LAB ASSIGNMENT-3

CSN-361
Computer Networks Laboratory

Submitted by - Prateek Mali
Enrollment no. - 17114059 (CSE)

Problem Statements

1. Write a socket program in C to determine class, Network and Host ID of an IPv4 address.

2. Write a C program to demonstrate File Transfer using UDP.

3. Write a TCL code for network simulator NS2 to demonstrate the star topology among a set of computer nodes. Given N nodes, one node will be assigned as the central node and the other nodes will be connected to it to form the star. You have to set up a TCP connection between k pairs of nodes and demonstrate the packet transfer between them using Network Animator (NAM). Use File Transfer protocol (FTP) for the same. Each link should have

different color of packets to differentiate the packets transferred between each pair of nodes. The program should take the number of nodes (N) as input followed by k pairs of nodes.

4. Write a TCL code for network simulator NS2 to demonstrate the ring topology among a set of computer nodes. Given N nodes, each node will be connected to two other nodes in the form of a ring. You have to set up a TCP connection between k pairs of nodes and demonstrate packet transfer between them using Network Animator (NAM). Use File Transfer protocol (FTP) for the same. Each link should have different color of packets to differentiate the packets transferred between each pair of nodes. The program should take the number of nodes (N) as input followed by k pairs of nodes.

5. Write a TCL code for network simulator NS2 to demonstrate the bus topology among a set of computer nodes. Given N nodes, each node will be connected to a common link. You have to set up a TCP connection between k pairs of nodes and demonstrate packet transfer between them using Network Animator (NAM). Use File Transfer protocol (FTP) for the same. Each link should have different color of packets to differentiate the packets transferred between each pair of nodes. The program should take the number of nodes (N) as input followed by k pairs of nodes.

1

Implementations details

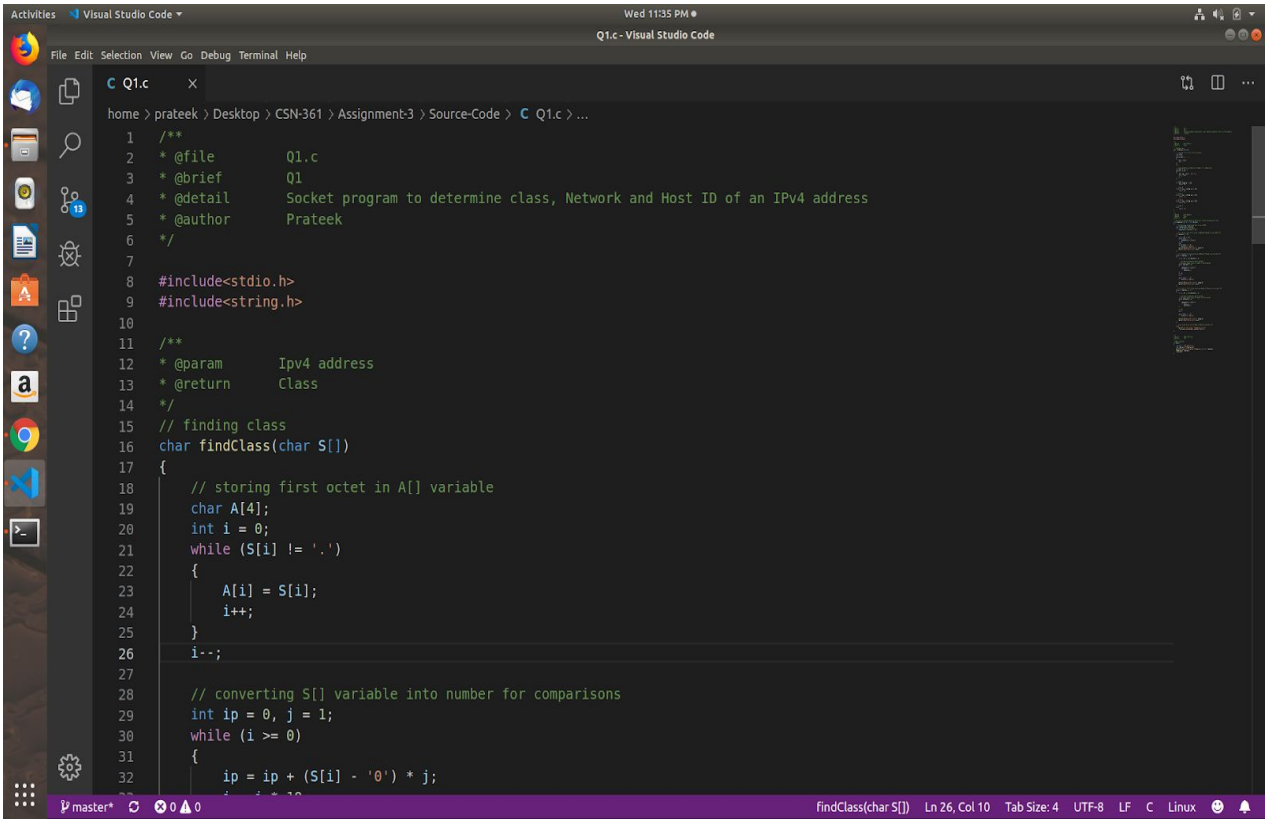
1. For determining the class we checked first octet of IP address. As we know, for class A first octet will range from 1 – 126, for class B first octet will range from 128 – 191, for class C first octet will range from 192-223, for class D first octet will range from 224 – 239, for class E first octet will range from 240 – 255.

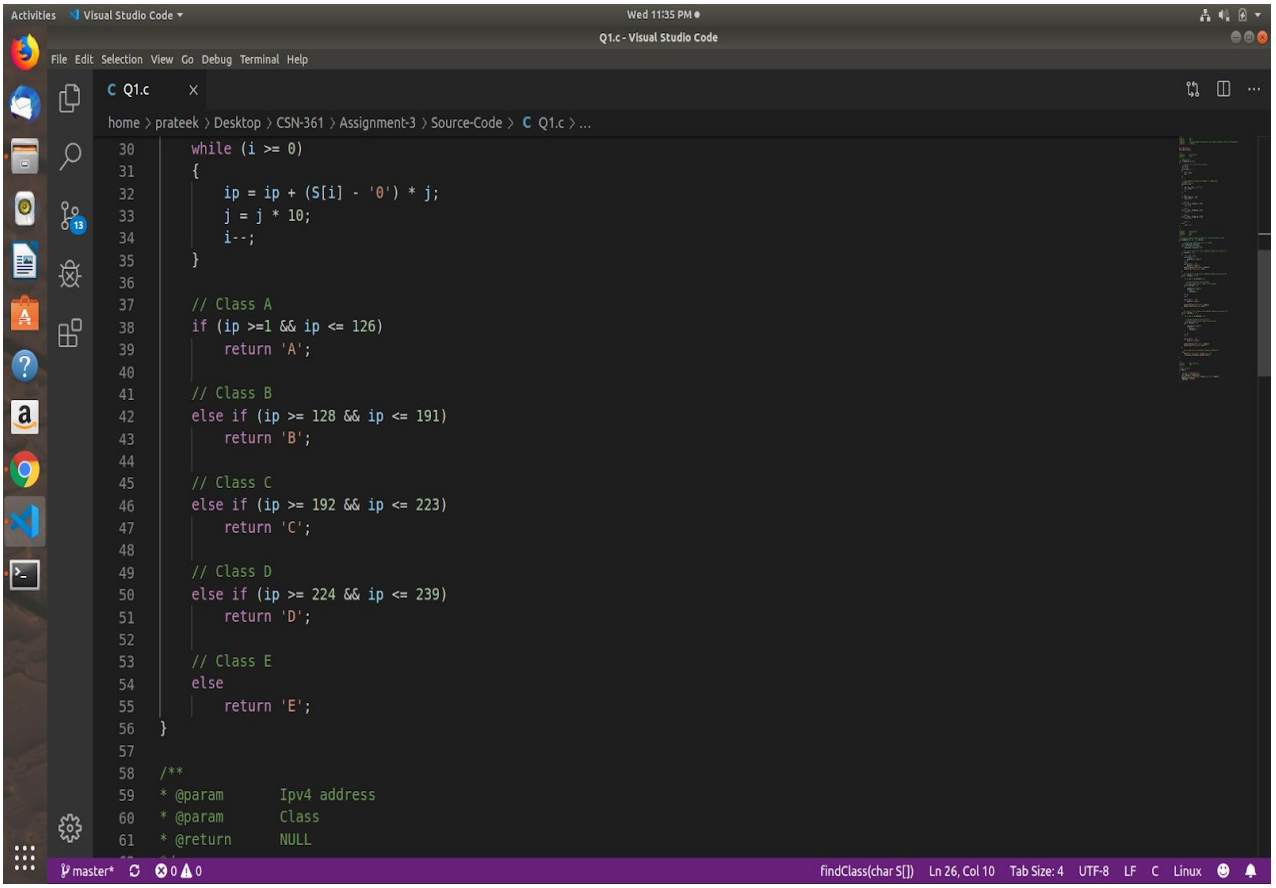
2. For determining the Network and Host ID: We know that Subnet Mask for Class A is 8, for Class B is 16 and for Class C is 24 whereas Class D and E is not divided into Network and Host ID.

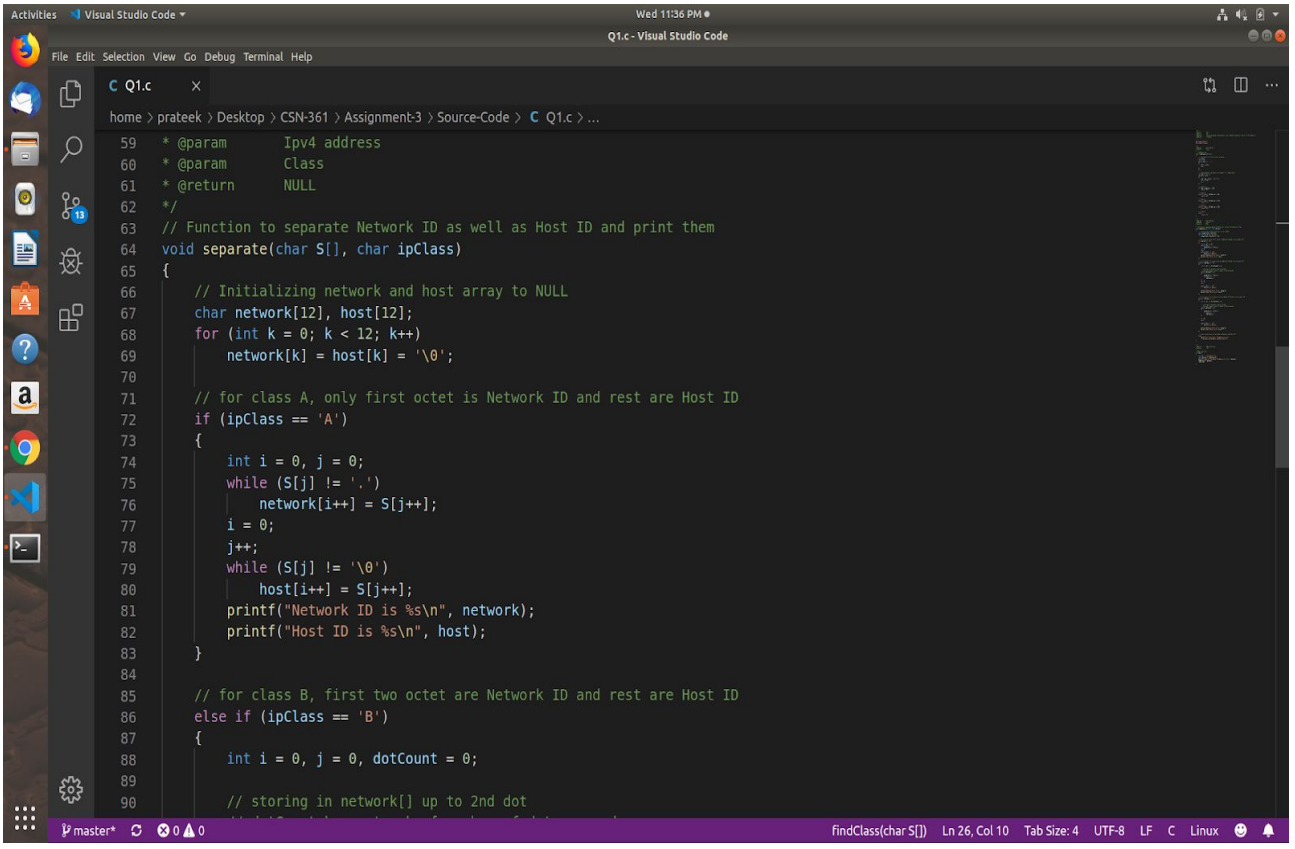
Data Structures used

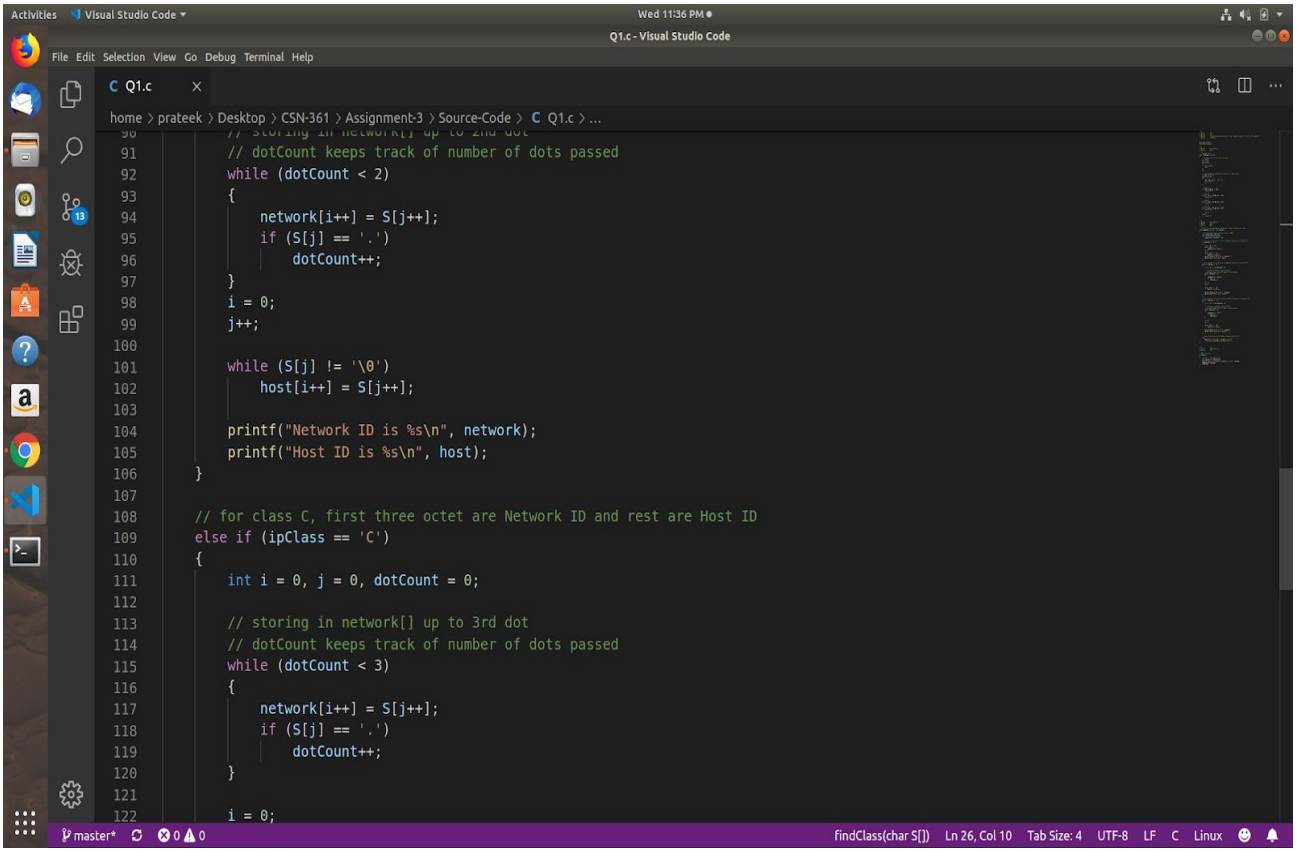
1. Array has been used as the major data structure.

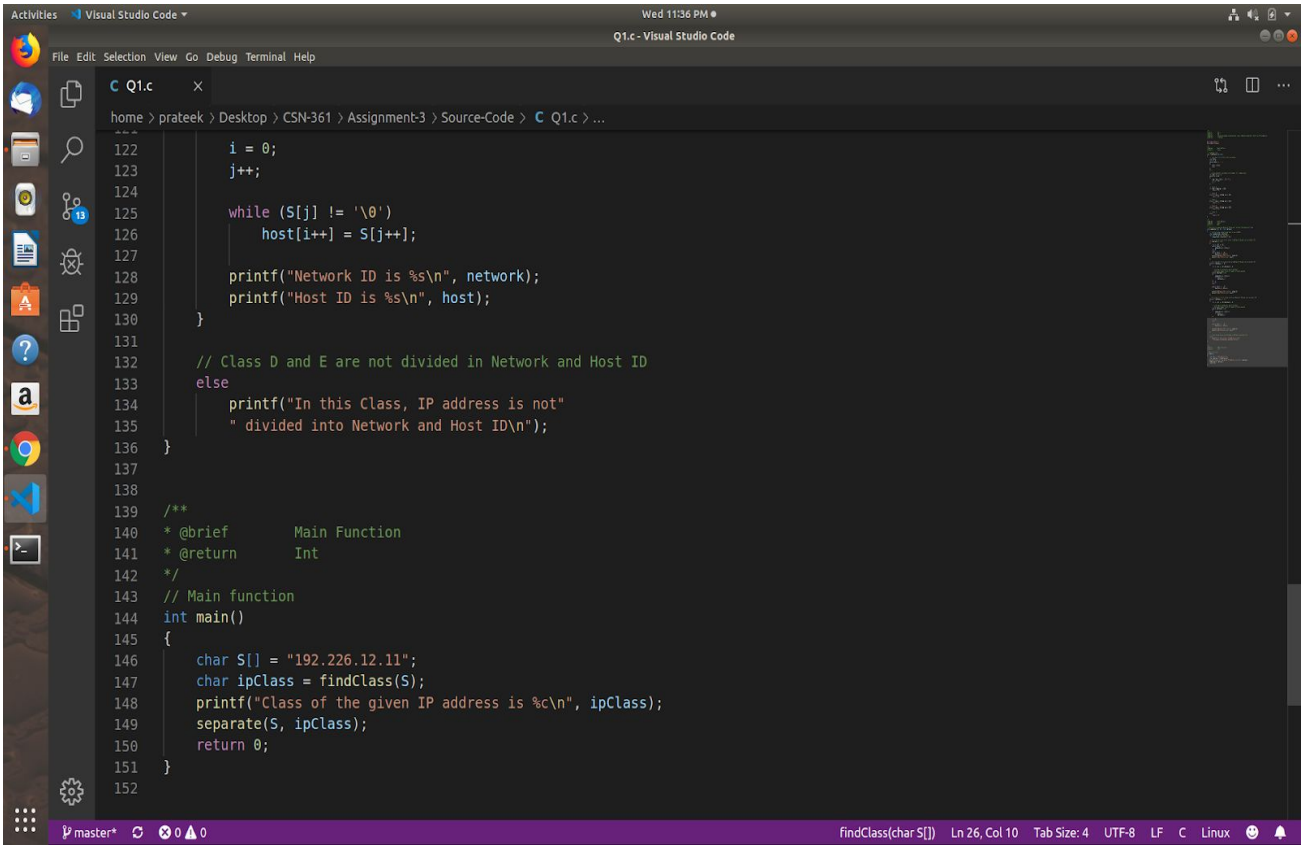
Code Snippets



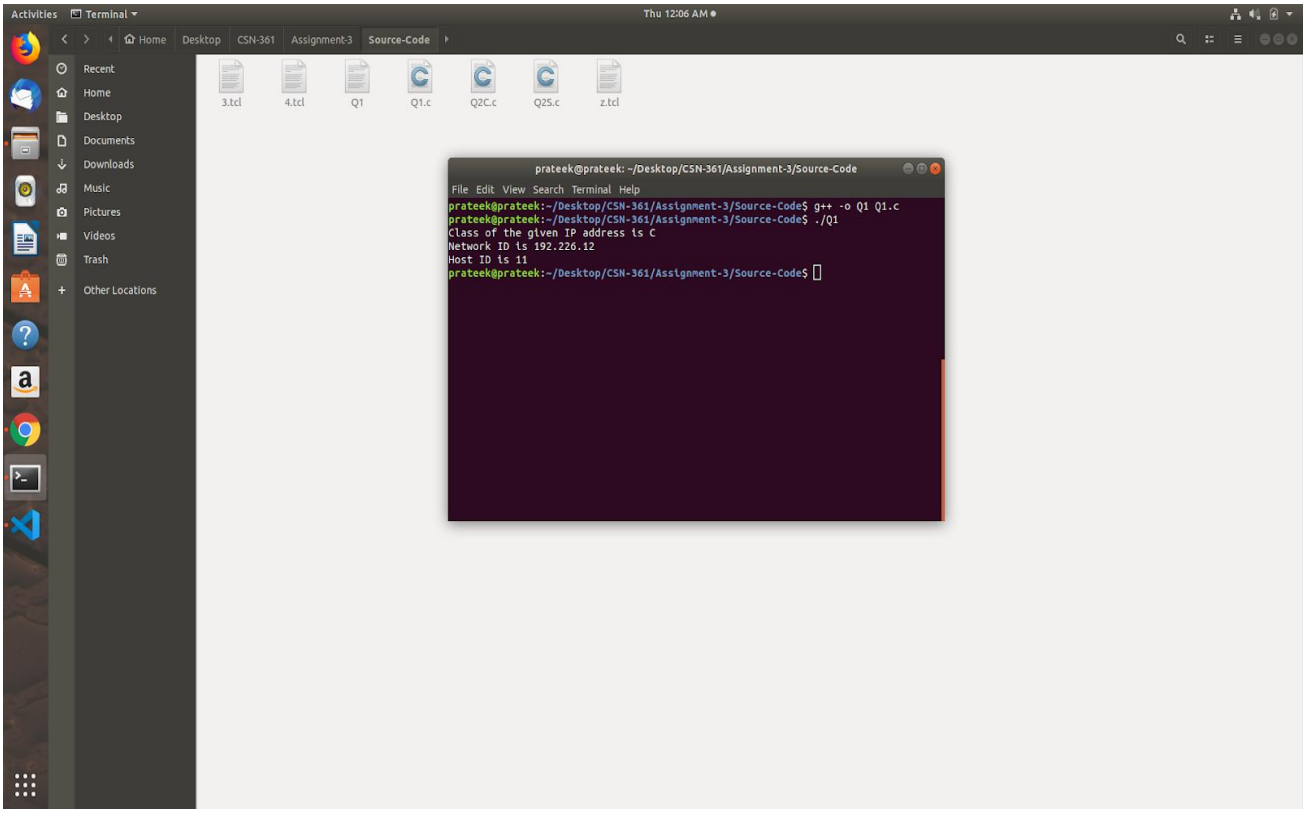



Activities
Visual Studio Code
Wed 11:36 PM
Q1.c - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
Q1.c
home › prateek › Desktop › CSN-361 › Assignment-3 › Source-Code › Q1.c › ...
* @param Ipv4 address
* @param Class
* @return NULL
*/
// Function to separate Network ID as well as Host ID and print them
void separate(char S[], char ipClass)
{
// Initializing network and host array to NULL
char network[12], host[12];
for (int k = 0; k < 12; k++)
network[k] = host[k] = '\0';
// for class A, only first octet is Network ID and rest are Host ID
if (ipClass == 'A')
{
int i = 0, j = 0;
while (S[j] != '.')
network[i++] = S[j++];
i = 0;
j++;
while (S[j] != '\0')
host[i++] = S[j++];
printf("Network ID is %s\n", network);
printf("Host ID is %s\n", host);
}
// for class B, first two octet are Network ID and rest are Host ID
else if (ipClass == 'B')
{
int i = 0, j = 0, dotCount = 0;
// storing in network[] up to 2nd dot
master*
0 0
findClass(char S[])
Ln 26, Col 10
Tab Size: 4
UTF-8
LF
C
Linux


Activities
Visual Studio Code
Wed 11:36 PM
Q1.c - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
Q1.c
home › prateek › Desktop › CSN-361 › Assignment-3 › Source-Code › Q1.c › ...
// dotCount keeps track of number of dots passed
while (dotCount < 2)
{
network[i++] = S[j++];
if (S[j] == '.')
dotCount++;
}
i = 0;
j++;
while (S[j] != '\0')
host[i++] = S[j++];
printf("Network ID is %s\n", network);
printf("Host ID is %s\n", host);
}
// for class C, first three octet are Network ID and rest are Host ID
else if (ipClass == 'C')
{
int i = 0, j = 0, dotCount = 0;
// storing in network[] up to 3rd dot
// dotCount keeps track of number of dots passed
while (dotCount < 3)
{
network[i++] = S[j++];
if (S[j] == '.')
dotCount++;
}
i = 0;
master*
0 0
findClass(char S[])
Ln 26, Col 10
Tab Size: 4
UTF-8
LF
C
Linux



Snapshot of running code



2

Implementations details (Algorithm)

1. The server starts and waits for filename.

2. The client sends a filename.

3. The server receives filename. If file is present, server starts reading file and continues to send a buffer filled with file contents encrypted until file-end is reached.

4. End is marked by EOF.

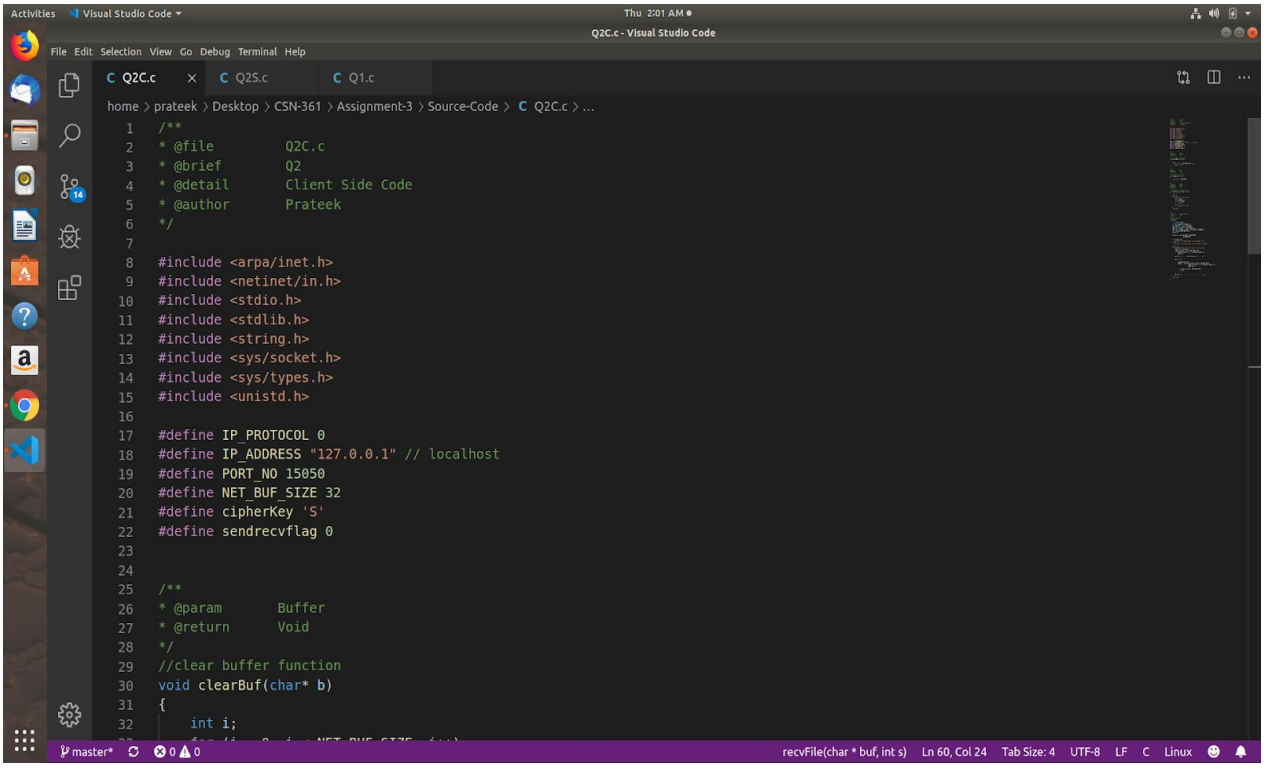
5. File is received as buffers until EOF is received. Then it is decrypted.

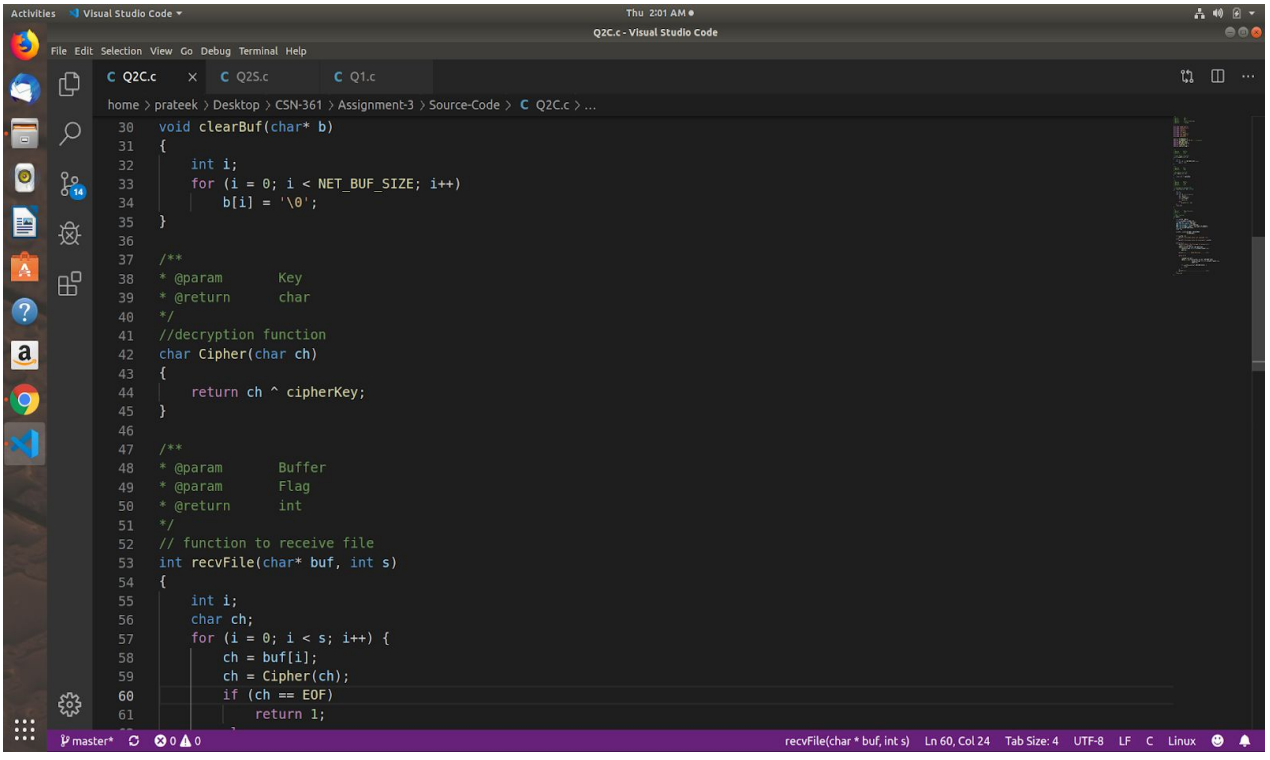
6. If Not present, a file not found is sent.

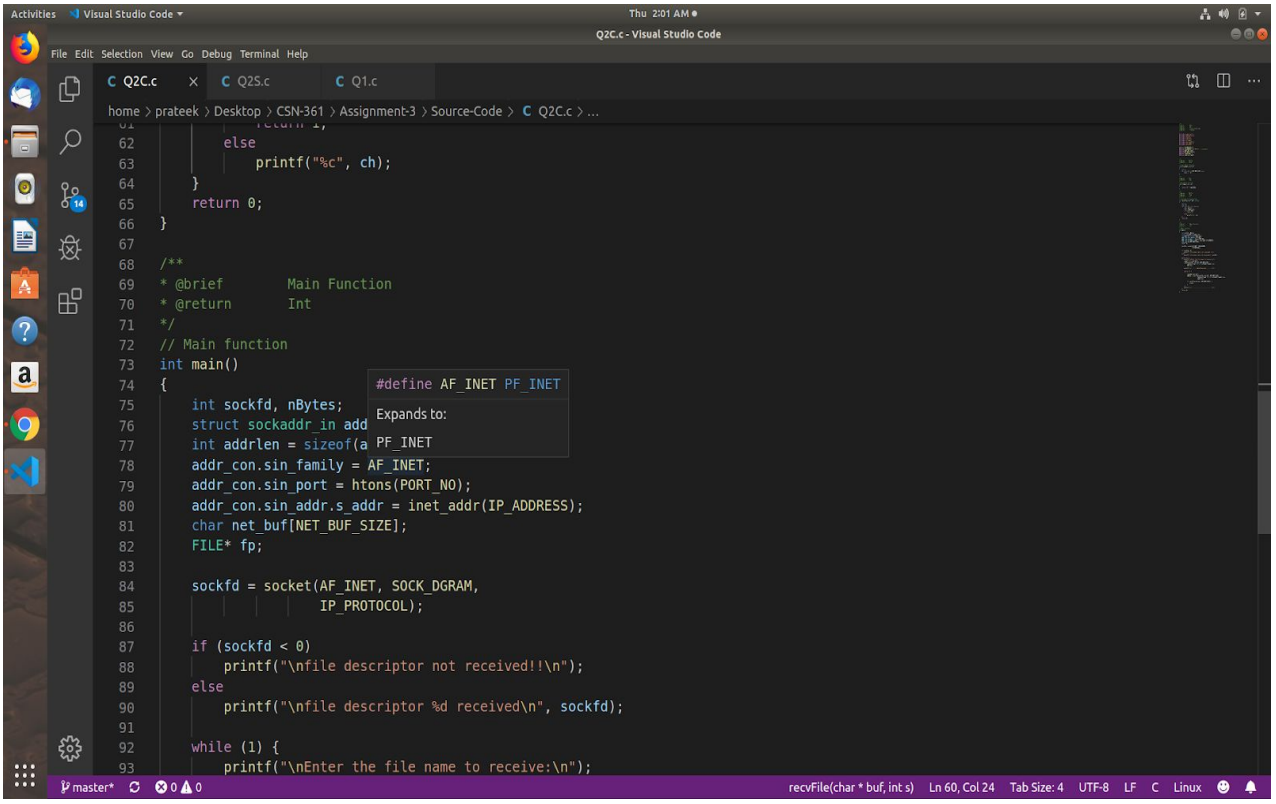
Data Structures used

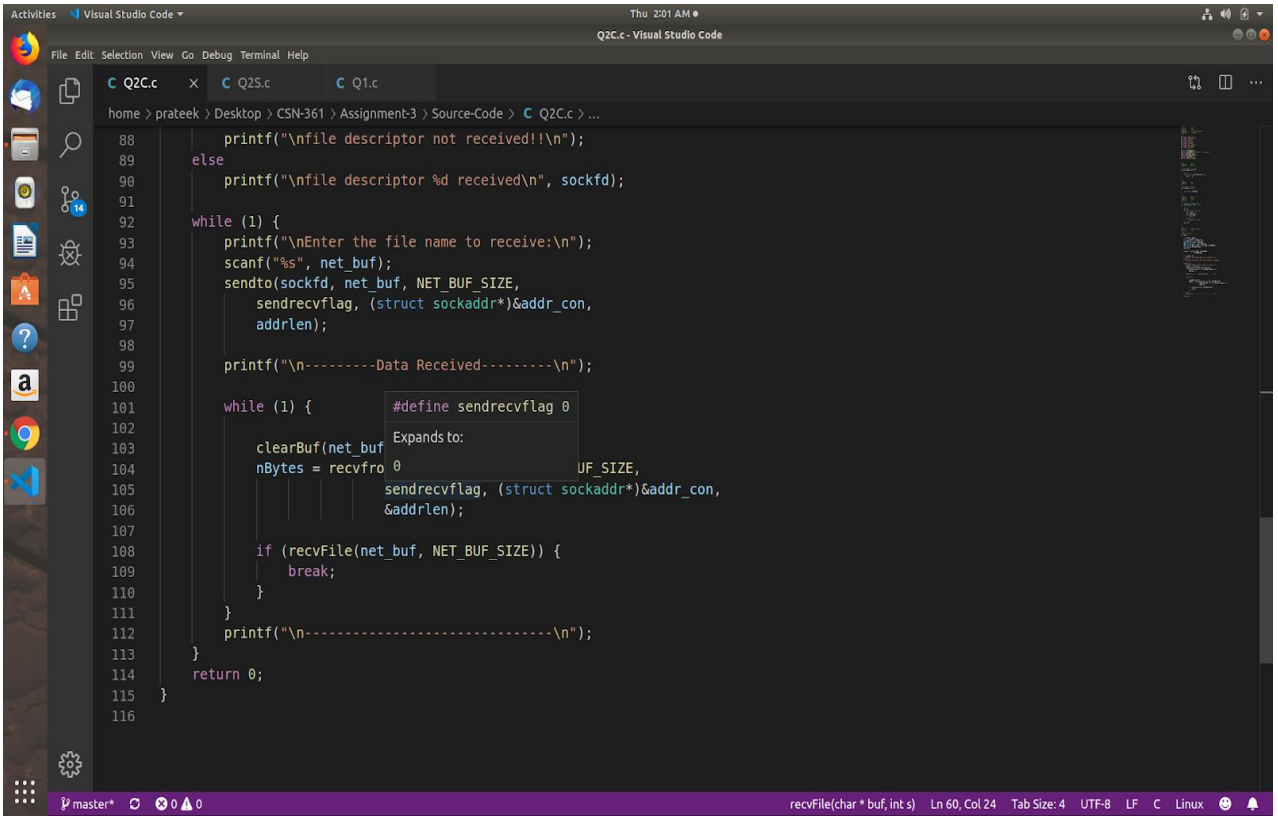
1. Buffer has been used as the major data structure.

Client-Side Code Snippets

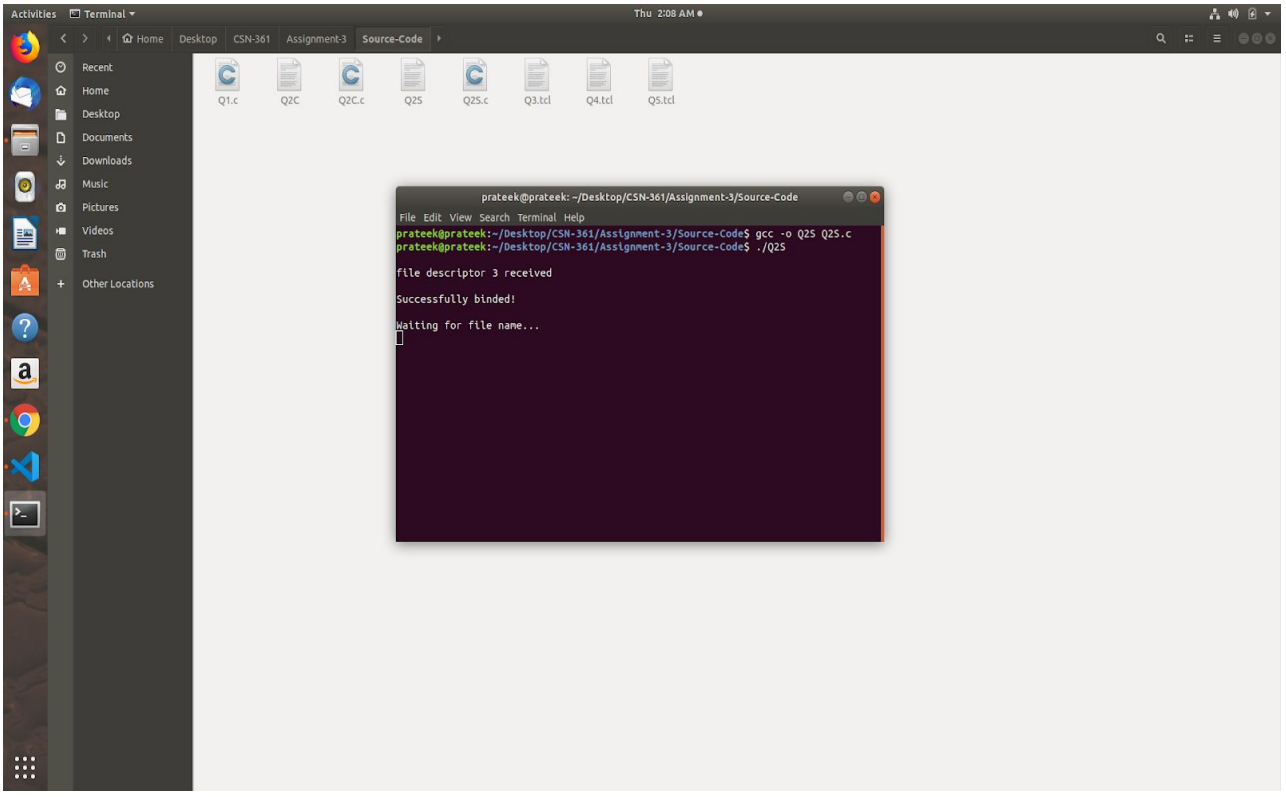

Activities
Visual Studio Code
Thu 2:01 AM
Q2C.c - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
Q2C.c
Q2S.c
Q1.c
home › prateek › Desktop › CSN-361 › Assignment-3 › Source-Code › Q2C.c › ...
/**
* @file Q2C.c
* @brief Q2
* @detail Client Side Code
* @author Prateek
*/
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#define IP_PROTOCOL 0
#define IP_ADDRESS "127.0.0.1" // localhost
#define PORT_NO 15050
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0
/**
* @param Buffer
* @return Void
*/
//clear buffer function
void clearBuf(char* b)
{
int i;
master*
recvFile(char * buf, int s)
Ln 60, Col 24
Tab Size: 4
UTF-8
LF
C
Linux

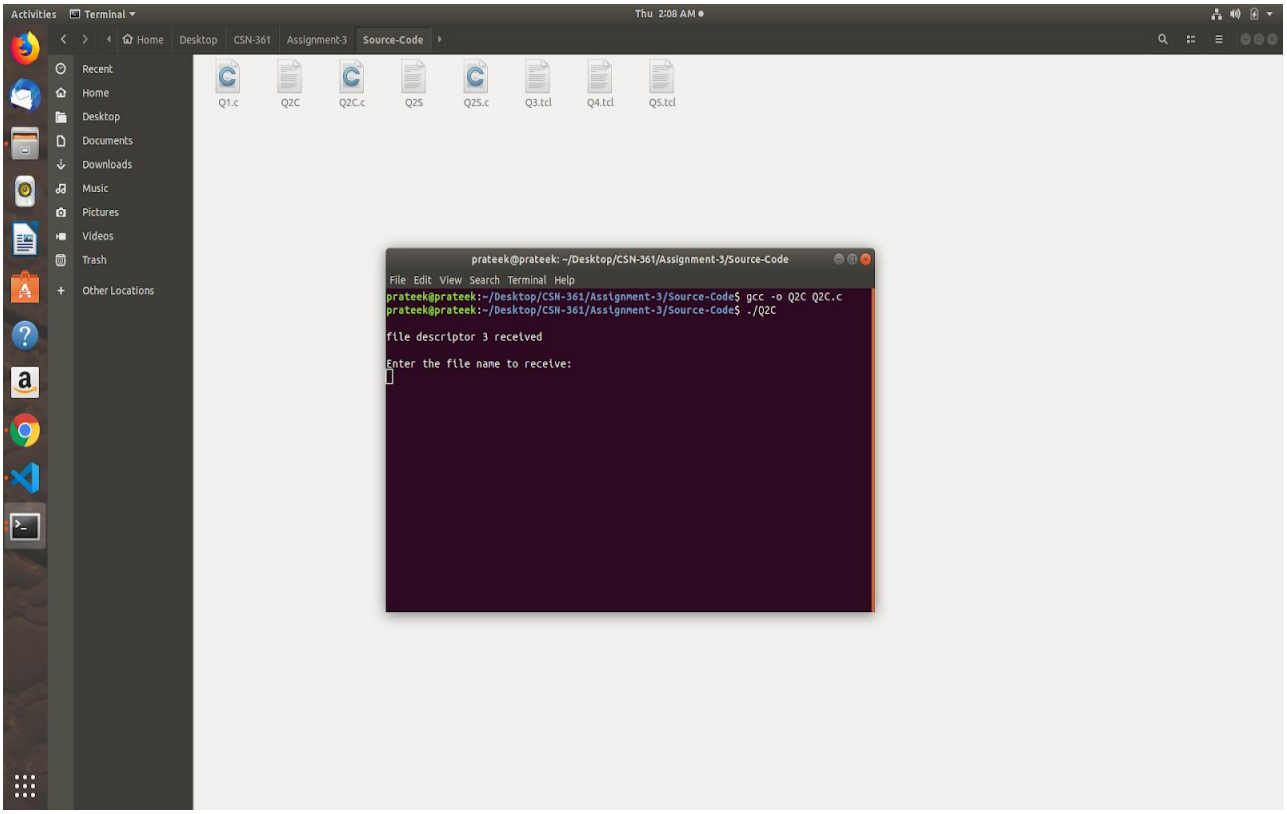

Activities
Visual Studio Code
Thu 2:01 AM
Q2C.c - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
Q2C.c
Q2S.c
Q1.c
home › prateek › Desktop › CSN-361 › Assignment-3 › Source-Code › Q2C.c › ...
void clearBuf(char* b)
{
int i;
for (i = 0; i < NET_BUF_SIZE; i++)
b[i] = '\0';
}
/**
* @param Key
* @return char
*/
//decryption function
char Cipher(char ch)
{
return ch ^ cipherKey;
}
/**
* @param Buffer
* @param Flag
* @return int
*/
// function to receive file
int recvFile(char* buf, int s)
{
int i;
char ch;
for (i = 0; i < s; i++) {
ch = buf[i];
ch = Cipher(ch);
if (ch == EOF)
return 1;
master*
recvFile(char * buf, int s)
Ln 60, Col 24
Tab Size: 4
UTF-8
LF
C
Linux

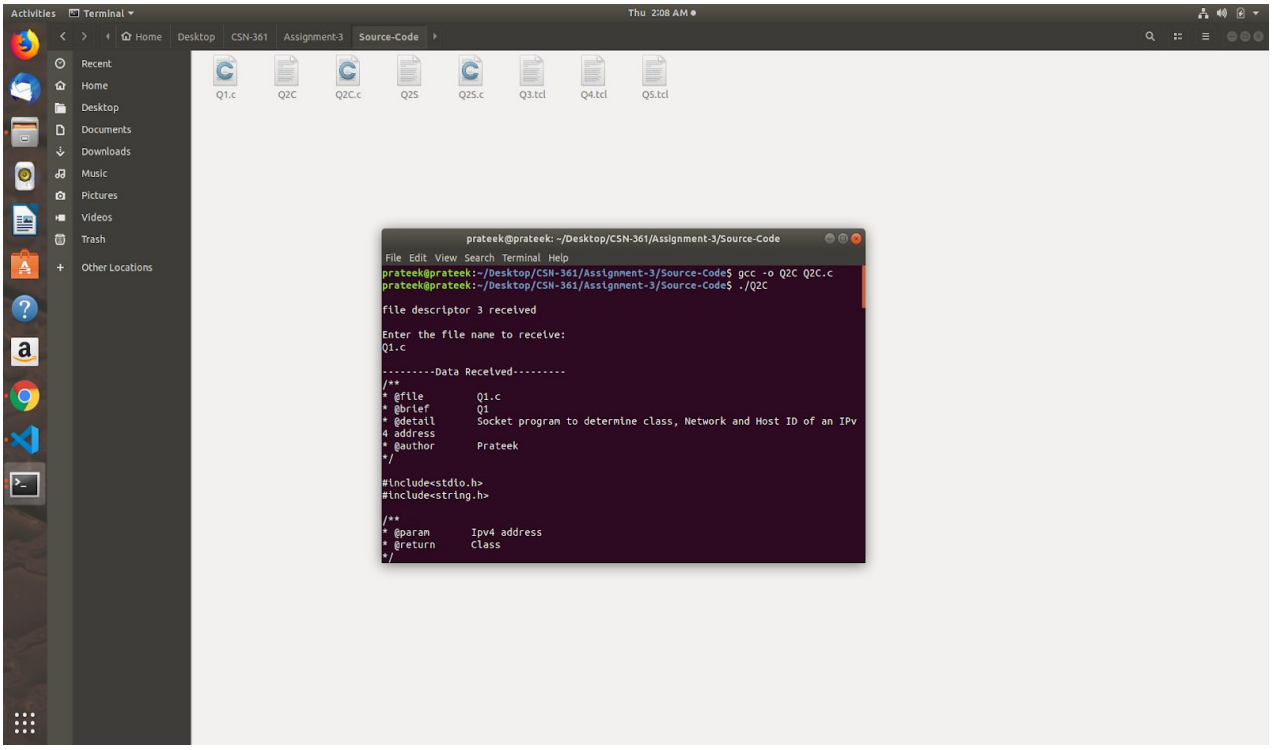

Activities
Visual Studio Code
Thu 2:01 AM
Q2C.c - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
Q2C.c
Q2S.c
Q1.c
home > prateek > Desktop > CSN-361 > Assignment-3 > Source-Code > Q2C.c > ...
else
printf("%c", ch);
}
return 0;
}
/**
* @brief Main Function
* @return Int
*/
// Main function
int main()
{
#define AF_INET PF_INET
Expands to:
PF_INET
int sockfd, nBytes;
struct sockaddr_in add
int addrlen = sizeof(a
addr_con.sin_family = AF_INET;
addr_con.sin_port = htons(PORT_NO);
addr_con.sin_addr.s_addr = inet_addr(IP_ADDRESS);
char net_buf[NET_BUF_SIZE];
FILE* fp;
sockfd = socket(AF_INET, SOCK_DGRAM,
IP_PROTOCOL);
if (sockfd < 0)
printf("\nfile descriptor not received!!\n");
else
printf("\nfile descriptor %d received\n", sockfd);
while (1) {
printf("\nEnter the file name to receive:\n");
master*
0 0
recvFile(char * buf, int s)
Ln 60, Col 24
Tab Size: 4
UTF-8
LF
C
Linux

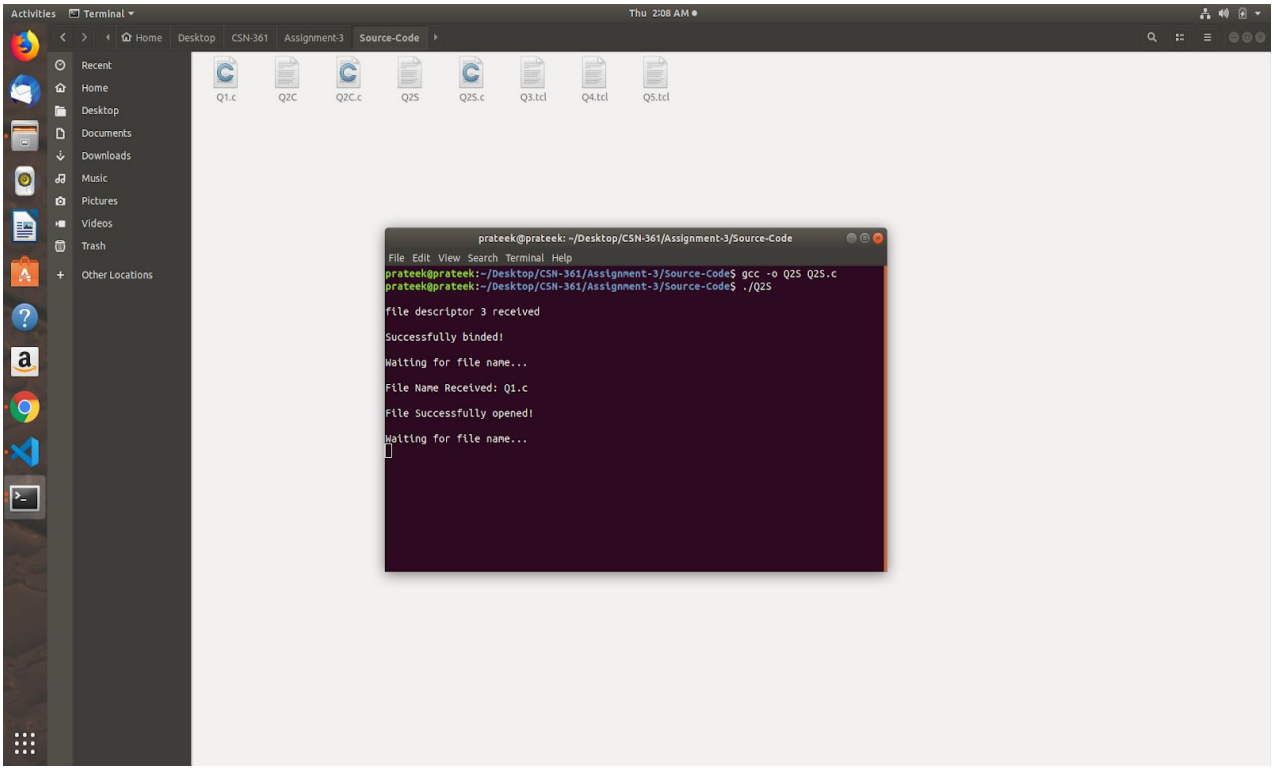

Activities
Visual Studio Code
Thu 2:01 AM
Q2C.c - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
Q2C.c
Q2S.c
Q1.c
home > prateek > Desktop > CSN-361 > Assignment-3 > Source-Code > Q2C.c > ...
printf("\nfile descriptor not received!!\n");
else
printf("\nfile descriptor %d received\n", sockfd);
while (1) {
printf("\nEnter the file name to receive:\n");
scanf("%s", net_buf);
sendto(sockfd, net_buf, NET_BUF_SIZE,
sendrecvflag, (struct sockaddr*)&addr_con,
addrlen);
printf("\n---------Data Received---------\n");
while (1) {
#define sendrecvflag 0
Expands to:
0
clearBuf(net_buf
nBytes = recvfro
UF_SIZE,
sendrecvflag, (struct sockaddr*)&addr_con,
&addrlen);
if (recvFile(net_buf, NET_BUF_SIZE)) {
break;
}
}
printf("\n------------------------------\n");
}
return 0;
}
master*
0 0
recvFile(char * buf, int s)
Ln 60, Col 24
Tab Size: 4
UTF-8
LF
C
Linux

Snapshots of running code







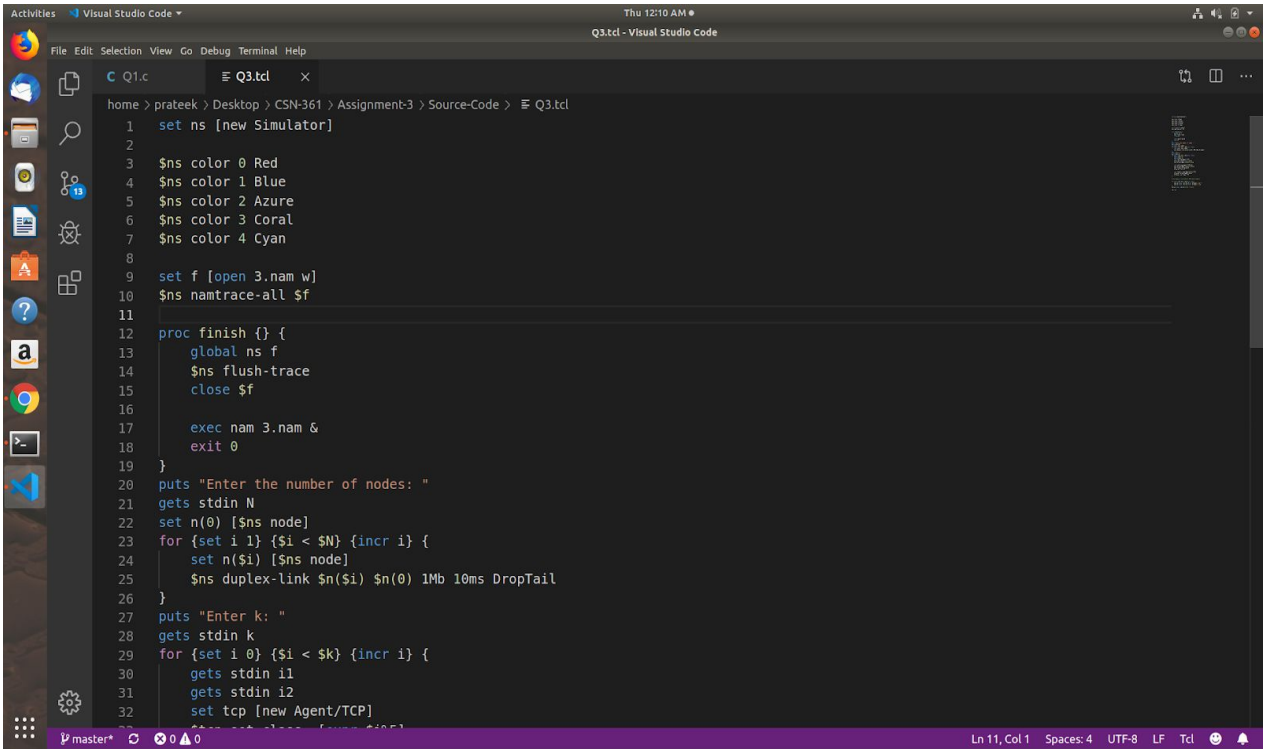


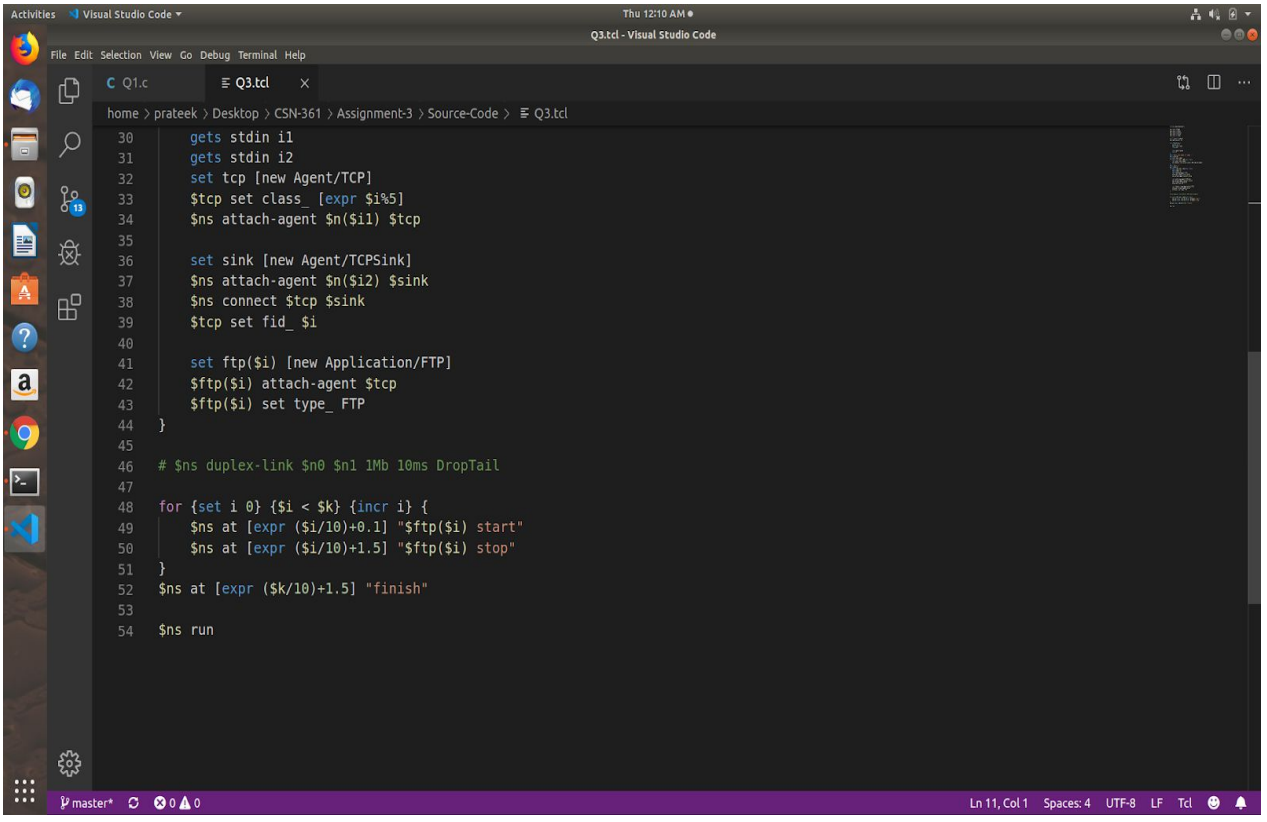
3

Implementations details

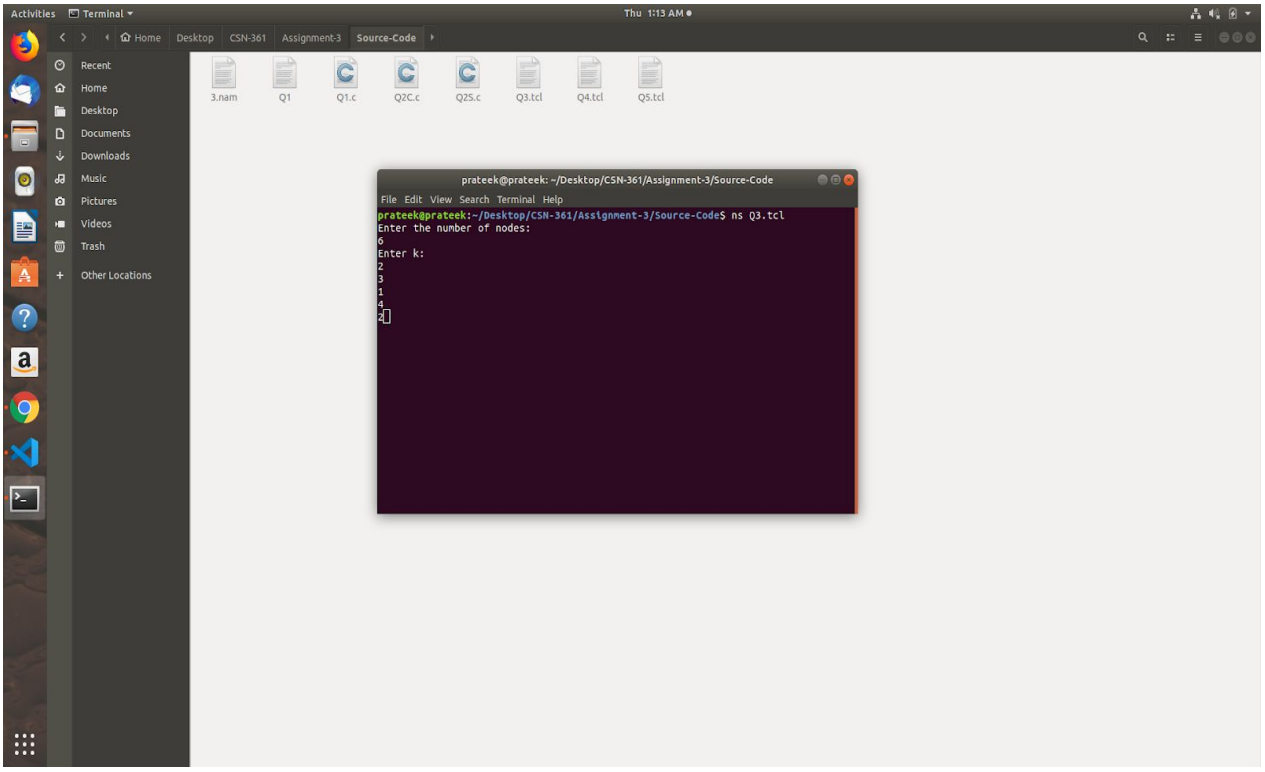
We have implemented star topology. A star topology is a topology for a Local Area Network (LAN) in which all nodes are individually connected to a central connection point, like a hub or a switch.

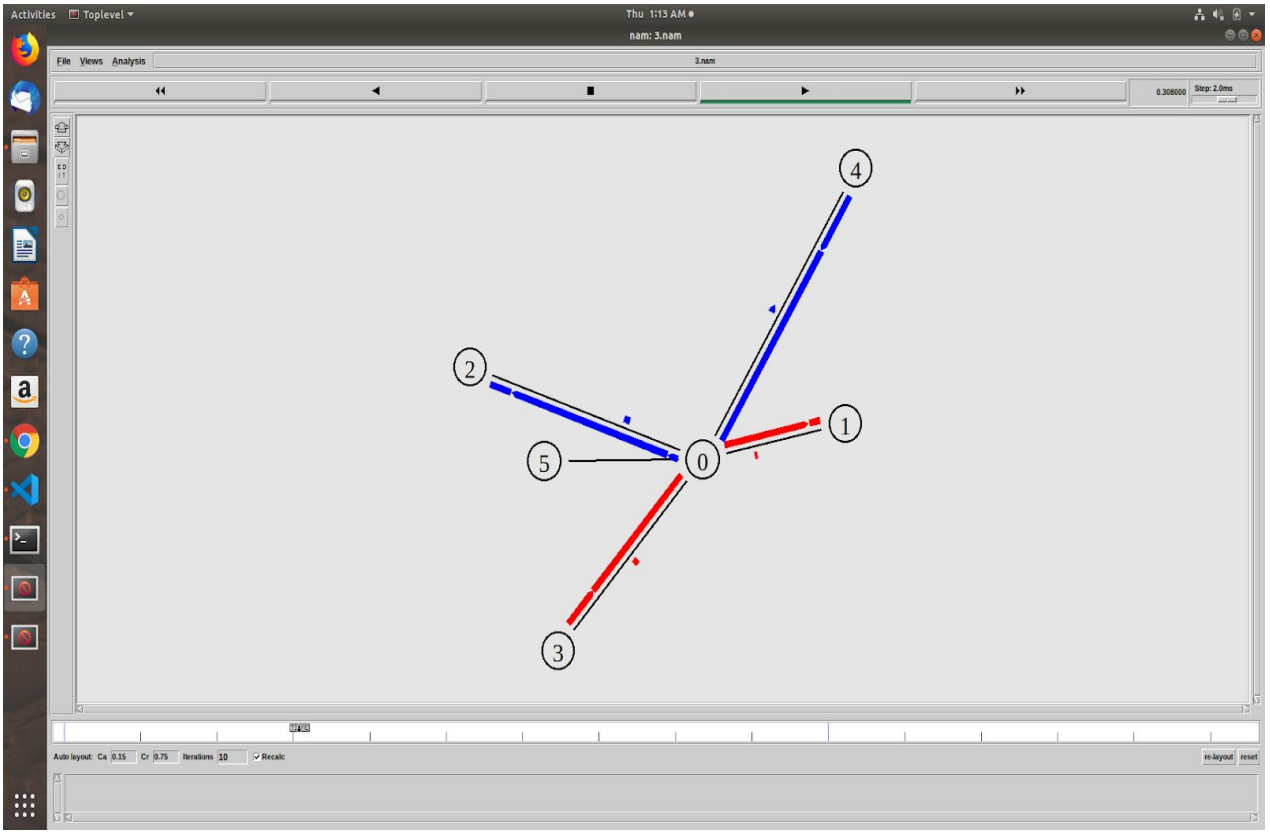
Code Snippets





Snapshots of running code



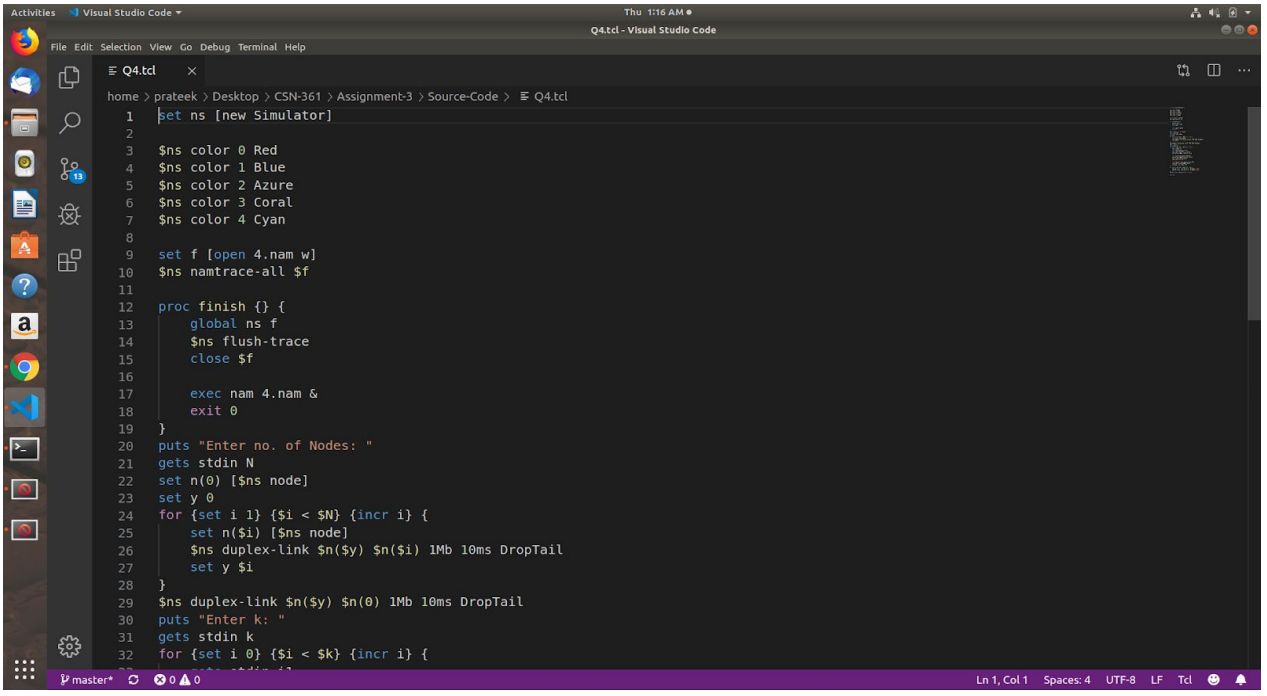


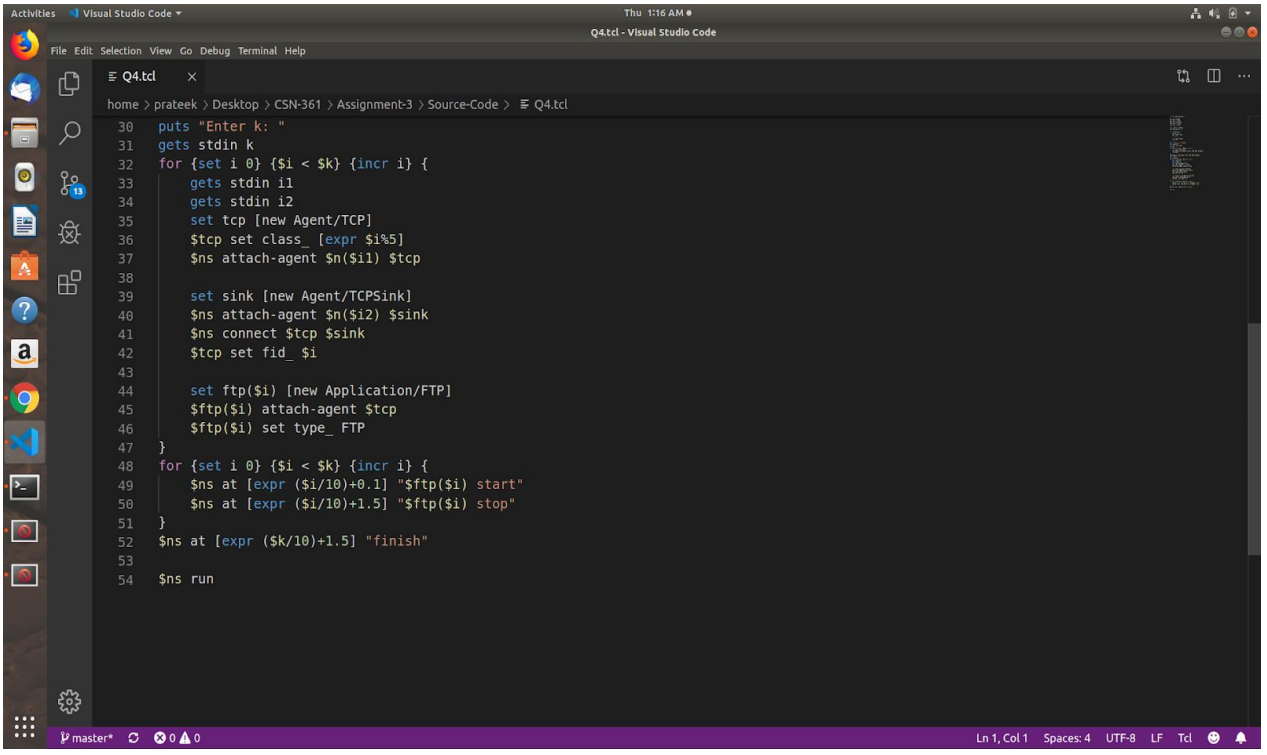
4

Implementations details

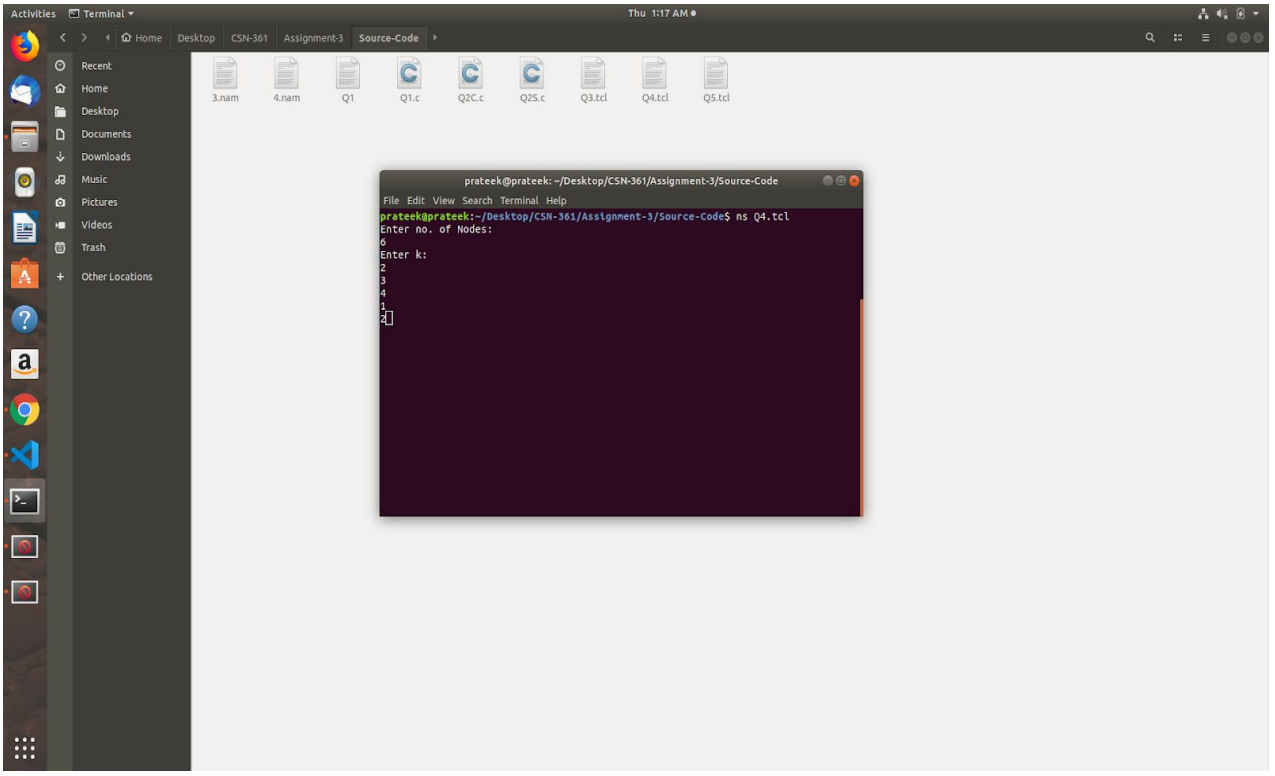
We have implemented ring topology. A ring topology is a network configuration in which device connections create a circular data path. Each networked device is connected to two others, like points on a circle.

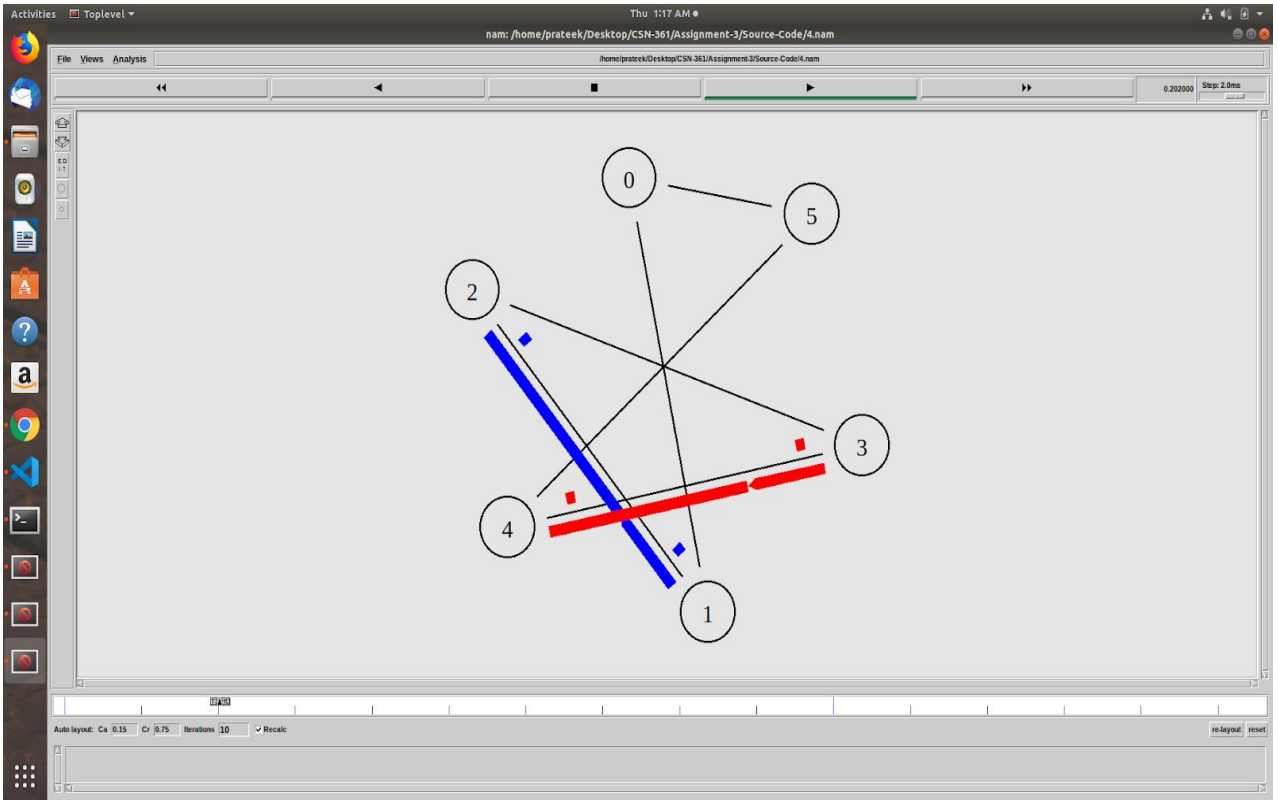
Code Snippets





Snapshots of running code



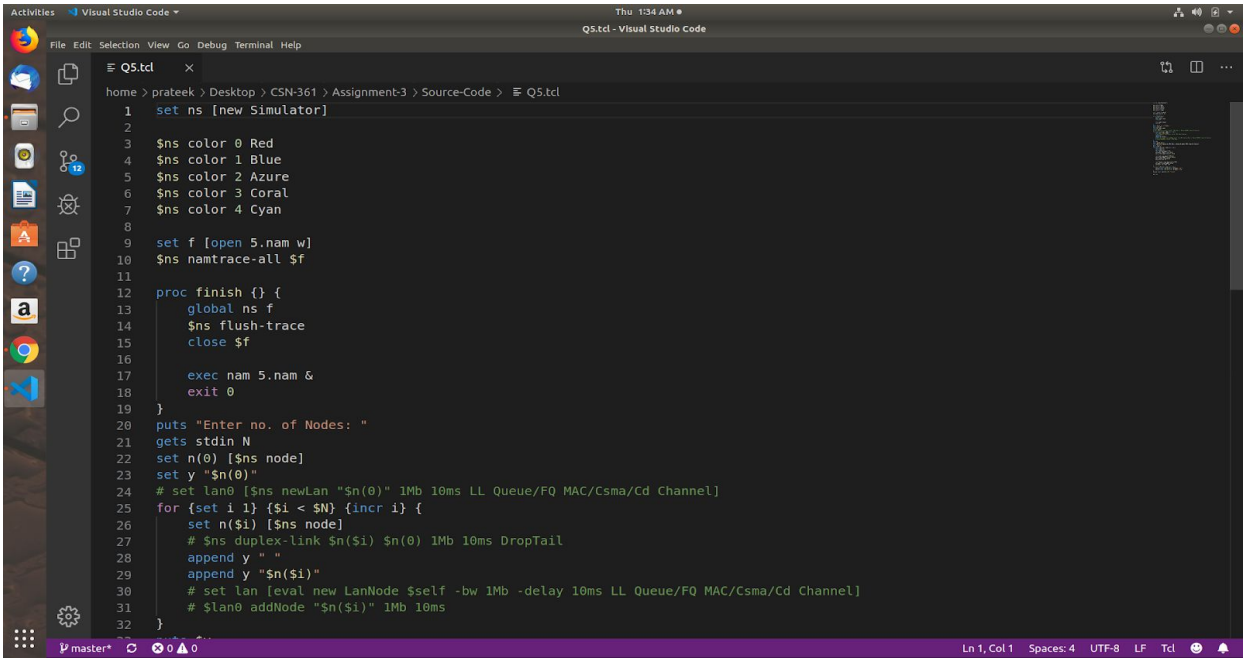


5

Implementations details

We have implemented bus topology. A bus topology is a topology for a Local Area Network (LAN) in which all the nodes are connected to a single cable.

Code Snippets



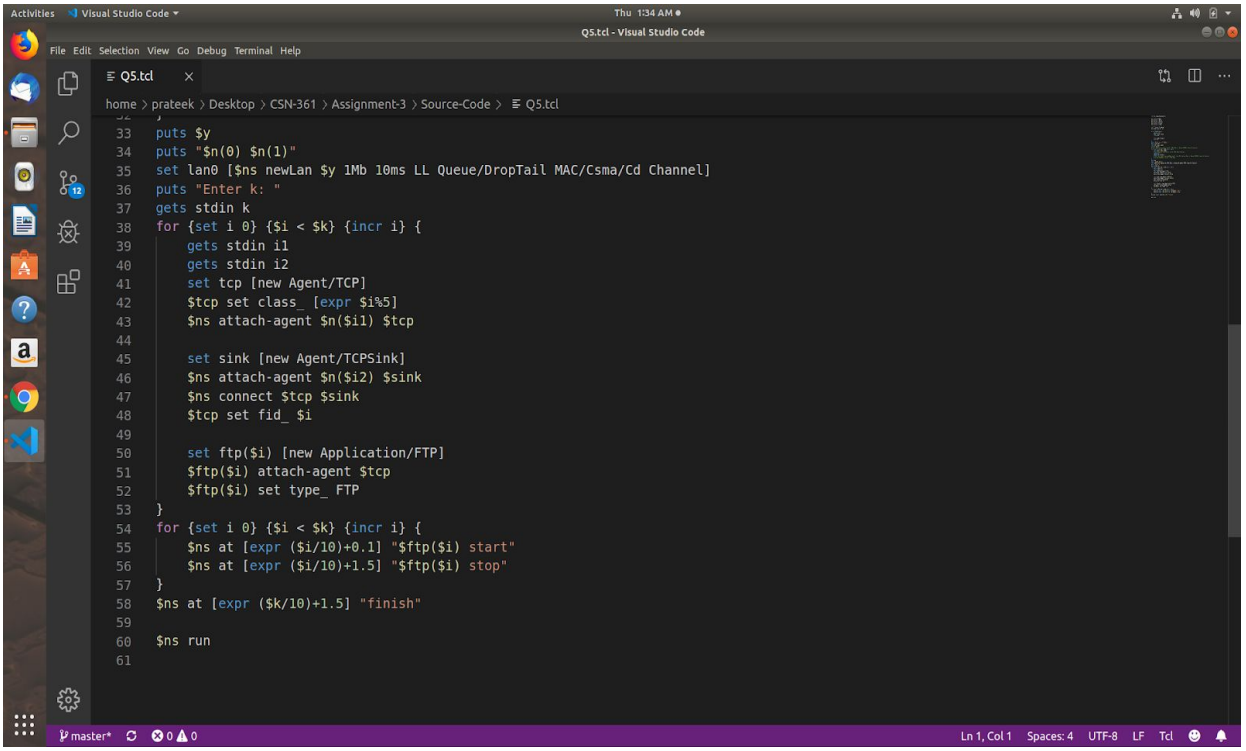

```tcl
set ns [new Simulator]

$ns color 0 Red
$ns color 1 Blue
$ns color 2 Azure
$ns color 3 Coral
$ns color 4 Cyan

set f [open 5.nam w]
$ns namtrace-all $f

proc finish {} {
    global ns f
    $ns flush-trace
    close $f

    exec nam 5.nam &
    exit 0
}
puts "Enter no. of Nodes: "
gets stdin N
set n(0) [$ns node]
set y "$n(0)"
# set lan0 [$ns newLan "$n(0)" 1Mb 10ms LL Queue/FQ MAC/Csma/Cd Channel]
for {set i 1} {$i < $N} {incr i} {
    set n($i) [$ns node]
    # $ns duplex-link $n($i) $n(0) 1Mb 10ms DropTail
    append y " "
    append y "$n($i)"
    # set lan [eval new LanNode $self -bw 1Mb -delay 10ms LL Queue/FQ MAC/Csma/Cd Channel]
    # $lan0 addNode "$n($i)" 1Mb 10ms
}
```




```tcl
puts $y
puts "$n(0) $n(1)"
set lan0 [$ns newLan $y 1Mb 10ms LL Queue/DropTail MAC/Csma/Cd Channel]
puts "Enter k: "
gets stdin k
for {set i 0} {$i < $k} {incr i} {
    gets stdin i1
    gets stdin i2
    set tcp [new Agent/TCP]
    $tcp set class_ [expr $i%5]
    $ns attach-agent $n($i1) $tcp

    set sink [new Agent/TCPSink]
    $ns attach-agent $n($i2) $sink
    $ns connect $tcp $sink
    $tcp set fid_ $i

    set ftp($i) [new Application/FTP]
    $ftp($i) attach-agent $tcp
    $ftp($i) set type_ FTP
}
for {set i 0} {$i < $k} {incr i} {
    $ns at [expr ($i/10)+0.1] "$ftp($i) start"
    $ns at [expr ($i/10)+1.5] "$ftp($i) stop"
}
$ns at [expr ($k/10)+1.5] "finish"

$ns run
```

Snapshots of running code