

Assignment 2
Of
Network & Distributed System Lab (CS2051)
Masters of Technology in Computer Science And Engineering

submitted to
Dr Sujoy Saha
Assistant Professor
&
Dr Suvrojit Das
Associate Professor
Dept. of CSE



National Institute of Technology, Durgapur

submitted by
Arghya Bandyopadhyay
RollNo. 20CS4103

19 June 2021

1. Write TCP and UDP Chat Program.

Answer.

```
1 //This is the Server side implementation of TCP Chat
2
3 #include<string.h>
4 #include<sys/socket.h>
5 #include<netinet/in.h>
6 #include<unistd.h>
7 #include<stdio.h>
8 #include <stdbool.h>
9
10
11 int TCP_ChatServer(int client_desc)
12 {
13     const int BUFFER_SIZE = 4096;
14     char msg[BUFFER_SIZE];
15     int msg_len = 0;
16
17     while((msg_len = read(client_desc, msg, BUFFER_SIZE)) != 0)
18     {
19         printf("%s %d %s","[User",client_desc,"] Msg Received :");
20         fflush(stdout);
21
22         write(fileno(stdout), msg, msg_len);
23
24         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
25             return 0;
26
27         printf("%s %d %s","[User",client_desc,"] Enter Response :");
28         fflush(stdout);
29
30
31         msg_len = read(fileno(stdin), msg, BUFFER_SIZE);
32
33         write(client_desc, msg, msg_len);
34
35         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
36             return 0;
37     }
38 }
39
40 return 0;
41 }
42
43 int main()
44 {
45     //Create Socket
46     int server_desc = socket(AF_INET,SOCK_STREAM,0);
47
48     //Create and Fill Address Structure for this Server
49     struct sockaddr_in server_addr;
```

```

50 server_addr.sin_family      = AF_INET;      //Address Family (AF_INET, AF_INET6, AF_LOCAL, ...)
51     server_addr.sin_addr.s_addr = INADDR_ANY; //Internet Address (INADDR_ANY-> Accept connection at any IP Address)
52     server_addr.sin_port      = htons(9000); //Port Number (htons -> h.HOST t.TO n.NETWORK s.SHORT , Ensures proper byte ordering)
53
54 //Bind Socket Descriptor and Address Structure together
55 int result = bind(server_desc, (struct sockaddr*) &server_addr, sizeof(server_addr));
56
57 //Start Listioning (Tell kernel to accept connections directed towards this socket) (Puts socket into passive mode)
58 listen(server_desc,4);
59
60
61
62
63 //Server Loop
64 bool RunServer = true;
65 while(RunServer)
66 {
67     //Accept a Connection (Puts process in sleep mode if Connection Queue is Empty)
68     int client_desc;
69     client_desc = accept(server_desc,NULL,NULL);          //Listening Socket
70
71     //Create Child Process to handle connection
72     int pid = fork();
73
74     if(pid > 0)                                           //Parent Process
75     {
76         //Close Client Socket
77         close(client_desc);
78         continue;
79     }
80     else
81     if(pid == 0)                                         //Child Process
82     {
83         //Close Listening Socket
84         close(server_desc);
85
86         TCP_ChatServer(client_desc);
87
88         //Close Connection
89         close(client_desc);
90         break;                                           //Work Done! Exit Child Process.
91     }
92     else
93     {
94         printf("%s ", "fork() Error!!!");
95         break;
96     }
97 }
98
99
100 return 0;
101 }

```

```

1 //This is the client side implementation of TCP Chat
2
3 #include<arpa/inet.h>
4 #include<sys/socket.h>
5 #include<sys/wait.h>
6 #include<netinet/in.h>
7 #include <stdbool.h>
8 #include<string.h>
9 #include<time.h>
10 #include<stdio.h>
11 #include<unistd.h>
12 #include<stdlib.h>
13 #include<string.h>
14 #include<signal.h>
15 #include<errno.h>
16
17
18 int TCP_ChatClient(int server_desc)
19 {
20     const int BUFFER_SIZE = 4096;
21     char msg[BUFFER_SIZE];
22     int len = 0;
23
24     while(true)
25     {
26         printf("%s ", "Input:"); fflush(stdout);
27         len = read(fileno(stdin), msg, BUFFER_SIZE);
28
29         if(len == 0) //EOF
30             return 0;
31
32         len = write(server_desc, msg, len);
33
34         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
35             return 0;
36
37         len = read(server_desc, msg, BUFFER_SIZE);
38
39         printf("%s ", "Response from server :"); fflush(stdout);
40         len = write(fileno(stdout), msg, len);
41
42         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
43             return 0;
44     }
45     return 0;
46 }
47
48
49 int main()
50 {
51     int sock = socket(AF_INET, SOCK_STREAM, 0);
52
53     struct sockaddr_in server_addr;

```

```
54  server_addr.sin_family      = AF_INET;
55  server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
56  server_addr.sin_port       = htons(9000);
57
58  int result = connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
59
60  TCP_Client(sock);
61
62
63  close(sock);
64  return 0;
65 }
```

```
arghya@Delton: /media/arghya/Development/Github...  
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/tcp$ ./tcpserver  
[User 4 ] Msg Received :hello world  
[User 4 ] Enter Response :hello world too  
[User 4 ] Msg Received :bye  
□
```

(a) TCPServer

```
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/tcp$ ./tcpclient  
Input: hello world  
Response from server : hello world too  
Input: bye  
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/tcp$ □
```

(b) TCPClient

Figure 1: Output:TCP

```

1 //This is the Server side implementation of UDP Chat
2
3 #include<sys/types.h>
4 #include<sys/socket.h>
5 #include<netinet/in.h>
6 #include<arpa/inet.h>
7 #include<netdb.h>
8 #include<stdio.h>
9 #include<unistd.h>
10 #include<string.h>
11
12 #define MAX_MSG 100
13 #define SERVER_ADDR "127.0.0.1"
14 #define SERVER_PORT 1500
15
16 int main()
17 {
18     int sd,rc,n,cliLen;
19     struct sockaddr x;
20     struct sockaddr_in cliAddr,servAddr;
21     char msg[MAX_MSG];
22
23     printf("\n sockaddr %ld",sizeof(x));
24     printf("\n long %ld",sizeof(long));
25     printf("\nint %ld",sizeof(int));
26     printf("\n sockaddr_in %ld",sizeof(cliAddr));
27     printf("\n short %ld\n",sizeof(short));
28
29     // build server address structure/*
30
31     bzero((char *)&servAddr,sizeof(servAddr));
32     servAddr.sin_family=AF_INET;
33     servAddr.sin_addr.s_addr=inet_addr(SERVER_ADDR);
34     servAddr.sin_port=htons(SERVER_PORT);
35     //CREATE DATAGRAM SOCKET
36
37     sd=socket(AF_INET,SOCK_DGRAM,0);
38     printf("datagram socket craeted successfully\n");
39     //BIND LOCAL PORT NUMBER
40
41
42     bind(sd,(struct sockaddr*)&servAddr,sizeof(servAddr));
43     printf("successfully bind local address\n");
44
45     printf("waiting for data on port UDP %u\n",SERVER_PORT);
46
47     while(1)
48     {
49         //init buffer
50
51         memset(msg,0x0,MAX_MSG);
52
53         //Receive data from client

```

```
54
55 cliLen=sizeof(cliAddr);
56
57 n=recvfrom(sd,msg,MAX_MSG,0,(struct sockaddr *) &cliAddr,&cliLen);
58
59 printf("from %s: UDP port %u: %s \n",inet_ntoa(cliAddr.sin_addr),ntohs(cliAddr.sin_port),msg);
60 printf("from %ld: UDP port %ld,in network byte ordering : %s \n",cliAddr.sin_addr,cliAddr.sin_port,msg);
61
62 }
63
64 return 0;
65
66 }
```



```

1 //This is the client side implementation of UDP Chat
2
3 #include<sys/types.h>
4 #include<sys/socket.h>
5 #include<netinet/in.h>
6 #include<arpa/inet.h>
7 #include<netdb.h>
8 #include<stdio.h>
9 #include<unistd.h>
10 #include<string.h>
11 #include<sys/time.h>
12
13
14 #define MAX_MSG 100
15 #define SERVER_ADDR "127.0.0.1"
16 #define SERVER_PORT 1500
17
18 int main()
19 {
20     int sd,rc,n,templen;
21     struct sockaddr x;
22     struct sockaddr_in cliAddr,tempAddr,remoteServAddr;
23     char msg[MAX_MSG];
24
25     bzero((char *)&remoteServAddr,sizeof(remoteServAddr));
26     remoteServAddr.sin_family=AF_INET;
27     remoteServAddr.sin_addr.s_addr=inet_addr(SERVER_ADDR);
28     remoteServAddr.sin_port=htons(SERVER_PORT);
29
30     sd=socket(AF_INET,SOCK_DGRAM,0);
31     printf("datagram socket craeted successfully\n");
32
33     do{
34         //send data to server
35
36         printf("Enter data to send:");
37         scanf("%s",msg);
38
39         sendto(sd,msg,strlen(msg)+1,0,(struct sockaddr *)&remoteServAddr,sizeof(remoteServAddr));
40
41     }while(strcmp(msg,"quit"));
42
43     close(sd);
44
45 }

```

```
arghya@Delton: /media/arghya/Development/Github Repo/MTechA...
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/udp$ ./udpserver

sockaddr 16
long 8
int 4
sockaddr_in 16
short 2
datagram socket craeted successfully
successfully bind local address
waiting for data on port UDP 1500
from 127.0.0.1: UDP port 36141: hello
from 16777343: UDP port 11661,in network byte ordering : hello
from 127.0.0.1: UDP port 36141: world
from 16777343: UDP port 11661,in network byte ordering : world
from 127.0.0.1: UDP port 36141: quit
from 16777343: UDP port 11661,in network byte ordering : quit
□
```

(a) UDPServer

```
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssi...
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/udp$ ./udpclient
datagram socket craeted successfully
Enter data to send:hello world
Enter data to send:Enter data to send:quit
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/udp$ □
```

(b) UDPClient

Figure 2: Output:UDP

2. Write a program to broadcast a message with UDP.

Answer.

```
1 //This is the server side implementation of UDP Broadcast service for sending message passed in the parameter to the clients
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <errno.h>
6 #include <string.h>
7 #include <sys/types.h>
8 #include <netinet/in.h>
9 #include <netdb.h>
10 #include <sys/socket.h>
11 #include <sys/wait.h>
12 #include <arpa/inet.h>
13
14 #define MAXBUFLen 100    /* the port users will be connecting to */
15
16 int main(int argc, char *argv[])
17 {
18     int sockfd;
19     struct sockaddr_in their_addr; /* connector's address information */
20     struct sockaddr_in my_addr;    /* connector's address information */
21     int numbytes;
22     int optval;                    /*Used to build the options for the broadcast */
23     int optlen;
24     char buf[MAXBUFLen];          /*The buffer that we read / write each time */
25     int addr_len;                 /* Address length for the network functions
26                                     that require that */
27     unsigned long int net_id;      /*The network id */
28     long int host_id;             /*The host id in the network */
29
30
31     if (argc != 3) {
32         fprintf(stderr, "usage: %s message port\n", argv[1]);
33         exit(1);
34     }
35
36     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) { /* The socket should be changed to broadcast */
37         /* This part demands root permissions */
38         perror("socket");
39         exit(1);
40     }
41
42     optval=1;                      /*Prepare the options of the socket for Broadcast */
43     optlen=sizeof(int);
44
45     if(setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, (char *) &optval, optlen)){
46         perror("Error setting socket to BROADCAST mode");
47         exit(1);
48     }
49 }
```

```

50     their_addr.sin_family = AF_INET;           /* Protocol family - host byte order */
51     their_addr.sin_port=htons((unsigned short) atoi(argv[2])); /* port - short, network byte order */
52     their_addr.sin_addr.s_addr=htonl(INADDR_BROADCAST); /* send to all */
53     bzero(&(their_addr.sin_zero), 8); /* zero the rest of the struct */
54
55     if ((numbytes=sendto(sockfd, argv[1], strlen(argv[1]), 0, \
56         (struct sockaddr *)&their_addr, sizeof(struct sockaddr))) == -1) {
57         perror("sendto");
58         exit(1);
59     }
60
61     printf("sent %d bytes to %s\n", numbytes, inet_ntoa(their_addr.sin_addr));
62
63     close(sockfd);
64
65     return 0;
66 }

```

```

1 //This is the client side implementation of UDP Broadcast service for receiving message passed by the server
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <errno.h>
6 #include <string.h>
7 #include <sys/types.h>
8 #include <netinet/in.h>
9 #include <sys/socket.h>
10 #include <sys/wait.h>
11 #include <sys/time.h>
12 #include <sys/unistd.h>
13 #include <arpa/inet.h>
14
15 #define MYPORT 5000 /* the port users will be sending to */
16
17 #define MAXBUFLEN 100
18
19 int main()
20 {
21     int sockfd;
22     struct sockaddr_in my_addr; /* my address information */
23     struct sockaddr_in their_addr; /* connector's address information */
24     int addr_len, numbytes;
25     char buf[MAXBUFLEN];
26     int option = 1;
27
28     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
29         perror("socket");
30         exit(1);
31     }
32     else
33         printf(" \n The socket got sockfd=%d \n ", sockfd);
34
35     setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option));
36
37     my_addr.sin_family = AF_INET; /* host byte order */
38     my_addr.sin_port = htons(MYPORT); /* short, network byte order */
39     my_addr.sin_addr.s_addr = INADDR_ANY; /* auto-fill with my IP */
40     bzero(&(my_addr.sin_zero), 8); /* zero the rest of the struct */
41
42
43     if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) /* The bind command makes the ability to wait for messages */
44         == -1) {
45         perror("bind");
46         exit(1);
47     }
48
49     printf("Wait for packet \n");
50
51     addr_len = sizeof(struct sockaddr);
52
53

```

```

54  if ((numbytes=recvfrom(sockfd, buf, MAXBUFLen, 0, \
55      (struct sockaddr *)&their_addr, &addr_len)) == -1) {
56      perror("recvfrom");
57      exit(1);
58  }
59
60  printf("got packet from %s ",inet_ntoa(their_addr.sin_addr));
61  printf("packet is %d bytes long ",numbytes);
62  buf[numbytes] = '\0';
63  printf("packet contains \"%s\\n\"",buf);
64
65
66  close(sockfd);
67
68  return 0;
69 }

```

```

1 //This is the client side implementation of UDP Broadcast service for receiving message passed from the server
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <errno.h>
6 #include <string.h>
7 #include <sys/types.h>
8 #include <netinet/in.h>
9 #include <sys/socket.h>
10 #include <sys/wait.h>
11 #include <sys/time.h>
12 #include <sys/unistd.h>
13 #include <arpa/inet.h>
14
15 #define MYPORT 5000 /* the port users will be sending to */
16
17 #define MAXBUFLEN 100
18
19 int main()
20 {
21     int sockfd;
22     struct sockaddr_in my_addr; /* my address information */
23     struct sockaddr_in their_addr; /* connector's address information */
24     int addr_len, numbytes;
25     char buf[MAXBUFLEN];
26     int option = 1;
27
28     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
29         perror("socket");
30         exit(1);
31     }
32     else
33         printf(" \n The socket got sockfd=%d \n ", sockfd);
34
35     setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option));
36
37     my_addr.sin_family = AF_INET; /* host byte order */
38     my_addr.sin_port = htons(MYPORT); /* short, network byte order */
39     my_addr.sin_addr.s_addr = INADDR_ANY; /* auto-fill with my IP */
40     bzero(&(my_addr.sin_zero), 8); /* zero the rest of the struct */
41
42
43     if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) /* The bind command makes the ability to wait for messages */
44         == -1) {
45         perror("bind");
46         exit(1);
47     }
48
49     printf("Wait for packet \n");
50
51     addr_len = sizeof(struct sockaddr);
52
53

```

```

54  if ((numbytes=recvfrom(sockfd, buf, MAXBUFLen, 0, \
55      (struct sockaddr *)&their_addr, &addr_len)) == -1) {
56      perror("recvfrom");
57      exit(1);
58  }
59
60  printf("got packet from %s ",inet_ntoa(their_addr.sin_addr));
61  printf("packet is %d bytes long ",numbytes);
62  buf[numbytes] = '\0';
63  printf("packet contains \"%s\\n\"",buf);
64
65
66  close(sockfd);
67
68  return 0;
69 }

```



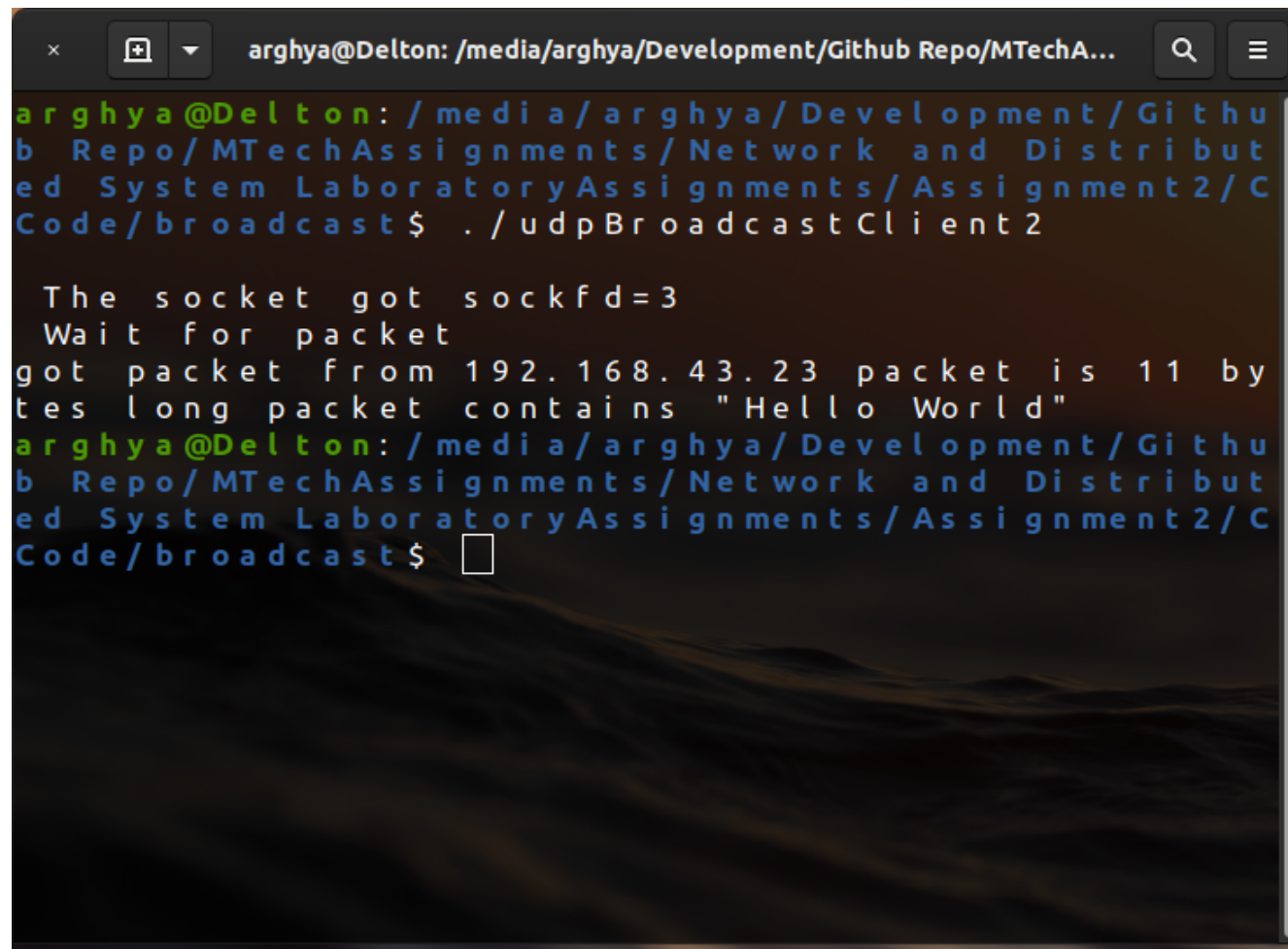
```
arghya@Delton: /media/arghya/Development/Github Repo/MTechA...
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/Code/broadcast$ ./udpBroadcastServer "Hello World" 5000
sent 11 bytes to 255.255.255.255
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/Code/broadcast$
```

(a) UDP Broadcast Server

```
arghya@Delton: /media/arghya/Development/Github Repo/MTechA...
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/Code/broadcast$ ./udpBroadcastClient1

The socket got sockfd=3
Wait for packet
got packet from 192.168.43.23 packet is 11 bytes long packet contains "Hello World"
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/Code/broadcast$
```

(b) UDP Broadcast Client1

A terminal window with a dark background and a desert landscape wallpaper. The window title is "arghya@Delton: /media/arghya/Development/Github Repo/MTechA...". The prompt is "arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System Laboratory Assignments/Assignment2/Code/broadcast\$". The user enters the command ". /udpBroadcastClient2". The output shows the program successfully opening a socket, waiting for a packet, and receiving a packet from 192.168.43.23 containing the text "Hello World". The prompt returns to the user.

```
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System Laboratory Assignments/Assignment2/Code/broadcast$ ./udpBroadcastClient2

The socket got sockfd=3
Wait for packet
got packet from 192.168.43.23 packet is 11 bytes long packet contains "Hello World"
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System Laboratory Assignments/Assignment2/Code/broadcast$
```

Figure 4: UDP Broadcast Client2