LAB ASSIGNMENT-3

CSN-361

Submitted by
Ritik Kumar 17114063

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

Algorithm

- Fetch the IP address
- Detect the class using the first octet of input
    - A 1 - 126
    - B 127 - 191
    - C 192 - 223
    - D 224 - 239
    - E 240 - 255
- According to the class, separate the Network and Host ID using their subnet masks

DataStructure

Int, char array, char, boolean

Code

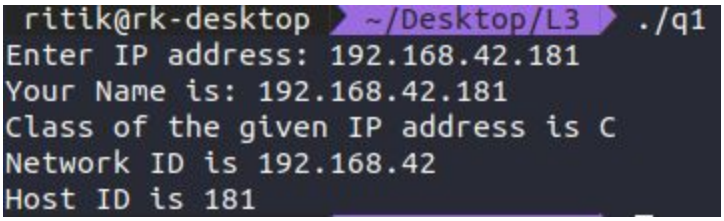
C Q1.c /home/ritik/Desktop/L3/Q1.c/...

```c
/** @file Q1.c
 *  @brief Determine class, Network and Host ID.
 *
 *  @author Ritik
 *
 * @date 8/21/2019
 */

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

/** @brief finding class.
 *
 *  @param str input IP address.
 *
 *  @return class.
 *
 */

char findClass(char str[])
{
    // storing first octet in arr[] variable
    char arr[4];
    int i = 0;
    while (str[i] != '.')
    {
        arr[i] = str[i];
        i++;
    }
    i--;
    // converting str[] variable into number for comparison
    int ip = 0, j = 1;
    while (i >= 0)
    {
        ip = ip + (str[i] - '0') * j;
        j = j * 10;
        i--;
    }
    // Class A
    if (ip >=1 && ip <= 126)
        return 'A';

    // Class B
    else if (ip >= 128 && ip <= 191)
        return 'B';
    // Class C
    else if (ip >= 192 && ip <= 223)
        return 'C';
    // Class D
    else if (ip >= 224 && ip <= 239)
        return 'D';
    // Class E
    else if (ip >= 240 && ip <= 255)
        return 'E';
    else return 'x';
}

//
/** @brief Function to separate Network ID as well as Host ID and print them.
 *
 *  @param str input IP address.
 *  @param ipClass class of IP.
 *
 */
```

```c
void separate(char str[], 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 (str[j] != '.')
            network[i++] = str[j++];
        i = 0;
        j++;
        while (str[j] != '\0')
            host[i++] = str[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
        // dotCount keeps track of number of
        // dots or octets passed
        while (dotCount < 2)
        {
            network[i++] = str[j++];
            if (str[j] == '.')
                dotCount++;
        }
        i = 0;
        j++;
        while (str[j] != '\0')
            host[i++] = str[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 or octets passed
        while (dotCount < 3)
        {
            network[i++] = str[j++];
            if (str[j] == '.')
                dotCount++;
        }
        i = 0;
        j++;
        while (str[j] != '\0')
            host[i++] = str[j++];
        printf("Network ID is %s\n", network);
        printf("Host ID is %s\n", host);
    }
    // Class D and E are not divided in Network
```

```c
/** @brief Q1 Client entrypoint for client.
 */
int main()
{
    char str[16];
    printf("Enter IP address: ");
    scanf("%s", str);
    printf("Your Name is: %s \n", str);
    // char str[] = "192.226.12.11";
    char ipClass = findClass(str);
    if(ipClass == 'x') {
        printf("Incorrect IP address");
        return 0;
    }
    printf("Class of the given IP address is %c\n", ipClass);
    separate(str, ipClass);
    return 0;
}
```

Running Code



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

Client

Algorithm

- Add the connection details like protocols, address, port
- Create a socket and wait for new connections
- Get the filename from the user
- Request the server for the file
- Receive the file from the server in a buffer
- Print the received status

DataStructure

Sockaddr_in, char array, File

Code

Q2_Client.c /home/ritik/Desktop/L3/Q2_Client.c/...

```c
/** @file Q2_Client.c
 *  @brief client code for UDP socket programming.
 *
 *  @author Ritik
 *
 * @date 8/21/2019
 */

#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 sendrecvflag 0

/** @brief function to clear buffer.
 *
 *  @param b pointer to buffer.
 *
 */
void clearBuf(char* b)
{
  int i;
  for (i = 0; i < NET_BUF_SIZE; i++)
    b[i] = '\0';
}

/** @brief function to receive file.
 *
 *  @param buf pointer to buffer.
 *  @param s size of buffer.
 *
 *  @return successful.
 *
 */

int recvFile(char* buf, int s)
{
  int i;
  char ch;
  for (i = 0; i < s; i++) {
    ch = buf[i];
    if (ch == EOF)
      return 1;
    else
      printf("%c", ch);
  }
  return 0;
}
```

```c
/** @brief Q2 Client entrypoint for client.
 */
int main()
{
  int sockfd, nBytes;
  struct 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 = inet_addr(IP_ADDRESS);
  char net_buf[NET_BUF_SIZE];
  FILE* fp;
  // socket()
  sockfd = socket(AF_INET, SOCK_DGRAM,
                  IP_PROTOCOL);

  if (sockfd < 0)
    printf("file descriptor not received!!\n");
  else
    printf("file descriptor %d received\n", sockfd);

  while (1) {
    printf("Please enter file name to receive:\n");
    scanf("%s", net_buf);
    sendto(sockfd, net_buf, NET_BUF_SIZE,
      sendrecvflag, (struct sockaddr*)&addr_con,
      addrlen);

    printf("---------Data Received---------\n");
    while (1) {
      // receive
      clearBuf(net_buf);
      nBytes = recvfrom(sockfd, net_buf, NET_BUF_SIZE,
                sendrecvflag, (struct sockaddr*)&addr_con,
                &addrlen);
      // process
      if (recvFile(net_buf, NET_BUF_SIZE)) {
        break;
      }
    }
    printf("-------------------------------\n");
  }
  return 0;
}
```

Running Code

```
✘ ritik@rk-desktop ~/Desktop/L3 ./q2c
file descriptor 3 received
Please enter file name to receive:
Q3.tcl
---------Data Received---------
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 3.nam w]
$ns namtrace-all $f

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

    exec nam 3.nam &
    exit 0
}
puts "Enter no. of Nodes: "
gets stdin N
set n(0) [$ns node]
$ns at 0.0 "$n(0) label node0"
```

Server

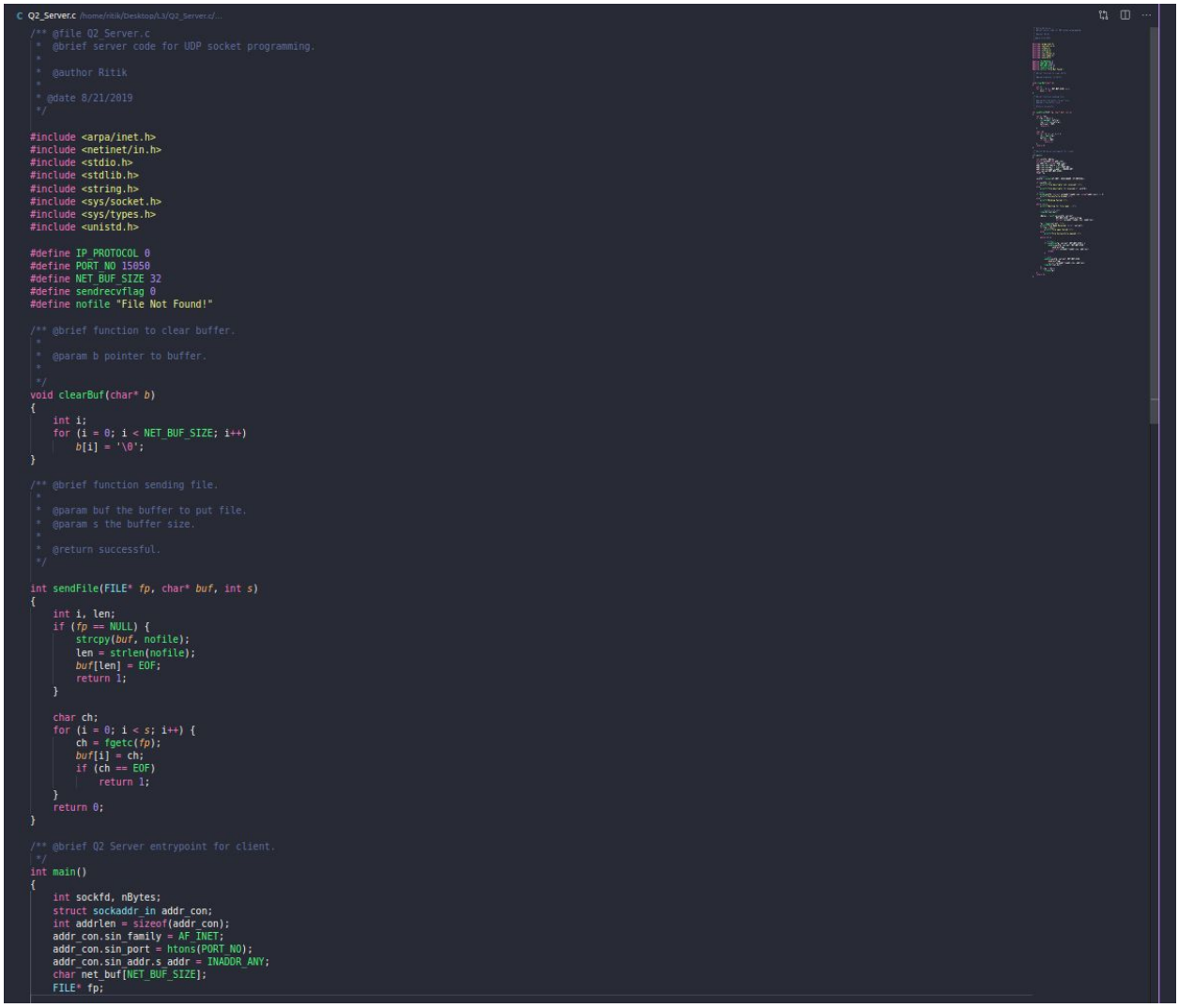
Algorithm

- Add the connection details like protocols, address, port.
- Create a socket, bind and wait for new connections
- Get the filename from the client
- If file is not present, send error
- Else send file in the buffer to the socket
- Close the socket

DataStructure

Sockaddr_in, char array, File

Code



```c
/** @file Q2_Server.c
 *  @brief server code for UDP socket programming.
 *
 *  @author Ritik
 *
 * @date 8/21/2019
 */

#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 PORT_NO 15050
#define NET_BUF_SIZE 32
#define sendrecvflag 0
#define nofile "File Not Found!"

/** @brief function to clear buffer.
 *
 *  @param b pointer to buffer.
 *
 */
void clearBuf(char* b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\0';
}

/** @brief function sending file.
 *
 *  @param buf the buffer to put file.
 *  @param s the buffer size.
 *
 *  @return successful.
 */

int sendFile(FILE* fp, char* buf, int s)
{
    int i, len;
    if (fp == NULL) {
        strcpy(buf, nofile);
        len = strlen(nofile);
        buf[len] = EOF;
        return 1;
    }

    char ch;
    for (i = 0; i < s; i++) {
        ch = fgetc(fp);
        buf[i] = ch;
        if (ch == EOF)
            return 1;
    }
    return 0;
}

/** @brief Q2 Server entrypoint for client.
 */
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;
```

```c
// socket()
sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);

if (sockfd < 0)
    printf("file descriptor not received!!\n");
else
    printf("file descriptor %d received\n", sockfd);

// bind()
if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(addr_con)) == 0)
    printf("Successfully binded!\n");
else
    printf("Binding Failed!\n");

while (1) {
    printf("Waiting for file name...\n");

    // receive file name
    clearBuf(net_buf);

    nBytes = recvfrom(sockfd, net_buf,
                      NET_BUF_SIZE, sendrecvflag,
                      (struct sockaddr*)&addr_con, &addrlen);

    fp = fopen(net_buf, "r");
    printf("File Name Received: %s\n", net_buf);
    if (fp == NULL)
        printf("File open failed!\n");
    else
        printf("File Successfully opened!\n");

    while (1) {

        // process
        if (sendFile(fp, net_buf, NET_BUF_SIZE)) {
            sendto(sockfd, net_buf, NET_BUF_SIZE,
                sendrecvflag,
                (struct sockaddr*)&addr_con, addrlen);
            break;
        }

        // send
        sendto(sockfd, net_buf, NET_BUF_SIZE,
            sendrecvflag,
            (struct sockaddr*)&addr_con, addrlen);
        clearBuf(net_buf);
    }
    if (fp != NULL)
        fclose(fp);
}
return 0;
}
```

Running Code

```
 ritik@rk-desktop  ~/Desktop/L3  gcc -o q2s Q2_Server.c
 ritik@rk-desktop  ~/Desktop/L3  ./q2s
file descriptor 3 received
Successfully binded!
Waiting for file name...
File Name Received: Q1.c
File Successfully opened!
Waiting for file name...
```

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 the File Transfer Protocol (FTP) for the same. Each link should have a different colour 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.

Algorithm

- Get ns object
- Set colour codes
- Open namtrace-file
- Get N, K value from user and assign to node array
- Assign node properties and link properties to create **star** topology
- Establish TCP agent and TCPSink agent and FTP application
- Start and stop at different times
- Finish and close operations

DataStructure

Simulator, file, Nodes, Nides array, Agent, Application, iterator

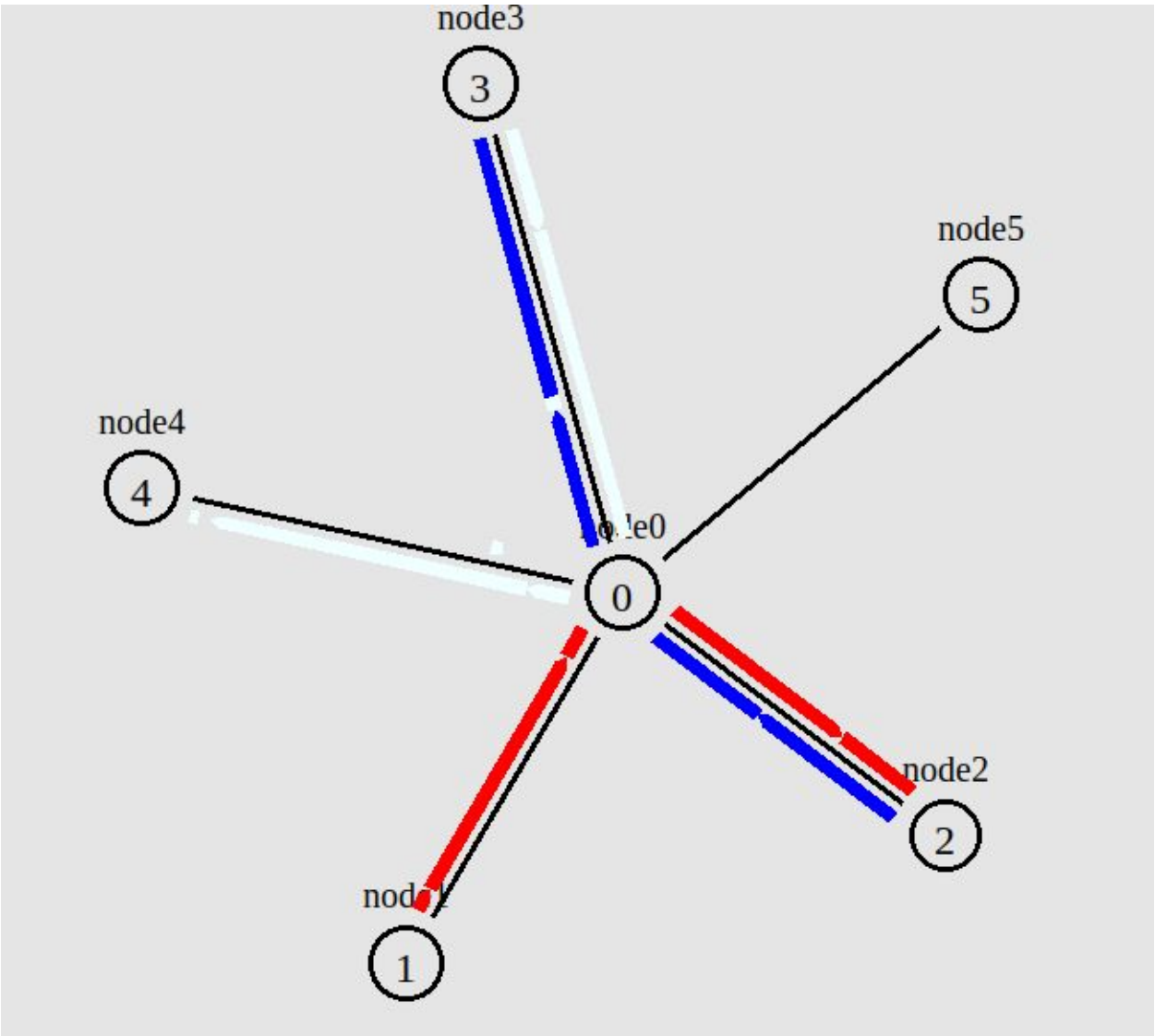
Code

Q3.tcl /home/ritik/Desktop/L3/Q3.tcl

```tcl
## \file Q3.tcl
# File Tcl file for Star topography.
# Author Ritik

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 3.nam w]
$ns namtrace-all $f

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

    exec nam 3.nam &
    exit 0
}
puts "Enter no. of Nodes: "
gets stdin N
set n(0) [$ns node]
$ns at 0.0 "$n(0) label node0"

for {set i 1} {$i < $N} {incr i} {
    set n($i) [$ns node]
    $ns at 0.0 "$n($i) label node$i"
    $ns duplex-link $n($i) $n(0) 1Mb 10ms DropTail
}
puts "Enter k: "
gets stdin k
for {set i 0} {$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
}
# $ns duplex-link $n0 $n1 1Mb 10ms DropTail

for {set i 0} {$i < $k} {incr i} {
    $ns at [expr ($i/5)] "$ftp($i) start"
    $ns at [expr ($i/5)+0.5] "$ftp($i) stop"
}
$ns at [expr ($k)+0.5] "finish"

$ns run
```

Running Code

```
ritik@rk-desktop ~/Desktop/L3 ns Q3.tcl
Enter no. of Nodes:
6
Enter k:
3
1
2
2
3
3
4
```



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 the File Transfer Protocol (FTP) for the same. Each link should have a different colour 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.

Algorithm

- Get ns object
- Set colour codes
- Open namtrace-file
- Get N, K value from user and assign to node array
- Assign node properties and link properties to create **ring** topology
- Establish TCP agent and TCPSink agent and FTP application
- Start and stop at different times
- Finish and close operations

DataStructure

Simulator, file, Nodes, Nides array, Agent, Application, iterator

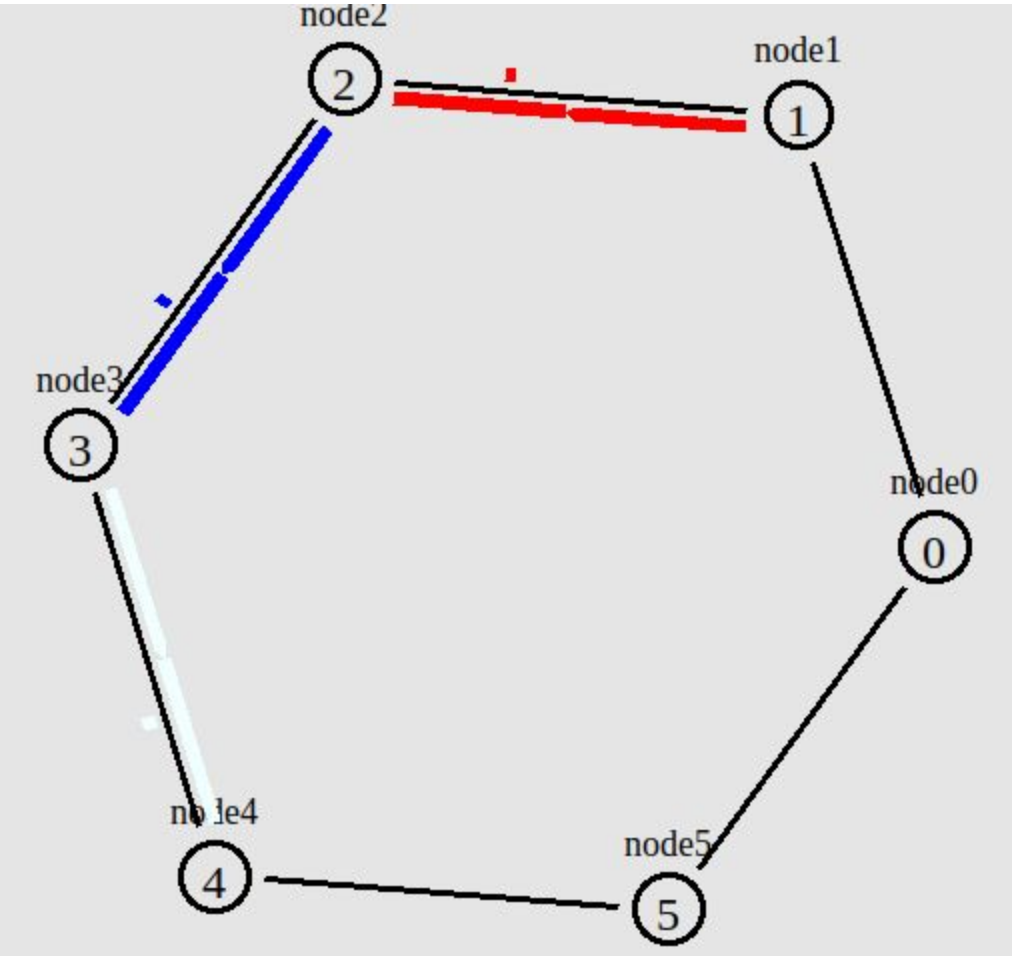
Code

Q4.tcl /home/ritik/Desktop/L3/Q4.tcl

```tcl
## \file Q3.tcl
# File Tcl file for Ring topography.
# Author Ritik

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 4.nam w]
$ns namtrace-all $f

set t [open 4.tr w]
$ns trace-all $t

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

    exec nam 4.nam &
    exit 0
}
puts "Enter no. of Nodes: "
gets stdin N
set n(0) [$ns node]
$ns at 0.0 "$n(0) label node0"
set y 0
for {set i 1} {$i < $N} {incr i} {
    set n($i) [$ns node]
    $ns at 0.0 "$n($i) label node$i"
    $ns duplex-link $n($y) $n($i) 1Mb 10ms DropTail
    set y $i
}
$ns duplex-link $n($y) $n(0) 1Mb 10ms DropTail
puts "Enter k: "
gets stdin k
for {set i 0} {$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/5)] "$ftp($i) start"
    $ns at [expr ($i/5)+0.5] "$ftp($i) stop"
}
$ns at [expr ($k/5)+0.5] "finish"

$ns run
```

Running Code

```
ritik@rk-desktop ~/Desktop/L3 ns Q4.tcl
Enter no. of Nodes:
6
Enter k:
3
1
2
2
3
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 the File Transfer Protocol (FTP) for the same. Each link should have a different colour 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.

Algorithm

- Get ns object
- Set colour codes
- Open namtrace-file
- Get N, K value from user and assign to node array
- Assign node properties and link properties to create **bus** topology
- Establish TCP agent and TCPSink agent and FTP application
- Start and stop at different times
- Finish and close operations

DataStructure

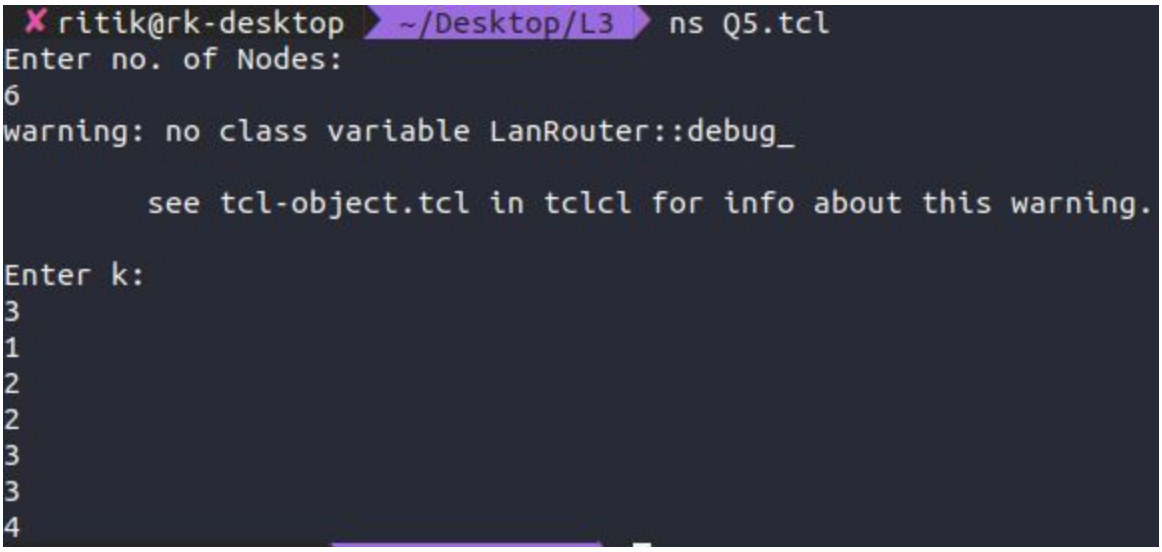
Simulator, file, Nodes, Nides array, Agent, Application, iterator

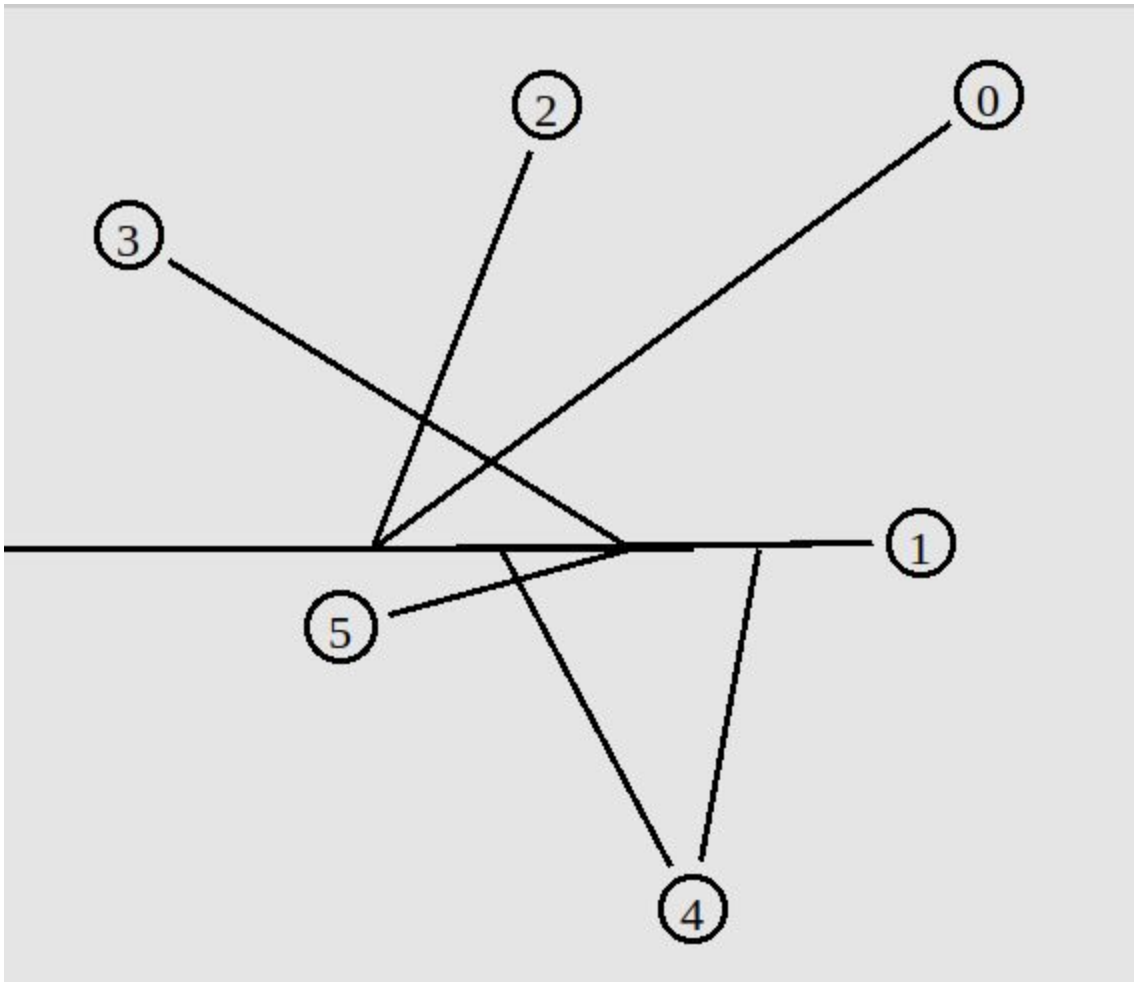
Code

Q5.tcl /home/ritik/Desktop/L3/Q5.tcl

```tcl
## \file Q3.tcl
# File Tcl file for Bus topography.
# Author Ritik

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]
$ns at 0.0 "$n(0) label node0"
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]
    $ns at 0.0 "$n($i) label node$i"
    # $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($i)" 1Mb 10ms
}
set lan0 [$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)+0.1] "$ftp($i) start"
    $ns at [expr ($i)+0.9] "$ftp($i) stop"
}
$ns at [expr ($k/10)+1.5] "finish"

$ns run
```

Running Code




2
0
3
1
5
4