

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 // TCP Chat Server (Concurrent)
2
3 // #include <iostream.h>
4 #include <string.h>
5 #include <sys/socket.h>
6 #include <netinet/in.h> //For sockaddr_in
7 #include <unistd.h>
8 //using namespace std;
9 #include <stdio.h>
10
11 // #include "Common.h"
12 #include <stdbool.h>
13
14
15 int TCP_ChatServer(int client_desc)
16 {
17     const int BUFFER_SIZE = 4096;
18     char msg[BUFFER_SIZE];
19     int msg_len = 0;
20
21     while((msg_len = read(client_desc, msg, BUFFER_SIZE)) != 0)
22     {
23         // cout << "[User " << client_desc << "] Msg Recieved : ";
24         printf("%s %d %s", "[User", client_desc, "] Msg Received :");
25         fflush(stdout);
26
27         write(fileno(stdout), msg, msg_len);
28
29         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
30             return 0;
31
32         // cout << "[User " << client_desc << "] Enter Responce : ";
33         printf("%s %d %s", "[User", client_desc, "] Enter Response :");
34
35         fflush(stdout);
36
37
38         msg_len = read(fileno(stdin), msg, BUFFER_SIZE);
39
40         write(client_desc, msg, msg_len);
41
42         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
43             return 0;
44     }
45
46     // EOF
47     return 0;
48 }
49
```

```

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

```

```
103         break;                                //Work Done! Exit Child Process.
104     }
105     else
106     {
107         printf("%s ", "fork() Error!!!");
108         break;
109     }
110 }
111
112
113 return 0;
114 }
```

```

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

```

```

54     return 0;
55 }
56
57 int main()
58 {
59     int sock = socket(AF_INET, SOCK_STREAM, 0);
60     //CheckError(sock, "socket()");
61
62     struct sockaddr_in server_addr;
63     server_addr.sin_family      = AF_INET;
64     server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
65     server_addr.sin_port       = htons(9000);
66
67     int result = connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
68     //CheckError(result, "connect()");
69
70     TCP_ChatClient(sock);
71
72
73     close(sock);
74     return 0;
75 }

```

```

arghya@Delton: /media/arghya/Development/Git
tributed System Laboratory Assignments/Assig
This is Client.java

Prioritizing files...

Server does not have mojave_dynamic_1.jpeg

Now begin sending...
Sending: mojave_dynamic_1.jpeg

Sender thread started.
File Info: mojave_dynamic_1.jpeg, 242547
Sending info...
Got --ACK--
File Info Sent.

Sending file contents...
Got ACK#1
Bytes Sent: 64999
Got ACK#2
Bytes Sent: 129999
Got ACK#3
Bytes Sent: 194999
Final Bytes Sent: 242547

For this file:
No. of packets req to send: 4
No. of packets sent: 4
No. of ACK received: 4
No. of packets lost: 0

```

(a) TCPServer

```

Bytes Sent: 129999
Got ACK#3
Bytes Sent: 194999
Final Bytes Sent: 242547

For this file:
No. of packets req to send: 4
No. of packets sent: 4
No. of ACK received: 4
No. of packets lost: 0

Sender thread done!

Receiver thread listening...
File Info Recv.
File Info: Question1Diagram.png, 40460

Receiving file contents...
Progress: 0.0
Progress: 100.0 Packet #1

File: Question1Diagram.png received.

For this file:
No. of packets recv: 2
No. of ACK sent: 2

Closing receiver due to in-activity.

Receiver done.

```

(b) TCPClient

Figure 1: Output:TCP

```

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

```



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

```

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

```

```

arghya@Delton: /media/arghya/Development/Git
tributed System Laboratory Assignments/Assignments
This is Server.java

Receiver thread listening...

Prioritizing files...

Client does not have Question1Diagram.png

Now begin sending...
Sending: Question1Diagram.png

Sender thread started.
File Info: Question1Diagram.png, 40460
Sending info...
File Info Recv.
File Info: mojave_dynamic_1.jpeg, 242547

Receiving file contents...
Progress: 0.0
Progress: 26.0 Packet #1
Progress: 53.0 Packet #2
Progress: 80.0 Packet #3
Progress: 100.0 Packet #4

File: mojave_dynamic_1.jpeg received.

For this file:
No. of packets recv: 5
No. of ACK sent: 5
ACK not received. Re-sending...

```

(a) UDPServer

```

Receiving file contents...
Progress: 0.0
Progress: 26.0 Packet #1
Progress: 53.0 Packet #2
Progress: 80.0 Packet #3
Progress: 100.0 Packet #4

File: mojave_dynamic_1.jpeg received.

For this file:
No. of packets recv: 5
No. of ACK sent: 5
ACK not received. Re-sending...
Sending info...
Got --ACK--
File Info Sent.

Sending file contents...
Final Bytes Sent: 40460

For this file:
No. of packets req to send: 1
No. of packets sent: 2
No. of ACK received: 1
No. of packets lost: 1

Sender thread done!

Closing receiver due to in-activity.

Receiver done.

```

(b) UDPClient

Figure 2: Output:UDP

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

Answer.

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

```

50         exit(1);
51     }
52
53     optval=1; /*Prepare the options of the socket for Broadcast */
54     optlen=sizeof(int);
55
56     if(setsockopt(sockfd,SOL_SOCKET,SO_BROADCAST,(char *) &optval,optlen)){
57         perror("Error setting socket to BROADCAST mode");
58         exit(1);
59     }
60
61     their_addr.sin_family = AF_INET;          /* Protocol family - host byte order */
62     their_addr.sin_port=htons((unsigned short) atoi(argv[2])); /* port - short, network byte order */
63     their_addr.sin_addr.s_addr=htonl(INADDR_BROADCAST); /* send to all */
64     /*their_addr.sin_addr.s_addr=inet_addr("enter here the IP address in dot notation"); */ /* send to all */
65     bzero(&(their_addr.sin_zero), 8); /* zero the rest of the struct */
66
67     if ((numbytes=sendto(sockfd, argv[1], strlen(argv[1]), 0, \
68         (struct sockaddr *)&their_addr, sizeof(struct sockaddr))) == -1) {
69         perror("sendto");
70         exit(1);
71     }
72
73     printf("sent %d bytes to %s\n",numbytes,inet_ntoa(their_addr.sin_addr));
74
75     close(sockfd);
76
77     return 0;
78 }

```

```

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

```

```

54
55 printf("Wait for packet \n");
56
57     addr_len = sizeof(struct sockaddr);
58
59
60 if ((numbytes=recvfrom(sockfd, buf, MAXBUFLen, 0, \
61                     (struct sockaddr *)&their_addr, &addr_len)) == -1) {
62     perror("recvfrom");
63     exit(1);
64 }
65
66     printf("got packet from %s ",inet_ntoa(their_addr.sin_addr));
67     printf("packet is %d bytes long ",numbytes);
68     buf[numbytes] = '\0';
69     printf("packet contains \"%s\"\n",buf);
70
71
72     close(sockfd);
73
74 return 0;
75 }

```

```

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

```



```

54
55 printf("Wait for packet \n");
56
57     addr_len = sizeof(struct sockaddr);
58
59
60     if ((numbytes=recvfrom(sockfd, buf, MAXBUFLen, 0, \
61                             (struct sockaddr *)&their_addr, &addr_len)) == -1) {
62         perror("recvfrom");
63         exit(1);
64     }
65
66     printf("got packet from %s ",inet_ntoa(their_addr.sin_addr));
67     printf("packet is %d bytes long ",numbytes);
68     buf[numbytes] = '\0';
69     printf("packet contains \"%s\"\n",buf);
70
71
72     close(sockfd);
73
74     return 0;
75 }

```

```

arghya@Delton: /media/arghya/Development/Git
tributed System LaboratoryAssignments/Assig
This is Server.java

Receiver thread listening...

Prioritizing files...

Client does not have Question1Diagram.png

Now begin sending...
Sending: Question1Diagram.png

Sender thread started.
File Info: Question1Diagram.png,40460
Sending info...
File Info Recv.
File Info: mojava_dynamic_1.jpeg,242547

Receiving file contents...
Progress: 0.0
Progress: 26.0 Packet #1
Progress: 53.0 Packet #2
Progress: 80.0 Packet #3
Progress: 100.0 Packet #4

File: mojava_dynamic_1.jpeg received.

For this file:
No. of packets recv: 5
No. of ACK sent: 5
ACK not received. Re-sending...
Sending info...
Got --ACK--
File Info Sent.

Sending file contents...
Final Bytes Sent: 40460

For this file:
No. of packets req to send: 1
No. of packets sent: 2
No. of ACK received: 1
No. of packets lost: 1

Sender thread done!

Closing receiver due to in-activity.

Receiver done.

```

(a) UDP Broadcast Server

```

Receiving file contents...
Progress: 0.0
Progress: 26.0 Packet #1
Progress: 53.0 Packet #2
Progress: 80.0 Packet #3
Progress: 100.0 Packet #4

File: mojava_dynamic_1.jpeg received.

For this file:
No. of packets recv: 5
No. of ACK sent: 5
ACK not received. Re-sending...
Sending info...
Got --ACK--
File Info Sent.

Sending file contents...
Final Bytes Sent: 40460

For this file:
No. of packets req to send: 1
No. of packets sent: 2
No. of ACK received: 1
No. of packets lost: 1

Sender thread done!

Closing receiver due to in-activity.

Receiver done.

```

(b) UDP Broadcast Client1

```

Receiving file contents...
Progress: 0.0
Progress: 26.0 Packet #1
Progress: 53.0 Packet #2
Progress: 80.0 Packet #3
Progress: 100.0 Packet #4

File: mojava_dynamic_1.jpeg received.

For this file:
No. of packets recv: 5
No. of ACK sent: 5
ACK not received. Re-sending...
Sending info...
Got --ACK--
File Info Sent.

Sending file contents...
Final Bytes Sent: 40460

For this file:
No. of packets req to send: 1
No. of packets sent: 2
No. of ACK received: 1
No. of packets lost: 1

Sender thread done!

Closing receiver due to in-activity.

Receiver done.

```

(c) UDP Broadcast Client2

Figure 3: Output:UDP Broadcast