Lab Assignment- 3

CSN-361: Computer Networks Laboratory

August 22, 2019

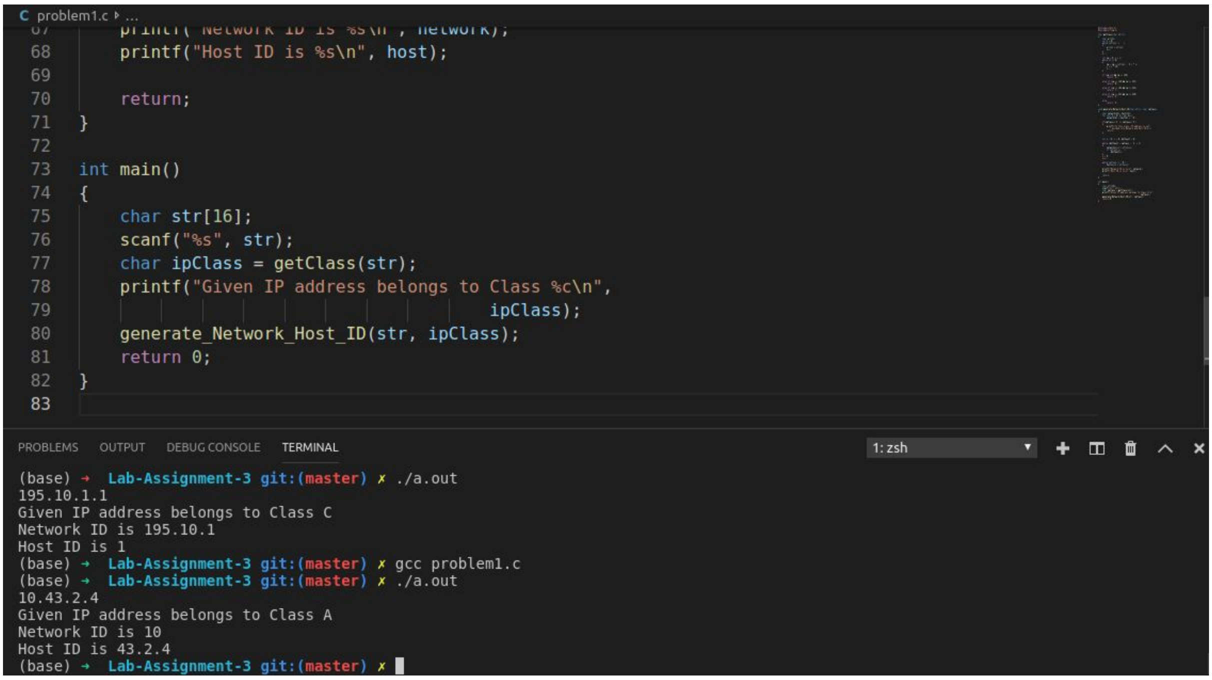
Gopi Kishan
Enrolment No. : 17114035
B.Tech, 3rd Yr
Computer Science and Engineering (CSE)

Problem Statement 1

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

- **Algorithms and data structures used in the implementation**
  - Input the IP Address.
  - Check in the ranges for the class.
  - Parse the string according to the class to get the Host ID and Network ID.

- **Snapshots of running the codes for each of the problems**



```c
    printf("Network ID is %s\n", network);
    printf("Host ID is %s\n", host);

    return;
}

int main()
{
    char str[16];
    scanf("%s", str);
    char ipClass = getClass(str);
    printf("Given IP address belongs to Class %c\n",
                                            ipClass);
    generate_Network_Host_ID(str, ipClass);
    return 0;
}
```

```
(base) → Lab-Assignment-3 git:(master) ✗ ./a.out
195.10.1.1
Given IP address belongs to Class C
Network ID is 195.10.1
Host ID is 1
(base) → Lab-Assignment-3 git:(master) ✗ gcc problem1.c
(base) → Lab-Assignment-3 git:(master) ✗ ./a.out
10.43.2.4
Given IP address belongs to Class A
Network ID is 10
Host ID is 43.2.4
(base) → Lab-Assignment-3 git:(master) ✗ 
```

Problem Statement 2

Write a C program to demonstrate File Transfer using UDP.

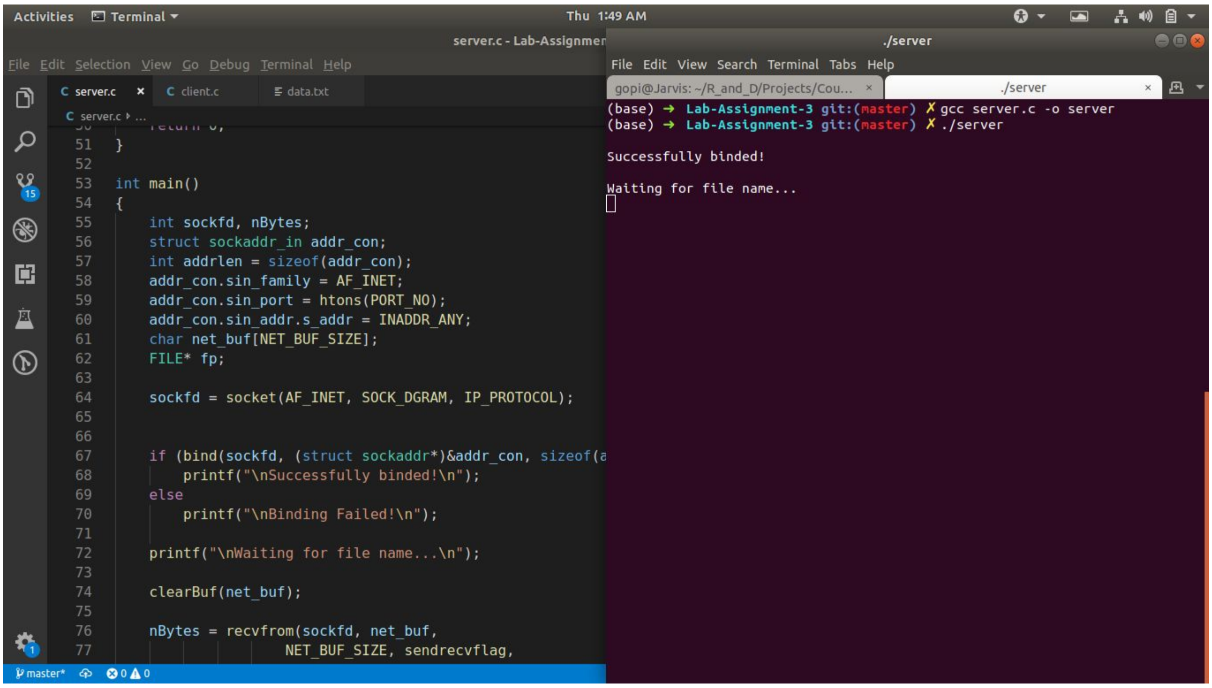
- **Algorithms and data structures used in the implementation**
  UDP Server :
    - Create UDP socket.
    - Bind the socket to server address.
    - Wait until datagram packet arrives from client.
    - Process the datagram packet and send a reply to client.

  UDP Client :
    - Create UDP socket.
    - Send message to server.
    - Wait until response from server is recieved.
    - Process reply and go back to step 2, if necessary.
    - Close socket descriptor and exit.

  Data structures used :

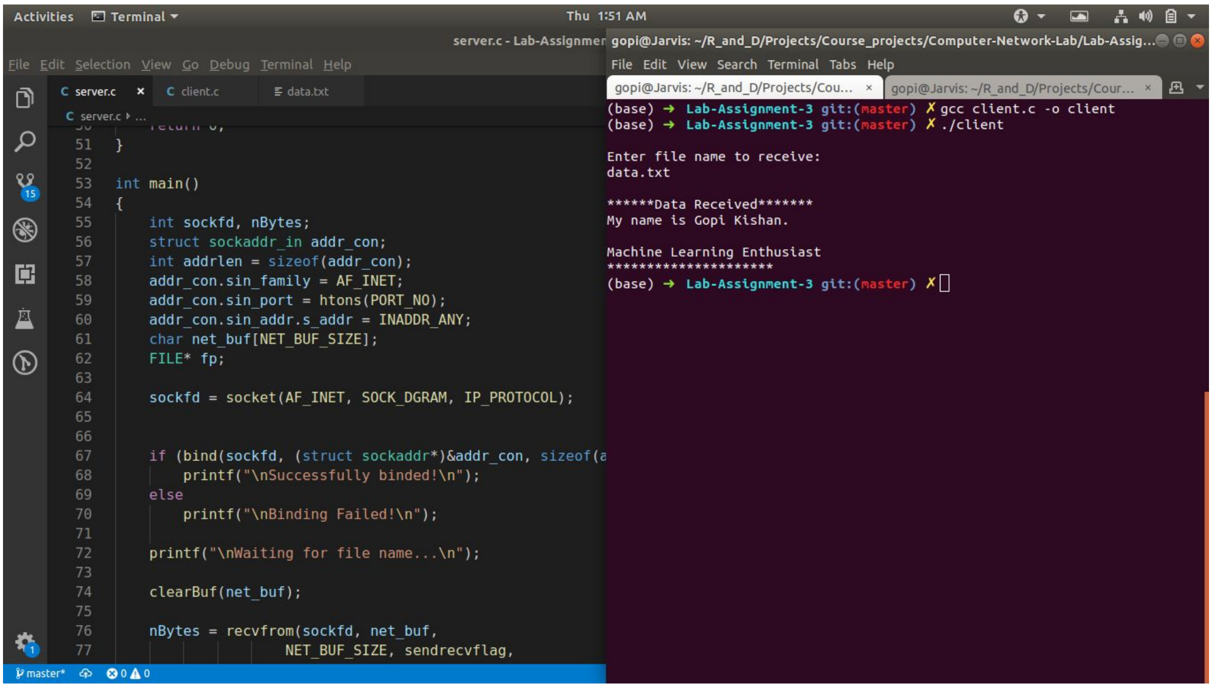
- **sockfd –** File descriptor of socket
- **buf –** Application buffer containing the data to be sent
- **len –** Size of *buf* application buffer
- **flags –** Bitwise OR of flags to modify socket behaviour
- **dest_addr –** Structure containing address of destination
- **addrlen –** Size of *dest_addr* structure

- **Snapshots of running the codes for each of the problems**


Activities
Terminal
Thu 1:49 AM
server.c - Lab-Assignmen
./server
File Edit Selection View Go Debug Terminal Help
server.c
client.c
data.txt
server.c ▸ ...
File Edit View Search Terminal Tabs Help
gopi@Jarvis: ~/R_and_D/Projects/Cou...
./server
master*
0 0

```c
}

int main()
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    int addrlen = sizeof(addr_con);
    addr_con.sin_family = AF_INET;
    addr_con.sin_port = htons(PORT_NO);
    addr_con.sin_addr.s_addr = INADDR_ANY;
    char net_buf[NET_BUF_SIZE];
    FILE* fp;

    sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);


    if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(a
        printf("\nSuccessfully binded!\n");
    else
        printf("\nBinding Failed!\n");

    printf("\nWaiting for file name...\n");

    clearBuf(net_buf);

    nBytes = recvfrom(sockfd, net_buf,
                      NET_BUF_SIZE, sendrecvflag,
```

```
(base) ➜ Lab-Assignment-3 git:(master) ✗ gcc server.c -o server
(base) ➜ Lab-Assignment-3 git:(master) ✗ ./server

Successfully binded!

Waiting for file name...
```


Activities
Terminal
Thu 1:51 AM
server.c - Lab-Assignmen
gopi@Jarvis: ~/R_and_D/Projects/Course_projects/Computer-Network-Lab/Lab-Assig...
File Edit Selection View Go Debug Terminal Help
server.c
client.c
data.txt
server.c ▸ ...
File Edit View Search Terminal Tabs Help
gopi@Jarvis: ~/R_and_D/Projects/Cou...
gopi@Jarvis: ~/R_and_D/Projects/Cour...
master*
0 0

```c
}

int main()
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    int addrlen = sizeof(addr_con);
    addr_con.sin_family = AF_INET;
    addr_con.sin_port = htons(PORT_NO);
    addr_con.sin_addr.s_addr = INADDR_ANY;
    char net_buf[NET_BUF_SIZE];
    FILE* fp;

    sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);


    if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(a
        printf("\nSuccessfully binded!\n");
    else
        printf("\nBinding Failed!\n");

    printf("\nWaiting for file name...\n");

    clearBuf(net_buf);

    nBytes = recvfrom(sockfd, net_buf,
                      NET_BUF_SIZE, sendrecvflag,
```

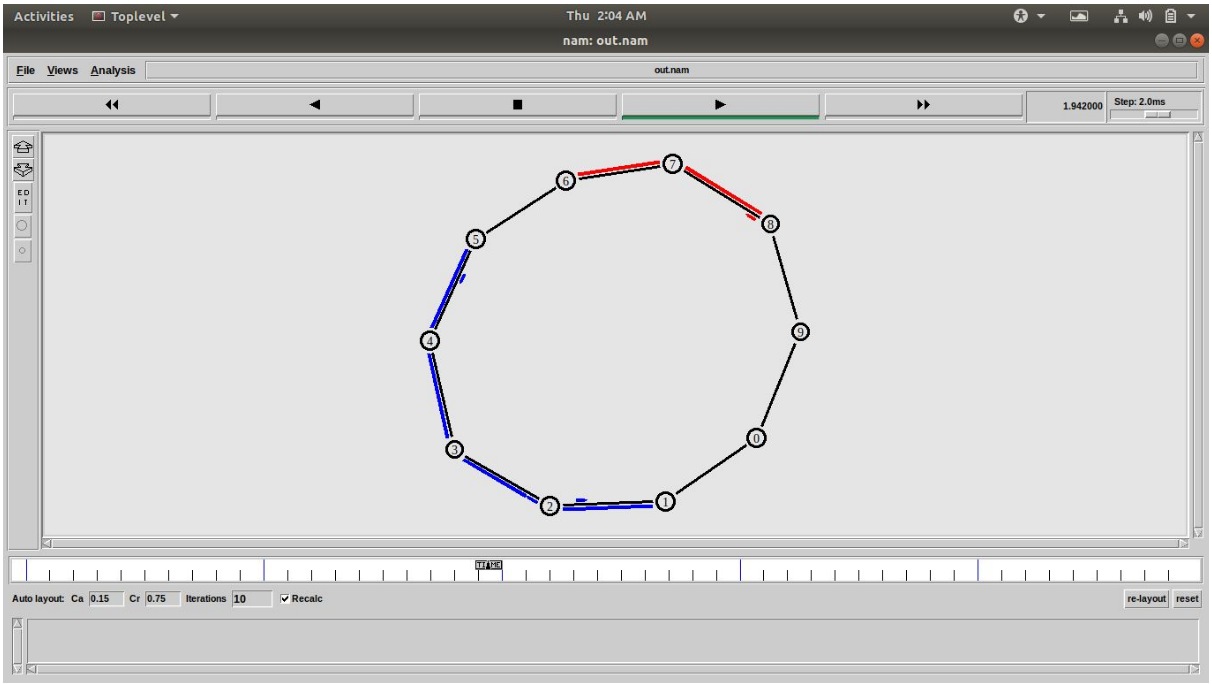
```
(base) ➜ Lab-Assignment-3 git:(master) ✗ gcc client.c -o client
(base) ➜ Lab-Assignment-3 git:(master) ✗ ./client

Enter file name to receive:
data.txt

******Data Received*******
My name is Gopi Kishan.

Machine Learning Enthusiast
*********************
(base) ➜ Lab-Assignment-3 git:(master) ✗
```

Problem Statement 3

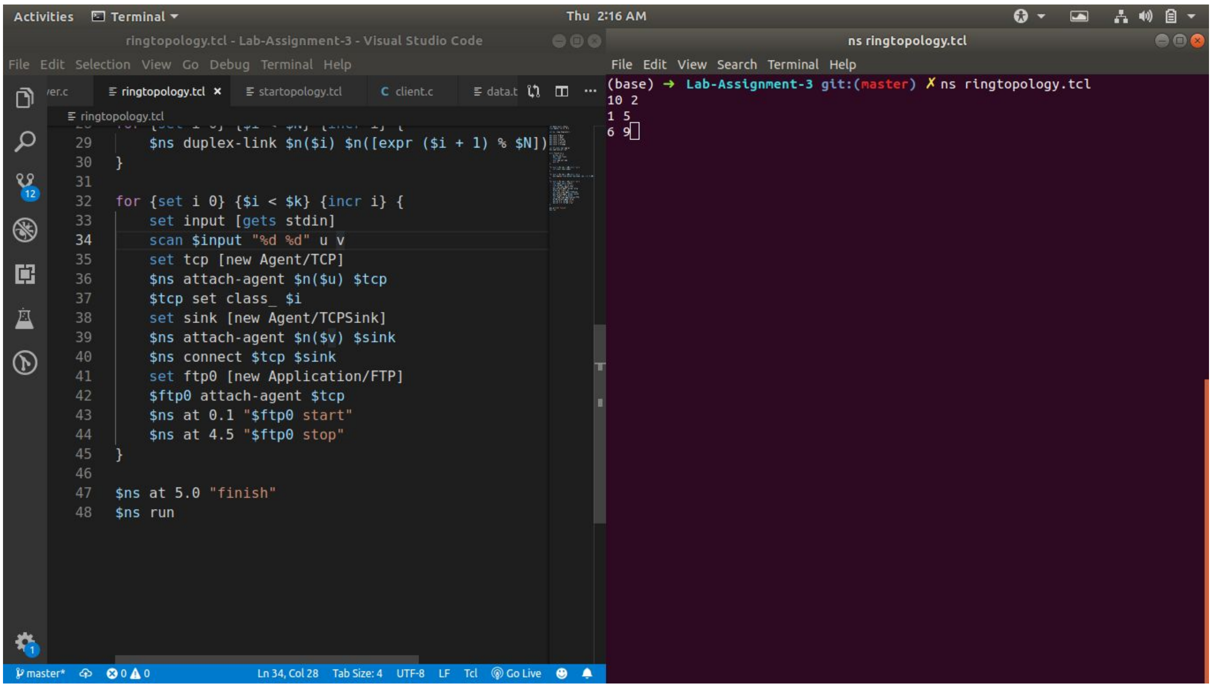
Write a TCL code for network simulator NS2 to demonstrate the star topology among a set of computer nodes. Given N nodes, one node will be assigned as the central node and the other nodes will be connected to it to form the star. You have to set up a TCP connection between k pairs of nodes and demonstrate 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.

- **Algorithms and data structures used in the implementation**
- Create N node using "`set n($i) [$ns node]`"
- Link nodes in a circular way using "`$ns duplex-link $n($i) $n([expr ($i + 1) % $N])`"
- Read input nodes among which package is to be transmitted
- Then, set tcp [new Agent/TCP] ; $ns attach-agent $n($u) $tcp

- **Snapshots of running the codes for each of the problems**



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc



Activities
Terminal
Thu 2:16 AM
ringtopology.tcl - Lab-Assignment-3 - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
ver.c
ringtopology.tcl
startopology.tcl
client.c
data.t
ringtopology.tcl
master*
0 0
Ln 34, Col 28 Tab Size: 4 UTF-8 LF Tcl Go Live
ns ringtopology.tcl
File Edit View Search Terminal Help

```
for {set i 0} {$i < $N} {incr i} {
    $ns duplex-link $n($i) $n([expr ($i + 1) % $N])
}

for {set i 0} {$i < $k} {incr i} {
    set input [gets stdin]
    scan $input "%d %d" u v
    set tcp [new Agent/TCP]
    $ns attach-agent $n($u) $tcp
    $tcp set class_ $i
    set sink [new Agent/TCPSink]
    $ns attach-agent $n($v) $sink
    $ns connect $tcp $sink
    set ftp0 [new Application/FTP]
    $ftp0 attach-agent $tcp
    $ns at 0.1 "$ftp0 start"
    $ns at 4.5 "$ftp0 stop"
}

$ns at 5.0 "finish"
$ns run
```

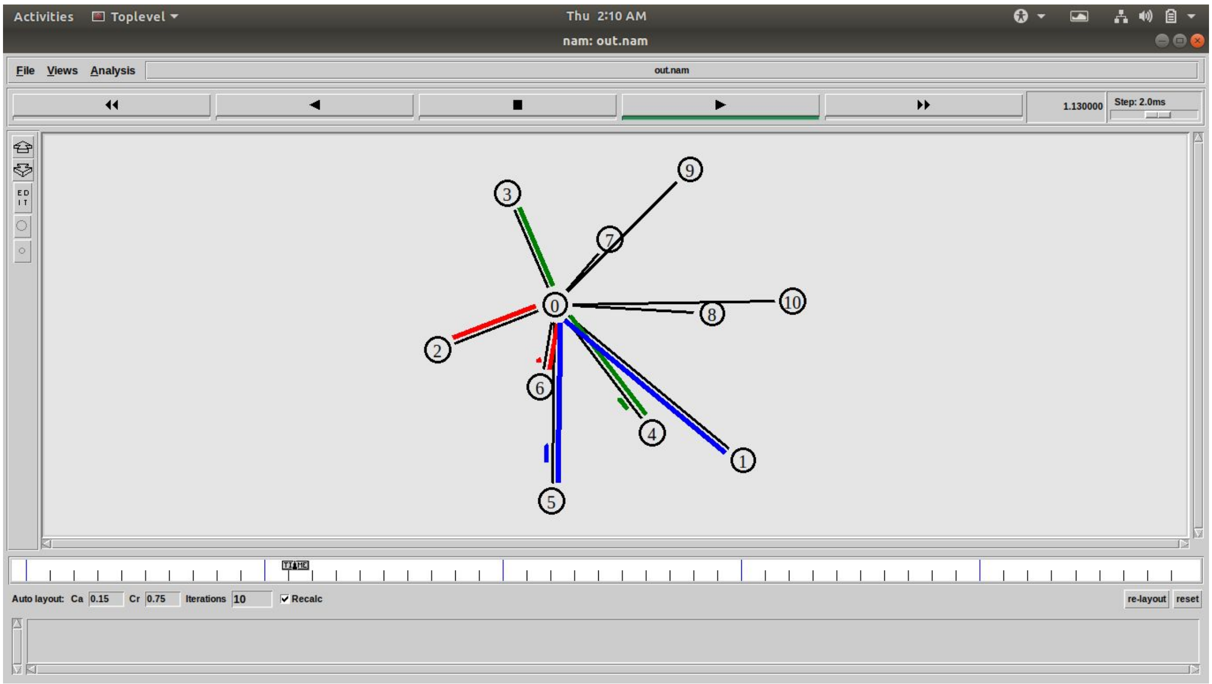
```
(base) ➜ Lab-Assignment-3 git:(master) ✗ ns ringtopology.tcl
10 2
1 5
6 9
```

Problem Statement 4

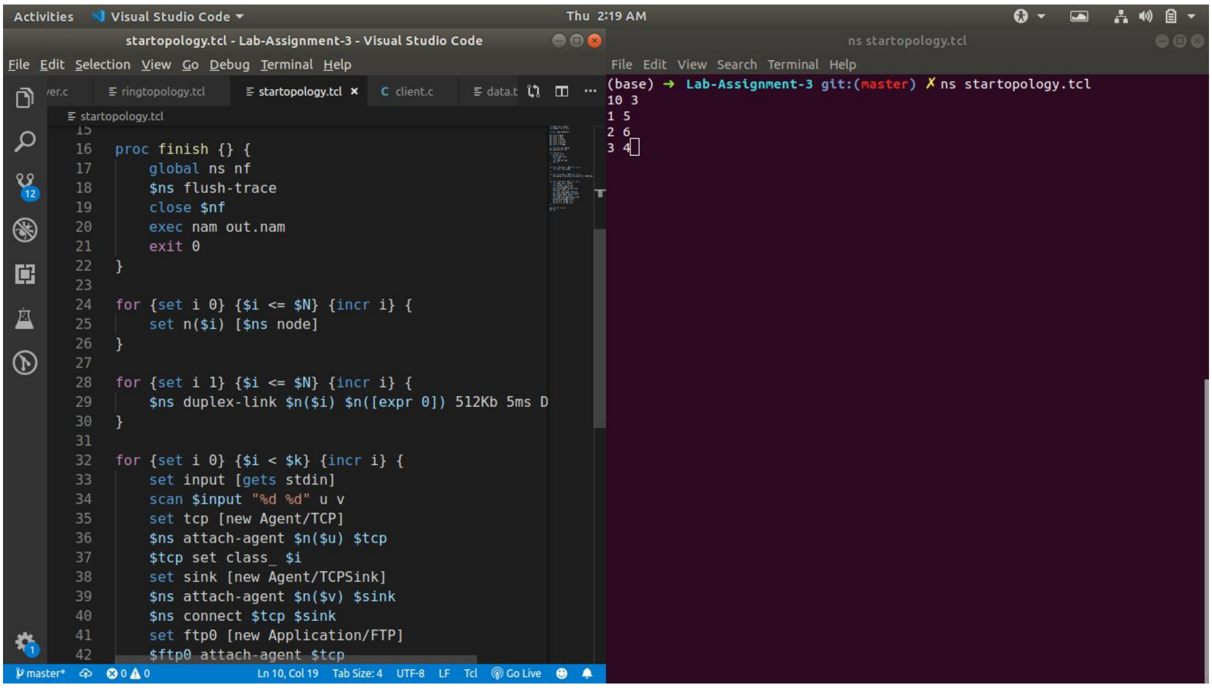
Write a TCL code for network simulator NS2 to demonstrate the ring topology among a set of computer nodes. Given N nodes, each node will be connected to two other nodes in the form of a ring. You have to set up a TCP connection between k pairs of nodes and demonstrate packet transfer between them using 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.

- **Algorithms and data structures used in the implementation**
- Create N node using “`set n($i) [$ns node]`”
- Set node $n0 as center node.
- Link all nodes to $no using “`$ns duplex-link $n($i) $n([expr0])`”
- Read input nodes among which package is to be transmitted
- Then, set tcp [new Agent/TCP] ; $ns attach-agent $n($u) $tcp

- **Snapshots of running the codes for each of the problems**



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc



```
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam
    exit 0
}

for {set i 0} {$i <= $N} {incr i} {
    set n($i) [$ns node]
}

for {set i 1} {$i <= $N} {incr i} {
    $ns duplex-link $n($i) $n([expr 0]) 512Kb 5ms D
}

for {set i 0} {$i < $k} {incr i} {
    set input [gets stdin]
    scan $input "%d %d" u v
    set tcp [new Agent/TCP]
    $ns attach-agent $n($u) $tcp
    $tcp set class_ $i
    set sink [new Agent/TCPSink]
    $ns attach-agent $n($v) $sink
    $ns connect $tcp $sink
    set ftp0 [new Application/FTP]
    $ftp0 attach-agent $tcp
```

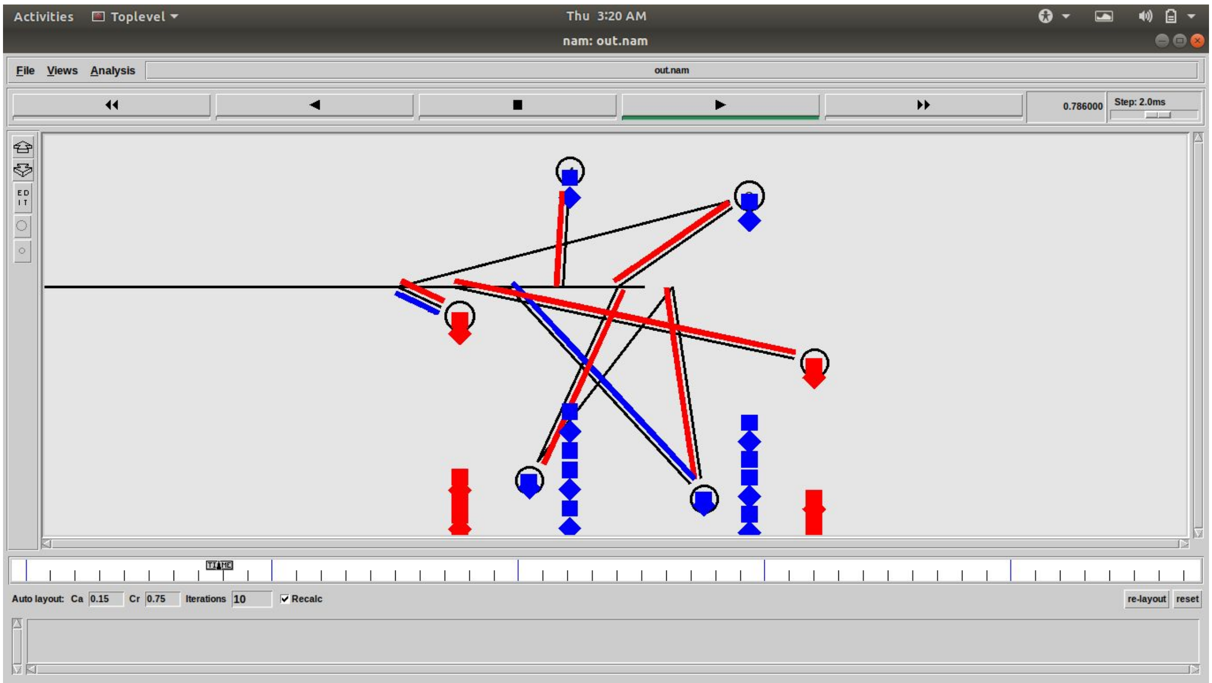
```
(base) ➜ Lab-Assignment-3 git:(master) ✗ ns startopology.tcl
10 3
1 5
2 6
3 4
```

Problem Statement 5

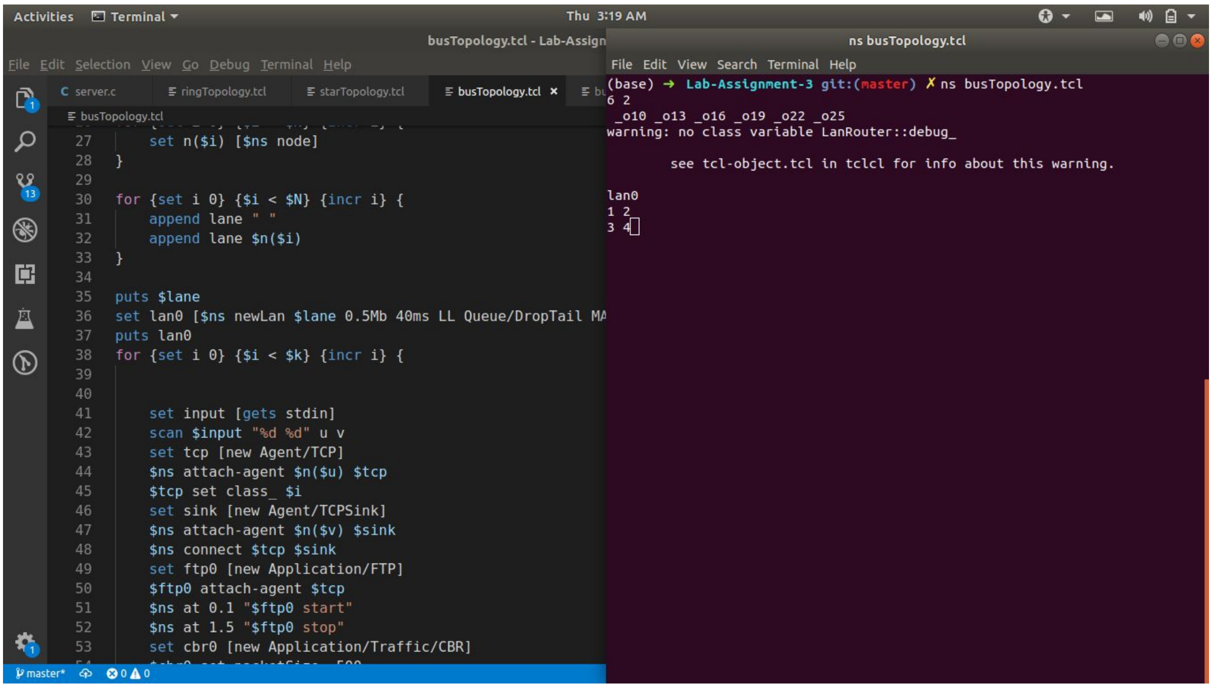
Write a TCL code for network simulator NS2 to demonstrate the bus topology among a set of computer nodes. Given N nodes, each node will be connected to a common link. You have to set up a TCP connection between k pairs of nodes and demonstrate packet transfer between them using Network Animator (NAM). 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.

- **Algorithms and data structures used in the implementation**
- Create N node using “`set n($i) [$ns node]`”
- Append all nodes to “lane”
- Create a lan-cable and attach all nodes to it using “`set lan0 [$ns newLan $lane]`”
- Read input nodes among which package is to be transmitted
- Then, set tcp [new Agent/TCP] ; $ns attach-agent $n($u) $tcp

- **Snapshots of running the codes for each of the problems**



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc



```
    set n($i) [$ns node]
}

for {set i 0} {$i < $N} {incr i} {
    append lane " "
    append lane $n($i)
}

puts $lane
set lan0 [$ns newLan $lane 0.5Mb 40ms LL Queue/DropTail MA
puts lan0
for {set i 0} {$i < $k} {incr i} {


    set input [gets stdin]
    scan $input "%d %d" u v
    set tcp [new Agent/TCP]
    $ns attach-agent $n($u) $tcp
    $tcp set class_ $i
    set sink [new Agent/TCPSink]
    $ns attach-agent $n($v) $sink
    $ns connect $tcp $sink
    set ftp0 [new Application/FTP]
    $ftp0 attach-agent $tcp
    $ns at 0.1 "$ftp0 start"
    $ns at 1.5 "$ftp0 stop"
    set cbr0 [new Application/Traffic/CBR]
```

```
(base) → Lab-Assignment-3 git:(master) ✗ ns busTopology.tcl
6 2
 _o10 _o13 _o16 _o19 _o22 _o25
warning: no class variable LanRouter::debug_

        see tcl-object.tcl in tclcl for info about this warning.

lan0
1 2
3 4
```