UCS1511 – Networks Laboratory
University Practical Examination

Shivanirudh S G
CSE – C
185001146

1.Write a socket program for simple client server connectivity using TCP.

- Client will send an ip address (classful addressing). Server will find, to which class it belongs to for classful addressing, the default mask value and sends it back to client.
- Client will send an ip address with prefix length (classless addressing). Server will find the first address and last address of the network.
- Server will find how many host machines can be connected to this ip address.

Program:

Functions:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<unistd.h>
#include<netinet/in.h>

//Return 1 for classful, 0 for classless addressing
int find_cat(char *addr){
for(int i = 0; addr[i]; i++){
if(addr[i] == '/')
return 0;
}
return 1;
}

int checkIP(char *ip){

char *addr = (char*)calloc(100, sizeof(char));
strcpy(addr, ip);


int cat = find_cat(addr);
if(!cat){
for(int i = 0; addr[i]; i++){
if(addr[i] == '/'){
```

```
addr[i] = '\0';
break;
}
}
}

int count = 0;
for(int i = 0; addr[i]; i++){
if(addr[i] == '.')
count++;
}
if(count != 3)
return 0;

char *token = strtok(addr, ".");
while(token){
int val = atoi(token);
if(val < 0 || val > 255)
return 0;
token = strtok(NULL, ".");
}
return 1;
}

//Find class
char* find_class(char *addr){
char *copy = (char*)calloc(100, sizeof(char));
strcpy(copy, addr);

char *class_name = (char*)calloc(100, sizeof(char));
char *token = strtok(addr, ".");
if(token){
int val = atoi(token);
if(val >= 0 && val <= 127)
strcpy(class_name, "A");
else if(val <= 191)
strcpy(class_name, "B");
else if(val <= 223)
strcpy(class_name, "C");
else if(val <= 239)
strcpy(class_name, "D");
else
strcpy(class_name, "E");
}
return class_name;
}

//Find default mask
char* find_def_mask(char *class_name, char *addr){
char *mask = (char*)calloc(100, sizeof(char));
if(!strcmp(class_name, "A"))
```

```
strcpy(mask, "255.0.0.0");
else if(!strcmp(class_name, "B"))
strcpy(mask, "255.255.0.0");
else if(!strcmp(class_name, "C"))
strcpy(mask, "255.255.255.0");
else{
strcpy(mask, "\0");
}
}

/*
____________________________________________________________________________________________________________________________________
*/
int power(int num, int exp){
int pdt = 1;
while(exp--){
pdt *= num;
}
return pdt;
}

void strrev(char* s){
int len = strlen(s);
int i = 0;
int j = len-1;
while(i<j){
char tmp = s[i];
s[i] = s[j];
s[j] = tmp;
i++;
j--;
}
}

int conv_to_dec(char *number){
int num = 0;
char *copy = (char*)calloc(100, sizeof(char));
strcpy(copy, number);
for(int i = 0; copy[i]; i++){
if(copy[i] == '1')
num += power(2, i);
}
return num;
}

char* conv_to_bin(int number){
char *bin = (char*)calloc(100, sizeof(char));
int n = number;
int pos=0;
while(n>0){
```

```
bin[pos++] = ('0'+(n%2));
n /= 2;
}
bin[pos] = '\0';
strrev(bin);
return bin;
}
/*
________________________________________________________________________
_____________________________
*/

char* find_first(char *addr){
char *copy = (char*)calloc(100, sizeof(char));
strcpy(copy, addr);
int host_bits = 0;

char *token = strtok(copy, "/");
while(token){
host_bits = atoi(token);
token = strtok(NULL, "/");
}

for(int i = 0; copy[i]; i++){
if(copy[i] == '/'){
copy[i] = '\0';
break;
}
}
int ip[4];
int ctr = 0;
token = strtok(copy, ".");
while(token){
ip[ctr++] = atoi(token);
token = strtok(NULL, ".");
}
int q, r, num;
q = host_bits/8;
r = host_bits%8;
for(int i = 0; i<4;i++){
if(i == q){
num = ip[i];
num = num >> (8-r);
num = num << (8-r);
ip[i] = num;
}
else if(i>q)
ip[i] = 0;
}
char *v = (char*)calloc(100, sizeof(char));
char *first = (char*)calloc(100, sizeof(char));
```

```
for(int i = 0;i < 4; i++){
sprintf(v, "%d", ip[i]);
strcat(first, v);strcat(first, ".");
}
return first;
}

char* find_last(char *addr){
char *copy = (char*)calloc(100, sizeof(char));
strcpy(copy, addr);
int host_bits = 0;
char *token = strtok(copy, "/");
while(token){
host_bits = atoi(token);
token = strtok(NULL, "/");
}
for(int i = 0; copy[i]; i++){
if(copy[i] == '/'){
copy[i] = '\0';
break;
}
}
int ip[4];
int ctr = 0;
token = strtok(copy, ".");
while(token){
ip[ctr++] = atoi(token);
token = strtok(NULL, ".");
}
int q, r, num;
q = host_bits/8;
r = host_bits%8;
for(int i = 0; i<4;i++){
if(i>q){
ip[i] = 255;
}
else if(i == q){
num = ip[i];
num = num >> (8-r);
num = num << (8-r);
ip[i] = num + power(2,(8-r)) - 1;
}
}
char *v = (char*)calloc(100, sizeof(char));
char *last = (char*)calloc(100, sizeof(char));
for(int i = 0;i < 4; i++){
sprintf(v, "%d", ip[i]);
strcat(last, v);strcat(last, ".");
}
return last;
}
```

```
char* find_hosts(char *addr){
char *copy = (char*)calloc(100, sizeof(char));
strcpy(copy, addr);
int host_bits = 0;
char *token = strtok(copy, "/");
while(token){
host_bits = atoi(token);
token = strtok(NULL, "/");
}
for(int i = 0; copy[i]; i++){
if(copy[i] == '/'){
copy[i] = '\0';
break;
}
}
host_bits = 32 - host_bits;
char *host_count = (char*)calloc(100, sizeof(char));
int count = power(2, host_bits);
count -= 2;
sprintf(host_count, "%d", count);
return host_count;
}
```

Server:

```
#include "Addressing.h"

int main(int argc, char **argv){
if(argc > 1){
perror("\nError: No arguments needed for server");
exit(1);
}
struct sockaddr_in server, client;
char buffer[1024];

int sockfd = socket(AF_INET, SOCK_STREAM, 0);
if(sockfd < 0){
perror("\nError: Socket");
exit(1);
}

bzero(&server, sizeof(server));

server.sin_family = AF_INET;
server.sin_port = htons(7002);
server.sin_addr.s_addr = INADDR_ANY;

if(bind(sockfd, (struct sockaddr *)&server, sizeof(server)) < 0){
perror("\nError:Bind");
exit(1);
}

listen(sockfd, 2);

int len = sizeof(client);

int newfd = accept(sockfd, (struct sockaddr *)&client, &len);
while(1){
bzero(&buffer, sizeof(buffer));

read(newfd, buffer, sizeof(buffer));
if(!strcmp(buffer, "end")){
break;
}
printf("\n-----------------------------------------------\n");
printf("\nReceived IP address: %s\n", buffer);

int cat = find_cat(buffer);

//Classful addressing
if(cat){
char *class_name = (char*)calloc(100, sizeof(char));
strcpy(class_name, find_class(buffer));
char *default_mask = (char*)calloc(100, sizeof(buffer));
strcpy(default_mask, find_def_mask(class_name, buffer));

bzero(&buffer, sizeof(buffer));
strcat(buffer, "1");strcat(buffer, " ");
strcat(buffer, class_name);strcat(buffer, " ");
strcat(buffer, default_mask);
}
//Classless addressing
else{
char *first_addr = (char*)calloc(100, sizeof(char));
char *last_addr = (char*)calloc(100, sizeof(char));
strcpy(first_addr, find_first(buffer));
strcpy(last_addr, find_last(buffer));
char *hosts_count = (char*)calloc(100, sizeof(char));
strcpy(hosts_count, find_hosts(buffer));

bzero(&buffer, sizeof(buffer));
strcat(buffer, "0");strcat(buffer, " ");
strcat(buffer, first_addr);strcat(buffer, " ");
strcat(buffer, last_addr);strcat(buffer, " ");
strcat(buffer, hosts_count);strcat(buffer, " ");
}
printf("\nMessage to be sent: %s\n", buffer);
write(newfd, buffer, sizeof(buffer));
printf("\n-----------------------------------------------\n");
}
```

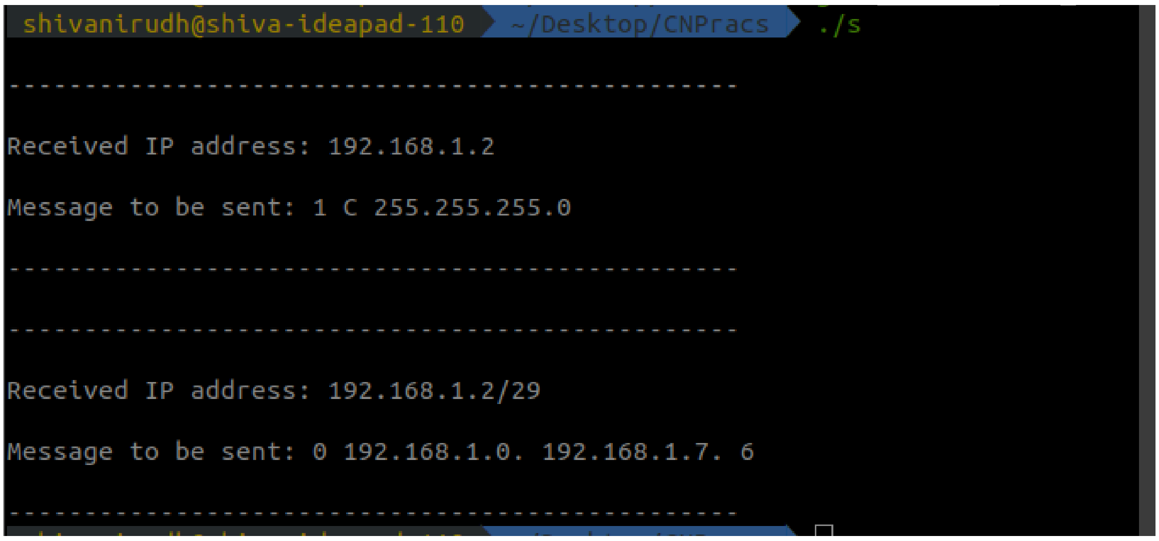
```
close(newfd);
close(sockfd);

}
```

Client:

```
#include "Addressing.h"

int main(int argc, char **argv){
if(argc != 2){
perror("\nError: Server IP address needed for client");
exit(1);
}
struct sockaddr_in server;
char buffer[1024];

int sockfd = socket(AF_INET, SOCK_STREAM, 0);
if(sockfd < 0){
perror("\nError: Socket");
exit(1);
}

bzero(&server, sizeof(server));

server.sin_family = AF_INET;
server.sin_port = htons(7002);
server.sin_addr.s_addr = inet_addr(argv[1]);

connect(sockfd, (struct sockaddr*)&server, sizeof(server));
while(1){
printf("\n----------------------------------------------\n");
bzero(&buffer, sizeof(buffer));
printf("\nEnter an IP address: ");scanf(" %[^\n]", buffer);
if(!strcmp(buffer, "end")){
write(sockfd, buffer, sizeof(buffer));
break;
}
while(!checkIP(buffer)){
printf("\nInvalid IP address. Please re-enter: ");
scanf(" %[^\n]", buffer);
}
write(sockfd, buffer, sizeof(buffer));
bzero(&buffer, sizeof(buffer));

read(sockfd, buffer, sizeof(buffer));
char *token = strtok(buffer, " ");
while(token){
if(!strcmp(token, "1")){
token = strtok(NULL, " ");
printf("\nClass: %s", token);
token = strtok(NULL, " ");
if(token[0])
printf("\nDefault mask: %s\n", token);
else
printf("\nClasses D and E do not have default masks.\n");
token = strtok(NULL, " ");
}
else{
token = strtok(NULL, " ");
printf("\nFirst address: %s", token);
token = strtok(NULL, " ");
printf("\nLast address: %s\n", token);
token = strtok(NULL, " ");
printf("\nNumber of hosts: %s\n", token);
token = strtok(NULL, " ");
}
}
printf("\n-------------------------------------------------\n");
}
close(sockfd);
}
```

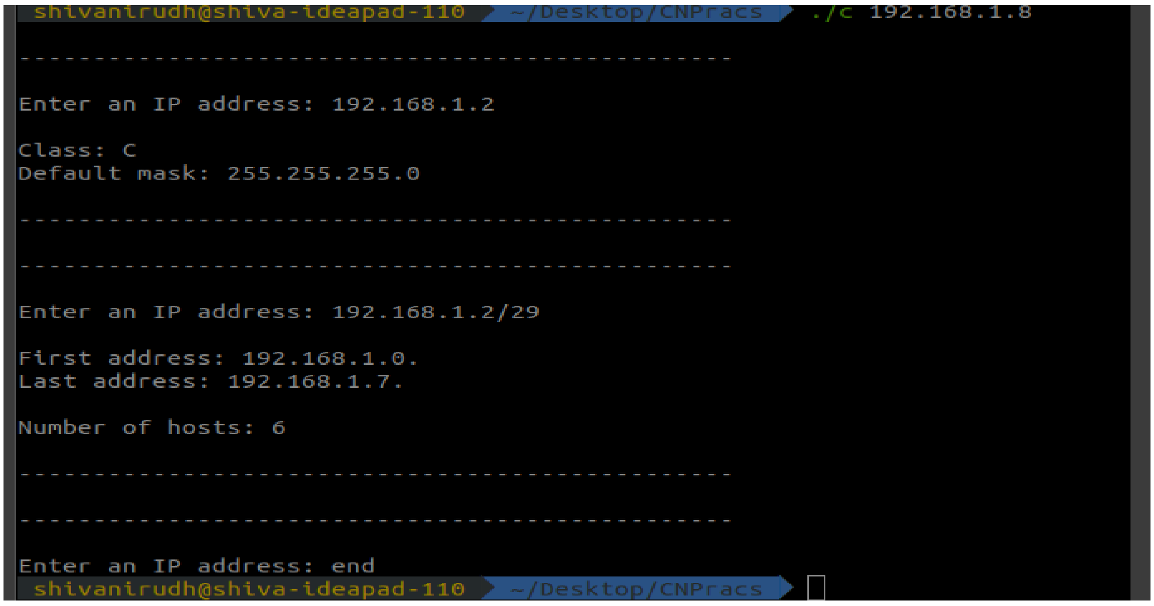
Output:

Server:



```
shivanirudh@shiva-ideapad-110  ~/Desktop/CNPracs  ./s

------------------------------------------------

Received IP address: 192.168.1.2

Message to be sent: 1 C 255.255.255.0

------------------------------------------------


------------------------------------------------

Received IP address: 192.168.1.2/29

Message to be sent: 0 192.168.1.0. 192.168.1.7. 6

------------------------------------------------
```

Client:



```
 shivanirudh@shiva-ideapad-110  ~/Desktop/CNPracs  ./c 192.168.1.8

-----------------------------------------------

Enter an IP address: 192.168.1.2

Class: C
Default mask: 255.255.255.0

-----------------------------------------------

-----------------------------------------------

Enter an IP address: 192.168.1.2/29

First address: 192.168.1.0.
Last address: 192.168.1.7.

Number of hosts: 6

-----------------------------------------------

-----------------------------------------------

Enter an IP address: end
 shivanirudh@shiva-ideapad-110  ~/Desktop/CNPracs 
```

Result:

A TCP server client implementation is used to identify mask and class for a classful IP address, and the first, and last addresses, number of hosts in a classless IP address.