



# Computer Network Lab(CSN-361)

Assignment - 3

22.08.2019

---

Shivansh Bindal

17115088

Computer Science & Engineering

3rd yr

## Problem Statement 1:

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

### Code

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

char findIPClass(char str[])
{
    char arr[4];
    int i = 0;
    while (str[i] != '.')
    {
        arr[i] = str[i];
        i++;
    }
    i--;

    int ip = 0, j = 1;
    while (i >= 0)
    {
        ip = ip + (str[i] - '0') * j;
        j = j * 10;
        i--;
    }
}
```

```
if (ip >= 1 && ip <= 126)
    return 'A';

else if (ip >= 128 && ip <= 191)
    return 'B';

else if (ip >= 192 && ip <= 223)
    return 'C';

else if (ip >= 224 && ip <= 239)
    return 'D';

else
    return 'E';
}

void separate(char str[], char ipClass)
{
    char network[12], host[12];
    for (int k = 0; k < 12; k++)
        network[k] = host[k] = '\\0';

    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);
}

else if (ipClass == 'B')
{
    int i = 0, j = 0, dotCount = 0;

    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);
}

else if (ipClass == 'C')
{
```

```
int i = 0, j = 0, dotCount = 0;

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);
}

else
    printf("In this Class, IP address is not"
           " divided into Network and Host ID\n");
}

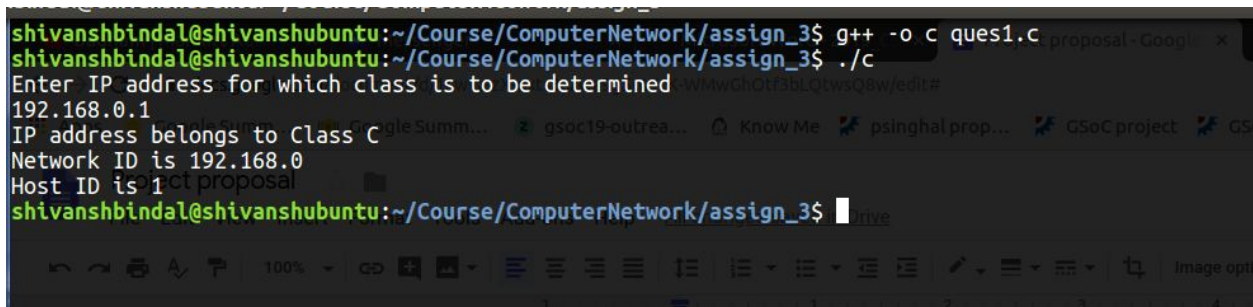
int main()
{
    char str[100];
    printf("Enter IP address for which class is to be determined \n");
    scanf("%s", str);
```

```

    char ipClass = findIPClass(str);
    printf("IP address belongs to Class %c\n",
           ipClass);
    separate(str, ipClass);
    return 0;
}

```

## Screenshot of running code



```

shivanshbindal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ g++ -o c ques1.c
shivanshbindal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ ./c
Enter IP address for which class is to be determined
192.168.0.1
IP address belongs to Class C
Network ID is 192.168.0
Host ID is 1
shivanshbindal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$

```

## Problem Statement 2:

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

### Code

#### Server

```

#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 cipherKey 'S'
#define sendrecvflag 0
#define nofile "File Not Found!"

void clearBuf(char *b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\\0';
}

char Cipher(char ch)
{
    return ch ^ cipherKey;
}

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

```
        for (i = 0; i <= len; i++)
            buf[i] = Cipher(buf[i]);
        return 1;
    }

    char ch, ch2;
    for (i = 0; i < s; i++)
    {
        ch = fgetc(fp);
        ch2 = Cipher(ch);
        buf[i] = ch2;
        if (ch == EOF)
            return 1;
    }
    return 0;
}

int main()
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    socklen_t 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 (sockfd < 0)
    printf("\nThe file descriptor is not received!!\n");
else
    printf("\nThe file descriptor %d is received\n", sockfd);

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

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

    clearBuf(net_buf);

    nBytes = recvfrom(sockfd, net_buf,
                      NET_BUF_SIZE, 0,
                      (struct sockaddr *)&addr_con, &addrlen);

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

    while (1)
```

```
{

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

    sendto(sockfd, net_buf, NET_BUF_SIZE,
            sendrecvflag,
            (struct sockaddr *)&addr_con, addrlen);
    clearBuf(net_buf);
}

if (fp != NULL)
    fclose(fp);
}

return 0;
}
```

## Client

```
#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"
#define PORT_NO 15050
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0

void clearBuf(char *b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\\0';
}

char Cipher(char ch)
{
    return ch ^ cipherKey;
}

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;
        else
            printf("%c", ch);
    }
    return 0;
}

int main()
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    socklen_t 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;

    sockfd = socket(AF_INET, SOCK_DGRAM,
                    IP_PROTOCOL);

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

    while (1)
```

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

    printf("\nData Received\n");

    while (1)
    {
        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-----\n");
}
return 0;
}
```

## Screenshot of running code

```

shivanshbinal@shivanshubuntu: ~/Course/ComputerNetwork/assign_3
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ ls
c client out.nam ques1.c ques2_client.c ques2_server.c ques3.tcl ques4.tcl server
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ ls
c client out.nam ques1.c ques2_client.c ques2_server.c ques3.tcl ques4.tcl server
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ g++ -o client ques2_client.c
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ ./client
The file descriptor 3 is received
Enter file name to receive: ques1.c
Data Received
#include <stdio.h>
#include <string.h>

char findIPClass(char str[])
{
    char arr[4];
    int i = 0;
    while (str[i] != '.')
    {
        arr[i] = str[i];
        i++;
    }
    if (i == 0)
    {
        return 'A';
    }
    else if (i == 1)
    {
        return 'B';
    }
    else if (i == 2)
    {
        return 'C';
    }
    else if (i == 3)
    {
        return 'D';
    }
    else
    {
        return 'E';
    }
}

int main()
{
    int ip = 0, j = 1;
    while (i >= 0)
    {
        ip = ip + (str[i] - '0') * j;
        j = j * 10;
        i--;
    }
    if (ip >= 1 && ip <= 126)
    {
        return 'A';
    }
    else if (ip >= 128 && ip <= 191)
    {
        return 'B';
    }
    else if (ip >= 192 && ip <= 223)
    {
        return 'C';
    }
    else if (ip >= 224 && ip <= 239)
    {
        return 'D';
    }
    else
    {
        return 'E';
    }
}

```

```

shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ ls
a.cpp Arduino arduino-1.8.9 arduino-1.8.9-linux64.tar.xz CGAL chromedriver chromedriver_linux64.zip cms2.0 cms-migration-scripts cms-shivansh Course Desktop docker_tutorials
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ cd Course/
shivanshbinal@shivanshubuntu:~/Course$ ls
Artificial Intelligence ComputerNetwork DAA MTN 105 operating_systems se
shivanshbinal@shivanshubuntu:~/Course$ cd ComputerNetwork/
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork$ ls
17115088_ShivanshBinal assign_3 Q1.cpp Q22.cpp Q3.cpp
17115088_ShivanshBinal.zip c Q2 Q2.cpp Q4
assign_2 Q1 Q22 Q3 Q4.cpp
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork$ cd assign_3/
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ ls
c client out.nam ques2_client.c ques3.tcl server
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ g++ -o server ques2_server.c ques3.tcl ques4.tcl
shivanshbinal@shivanshubuntu:~/Course/ComputerNetwork/assign_3$ ./server
The file descriptor 3 is received
Successfully binded!
Waiting for file name...
File Name Received: ques1.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 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.

#### Code

```
set input [gets stdin]
scan $input "%d %d" N k

set ns [new Simulator]

$ns rtproto DV

set nf [open out.nam w]
$ns namtrace-all $nf

proc finish {} {
```

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

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

for {set i 1} {$i < $N} {incr i} {
    $ns duplex-link $node($i) $node(0) 1Mb 10ms DropTail
}

set colors(0) Yellow
set colors(1) Green
set colors(2) Orange
set colors(3) Pink
set colors(4) Red
set colors(5) Blue

for {set i 0} {$i < $k} {incr i} {
    set input [gets stdin]
    scan $input "%d %d" u v

    set tcp_con [new Agent/TCP]
    $ns attach-agent $node($u) $tcp_con
```



```
$tcp_con set class_ $i

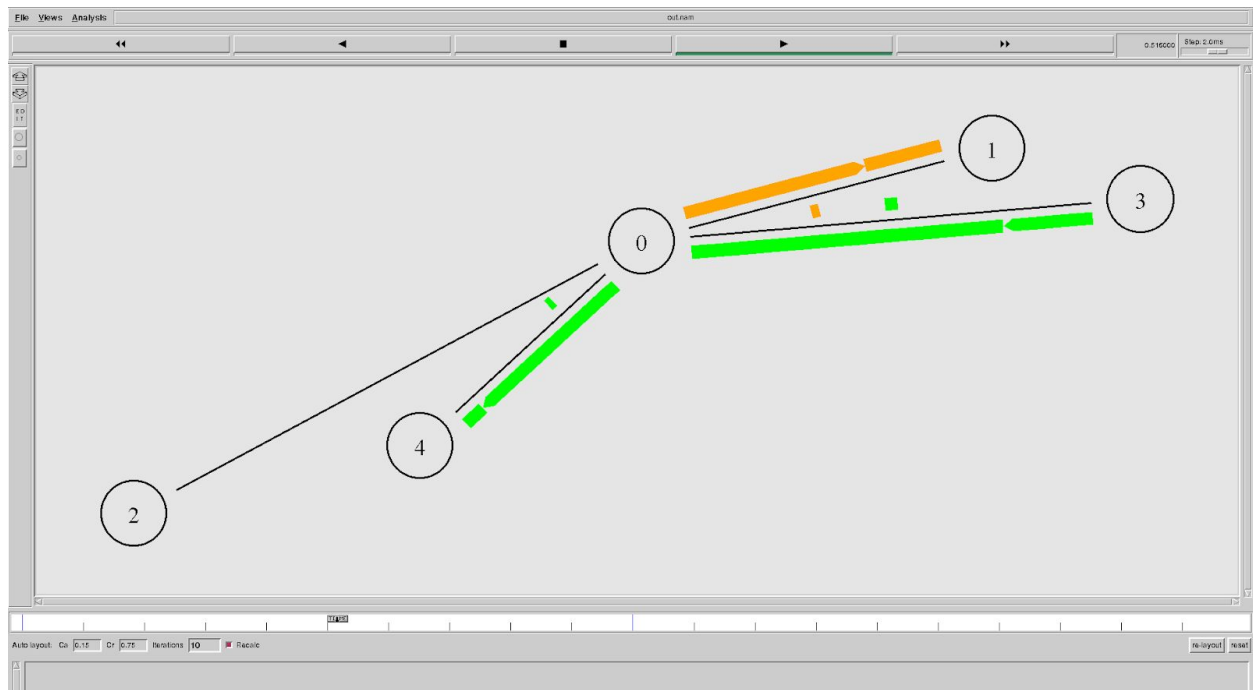
set sink_node [new Agent/TCPSink]
$ns attach-agent $node($v) $sink_node
$ns connect $tcp_con $sink_node

$ns color $i $colors([expr ($i) % 6])
$tcp_con set fid_ $i

set ftp_con [new Application/FTP]
$ftp_con attach-agent $tcp_con
$ns at 0.1 "$ftp_con start"
$ns at 1.5 "$ftp_con stop"
}

$ns at 2.0 "finish"
$ns run
```

## Screenshot of running code



### 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.**

### Code

```
set input [gets stdin]
scan $input "%d %d" N k

set ns [new Simulator]

$ns rtproto DV

set nf [open out.nam w]
$ns namtrace-all $nf

proc finish {} {
```

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

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

for {set i 0} {$i < $N} {incr i} {
    $ns duplex-link $node($i) $node([expr ($i + 1) % $N]) 512Kb 5ms
    DropTail
}

set colors(0) Yellow
set colors(1) Green
set colors(2) Orange
set colors(3) Pink
set colors(4) Red
set colors(5) Blue

for {set i 0} {$i < $k} {incr i} {
    set input [gets stdin]
    scan $input "%d %d" u v

    set tcp_con [new Agent/TCP]
    $ns attach-agent $node($u) $tcp_con
    $tcp_con set class_ $i
}
```

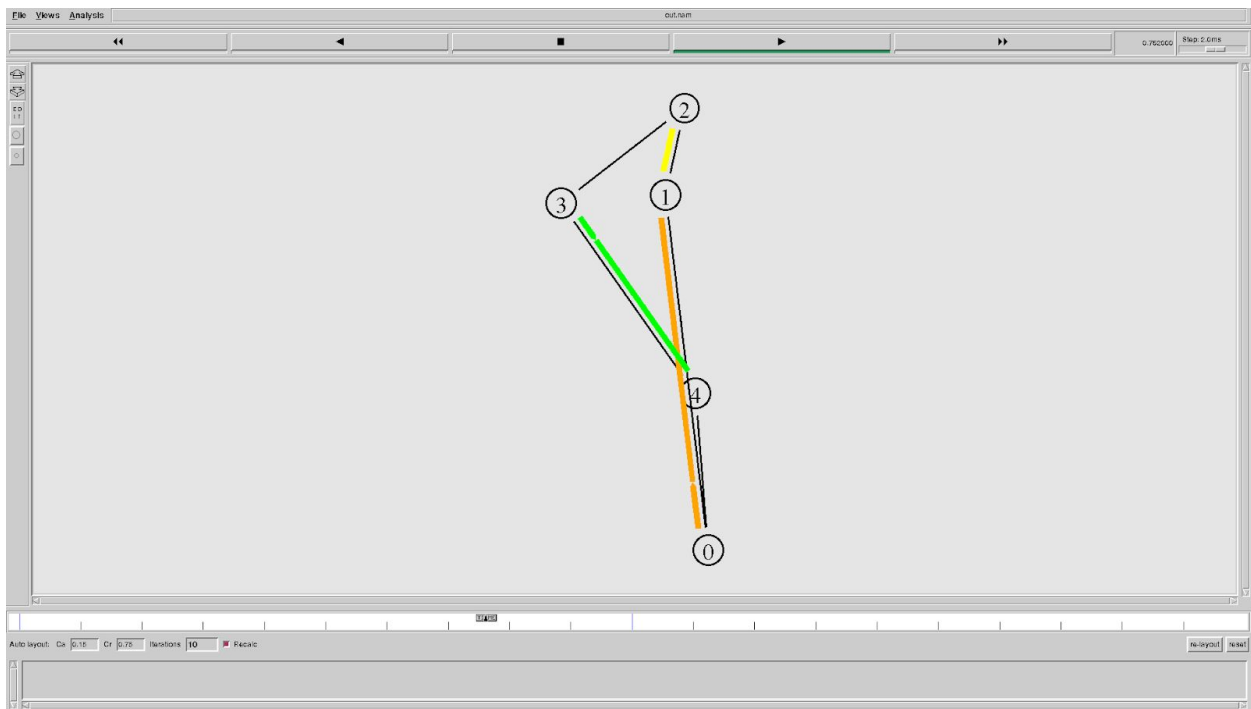
```
set sink_node [new Agent/TCPSink]
$ns attach-agent $node($v) $sink_node
$ns connect $tcp_con $sink_node

$ns color $i $colors([expr ($i) % 6])
$tcp_con set fid_ $i

set ftp_con [new Application/FTP]
$ftp_con attach-agent $tcp_con
$ns at 0.1 "$ftp_con start"
$ns at 1.5 "$ftp_con stop"
}

$ns at 2.0 "finish"
$ns run
```

Screenshot of running code



### 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.

## Code

```
set input [gets stdin]
scan $input "%d %d" N k

set ns [new Simulator]

# $ns rtproto DV

set nf [open out.nam w]
$ns namtrace-all $nf

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

set node(0) [$ns node]
set y "$node(0)"
for {set i 1} {$i < $N} {incr i} {
    set node($i) [$ns node]
    append y " "
    append y "$node($i)"
}

$ns make-lan $y 0.5Mb 40ms LL Queue/DropTail Mac/802_3
```

```
set colors(0) Yellow
set colors(1) Green
set colors(2) Orange
set colors(3) Pink
set colors(4) Red
set colors(5) Blue

for {set i 0} {$i < $k} {incr i} {
    set input [gets stdin]
    scan $input "%d %d" u v

    set tcp_con [new Agent/TCP]
    $ns attach-agent $node($u) $tcp_con
    $tcp_con set class_ $i

    set sink_node [new Agent/TCPSink]
    $ns attach-agent $node($v) $sink_node
    $ns connect $tcp_con $sink_node

    $ns color $i $colors([expr ($i) % 6])
    $tcp_con set fid_ $i

    set ftp_con [new Application/FTP]
    $ftp_con attach-agent $tcp_con
    $ns at 0.1 "$ftp_con start"
    $ns at 1.5 "$ftp_con stop"
}
```



```
$ns at 2.0 "finish"
```

```
$ns run
```

## Screenshot of running code

