

CSN-361: Computer Networks Laboratory

Lab Assignment (3)

Name : David Gokimmung
Enrollment no. : 17114023

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

Algorithm:

1. For determining the class: The idea is to check 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.

```
david@david:~/17114023$ gcc 1.c -o 1
david@david:~/17114023$ ./1
125.1.1.1
Given IP address belongs to Class A
Network ID is 125
Host ID is 1.1.1

david@david:~/17114023$
```

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

Algorithm:

1. The server starts and waits for filename.
2. The client sends a filename.
3. The server receives filename. If the 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.

The image shows two side-by-side code editors and their terminal outputs. The left editor, titled 'C server.c', contains the server-side code. It includes headers for `<arpa/inet.h>`, `<netinet/in.h>`, `<stdio.h>`, `<stdlib.h>`, `<string.h>`, `<sys/socket.h>`, `<sys/types.h>`, and `<unistd.h>`. It defines `IP_PROTOCOL 0`, `PORT_NO 15050`, `NET_BUF_SIZE 32`, `cipherKey 'S'`, `sendrecvflag 0`, and `nofile "File Not Found!"`. Functions include `clearBuf` to zero out a buffer, `Cipher` to XOR a character with the key, and `main` which binds to `127.0.0.1:15050`, receives a filename, reads the file, and sends encrypted buffers until EOF. The right editor, titled 'C client.c', contains the client-side code. It includes the same headers and defines `IP_PROTOCOL 0`, `IP_ADDRESS "127.0.0.1" // localhost`, `PORT_NO 15050`, `NET_BUF_SIZE 32`, `cipherKey 'S'`, and `sendrecvflag 0`. Functions include `clearBuf`, `Cipher`, and `main` which connects to the server, sends the filename, receives encrypted buffers, and prints the decrypted output. The terminal for the server shows it successfully binds and receives the filename 'filename.txt'. The terminal for the client shows it connects, sends the filename, and receives the encrypted data 'Hello World!'.

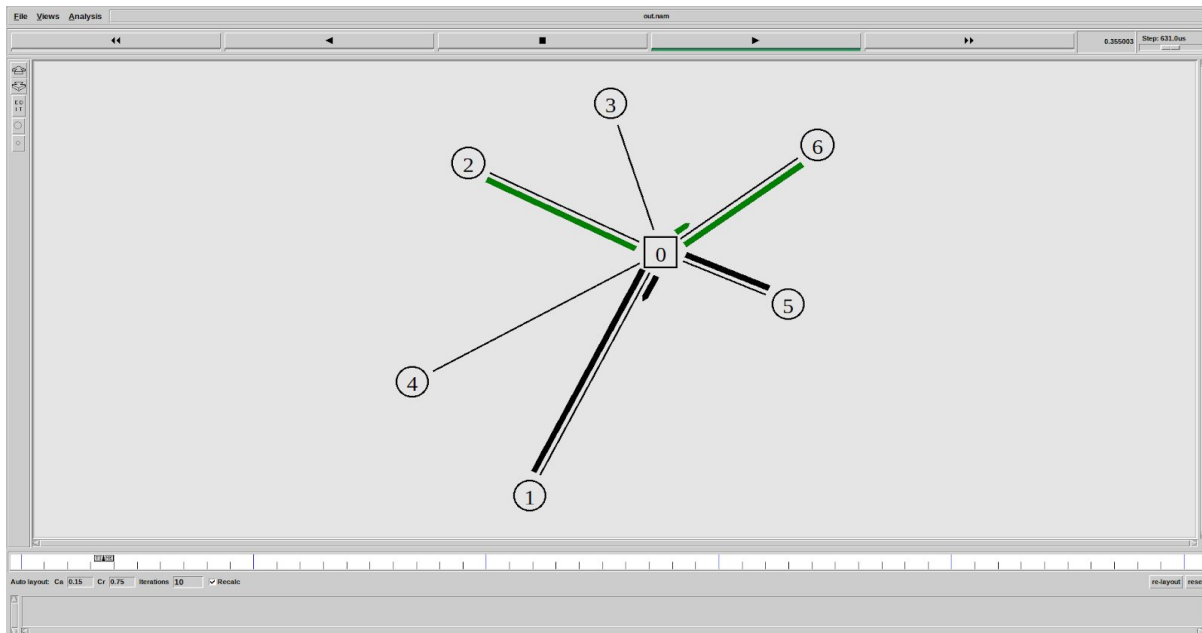
```
C server.c
1 // server code for UDP socket programming
2 #include <arpa/inet.h>
3 #include <netinet/in.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <sys/socket.h>
8 #include <sys/types.h>
9 #include <unistd.h>
10
11 #define IP_PROTOCOL 0
12 #define PORT_NO 15050
13 #define NET_BUF_SIZE 32
14 #define cipherKey 'S'
15 #define sendrecvflag 0
16 #define nofile "File Not Found!"
17
18 // function to clear buffer
19 void clearBuf(char* b)
20 {
21     int i;
22     for (i = 0; i < NET_BUF_SIZE; i++)
23         b[i] = '\0';
24 }
25
26 // function to encrypt
27 char Cipher(char ch)
28 {
29     return ch ^ cipherKey;
30 }
31
32 int main()
33 {
34     struct sockaddr_in server, client;
35     int sockfd, n;
36     char buf[NET_BUF_SIZE];
37     char filename[100];
38     FILE* f;
39
40     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
41     if (sockfd < 0) {
42         perror("socket");
43         return 1;
44     }
45
46     server.sin_family = AF_INET;
47     server.sin_port = htons(PORT_NO);
48     server.sin_addr.s_addr = INADDR_ANY;
49
50     if (bind(sockfd, (struct sockaddr*)&server, sizeof(server)) < 0) {
51         perror("bind");
52         return 1;
53     }
54
55     printf("file descriptor 3 received\n");
56     printf("Successfully binded!\n");
57     printf("Waiting for file name...\n");
58
59     n = recvfrom(sockfd, filename, sizeof(filename), 0, (struct sockaddr*)&client, NULL);
60     if (n < 0) {
61         perror("recvfrom");
62         return 1;
63     }
64
65     printf("File Name Received: %s\n", filename);
66     f = fopen(filename, "r");
67     if (f == NULL) {
68         printf("File Successfully opened!\n");
69         return 1;
70     }
71
72     printf("Waiting for file name...\n");
73     return 0;
74 }
```

```
C client.c
1 // client code for UDP socket programming
2 #include <arpa/inet.h>
3 #include <netinet/in.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <sys/socket.h>
8 #include <sys/types.h>
9 #include <unistd.h>
10
11 #define IP_PROTOCOL 0
12 #define IP_ADDRESS "127.0.0.1" // localhost
13 #define PORT_NO 15050
14 #define NET_BUF_SIZE 32
15 #define cipherKey 'S'
16 #define sendrecvflag 0
17
18 // function to clear buffer
19 void clearBuf(char* b)
20 {
21     int i;
22     for (i = 0; i < NET_BUF_SIZE; i++)
23         b[i] = '\0';
24 }
25
26 // function for decryption
27 char Cipher(char ch)
28 {
29     return ch ^ cipherKey;
30 }
31
32 int main()
33 {
34     struct sockaddr_in server, client;
35     int sockfd, n;
36     char buf[NET_BUF_SIZE];
37     char filename[100];
38
39     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
40     if (sockfd < 0) {
41         perror("socket");
42         return 1;
43     }
44
45     client.sin_family = AF_INET;
46     client.sin_port = htons(PORT_NO);
47     inet_aton(IP_ADDRESS, &client.sin_addr);
48
49     if (connect(sockfd, (struct sockaddr*)&client, sizeof(client)) < 0) {
50         perror("connect");
51         return 1;
52     }
53
54     printf("file descriptor 3 received\n");
55     printf("Please enter file name to receive:\n");
56     scanf("%s", filename);
57
58     printf("-----Data Received-----\n");
59     n = recv(sockfd, buf, sizeof(buf), 0);
60     if (n < 0) {
61         perror("recv");
62         return 1;
63     }
64
65     printf("-----\n");
66     printf("Please enter file name to receive:\n");
67     return 0;
68 }
```

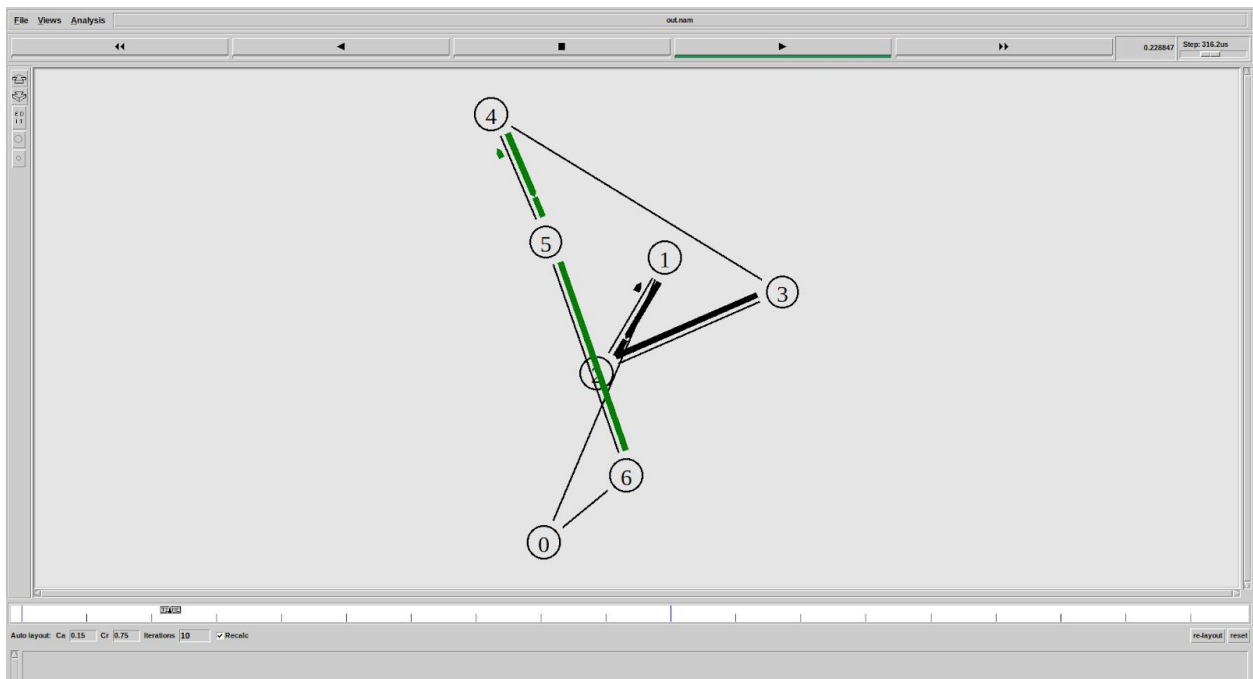
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
david@david:~/17114023$ gcc server.c -o server
david@david:~/17114023$ ./server
file descriptor 3 received
Successfully binded!
Waiting for file name...
File Name Received: filename.txt
File Successfully opened!
Waiting for file name...
[]

david@david:~/17114023$ gcc client.c -o client
david@david:~/17114023$ ./client
file descriptor 3 received
Please enter file name to receive:
filename.txt
-----Data Received-----
Hello World!
-----
Please enter file name to receive:
[]
```

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.



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.



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.

