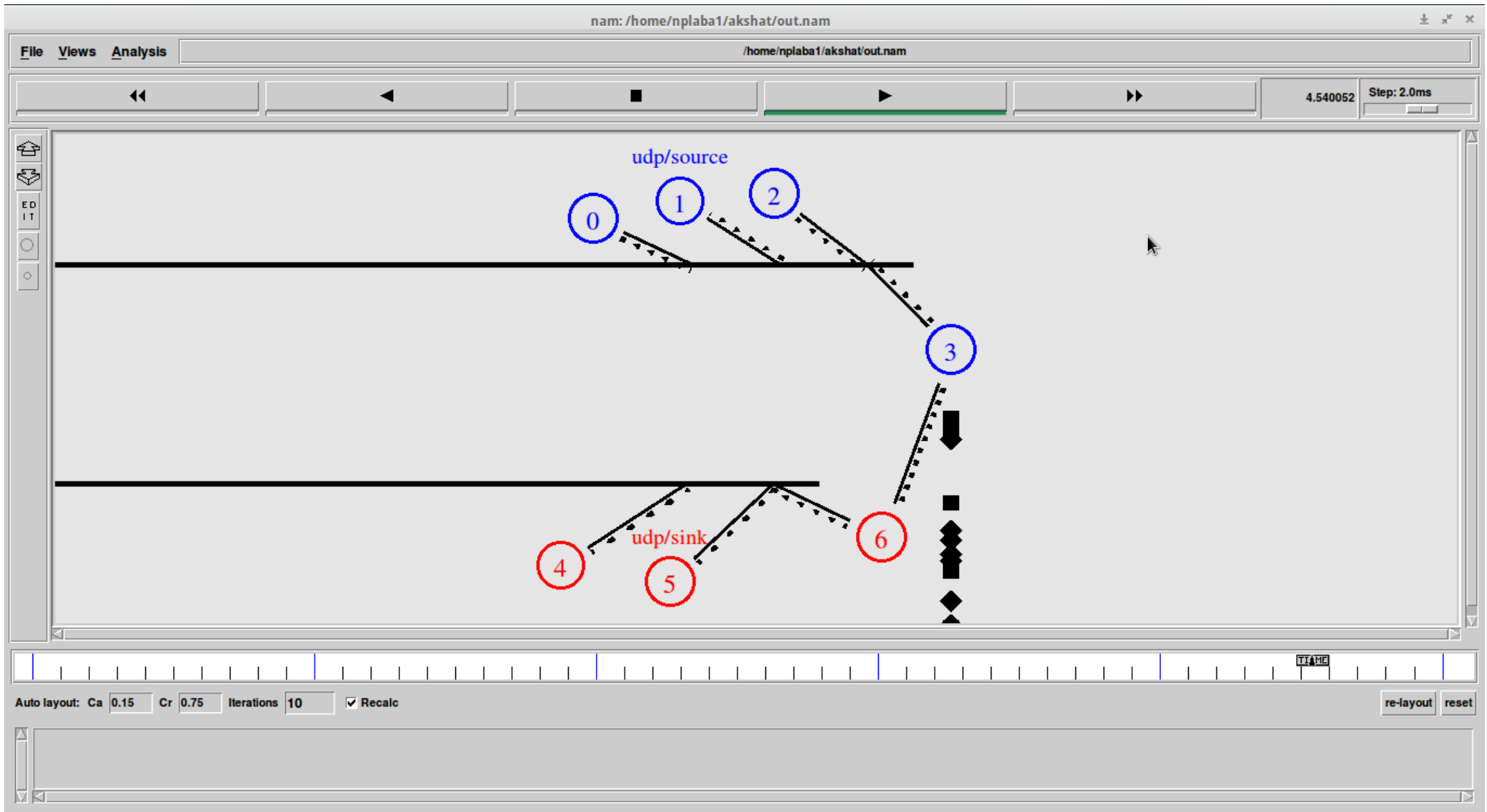
1 # Author : Akshat Agarwal
2
3 # 3. Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate
  and compare the throughput.
4
5
6 # Declare a new Simulator
7 set ns [new Simulator]
8
9 # Open nam and trace file in write mode
10 set tf [open out.tr w]
11 set nf [open out.nam w]
12 $ns trace-all $tf
13 $ns namtrace-all $nf
14
15 # Take value of error rate and data rate from std input
16 puts "Enter error rate (<1) : "
17 gets stdin erate
18
19 puts "Enter data rate (in Mbps) : "
20 gets stdin drate
21
22 ##### Select a topology #####
23 #
24 #           [udp1]                duplex-link
25 #.   [n0]   [n1] [n2]   [n3]-----
26 #       |       |       |       |
27 #       |       |       |       |
28 # -----lan7
29 #
30 # -----lan8
31 #       |       |       |
32 #       |       |       |
33 #   [n4]   [n5]   [n6]-----
34 #           [null5]
35 #
36 #####
37
38 # Create nodes
39 set n0 [$ns node]
40 set n1 [$ns node]
41 set n2 [$ns node]
42 set n3 [$ns node]
43 set n4 [$ns node]
44 set n5 [$ns node]
45 set n6 [$ns node]
46
47 # set label and color (OPTIONAL)
48 $n1 label "udp/source"
49 $n5 label "udp/null"
50 $n0 color "blue"
51 $n1 color "blue"
52 $n2 color "blue"
53 $n3 color "blue"
54 $n4 color "red"
55 $n5 color "red"
56 $n6 color "red"
57
58 # Create two lans
59 $ns make-lan "$n0 $n1 $n2 $n3" 10Mb 10ms LL Queue/DropTail Mac/802_3

```

```

60 $ns make-lan "$n4 $n5 $n6" 10Mb 10ms LL Queue/DropTail Mac/802_3
61
62 # Setup Links
63 $ns duplex-link $n3 $n6 10Mb 10ms DropTail
64
65 # Declare the transport layer protocols
66 set udp1 [new Agent/UDP]
67 set null5 [new Agent/Null]
68 $ns attach-agent $n1 $udp1
69 $ns attach-agent $n5 $null5
70
71 # Declare the application layer protocol
72 set cbr1 [new Application/Traffic/CBR]
73 $cbr1 attach-agent $udp1
74
75 # Connect the source and destination
76 $ns connect $udp1 $null5
77
78 # Create error model
79 set err [new ErrorModel]
80 $ns lossmodel $err $n3 $n6
81 $err set rate_ $erate
82
83 # Define the data rate
84 $cbr1 set packetSize_ $drate.Mb
85 $cbr1 set interval_ 0.001
86
87 # Define procedure
88 proc finish { } {
89     global ns nf tf
90     $ns flush-trace
91     exec nam out.nam &
92     close $nf
93     close $tf
94
95     set count 0
96     set tr [open out.tr r]
97     while {[gets $tr line] != -1} {
98         # 8 denotes LAN at destination side and 5 denotes destination node
99         if {[string match "* 8 5 *" $line]} {
100             set count [expr $count+1]
101         }
102     }
103     set thr [expr $count/5]
104     puts "Throughput : $thr"
105     exit 0
106 }
107
108 $ns at 0.1 "$cbr1 start"
109 $ns at 5.1 "finish"
110 $ns run
111
112 ##### output-1 #####
113
114 # Enter error rate (<1) :
115 # 0.4
116 # Enter data rate (in Mbps) :
117 # 1000
118 # Throughput : 593
119
120 #####

```



```

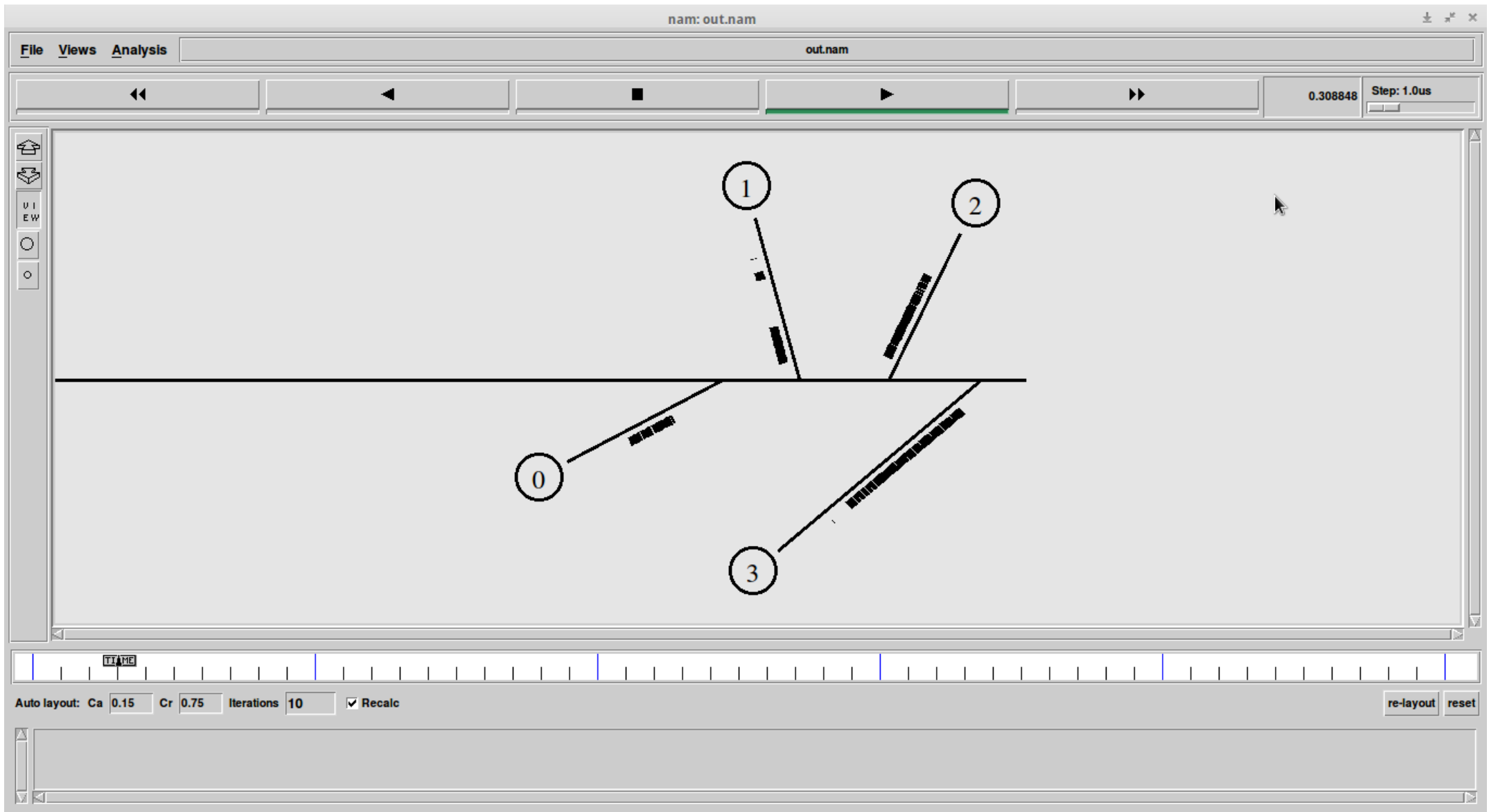
1 # Author : Akshat Agarwal
2
3 # 4. Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and
  determine the collision across different nodes.
4
5
6 # Declare a new Simulator
7 set ns [new Simulator]
8
9 # Open the trace and nam file in write mode
10 set tf [open out.tr w]
11 set nf [open out.nam w]
12 $ns trace-all $tf
13 $ns namtrace-all $nf
14
15 # Decide the topology: [tcp(0->2)], [udp(2->1)], [tcp(1->3)]
16 #
17 #      [tcp0]   [tcp1][null1]
18 #      [n0]     [n1]
19 #      |       |
20 #      |       |
21 # ----- lan4
22 #      |       |
23 #      |       |
24 #      [n3]     [n2]
25 #      [sink3]  [udp2][sink2]
26
27 # Create 4 nodes
28 set n0 [$ns node]
29 set n1 [$ns node]
30 set n2 [$ns node]
31 set n3 [$ns node]
32
33 # Create lan and setup the link
34 $ns make-lan -trace on "$n0 $n1 $n2 $n3" 100Mb 10ms LL Queue/DropTail Mac/802_3
35
36 # Declare the required transport layer Protocols
37 set tcp0 [new Agent/TCP]
38 set tcp1 [new Agent/TCP]
39 set udp2 [new Agent/UDP]
40 set null1 [new Agent/Null]
41 set sink2 [new Agent/TCPSink]
42 set sink3 [new Agent/TCPSink]
43
44 # Attach these Protocols to their respective nodes
45 $ns attach-agent $n0 $tcp0
46 $ns attach-agent $n1 $tcp1
47 $ns attach-agent $n2 $udp2
48 $ns attach-agent $n1 $null1
49 $ns attach-agent $n2 $sink2
50 $ns attach-agent $n3 $sink3
51
52 # Declare Application layer protocols and attach them with their transport layer
  protocols
53 set ftp0 [new Application/FTP]
54 set ftp1 [new Application/FTP]
55 set cbr2 [new Application/Traffic/CBR]
56 $ftp0 attach-agent $tcp0
57 $ftp1 attach-agent $tcp1
58 $cbr2 attach-agent $udp2

```

```

59
60 # connect source to destination
61 $ns connect $tcp0 $sink2
62 $ns connect $udp2 $null1
63 $ns connect $tcp1 $sink3
64
65 # set the interval
66 $ftp0 set interval_ 0.001
67 $ftp1 set interval_ 0.001
68 $cbr2 set interval_ 0.01
69
70 # define finish procedure
71 proc finish {} {
72     global ns nf tf
73     $ns flush-trace
74     exec nam out.nam &
75     close $tf
76     close $nf
77
78     set count 0
79     set tr [open out.tr r]
80     while {[gets $tr line] !=-1 } {
81         if { [string match "c*" $line] } {
82             set count [expr $count + 1]
83         }
84     }
85     puts "No of packets collided: $count"
86     exit 0
87 }
88
89 # schedule the events
90 $ns at 0.1 "$cbr2 start"
91 $ns at 0.1 "$ftp0 start"
92 $ns at 0.1 "$ftp1 start"
93 $ns at 5.0 "finish"
94 $ns run
95
96 ##### output #####
97
98 # No of packets collided: 242
99
100 #####

```



```

1 # Author : Akshat Agarwal
2
3 # 5. Simulate the transmission of ping messages over a network topology consisting
4 # of 6 nodes and find the number of packets dropped due to congestion.
5
6 # Declare new Simulator
7 set ns [new Simulator]
8
9 # Open trace and nam file in write mode
10 set tf [open out.tr w]
11 set nf [open out.nam w]
12 $ns trace-all $tf
13 $ns namtrace-all $nf
14
15 # Decide the topology
16 #
17 #   [s0][ping]       [ping]       [ping]
18 #     [n0]           [n1]         [n3]
19 #
20 #
21 #
22 #
23 #
24 #           [n2]
25 #
26 #
27 #
28 #
29 #
30 #
31 #   [n4]           [n5]         [n6]
32 #   [ping][d0]    [s1][ping]    [ping][d1]
33
34 # Create the nodes
35 set n0 [$ns node]
36 set n1 [$ns node]
37 set n2 [$ns node]
38 set n3 [$ns node]
39 set n4 [$ns node]
40 set n5 [$ns node]
41 set n6 [$ns node]
42
43 # set up links
44 $ns duplex-link $n0 $n2 100Mb 300ms DropTail
45 $ns duplex-link $n5 $n2 100Mb 300ms DropTail
46 $ns duplex-link $n1 $n2 1Mb 300ms DropTail
47 $ns duplex-link $n3 $n2 1Mb 300ms DropTail
48 $ns duplex-link $n2 $n4 1Mb 300ms DropTail
49 $ns duplex-link $n2 $n6 1Mb 300ms DropTail
50
51 # set up queue size
52 $ns queue-limit $n0 $n2 5
53 $ns queue-limit $n5 $n2 5
54 $ns queue-limit $n2 $n4 3
55 $ns queue-limit $n2 $n6 2
56
57 # Declare the agents/protocols
58 set ping0 [new Agent/Ping]
59 set ping4 [new Agent/Ping]

```

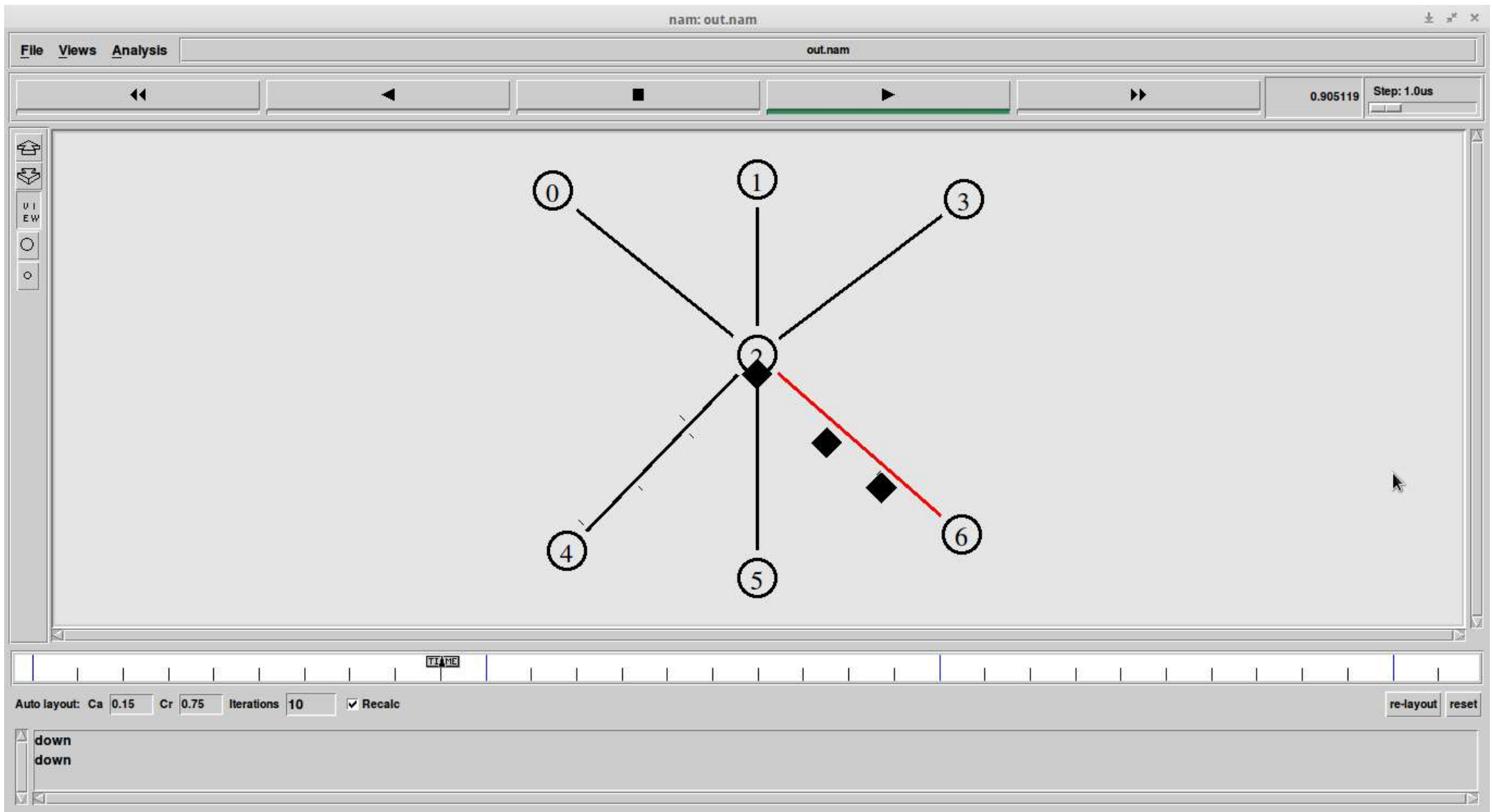


```

60 set ping5 [new Agent/Ping]
61 set ping6 [new Agent/Ping]
62
63 # Attach the ping with the respective nodes
64 $ns attach-agent $n0 $ping0
65 $ns attach-agent $n4 $ping4
66 $ns attach-agent $n5 $ping5
67 $ns attach-agent $n6 $ping6
68
69 # Connect the ping from source to destination
70 $ns connect $ping0 $ping4
71 $ns connect $ping5 $ping6
72
73 # Write proc for ping agent
74 Agent/Ping instproc recv {from rtt} {
75     $self instvar node_
76     puts "The node [$node_ id] recieved $from with round trip time $rtt"
77 }
78
79 # Write the proc function
80 proc finish { } {
81     global ns nf tf
82     $ns flush-trace
83     exec nam out.nam &
84     close $nf
85     close $tf
86
87     set count 0
88     set tr [open out.tr r]
89     while {[gets $tr line] != -1} {
90         if {[string match "d*" $line]} {
91             set count [expr $count + 1]
92         }
93     }
94     puts "No. of packet dropped : $count"
95     exit 0
96 }
97
98 $ns rtmodel-at 0.9 down $n2 $n6
99 $ns rtmodel-at 1.9 up $n2 $n6
100
101 # schedule events
102 for {set i 0.1} {$i < 2} {set i [expr $i + 0.1]} {
103     $ns at $i "$ping0 send"
104     $ns at $i "$ping5 send"
105 }
106
107 $ns at 5.0 "finish"
108 $ns run
109
110 ##### output #####
111 # The node 0 recieved 2 with round trip time 1201.0
112 # The node 0 recieved 2 with round trip time 1201.0
113 # The node 0 recieved 2 with round trip time 1201.0
114 # The node 0 recieved 2 with round trip time 1201.0
115 # The node 0 recieved 2 with round trip time 1201.0
116 # The node 0 recieved 2 with round trip time 1201.0
117 # The node 0 recieved 2 with round trip time 1201.0
118 # The node 0 recieved 2 with round trip time 1201.0
119 # The node 0 recieved 2 with round trip time 1201.0
120 # The node 0 recieved 2 with round trip time 1201.0

```

```
121 # The node 0 recieved 2 with round trip time 1201.0
122 # The node 0 recieved 2 with round trip time 1201.0
123 # The node 0 recieved 2 with round trip time 1201.0
124 # The node 0 recieved 2 with round trip time 1201.0
125 # The node 0 recieved 2 with round trip time 1201.0
126 # The node 0 recieved 2 with round trip time 1201.0
127 # The node 3 recieved 4 with round trip time 1201.0
128 # The node 0 recieved 2 with round trip time 1201.0
129 # The node 3 recieved 4 with round trip time 1201.0
130 # The node 0 recieved 2 with round trip time 1201.0
131 # The node 3 recieved 4 with round trip time 1201.0
132 # The node 0 recieved 2 with round trip time 1201.0
133 # The node 3 recieved 4 with round trip time 1201.0
134 # No. of packet dropped : 5
135 #####
```



```

1 # Author : Akshat Agarwal
2
3 # 6. A Simple ESS with transmitting nodes in Wireless LAN
4 # 7. A simple ad-hoc network with transmitting nodes
5
6
7 # Declare new Simulator
8 set ns [new Simulator]
9
10 # Open the trace file in write mode
11 set tf [open out.tr w]
12 $ns trace-all $tf
13
14 # Set name-trace for wireless network
15 set nf [open out.nam w]
16 $ns namtrace-all-wireless $nf 500 500
17
18 # Set new topography
19 set topo [new Topography]
20 $topo load_flatgrid 500 500
21
22 # Configure for a wireless node.
23 $ns node-config -adhocRouting DSDV \
24 -llType LL \
25 -macType Mac/802_11 \
26 -ifqType Queue/DropTail \
27 -ifqLen 50 \
28 -phyType Phy/WirelessPhy \
29 -channelType Channel/WirelessChannel \
30 -propType Propagation/TwoRayGround \
31 -antType Antenna/OmniAntenna \
32 -topoInstance $topo \
33 -agentTrace ON \
34 -routerTrace ON \
35 -macTrace OFF
36
37 # Create a god object
38 create-god 3
39
40 ##### Decide the topology #####
41 # 500
42 # |
43 # |
44 # |
45 # 400 [sink2]
46 # | [n2]
47 # |
48 # |
49 # |
50 # |
51 # |
52 # |
53 # 100 [n1]
54 # | .' [sink1]
55 # | .' [tcp1]
56 # 10 [n0] [ftp1]
57 # | [tcp0]
58 # | [ftp0]
59 # |
60 # |_____10_____100_____400_____500

```

```

61
62 # Create nodes
63 set n0 [$ns node]
64 set n1 [$ns node]
65 set n2 [$ns node]
66
67 # Locate the nodes on load_flatgrid
68 $n0 set X_ 10
69 $n0 set Y_ 10
70 $n0 set Z_ 0
71
72 $n1 set X_ 100
73 $n1 set Y_ 100
74 $n1 set Z_ 0
75
76 $n2 set X_ 400
77 $n2 set Y_ 400
78 $n2 set Z_ 0
79
80 # initial state
81 $ns at 0.0 "$n0 setdest 10 10 15"
82 $ns at 0.0 "$n1 setdest 100 100 15"
83 $ns at 0.0 "$n2 setdest 400 400 15"
84
85 # Declare and attach transport layer protocol
86 set tcp0 [new Agent/TCP]
87 set tcp1 [new Agent/TCP]
88 $ns attach-agent $n0 $tcp0
89 $ns attach-agent $n1 $tcp1
90
91 set sink1 [new Agent/TCPSink]
92 set sink2 [new Agent/TCPSink]
93 $ns attach-agent $n1 $sink1
94 $ns attach-agent $n2 $sink2
95
96 # Declare and attach application layer protocol
97 set ftp0 [new Application/FTP]
98 set ftp1 [new Application/FTP]
99 $ftp0 attach-agent $tcp0
100 $ftp1 attach-agent $tcp1
101
102 # connect source to destination
103 $ns connect $tcp0 $sink1
104 $ns connect $tcp1 $sink2
105
106
107 proc finish { } {
108     global ns nf tf
109     $ns flush-trace
110     exec nam out.nam &
111     close $tf
112
113     set ctr1 0
114     set ctr2 0
115     set tf [open out.tr r]
116
117     while {[gets $tf line] != -1} {
118         # r->received, _1_ -> destination node
119         if {[string match "r*_1_*AGT*" $line]} {
120             set ctr1 [expr $ctr1 + 1]
121         }

```

```

122         if {[string match "r*_2_*AGT*" $line]} {
123             set ctr2 [expr $ctr2 + 1]
124         }
125     }
126     puts "\nThroughput from n0 to n1: $ctr1"
127     puts "Throughput from n1 to n2: $ctr2"
128     exit 0
129 }
130
131 # schedule events
132
133 # move n1 near to node n2 at 50s and come back near to node n0 at 100s
134 $ns at 50 "$n1 setdest 300 300 15"
135 $ns at 100 "$n1 setdest 100 100 15"
136
137 # start ftp traffic
138 $ns at 1 "$ftp0 start"
139 $ns at 1 "$ftp1 start"
140 $ns at 150 "finish"
141 $ns run
142
143 ##### output #####
144
145 # num_nodes is set 3
146 # INITIALIZE THE LIST xListHead
147 # channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
148 # highestAntennaZ_ = 1.5, distCST_ = 550.0
149 # SORTING LISTS ...DONE!
150 #
151 # Throughput from n0 to n1: 8438
152 # Throughput from n1 to n2: 3000
153
154 #####
155

```

