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 Programing 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.

```c
/* Author : Gangadhar, Akshat

1. For the given network graph, write a program to
    * implement Link state routing algorithm
    * build a routing table for the given node. */

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define INFINITY 999
#define MAX 100

int cost[MAX][MAX]; // cost matrix
int distance[MAX]; // distance from source
int visited[MAX] = {0};
int parent[MAX];
int source;
int n; // number of nodes

void initialize()
{
    int i;
    visited[source] = 1;
    parent[source] = source;

    for(i=0; i<n; i++)
    {
        distance[i] = cost[source][i];
        if( cost[source][i] != INFINITY )
        {
            parent[i] = source;
        }
    }
}

/* Get minimum distant node not already in network */
int GetMin()
{
    int minIdx = -1;
    int minDist = INFINITY;

    int i;
    for(i=0; i<n; i++)
    {
        if( !visited[i] && minDist >= distance[i] )
        {
                minIdx = i;
                minDist = distance[i];
        }
    }
    return minIdx;
}

/* update distance for adjacent nodes */
void updateTable(int node)
{
    int i;
    for(i=0; i<n; i++)
    {
        if( cost[node][i] != INFINITY && distance[i] > distance[node]+cost[node][i]
)
        {
            distance[i] = distance[node] + cost[node][i];
            parent[i] = node;
        }
    }
}

void display()
{
    int i;
    int node;

    printf("\nNode \t Distance from source \t Path \n");
    for(i=0; i<n; i++)
    {
        printf("%d \t\t %d \t\t", i, distance[i]);

        // node <- parent[node] <- parent[parent[node]] <- ... <- source
        node = i;
        printf("%d", node);
        while( node != source)
        {
            printf(" <- %d", parent[node]);
            node = parent[node];
        }
        printf("\n");
    }
}

int main()
{
    int i, j, node;

    printf("Enter the number of nodes: ");
    scanf("%d", &n);

    printf("Enter the source node    : ");
    scanf("%d", &source);

    printf("\nEnter the cost matrix: \n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
        {
            scanf("%d", &cost[i][j]);
        }
    }

    initialize();

    for(i=0; i<n-1; i++) // for all remaining vertices(since source is already
visited)
    {
        node = GetMin();
        visited[node] = 1;
        updateTable(node);
    }
    display();
    return 0;
```

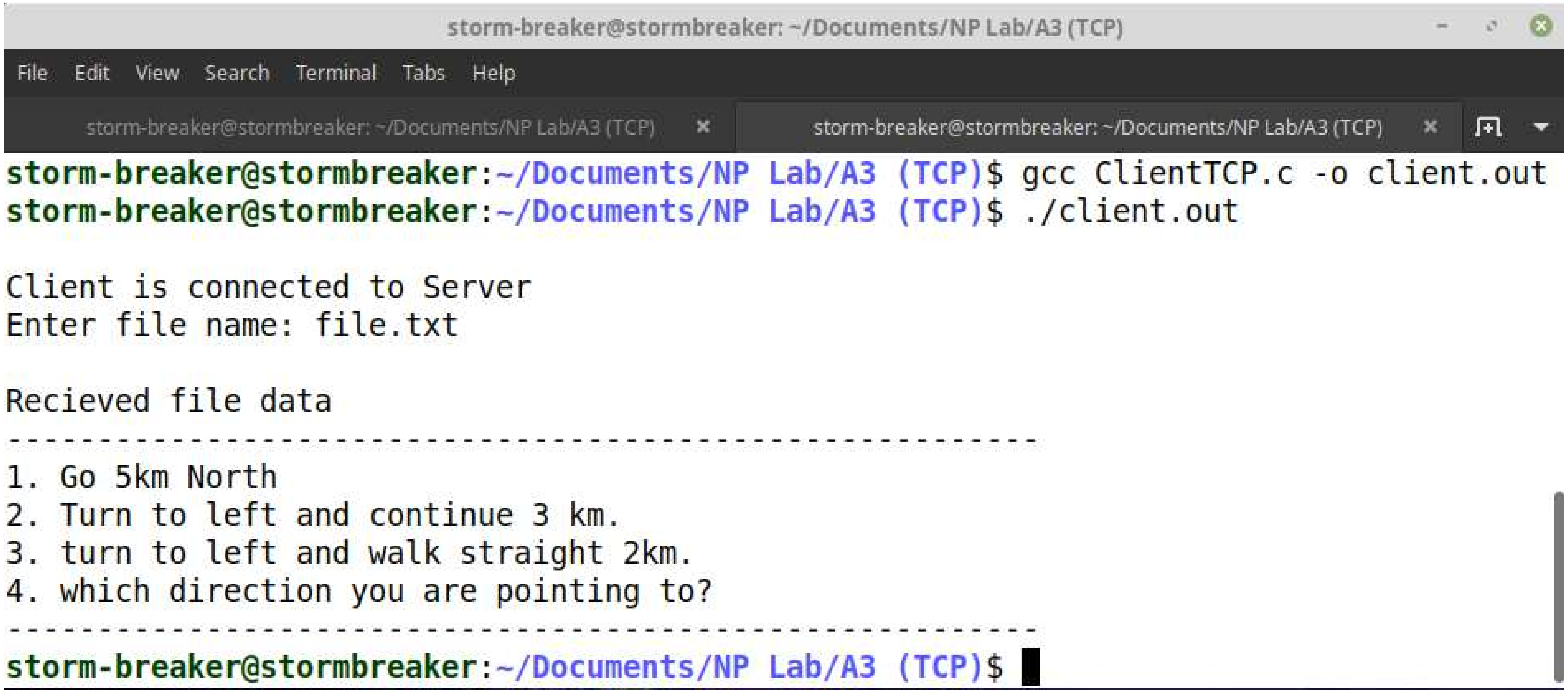
```
}


/**************** OUTPUT-1 ******************************
Enter the number of nodes: 9
Enter the source node    : 3

Enter the cost matrix:
0     4     999   999   999   999   999   8     999
4     0     8     999   999   999   999   11    999
999   8     0     7     999   4     999   999   2
999   999   7     0     9     14    999   999   999
999   999   999   9     0     10    999   999   999
999   999   4     14    10    0     2     999   999
999   999   999   999   999   2     0     1     6
8     11    999   999   999   999   1     0     7
999   999   2     999   999   999   6     7     0

Node     Distance from source     Path
0                19              0 <- 1 <- 2 <- 3
1                15              1 <- 2 <- 3
2                7               2 <- 3
3                0               3
4                9               4 <- 3
5                11              5 <- 2 <- 3
6                13              6 <- 5 <- 2 <- 3
7                14              7 <- 6 <- 5 <- 2 <- 3
8                9               8 <- 2 <- 3
*************************************************************/
```

```c
/* Author : Gangadhar, Akshat

2. Write a program to divide the message into variable length frames and sort them
and display the message at the receiving side. */

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

#define MAX 100

typedef struct{
    int id;
    char data[MAX];
}frame;

// Fisher yates algorithm to shuffle the frame
void shuffleFrame(frame f[MAX], int n)
{
    srand(time(NULL));

    int i;
    for(i=n; i>=0; i--)
    {
        int j = rand()%(i+1);

        frame temp = f[j];
        f[j] = f[i];
        f[i] = temp;
    }
}

// Insertion sort algorithm to sort frames based on id
void sortFrames(frame f[MAX], int n)
{
    int i, j;

    for(i=1; i<=n; i++)
    {
        frame t = f[i];
        j = i-1;
        while(j>=0 && f[j].id > t.id)
        {
            f[j+1] = f[j];
            j=j-1;
        }
        f[j+1] = t;
    }
}

int main()
{
    frame f[MAX];
    int n = -1;      // no of frames
    int fsize;       // size of frame

    char msg[MAX];
    int m = 0; // message iterator
    int i, j;

    printf("Enter a message : ");
    fgets(msg , MAX, stdin);
    msg[strlen(msg)-1] = '\0'; // to remove '\n' from string

    srand(time(NULL));
    // Divide the message into frames
    for(i=0 ; m < strlen(msg) ; i++)
    {
        f[i].id = i;

        n++; // count number of frames
        fsize = rand()%5+1; // variable Frame size in range [1,5]

        for(j=0 ; j<fsize && m < strlen(msg); j++)
        {
            f[i].data[j] = msg[m++];
        }
    }

    shuffleFrame(f, n);

    printf("\nShuffled frames:");
    printf("\nframe_id \t frame_data \n");
    printf("----------------------------\n");
    for(i=0 ; i <= n; i++)
    {
        printf("%d \t\t %s \n", f[i].id, f[i].data);
    }

    sortFrames(f, n);

    printf("\nSorted frames:");
    printf("\nframe_id \t frame_data \n");
    printf("----------------------------\n");
    for(i=0 ; i <= n; i++)
    {
        printf("%d \t\t %s \n", f[i].id, f[i].data);
    }

    printf("\nfinal message : ");
    for(i=0; i<= n; i++)
    {
        printf("%s", f[i].data);
    }

    printf("\n");
}


```

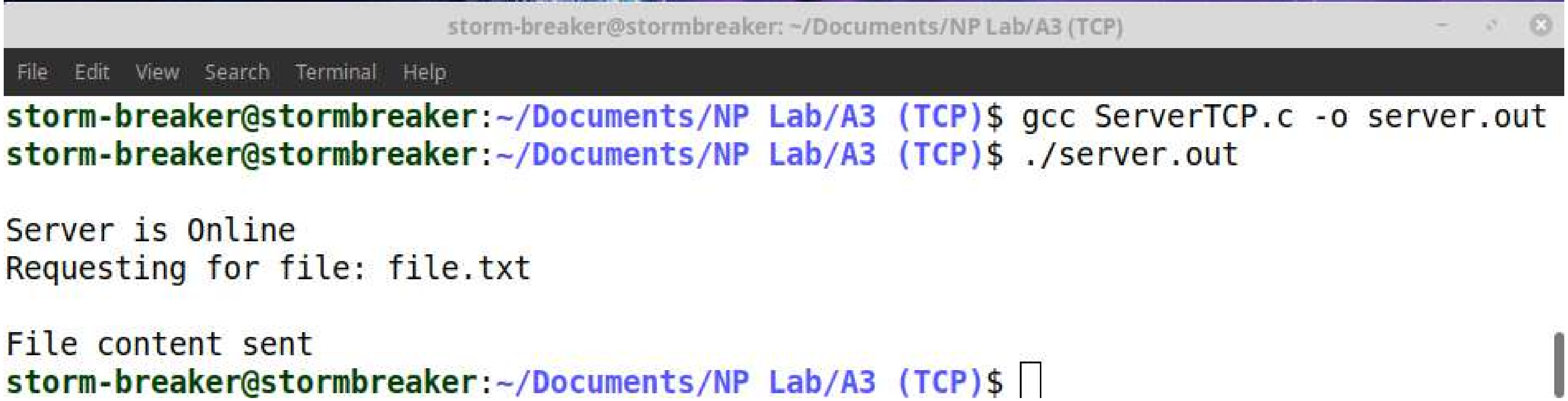
```
/**************** OUTPUT-1 ************************
Enter a message : hello beautiful world

Shuffled frames:
frame_id         frame_data
----------------------------
6                ld
1                lo be
0                hel
2                auti
5                wor
3                ful
4

Sorted frames:
frame_id         frame_data
----------------------------
0                hel
1                lo be
2                auti
3                ful
4
5                wor
6                ld

final message : hello beautiful world
**************************************************/
```

```c
/* Author : Akshat Agarwal

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. */

/* Server Program */

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <unistd.h>

int main()
{
    int sersock, sock, fd, n, reuse = 1;
    char buffer[1024], fname[50];

    /* sockfd = socket(domain, type, protocol) */
    sersock = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };

    // Forcefully connecting to same port everytime
  setsockopt(sersock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));

    /* attaching socket to port */
    bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
    printf("\nServer is Online");

    listen(sersock, 5); // listen(int sockfd, int backlog)
    sock = accept(sersock, NULL, NULL);

    /*  receive the filename */
    recv(sock, fname, 50, 0);
    printf("\nRequesting for file: %s\n", fname);

    /*  open the file in read-only mode */
    fd = open(fname, O_RDONLY);
    if (fd < 0)
    {
        send(sock, "\nFile not found\n", 15, 0); // strlen(\nFile not found)=15
    }
    else
    {
        while ((n = read(fd, buffer, sizeof(buffer))) > 0)
        {
            send(sock, buffer, n, 0);
        }
    }
    printf("\nFile content sent\n");

    close(fd);
    return 0;
}
```

```c
/* Author : Akshat Agarwal

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. */

/* Client Program */

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <unistd.h>

int main()
{
    int sock, n;
    char buffer[1024], fname[50];

    sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };

    /*  keep trying to esatablish connection with server */
    while(connect(sock, (struct sockaddr *) &addr, sizeof(addr))) ;
    printf("\nClient is connected to Server");

    /* send the filename to the server */
    printf("\nEnter file name: ");
    scanf("%s", fname);
    send(sock, fname, sizeof(fname), 0);

    printf("\nRecieved file data\n");
    printf("---------------------------------------------------------\n");

    /*  keep printing any data received from the server */
    while ((n = recv(sock, buffer, sizeof(buffer), 0)) > 0)
    {
        buffer[n] = '\0' ;
        printf("%s", buffer);
    }

  printf("---------------------------------------------------------\n");
    return 0;
}
```


storm-breaker@stormbreaker: ~/Documents/NP Lab/A3 (TCP)
File Edit View Search Terminal Tabs Help
storm-breaker@stormbreaker: ~/Documents/NP Lab/A3 (TCP)
storm-breaker@stormbreaker: ~/Documents/NP Lab/A3 (TCP)

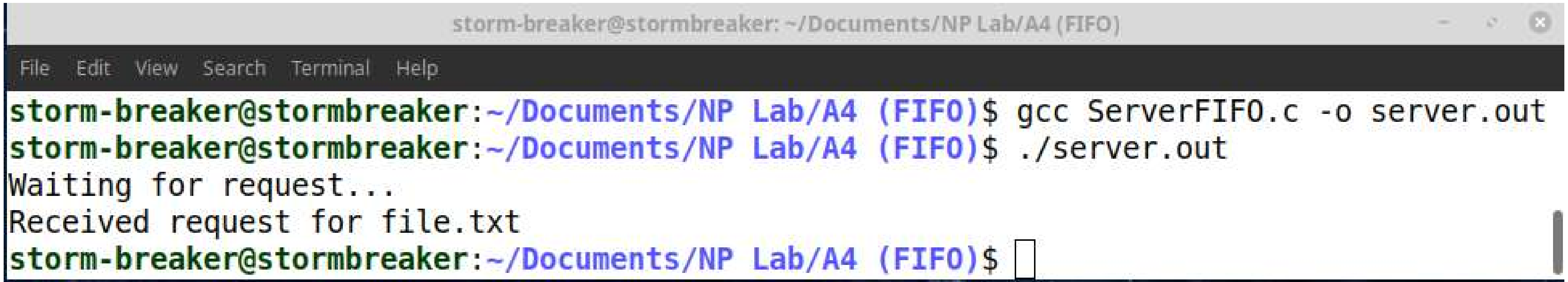
```
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ gcc ClientTCP.c -o client.out
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ ./client.out

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)
File Edit View Search Terminal Help

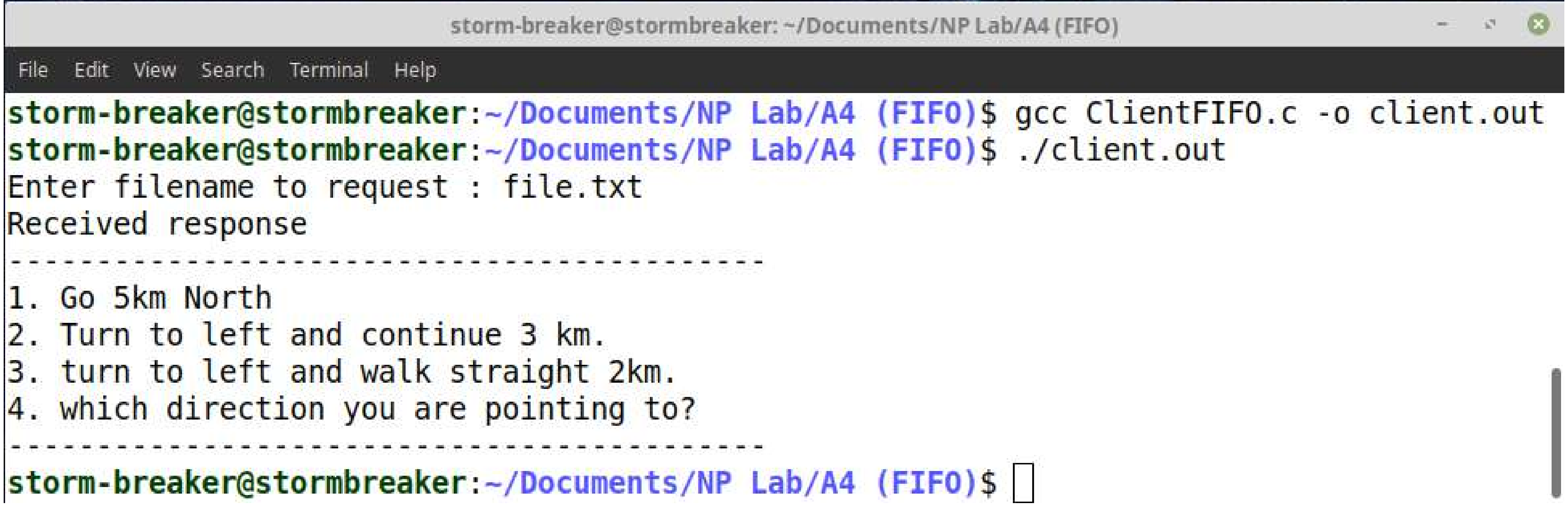
```
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ gcc ServerTCP.c -o server.out
storm-breaker@stormbreaker:~/Documents/NP Lab/A3 (TCP)$ ./server.out

Server is Online
Requesting for file: file.txt

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

```c
/* Author : Akshat Agarwal

4. Using FIFOs as IPC, write a client – server program,
    - the client sends the file name and
    - the server sends back the requested text file if present. */

/* Server Program */

#include <stdio.h>
#include <unistd.h>
#include <fcntl.h> // used for file handling
#include <sys/stat.h> // used for mkfifo function
#include <sys/types.h> // mkfifo() has dependency on both types.h and stat.h

int main()
{
    char fname[50], buffer[1025];
    int req, res, n, file;

    mkfifo("req.fifo", 0777);
    mkfifo("res.fifo", 0777);

    printf("Waiting for request...\n");
    req = open("req.fifo", O_RDONLY);
    res = open("res.fifo", O_WRONLY);

    read(req, fname, sizeof(fname));
    printf("Received request for %s\n", fname);

    file = open(fname, O_RDONLY);
    if (file < 0)
    {
        write(res, "File not found\n", 15);
    }
    else
    {
        while((n = read(file, buffer, sizeof(buffer))) > 0)
        {
            write(res, buffer, n);
        }
    }

    close(req);
    close(res);

    unlink("req.fifo");
    unlink("res.fifo");

    return 0;
}
```

```c
/* Author : Akshat Agarwal

4. Using FIFOs as IPC, write a client – server program,
    - the client sends the file name and
    - the server sends back the requested text file if present. */

/* Client Program */

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>

int main()
{
    char fname[50], buffer[1025];
    int req, res, n;

    req = open("req.fifo", O_WRONLY);
    res = open("res.fifo", O_RDONLY);

    if(req < 0 || res < 0)
    {
        printf("Please Start the server first\n");
        exit(-1);
    }

    printf("Enter filename to request : ");
    scanf("%s", fname);

    // write file name to request file
    write(req, fname, sizeof(fname));

    printf("Received response\n");
    printf("-------------------------------------------\n");
    while((n = read(res, buffer, sizeof(buffer)))>0)
    {
        printf("%s", buffer);
    }
    printf("-------------------------------------------\n");

    close(req);
    close(res);
    return 0;
}
```


storm-breaker@stormbreaker: ~/Documents/NP Lab/A4 (FIFO)
File Edit View Search Terminal Help

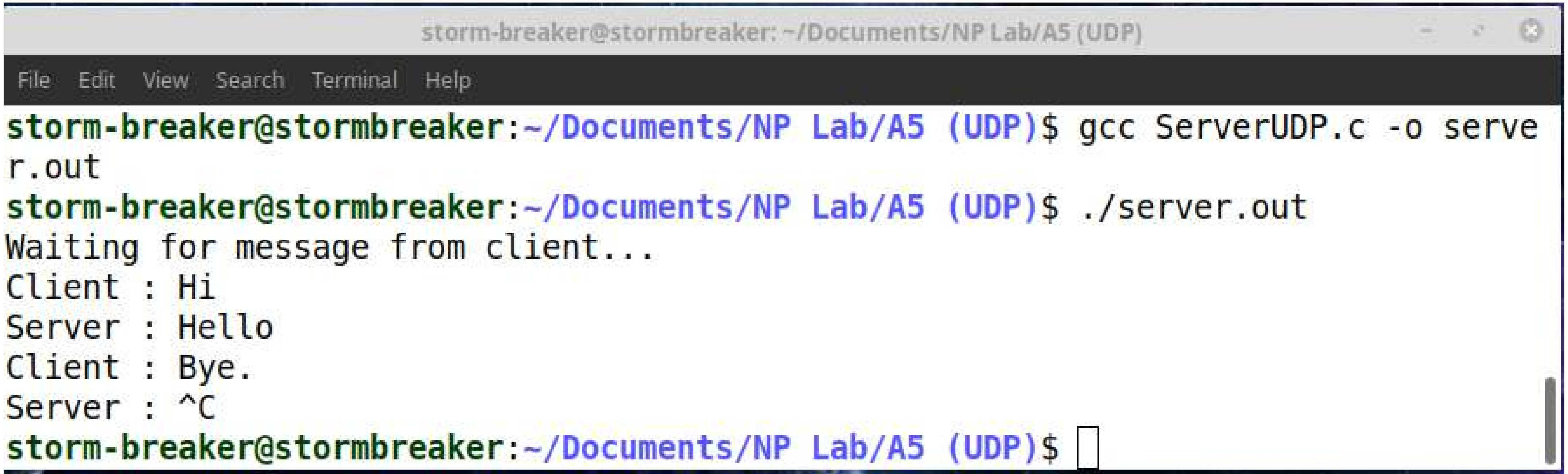
```
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$ gcc ServerFIFO.c -o server.out
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$ ./server.out
Waiting for request...
Received request for file.txt
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$
```


storm-breaker@stormbreaker: ~/Documents/NP Lab/A4 (FIFO)
File Edit View Search Terminal Help

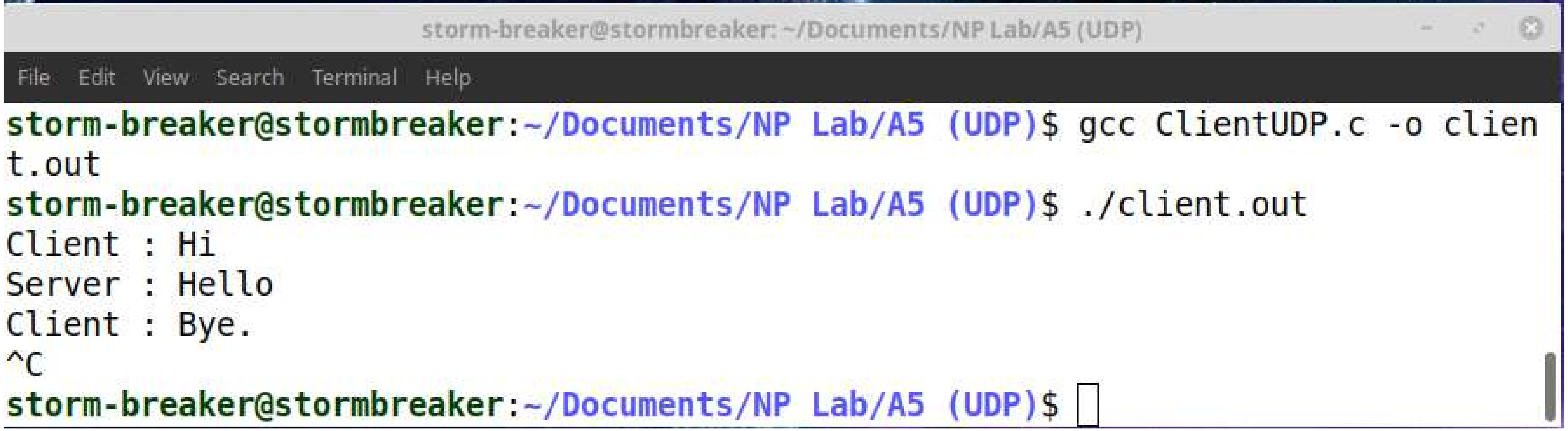
```
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$ gcc ClientFIFO.c -o client.out
storm-breaker@stormbreaker:~/Documents/NP Lab/A4 (FIFO)$ ./client.out
Enter filename to request : file.txt
Received response
-------------------------------------------
1. Go 5km North
2. Turn to left and continue 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)$
```

```c
/* Author : Akshat Agarwal

5. Using UDP, write a client-server program, to exchange messages between client and the server. */

/* Server Program */

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

#define MAX 1024

int main()
{
  int sock;
  int len, n, reuse = 1;
  char buffer[MAX], msg[MAX];
  struct sockaddr_in servaddr, cliaddr;

  sock = socket(AF_INET, SOCK_DGRAM, 0);
  // Forcefully connecting to same port everytime
  setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));

  // initialize servaddr and cliaddr to 0
  memset(&servaddr, 0, sizeof(servaddr));
  memset(&cliaddr, 0, sizeof(cliaddr));
  len = sizeof(cliaddr);

  // Filling server information
  servaddr.sin_family = AF_INET; // IPv4
  servaddr.sin_addr.s_addr = INADDR_ANY;
  servaddr.sin_port = htons(1234);

  // Bind the socket with the server address
  if (bind(sock, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0 )
  {
    printf("Binding error\n"); exit(0);
  }
  printf("Waiting for message from client...\n");

  while(1)
  {
    n = recvfrom(sock, (char *)buffer, sizeof(buffer), 0, ( struct sockaddr *)&cliaddr, &len);
    buffer[n] = '\0';
    printf("Client : %s", buffer);

    printf("Server : ");
    fgets(msg, MAX, stdin);
    sendto(sock, (const char *)msg, strlen(msg), 0, (const struct sockaddr *)&cliaddr, len);
  }
  return 0;
}
```

```c
/* Author : Akshat Agarwal

5. Using UDP, write a client – server program, to exchange messages between client
and the server. */

/* Client Program */

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

#define MAX 1024

int main()
{
  int n, len, sock;
  char buffer[MAX], msg[MAX];
  struct sockaddr_in servaddr;

  // Creating socket file descriptor
  sock = socket(AF_INET, SOCK_DGRAM, 0);

  memset(&servaddr, 0, sizeof(servaddr));
  // Filling server information
  servaddr.sin_family = AF_INET;
  servaddr.sin_port = htons(1234); // htons(port)
  servaddr.sin_addr.s_addr = INADDR_ANY;

  while( connect(sock, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0);
  printf("\nConnection Established") ;

  while(1)
  {
    printf("Client : ");
    fgets(msg, MAX, stdin);
    sendto(sock, (const char *)msg, strlen(msg), 0, (const struct sockaddr *)
&servaddr, sizeof(servaddr));

    n = recvfrom(sock, (char *)buffer, sizeof(buffer), 0, NULL, NULL);
    buffer[n] = '\0';
    printf("Server : %s", buffer);
  }
  return 0;
}
```


storm-breaker@stormbreaker: ~/Documents/NP Lab/A5 (UDP)
File Edit View Search Terminal Help

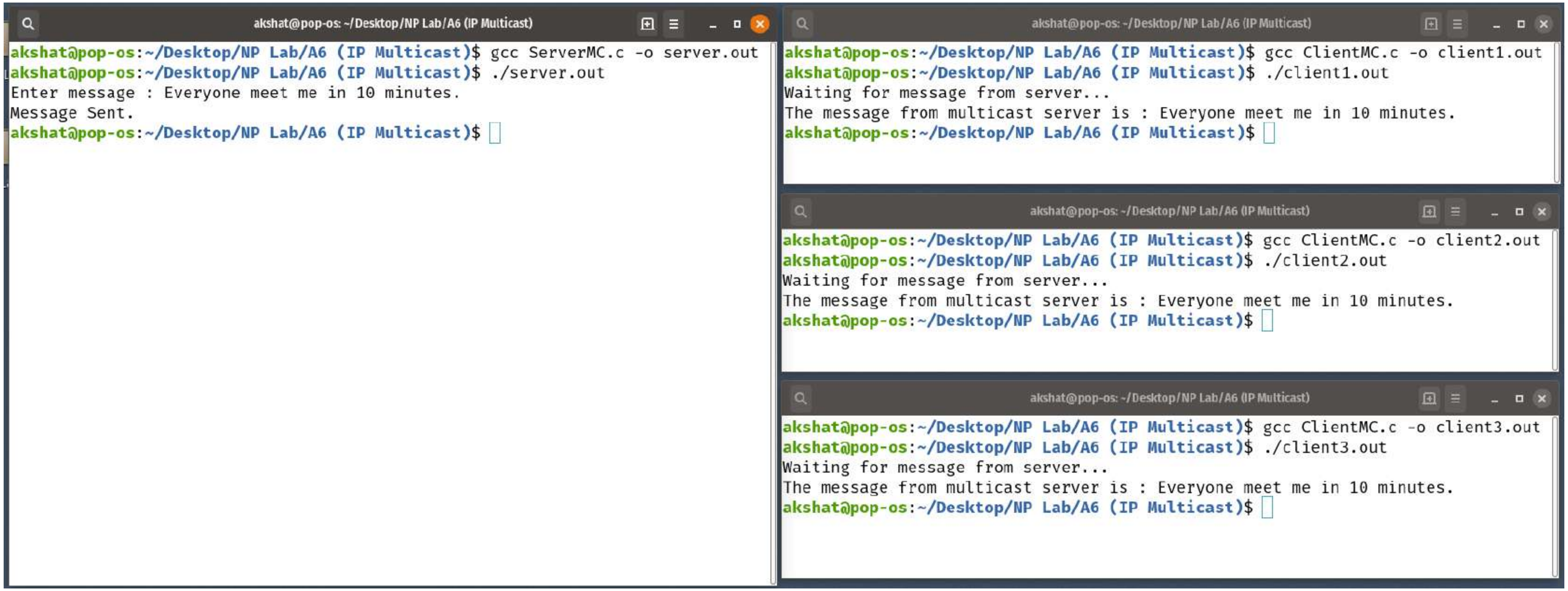
```
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$ gcc ServerUDP.c -o serve
r.out
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$ ./server.out
Waiting for message from client...
Client : Hi
Server : Hello
Client : Bye.
Server : ^C
storm-breaker@stormbreaker:~/Documents/NP Lab/A5 (UDP)$ 
```


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 clien
t.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)$ 
```

```c
/* Author : Akshat Agarwal

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. */

/**** server sends the message ****/

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

int main ()
{
    int sock;
    char msg[1024];
    struct sockaddr_in groupaddr;
    struct in_addr localInterface;

    sock = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&groupaddr, 0, sizeof(groupaddr));
    groupaddr.sin_family = AF_INET;
    groupaddr.sin_addr.s_addr = inet_addr("226.1.1.1");
    groupaddr.sin_port = htons(1234);

    localInterface.s_addr = inet_addr("127.0.0.1");  // or system ip address
    setsockopt(sock, IPPROTO_IP, IP_MULTICAST_IF, (char *)&localInterface, sizeof(localInterface));

    printf("Enter message : ");
    fgets(msg, 1024, stdin);
    msg[strlen(msg)-1] = '\0'; // to remove '\n' from string
    sendto(sock, msg, sizeof(msg), 0, (struct sockaddr*)&groupaddr, sizeof(groupaddr));
    printf("Message Sent.\n");

    return 0;
}
```

```c
/* Author : Akshat Agarwal

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. */

/**** clients recieves the message ****/

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

int main()
{
  int sock, reuse = 1;
  char msg[1024];
  struct sockaddr_in addr;
  struct ip_mreq group;

  sock = socket(AF_INET, SOCK_DGRAM, 0);

  setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));

  memset(&addr, 0, sizeof(addr));
  addr.sin_family = AF_INET;
  addr.sin_port = htons(1234);
  addr.sin_addr.s_addr = INADDR_ANY; // to listen on all available interfaces.

  if(bind(sock, (struct sockaddr*)&addr, sizeof(addr)) < 0)
  {
    printf("Binding failed");
    close(sock);
    exit(1);
  }

  group.imr_multiaddr.s_addr = inet_addr("226.1.1.1");
  group.imr_interface.s_addr = inet_addr("127.0.0.1");
  setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, (char *)&group, sizeof(group));
  printf("Waiting for message from server.....");

  read(sock, msg, sizeof(msg));
  printf("\nThe message from multicast server is : %s \n", msg);

  close(sock);
  return 0;
}
```



```
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)$
```

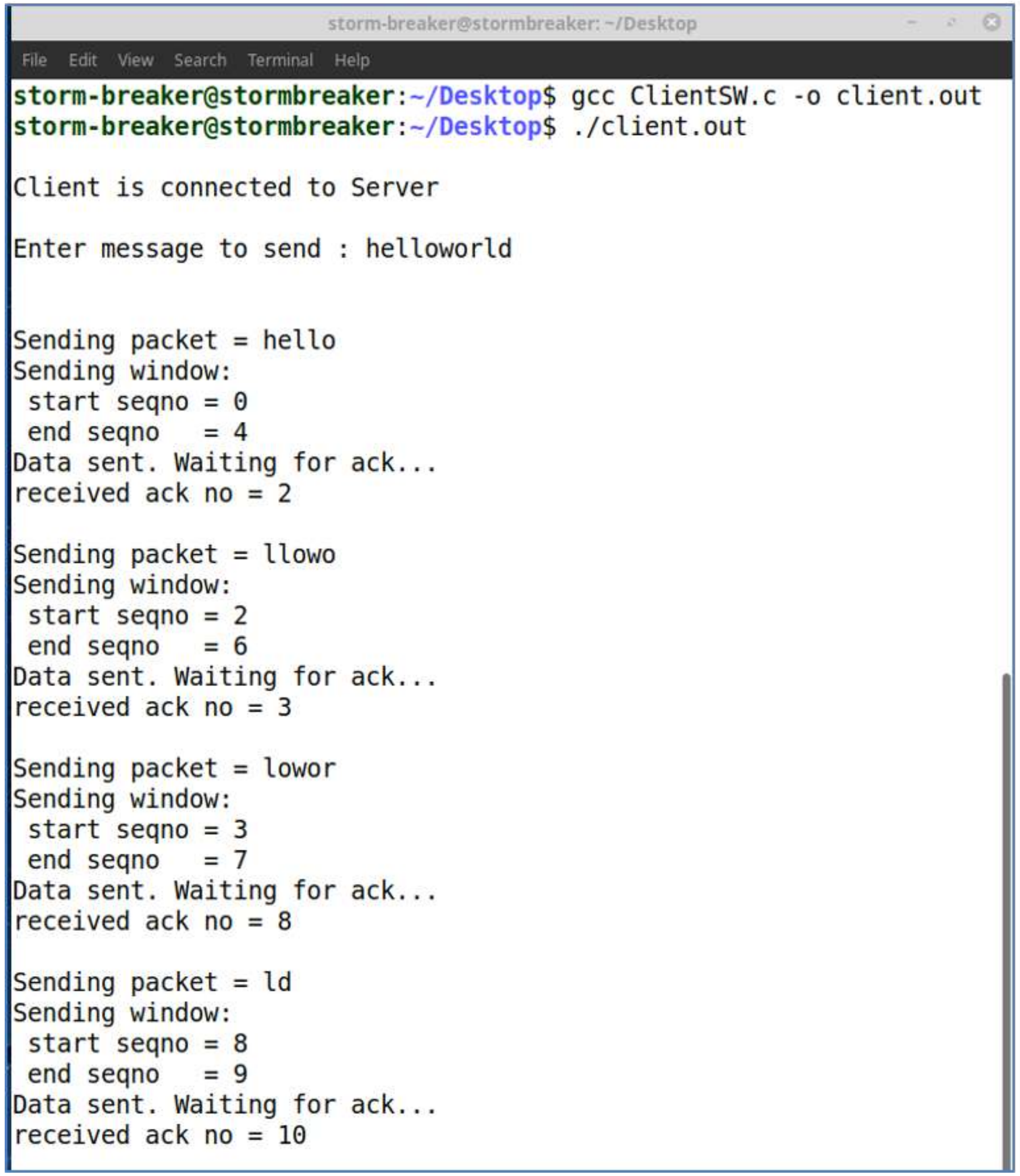
```
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ gcc ClientMC.c -o client1.out
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ ./client1.out
Waiting for message from server...
The message from multicast server is : Everyone meet me in 10 minutes.
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$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

```
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)$
```

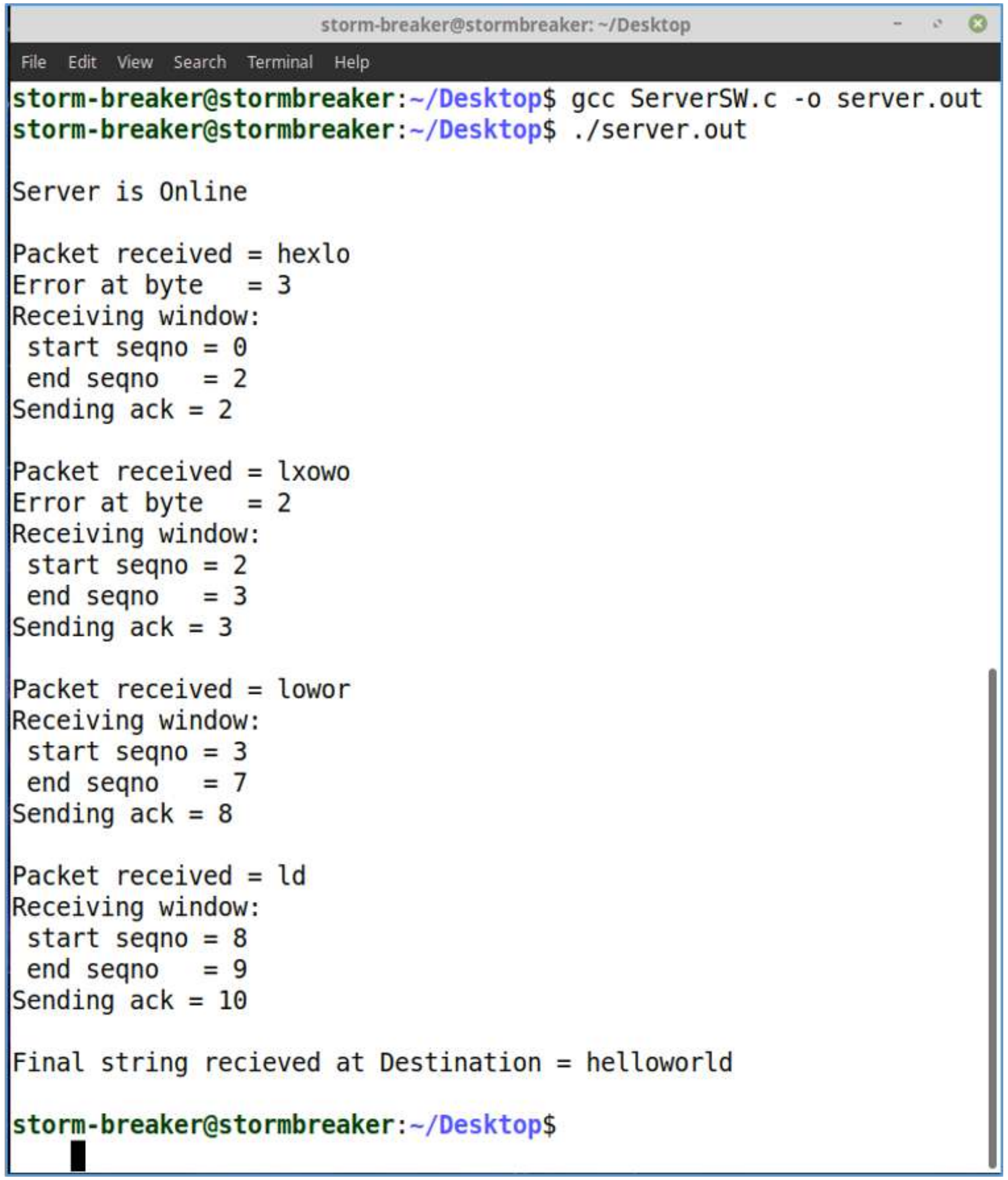
```
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ gcc ClientMC.c -o client3.out
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$ ./client3.out
Waiting for message from server...
The message from multicast server is : Everyone meet me in 10 minutes.
akshat@pop-os:~/Desktop/NP Lab/A6 (IP Multicast)$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

```c
/* Author : Akshat Agarwal

7.  to implement sliding window protocol, between two hosts(TCP Flow Control)
  - Client sends the frame
  - Server recieves the frame */

/* Server Program */

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>

#define MAX 20

int main()
{
  int sersock, sock, reuse=1;
  char frame[MAX];
  char res[MAX]; // to store all bytes that are recieved successfully
  int ack;

  sersock = socket(AF_INET, SOCK_STREAM, 0);

  struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };

  // Forcefully connecting to same port everytime
  setsockopt(sersock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));

  bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
  printf("\nServer is Online");

  listen(sersock, 5);
  sock = accept(sersock, NULL, NULL);

  int k=0; // iterator for res[MAX]
  srand(time(NULL));

  while(1)
  {
    int recvsize = 5;
    memset(frame, 0, MAX); // re-initialise frame buffer with 0

    recv(sock, frame, recvsize, 0);  // recv(socket, buffer, length, flag)

    if(strlen(frame) < recvsize)
    {
      recvsize = strlen(frame);
    }
    // at end exit frame is recieved from client
    if(strcmp(frame, "Exit") == 0) break;

    int err_idx = rand()%8; // probability of byte to get corrupted = 50%
    int j;

    if(err_idx < recvsize)
    {
      for(j=0; j<err_idx ; j++)
      {
        res[k++] = frame[j];
      }
      frame[err_idx]='x';
      printf("\n\nPacket received = %s", frame);
      printf("\nError at byte    = %d", err_idx+1);
      printf("\nReceiving window: ");
      printf("\n start seqno = %d", k-err_idx);
    }
    else
    {
      for(j=0; j<recvsize ; j++)
      {
        res[k++] = frame[j];
      }
      printf("\n\nPacket received = %s", frame);
      printf("\nReceiving window: ");
      printf("\n start seqno = %d", k-recvsize);
    }
    printf("\n end seqno    = %d", k-1);
    ack = k ;
    printf("\nSending ack = %d", ack);
    send(sock, &ack, sizeof(ack), 0) ;
  }

  res[k] = '\0';
  printf("\n\nFinal string recieved at Destination = ");
  fputs(res, stdout);

  printf("\n\n");
  close(sock); close(sersock);
}
```

```c
/* Author : Akshat Agarwal

7.  to implement sliding window protocol, between two hosts(TCP Flow Control)
  - Client sends the frame
  - Server recieves the frame */

/* Client Program */

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>

#define MAX 20

int main()
{
    int sock, ack;
    char msg[MAX], frame[MAX];

    sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };

    /*  keep trying to establish connection with server */
    while(connect(sock, (struct sockaddr *) &addr, sizeof(addr))) ;
    printf("\nClient is connected to Server\n");

    printf("\nEnter message to send : "); scanf("%s", msg);

    int i = 0;
    while(i<strlen(msg))
    {
        int sendsize = 5;
        memset(frame, 0, MAX); // re-initialise frame buffer with 0

        // strncpy(destination , source , length)
        strncpy(frame, msg+i, sendsize); //copy msg to frame
        if( sendsize > strlen(frame) )
        {
            sendsize = strlen(frame);
        }
        printf("\n\nSending packet = %s", frame);
        printf("\nSending window: ");
        printf("\n start seqno = %d", i);
        printf("\n end seqno   = %d", i+sendsize-1);

        send(sock, frame, strlen(frame), 0);
        printf("\nData sent. Waiting for ack...");

        recv(sock, &ack, sendsize, 0);
        printf("\nreceived ack no = %d ",ack);

        i = ack; // next data seq no = incoming ack no
    }
    send(sock, "Exit", strlen("Exit"), 0);
    close(sock); printf("\n\n");
}
```


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$
```

```c
/* Author : Akshat Agarwal

8. Write a program for Error Detection using CRC-CCITT(16 bits). */

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

# define MAX 30

/* crc(dividend , divisor, remainder) */
void crc(char *data, char *gen, char *rem)
{
    int i, j, k=0;
    char out[MAX]; // xored val after each step

    strcpy(out, data);

    /* Perform XOR on the msg */
    for(i=0; i<strlen(data)-strlen(gen)+1; i++)
    {
        if(out[i] == '1')
        {
            out[i] = '0' ;
            for(j=1; j<strlen(gen); j++)
            {
                out[i+j] = (out[i+j] == gen[j]) ? '0' : '1';
            }
        }
    }

    // size of output = strlen(gen)-1 = 16 bits
    int idx = strlen(out)-strlen(gen)+1; // last 16 bits of out array
    for(i=0; i<strlen(gen)-1; i++)
    {
        rem[i] = out[idx+i]; // last 16 bits of out array
    }
}

int main()
{
    int i, j;
    char dword[MAX]; // dataword
    char augWord[MAX]; // augmented dataword
    char cword[MAX]; // codeword
    char rem[MAX]; // remainder from crc
    char recv[MAX]; // received message
    char gen[MAX] = "10001000000100001\0";

    printf("\nCRC-16 Generator : x^16 + x^12 + x^5 + 1 ");
    printf("\nBinary Form      : %s", gen);

    printf("\n\nEnter Dataword   : ");
    scanf("%s", dword);

    strcpy(augWord, dword);
    for(i=0; i<strlen(gen)-1; i++)
    {
        strcat(augWord, "0");
    }
    printf("\nAugmented dataword is   : %s",augWord);

    crc(augWord, gen, rem);

    strcpy(cword, dword);
    strcat(cword, rem);
    printf("\n\nFinal data transmitted  : %s", cword);

    printf("\n\nEnter the data received : ");
    scanf("%s", recv);
    if(strlen(recv) < strlen(cword))
    {
        printf("\n Invalid input \n");
        exit(0);
    }

    crc(recv, gen, rem);

    printf("\nSyndrome = %s ", rem);
    for(i=0; i<strlen(rem); i++)
    {
        if(rem[i] == '1')
        {
            printf("\nError occured !!! Corrupted data received. \n");
            exit(0);
        }
    }
    printf("\nNo Error. Data received successfully.\n");
}

/*************** Output -1 *************************
CRC-16 Generator : x^16 + x^12 + x^5 + 1
Binary Form      : 10001000000100001

Enter Dataword    : 11110001

Augmented dataword is   : 111100010000000000000000

Final data transmitted  : 111100011111111100111110

Enter the data received : 111100011111111100111110

Syndrome = 0000000000000000
No Error. Data received successfully.
*****************************************************/

/*************** Output -2 *************************
CRC-16 Generator : x^16 + x^12 + x^5 + 1
Binary Form      : 10001000000100001

Enter Dataword    : 10101011

Augmented dataword is   : 101010110000000000000000

Final data transmitted  : 101010110000010010000001

Enter the data received : 101010110000000000000000

Syndrome = 0000010010000001
Error occured !!! Corrupted data received.
*****************************************************/
```

For Network simulation using NS2

1. Install NS2

```
sudo apt install ns2
```

2. Install NAM (Network animator)

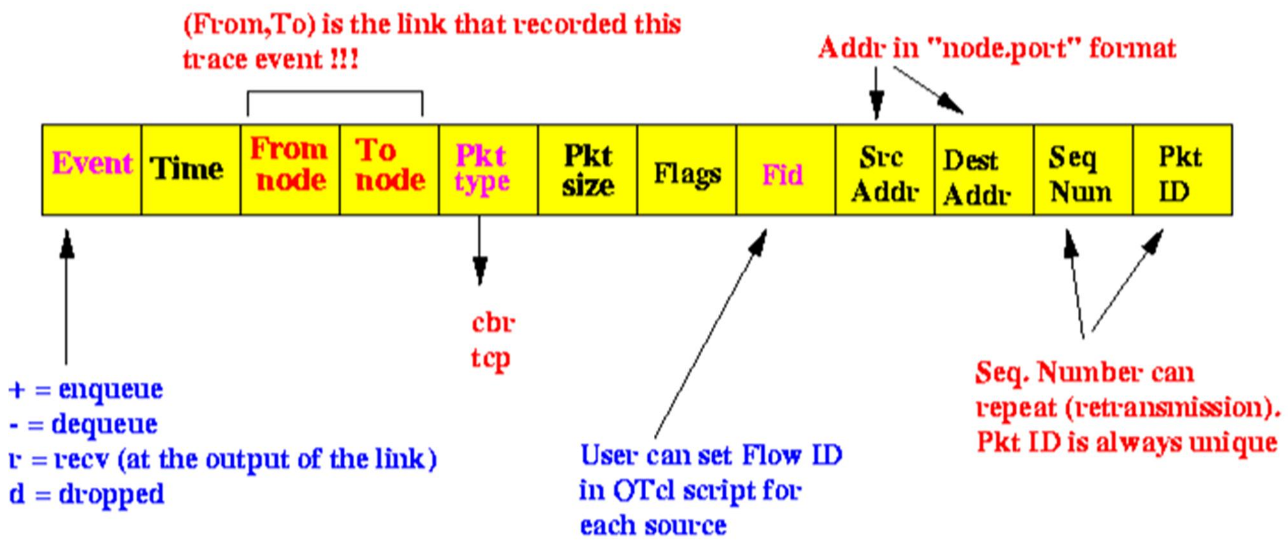
```
sudo apt install nam
```

3. Install tcl (Tool command language)

```
sudo apt install tcl
```

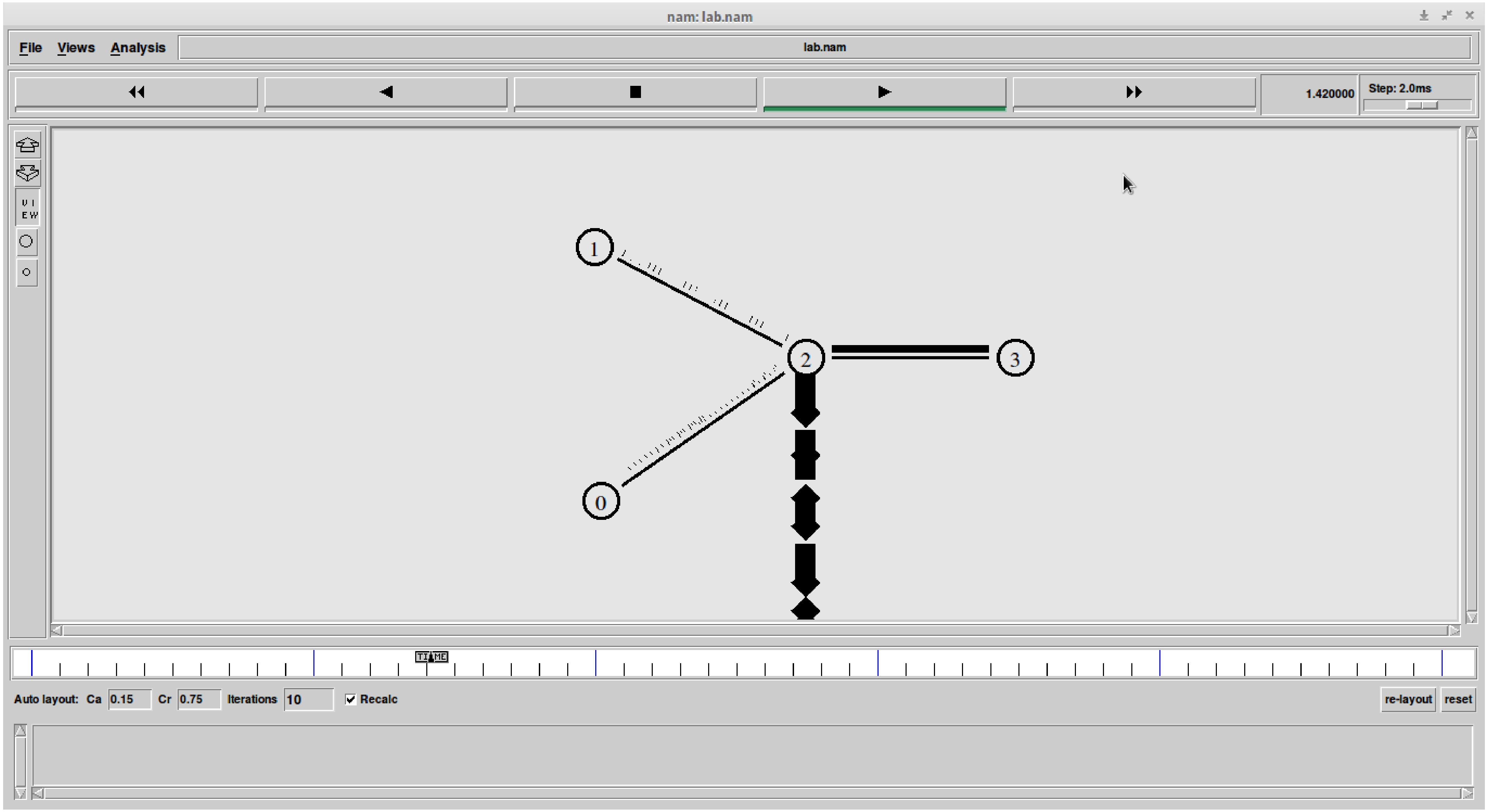
Structure of trace file

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


(From,To) is the link that recorded this trace event !!!
Addr in "node.port" format
Event
Time
From node
To node
Pkt type
Pkt size
Flags
Fid
Src Addr
Dest Addr
Seq Num
Pkt ID
cbr
tcp
+ = enqueue
- = dequeue
r = recv (at the output of the link)
d = dropped
User can set Flow ID in OTcl script for each source
Seq. Number can repeat (retransmission). Pkt ID is always unique

```
# Author : Akshat Agarwal

# 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.


# create a new simulator
set ns [new Simulator]

# open trace and NAM trace file in write mode
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf

##### Decide a topology #########
#
#   [udp][cbr]
#     [0]------
#             |      [null]
#            [2]------[3]
#             |
#     [1]------
#  [udp][cbr]
#
##################################

# create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# create duplex links between nodes
$ns duplex-link $n0 $n2 10Mb 300ms DropTail
$ns duplex-link $n1 $n2 10Mb 300ms DropTail
$ns duplex-link $n2 $n3 1Mb 300ms DropTail

# set up queue size
$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10
$ns queue-limit $n2 $n3 10

# setup udp connection for transport layer
set udp0 [new Agent/UDP]
set udp1 [new Agent/UDP]
set null3 [new Agent/Null]

$ns attach-agent $n0 $udp0
$ns attach-agent $n1 $udp1
$ns attach-agent $n3 $null3

# setup cbr(constant bit rate) over udp for application layer
set cbr0 [new Application/Traffic/CBR]
set cbr1 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr1 attach-agent $udp1

# connect source to destination
```

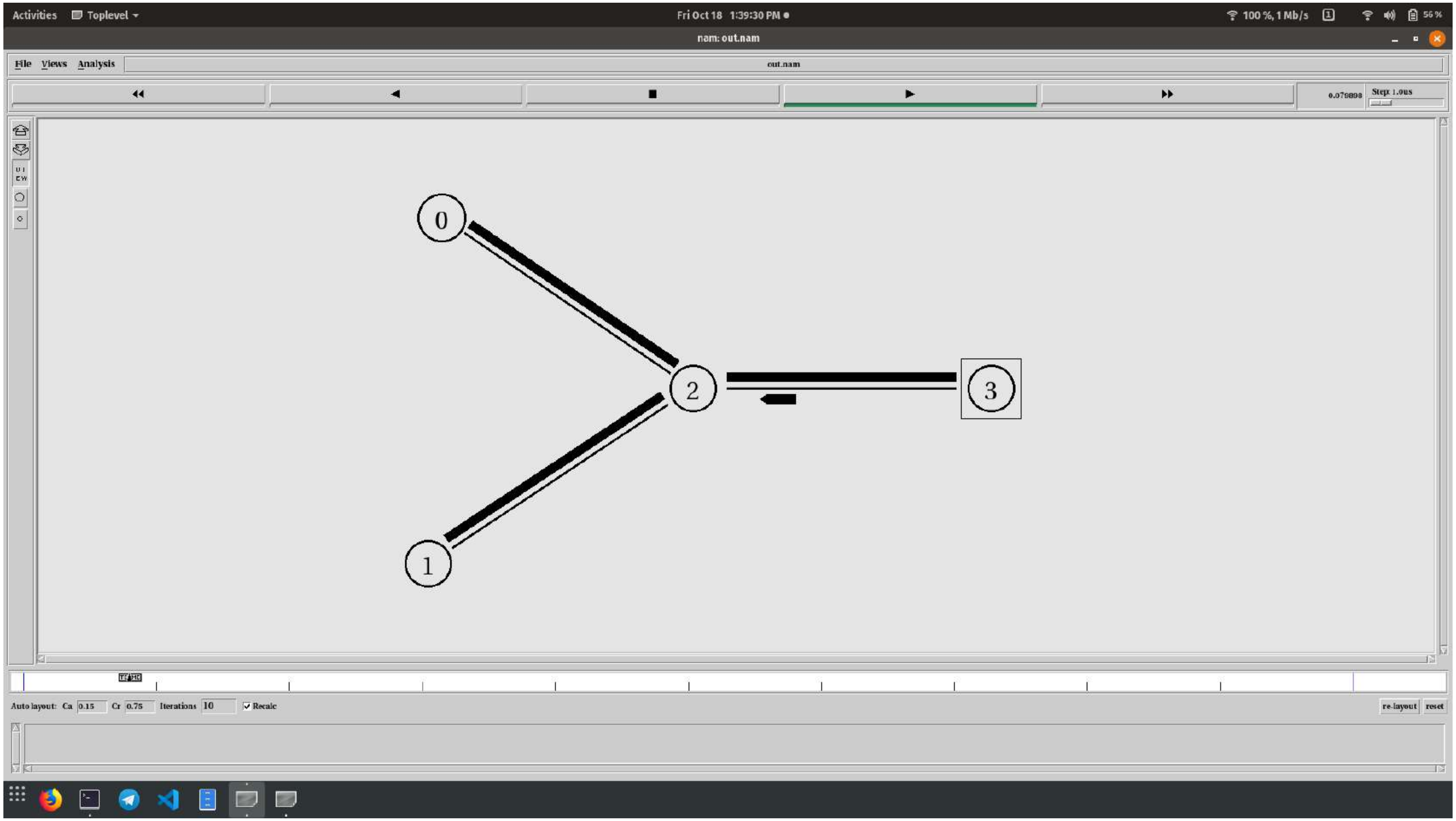
```
$ns connect $udp0 $null3
$ns connect $udp1 $null3

# set bandwidth (vary values for different output)
$cbr0 set packetSize_ 500Mb
$cbr1 set packetSize_ 500Mb
$cbr0 set interval_ 0.005
$cbr1 set interval_ 0.005

# define a finish procedure
proc finish {} {
  global ns nf tf
  $ns flush-trace
  exec nam out.nam &
  close $tf
  close $nf

  set count 0
  set tf [open out.tr r]
  while {[gets $tf line] != -1} {
    # d is event in the trace file which denotes dropped packets
    if { [string match "d*" $line] } {
      set count [expr $count + 1]
    }
  }
  puts "Number of packets dropped: $count"
  exit 0
}

# schedule events
$ns at 0.01 "$cbr0 start"
$ns at 0.01 "$cbr1 start"
$ns at 5.0 "finish"
$ns run

############# output #############

# Number of packets dropped: 700

##################################
```



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 [✓ Recalc]

```
# Author : Akshat Agarwal

# 2. Simulate the different types of Internet traffic such as FTP and TELNET over a network and analyze the throughput.


# create a new simulator
set ns [new Simulator]

# open trace and NAM trace file in write mode
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf

######## Decide a topology #######
#
#    [ftp]
#    [tcp]
#     [0]------
#              |          [sink0]
#             [2]------[3]
#              |          [sink1]
#      [1]------
#     [tcp]
#   [telnet]
#
##################################

# create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# create duplex links between nodes
$ns duplex-link $n0 $n2 2Mb 1ms DropTail
$ns duplex-link $n1 $n2 2Mb 1ms DropTail
$ns duplex-link $n2 $n3 2Mb 1ms DropTail

# set n0 and n1 as tcp source
set tcp0 [new Agent/TCP]
set tcp1 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
$ns attach-agent $n1 $tcp1

# set n3 as tcp destination for n0 and n1
set TCPS0 [new Agent/TCPSink]
set TCPS1 [new Agent/TCPSink]
$ns attach-agent $n3 $TCPS0
$ns attach-agent $n3 $TCPS1

# set ftp over tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

#set telnet over tcp1
set tel1 [new Application/Telnet]
$tel1 attach-agent $tcp1

```

```
$tel1 set packetSize_ 500Mb
$tel1 set interval_ 0.001

# connect source to destination
$ns connect $tcp0 $TCPS0
$ns connect $tcp1 $TCPS1

proc finish { } {
  global ns nf tf
  $ns flush-trace
  exec nam out.nam &
  close $tf
  close $nf

  # because time difference between start and finish is 2
  set time 2
  set fCount 0
  set tCount 0
  set tf [open out.tr r]
  while {[gets $tf line] != -1} {
    if { [string match "*tcp*0.0*3.0*" $line] } {
      set fCount [expr $fCount + 1]
    }
    if { [string match "*tcp*1.0*3.1*" $line] } {
      set tCount [expr $tCount + 1]
    }
  }
  puts "Throughput of FTP: [expr $fCount/$time]"
  puts "Throughput of TELNET: [expr $tCount/$time]"
  exit 0
}

# schedule events
$ns at 0.01 "$ftp0 start"
$ns at 0.01 "$tel1 start"
$ns at 2.01 "finish"
$ns run

############### output #################

# No of FTP packets: 767
# No of TELNET packets: 750

########################################
```

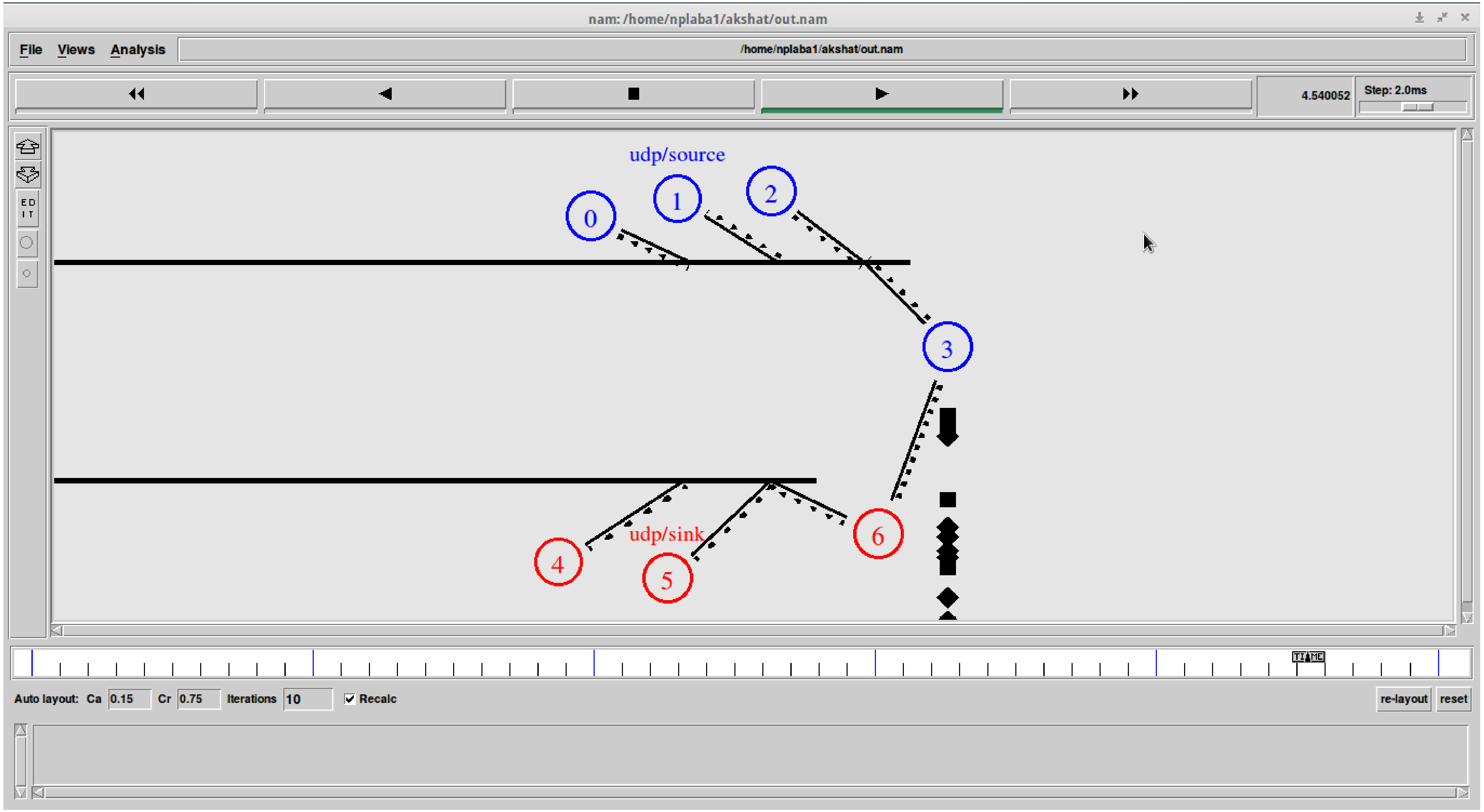


0.079898 Step 1.0us

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 [✓ Recalc]

```
# Author : Akshat Agarwal

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


# Declare a new Simulator
set ns [new Simulator]

# Open nam and trace file in write mode
set tf [open out.tr w]
set nf [open out.nam w]
$ns trace-all $tf
$ns namtrace-all $nf

# Take value of error rate and data rate from std input
puts "Enter error rate (<1) : "
gets stdin erate

puts "Enter data rate (in Mbps) : "
gets stdin drate

############## Select a topology #####################
#
#            [udp1]                duplex-link
#.     [n0]    [n1]  [n2]    [n3]-------------
#        |      |      |      |              |
#        |      |      |      |              |
#    -------------------------------lan7     |
#                                            |
#    -------------------------------lan8     |
#         |       |        |                 |
#         |       |        |                 |
#       [n4]    [n5]     [n6]----------------
#              [null5]
#
#####################################################

# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

# set label and color (OPTIONAL)
$n1 label "udp/source"
$n5 label "udp/null"
$n0 color "blue"
$n1 color "blue"
$n2 color "blue"
$n3 color "blue"
$n4 color "red"
$n5 color "red"
$n6 color "red"

# Create two lans
$ns make-lan "$n0 $n1 $n2 $n3" 10Mb 10ms LL Queue/DropTail Mac/802_3
```

```
$ns make-lan "$n4 $n5 $n6" 10Mb 10ms LL Queue/DropTail Mac/802_3

# Setup Links
$ns duplex-link $n3 $n6 10Mb 10ms DropTail

# Declare the transport layer protocols
set udp1 [new Agent/UDP]
set null5 [new Agent/Null]
$ns attach-agent $n1 $udp1
$ns attach-agent $n5 $null5

# Declare the application layer protocol
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

# Connect the source and destination
$ns connect $udp1 $null5

# Create error model
set err [new ErrorModel]
$ns lossmodel $err $n3 $n6
$err set rate_ $erate

# Define the data rate
$cbr1 set packetSize_ $drate.Mb
$cbr1 set interval_ 0.001

# Define procedure
proc finish { } {
  global ns nf tf
  $ns flush-trace
  exec nam out.nam &
  close $nf
  close $tf

  set count 0
  set tr [open out.tr r]
  while {[gets $tr line] != -1} {
    # 8 denotes LAN at destination side and 5 denotes destination node
    if {[string match "* 8 5 *" $line]} {
      set count [expr $count+1]
    }
  }
  set thr [expr $count/5]
  puts "Throughput : $thr"
  exit 0
}

$ns at 0.1 "$cbr1 start"
$ns at 5.1 "finish"
$ns run

######################### output-1 #####################

# Enter error rate (<1) :
# 0.4
# Enter data rate (in Mbps) :
# 1000
# Throughput : 593

########################################################
```

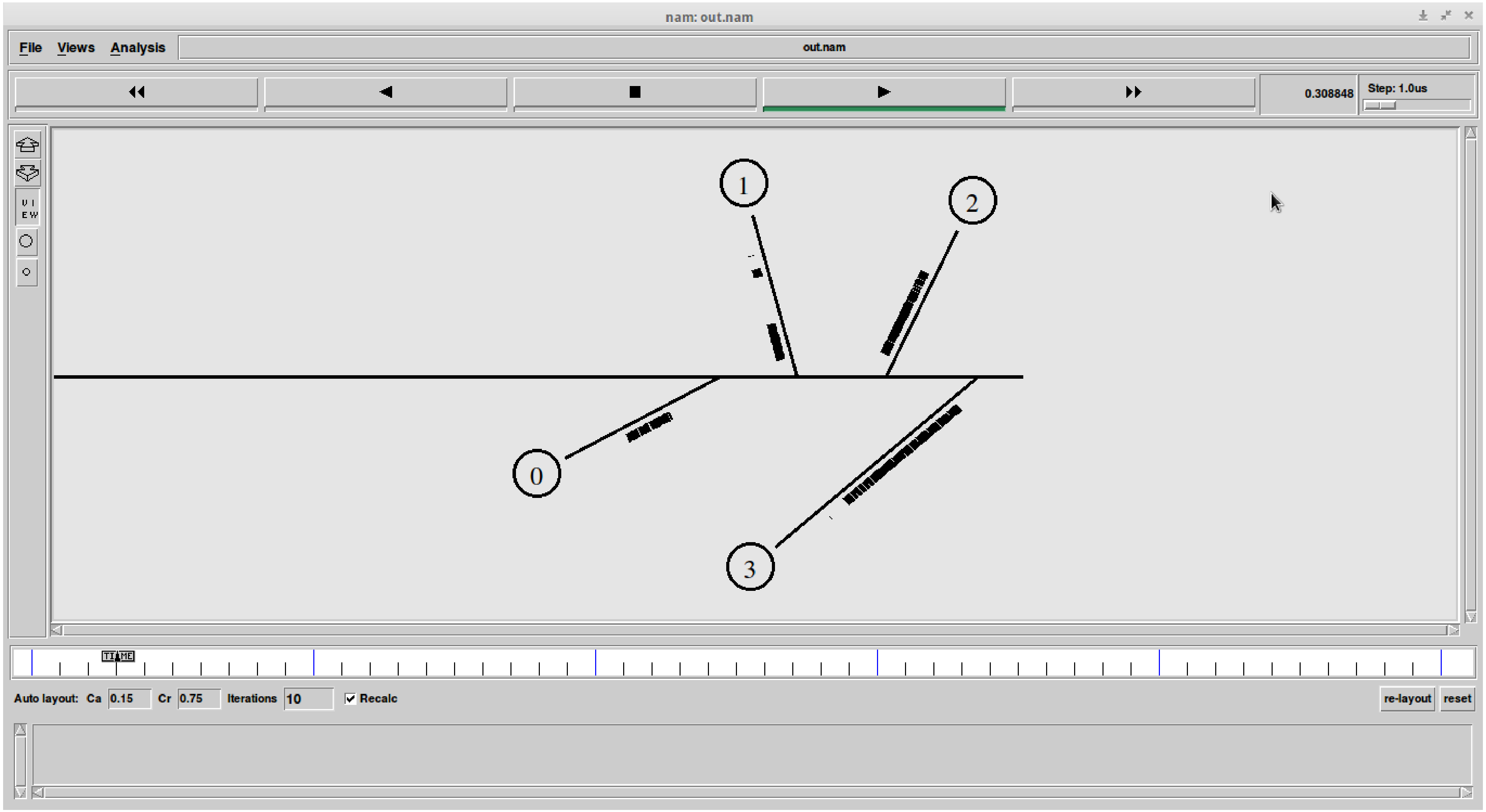


nam: /home/nplaba1/akshat/out.nam

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 [✓ Recalc]

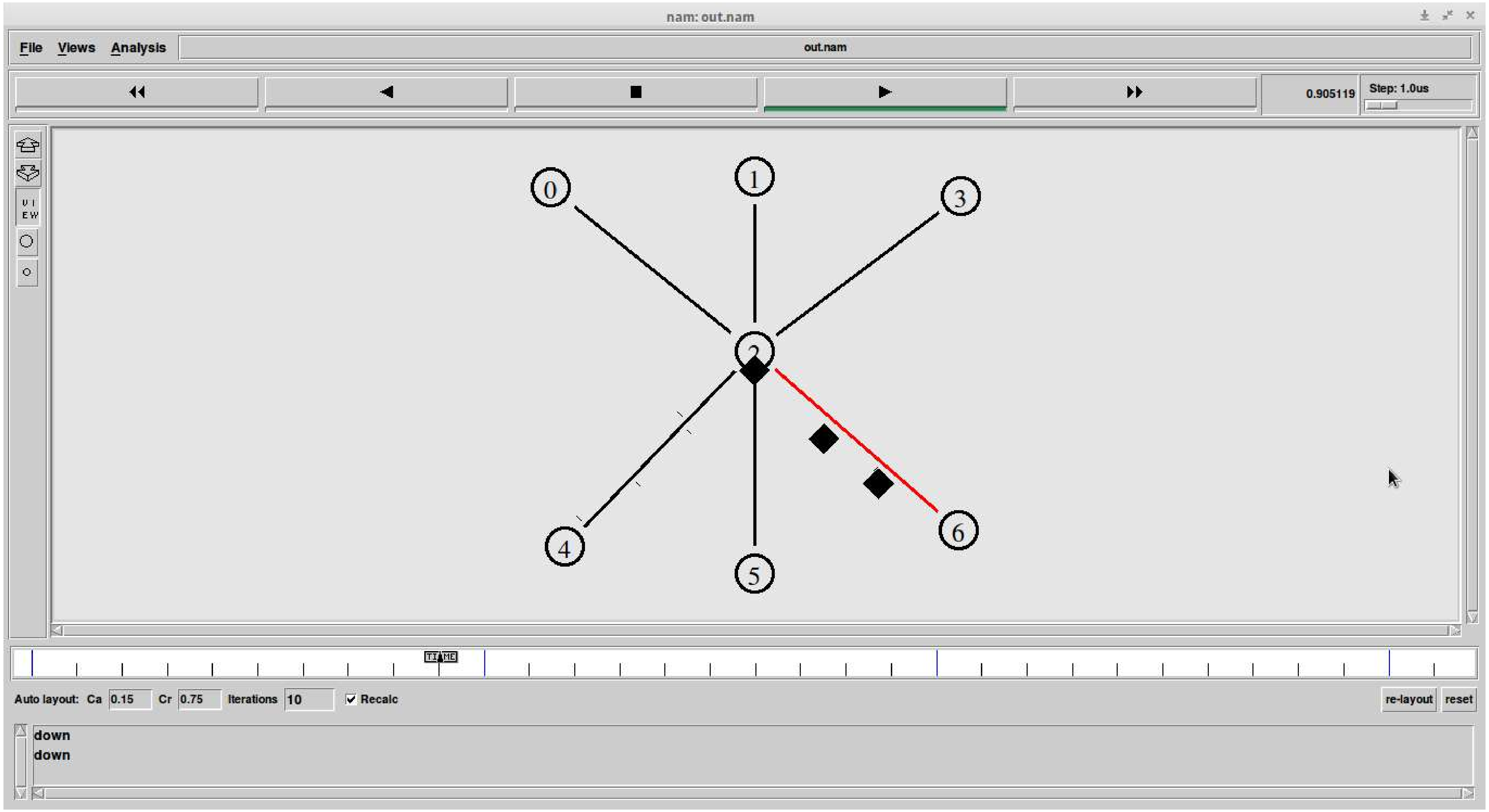
```
# Author : Akshat Agarwal

# 4. Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and determine the collision across different nodes.


# Declare a new Simulator
set ns [new Simulator]

# Open the trace and nam file in write mode
set tf [open out.tr w]
set nf [open out.nam w]
$ns trace-all $tf
$ns namtrace-all $nf

# Decide the topology: [tcp(0->2)], [udp(2->1)], [tcp(1->3)]
#
#      [tcp0]   [tcp1][null1]
#       [n0]        [n1]
#        |           |
#        |           |
#   ------------------------------ lan4
#            |             |
#            |             |
#           [n3]          [n2]
#         [sink3]     [udp2][sink2]

# Create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Create lan and setup the link
$ns make-lan -trace on "$n0 $n1 $n2 $n3" 100Mb 10ms LL Queue/DropTail Mac/802_3

# Declare the required transport layer Protocols
set tcp0 [new Agent/TCP]
set tcp1 [new Agent/TCP]
set udp2 [new Agent/UDP]
set null1 [new Agent/Null]
set sink2 [new Agent/TCPSink]
set sink3 [new Agent/TCPSink]

# Attach these Protocols to their respective nodes
$ns attach-agent $n0 $tcp0
$ns attach-agent $n1 $tcp1
$ns attach-agent $n2 $udp2
$ns attach-agent $n1 $null1
$ns attach-agent $n2 $sink2
$ns attach-agent $n3 $sink3

# Declare Application layer protocols and attach them with their transport layer protocols
set ftp0 [new Application/FTP]
set ftp1 [new Application/FTP]
set cbr2 [new Application/Traffic/CBR]
$ftp0 attach-agent $tcp0
$ftp1 attach-agent $tcp1
$cbr2 attach-agent $udp2
```

```

# connect source to destination
$ns connect $tcp0 $sink2
$ns connect $udp2 $null1
$ns connect $tcp1 $sink3

# set the interval
$ftp0 set interval_ 0.001
$ftp1 set interval_ 0.001
$cbr2 set interval_ 0.01

# define finish procedure
proc finish {} {
  global ns nf tf
  $ns flush-trace
  exec nam out.nam &
  close $tf
  close $nf

  set count 0
  set tr [open out.tr r]
  while {[gets $tr line] !=-1 } {
    if { [string match "c*" $line] } {
      set count [expr $count + 1]
    }
  }
  puts "No of packets collided: $count"
  exit 0
}

# schedule the events
$ns at 0.1 "$cbr2 start"
$ns at 0.1 "$ftp0 start"
$ns at 0.1 "$ftp1 start"
$ns at 5.0 "finish"
$ns run

############ output ############

# No of packets collided: 242

################################
```



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 [✓ Recalc]

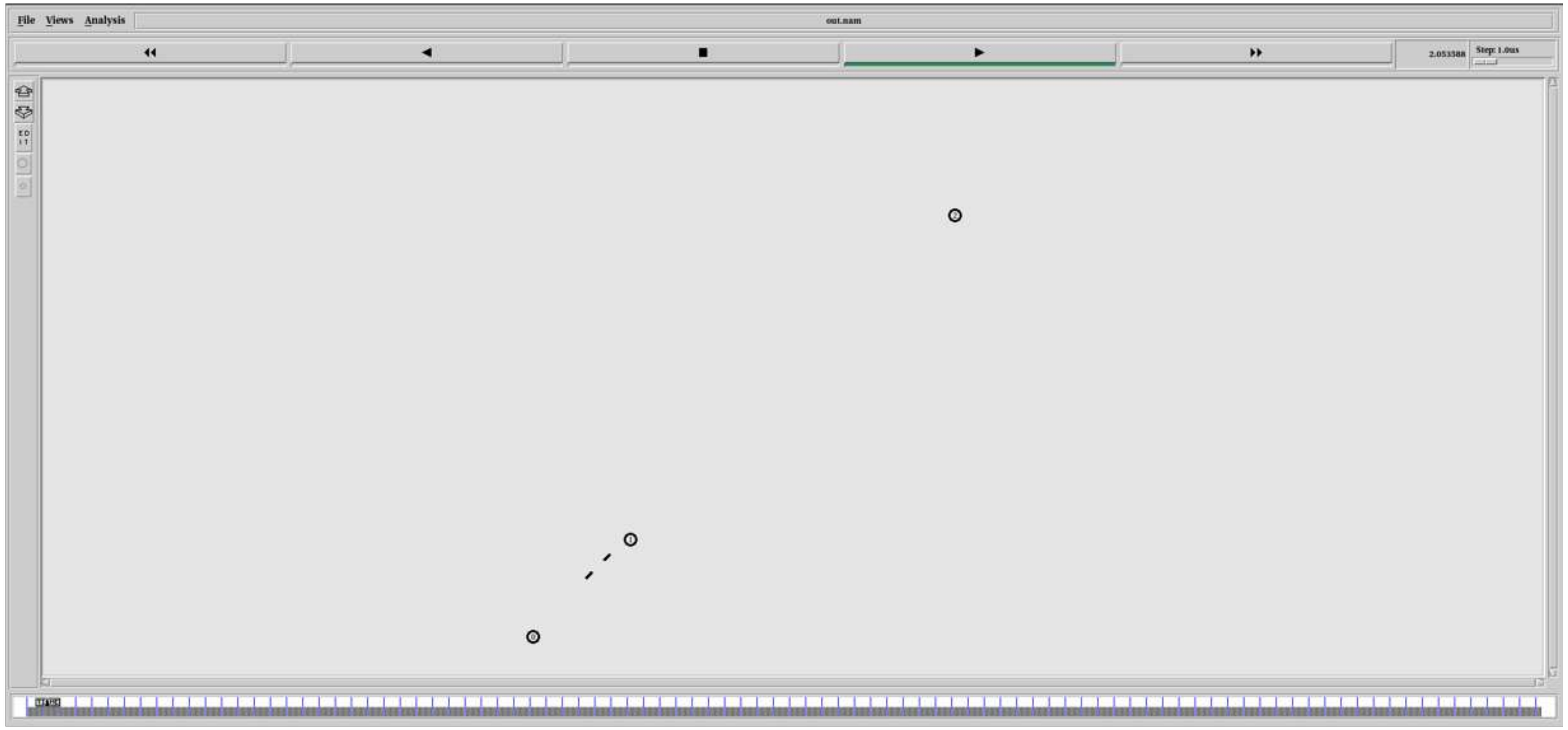
```
# Author : Akshat Agarwal

# 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.


# Declare new Simulator
set ns [new Simulator]

# Open trace and nam file in write mode
set tf [open out.tr w]
set nf [open out.nam w]
$ns trace-all $tf
$ns namtrace-all $nf

# Decide the topology
#
#   [s0][ping]      [ping]      [ping]
#      [n0]           [n1]        [n3]
#         '.           |         .'
#           '.         |       .'
#             '.       |     .'
#               '.     |   .'
#                 '.   | .'
#                    [n2]
#                  .' | '.
#                .'   |   '.
#              .'     |     '.
#            .'       |       '.
#          .'         |         '.
#        .'           |           '.
#      [n4]          [n5]          [n6]
#    [ping][d0]   [s1][ping]   [ping][d1]

# Create the nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

# set up links
$ns duplex-link $n0 $n2 100Mb 300ms DropTail
$ns duplex-link $n5 $n2 100Mb 300ms DropTail
$ns duplex-link $n1 $n2 1Mb 300ms DropTail
$ns duplex-link $n3 $n2 1Mb 300ms DropTail
$ns duplex-link $n2 $n4 1Mb 300ms DropTail
$ns duplex-link $n2 $n6 1Mb 300ms DropTail

# set up queue size
$ns queue-limit $n0 $n2 5
$ns queue-limit $n5 $n2 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n2 $n6 2

# Declare the agents/protocols
set ping0 [new Agent/Ping]
set ping4 [new Agent/Ping]
set ping5 [new Agent/Ping]
set ping6 [new Agent/Ping]

# Attach the ping with the respective nodes
$ns attach-agent $n0 $ping0
$ns attach-agent $n4 $ping4
$ns attach-agent $n5 $ping5
$ns attach-agent $n6 $ping6

# Connect the ping from source to destination
$ns connect $ping0 $ping4
$ns connect $ping5 $ping6

# Write proc for ping agent
Agent/Ping instproc recv {from rtt} {
  $self instvar node_
  puts "The node [$node_ id] recieved $from with round trip time $rtt"
}

# Write the proc function
proc finish { } {
  global ns nf tf
  $ns flush-trace
  exec nam out.nam &
  close $nf
  close $tf

  set count 0
  set tr [open out.tr r]
  while {[gets $tr line]!=-1} {
    if {[string match "d*" $line]} {
      set count [expr $count + 1]
    }
  }
  puts "No. of packet dropped : $count"
  exit 0
}

$ns rtmodel-at 0.9 down $n2 $n6
$ns rtmodel-at 1.9 up $n2 $n6

# schedule events
for {set i 0.1} {$i<2} {set i [expr $i+0.1]} {
    $ns at $i "$ping0 send"
    $ns at $i "$ping5 send"
}

$ns at 5.0 "finish"
$ns run

####################### output #####################
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 3 recieved 4 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 3 recieved 4 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 3 recieved 4 with round trip time 1201.0
# The node 0 recieved 2 with round trip time 1201.0
# The node 3 recieved 4 with round trip time 1201.0
# No. of packet dropped : 5
####################################################
```

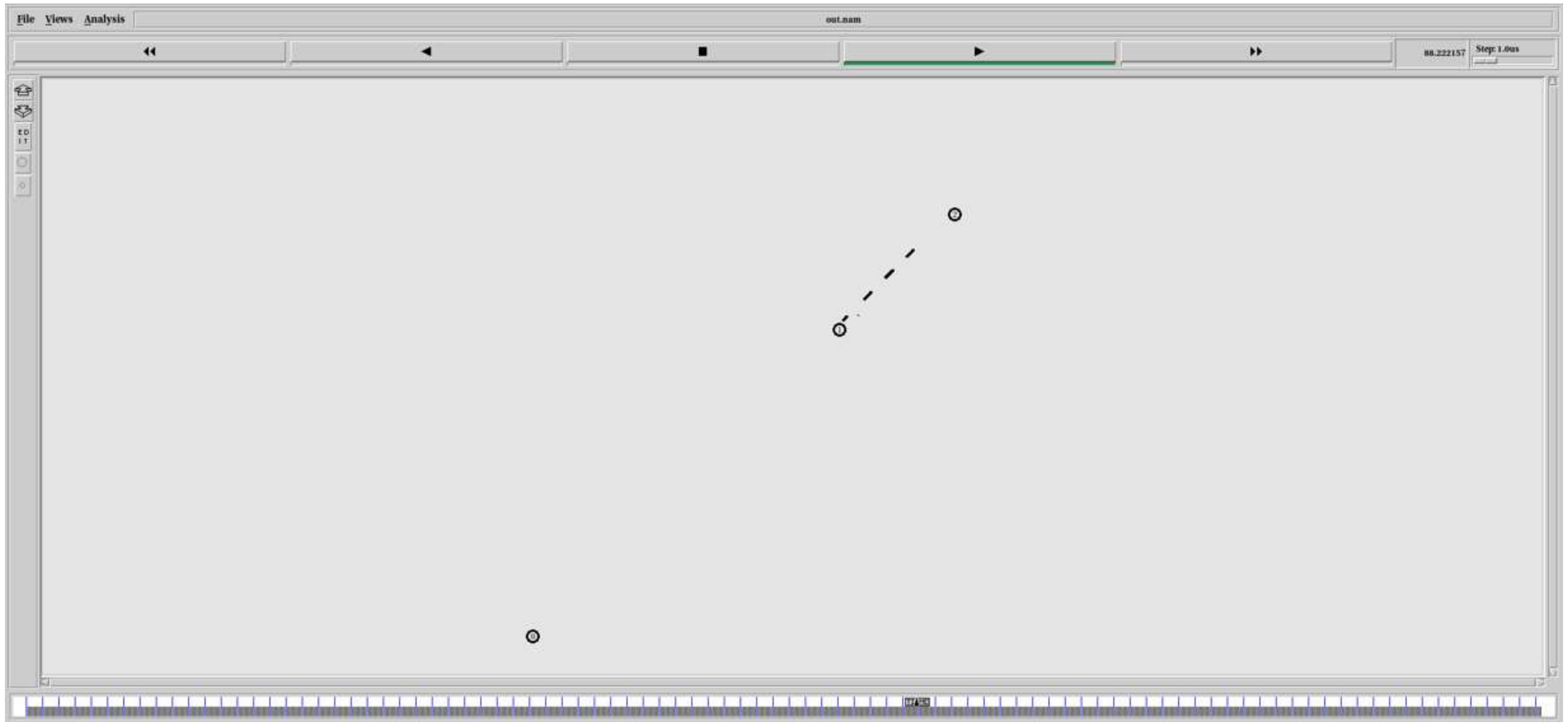


Auto layout: Ca 0.15 Cr 0.75 Iterations 10 [✓ Recalc]

```
down
down
```

```
# Author : Akshat Agarwal

# 6. A Simple ESS with transmitting nodes in Wireless LAN
# 7. A simple ad-hoc network with transmitting nodes


# Declare new Simulator
set ns [new Simulator]

# Open the trace file in write mode
set tf [open out.tr w]
$ns trace-all $tf

# Set name-trace for wireless network
set nf [open out.nam w]
$ns namtrace-all-wireless $nf 500 500

# Set new topography
set topo [new Topography]
$topo load_flatgrid 500 500

# Configure for a wireless node.
$ns node-config -adhocRouting DSDV \
-llType LL \
-macType Mac/802_11 \
-ifqType Queue/DropTail \
-ifqLen 50 \
-phyType Phy/WirelessPhy \
-channelType Channel/WirelessChannel \
-propType Propagation/TwoRayGround \
-antType Antenna/OmniAntenna \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF

# Create a god object
create-god 3

################### Decide the topology ###############
#   500
#   |
#   |
#   |
#   400                         [sink2]
#   |                            [n2]
#   |                           .'
#   |                         .'
#   |                       .'
#   |                     .'
#   |                   .'
#   |                 .'
#   100            [n1]
#   |          .' [sink1]
#   |        .'   [tcp1]
#   10   [n0]     [ftp1]
#   |  [tcp0]
#   |  [ftp0]
#   |
#   |____10______100_______________400_________500

# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

# Locate the nodes on load_flatgrid
$n0 set X_ 10
$n0 set Y_ 10
$n0 set Z_ 0

$n1 set X_ 100
$n1 set Y_ 100
$n1 set Z_ 0

$n2 set X_ 400
$n2 set Y_ 400
$n2 set Z_ 0

# initial state
$ns at 0.0 "$n0 setdest 10 10 15"
$ns at 0.0 "$n1 setdest 100 100 15"
$ns at 0.0 "$n2 setdest 400 400 15"

# Declare and attach transport layer protocol
set tcp0 [new Agent/TCP]
set tcp1 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
$ns attach-agent $n1 $tcp1

set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns attach-agent $n2 $sink2

# Declare and attach appliction layer protocol
set ftp0 [new Application/FTP]
set ftp1 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp1 attach-agent $tcp1

# connect source to destination
$ns connect $tcp0 $sink1
$ns connect $tcp1 $sink2


proc finish { } {
    global ns nf tf
    $ns flush-trace
    exec nam out.nam &
    close $tf

    set ctr1 0
    set ctr2 0
    set tf [open out.tr r]

    while {[gets $tf line] != -1} {
        # r->received, _1_ -> destination node
        if {[string match "r*_1_*AGT*" $line]} {
            set ctr1 [expr $ctr1 + 1]
        }
        if {[string match "r*_2_*AGT*" $line]} {
            set ctr2 [expr $ctr2 + 1]
        }
    }
    puts "\nThroughput from n0 to n1: $ctr1"
    puts "Throughput from n1 to n2: $ctr2"
    exit 0
}

# schedule events

# move n1 near to node n2 at 50s and come back near to node n0 at 100s
$ns at 50 "$n1 setdest 300 300 15"
$ns at 100 "$n1 setdest 100 100 15"

# start ftp traffic
$ns at 1 "$ftp0 start"
$ns at 1 "$ftp1 start"
$ns at 150 "finish"
$ns run

######################## output ###########################

# num_nodes is set 3
# INITIALIZE THE LIST xListHead
# channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
# highestAntennaZ_ = 1.5,  distCST_ = 550.0
# SORTING LISTS ...DONE!
#
# Throughput from n0 to n1: 8438
# Throughput from n1 to n2: 3000

############################################################

```


File
Views
Analysis
out.nam
2.053588
Step: 1.0us
0
1
2


File
Views
Analysis
out.nam
88.222157
Step: 1.0us
0
1
2