Experiment -3

UDP chat Service

3.1 AIM: To implement a simple chat service using UDP sockets.

3.2 DESCRIPTION/PROBLEM DEFINITION: Chat service is a simple application where two or more users communicate/exchange messages interactively. It can be implemented using with TCP or UDP. The following Figure 3.1 shows the connection-less client-server relationship.

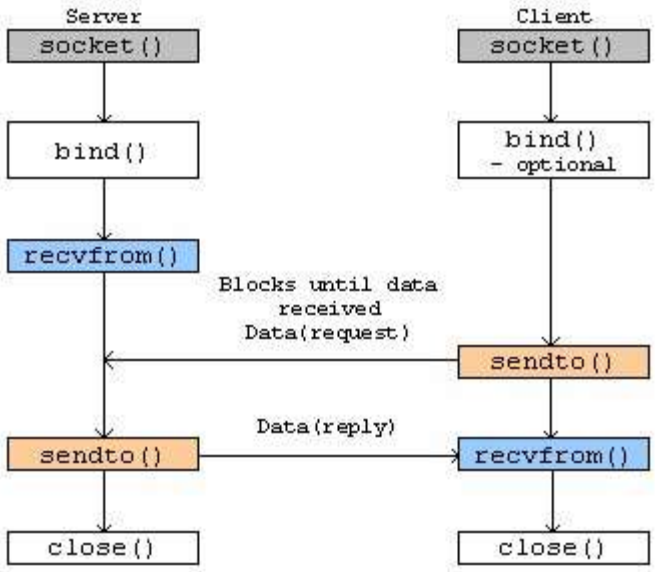

Figure 3.1: Connection-less Client-Server relationship with the socket API [1]

3.3 ALGORITHMS FOR CHAT SERVICE

Server

1. Start
2. Create a socket using SOCK_DGRAM option in ***socket()*** system call
3. Bind the server address using ***bind()*** system call
4. Receive message from the client and display on the servers' terminal [use ***recvfrom()*** *and* ***sendto()*** system calls]
5. Prompt the user to enter a response and send that response to the client over the socket (use ***sendto()*** system call)
6. Repeat steps 4 and 5 until the client quits by sending a message 'exit' or 'bye'.
7. Close the socket using **close()** system call
8. Stop

Client

1. Start
2. Create a socket using the SOCK_DGRAM option in ***socket()*** system call
3. Prompt the user to enter a message
4. Read a message from the user and send to the server over the socket (use **read()** and ***sendto()*** system call)
5. Read from the message from the server through the socket and write that message to the terminal (use **recvfrom()** and **write()** system calls).
6. Repeat steps 4 and 5 until the user types 'exit' or 'bye'.
7. Close the socket
8. Stop

3.4: REQUIREMENTS

a. System calls needed:

socket(), bind(), sendto(), recvfrom(),close()

b. Include directives:

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <stdlib.h>
```

3.5 IMPLEMENTATION

//Server Program to implement UDP chat service

```
int main(int argc, char *argv[])
{
  int ser_sfd, nbytes, i, addr_len;
  char buffer[1024];
  struct sockaddr_in sa,ca;

  ser_sfd = socket(AF_INET, SOCK_DGRAM, 0);          //Create UDP socket

  sa.sin_family =  AF_INET;                          //Fill the server address structure
  sa.sin_port = htons(atoi(argv[1]));
  sa.sin_addr.s_addr = inet_addr("172.16.0.6");
  len=sizeof(sa);

  bind(ser_sfd, (struct sockaddr *) &sa, len);       //register server address
  addr_len = sizeof (clientaddr);
```

```
//Client server interaction using UDP protocol
while(1){
    nbytes =recvfrom(ser_sfd, &buffer,1024,0,(struct sockaddr *)&ca, &addr_len);
    write(1,"CLIENT>  ",  8);
    write(1,&buff, nbytes);              //display the message received from the client
    write(1,"SERVER> ",9);               //prompt the user to enter a message
    fgets(buffer,1024,stdin);            //read a message from keyboard
    sendto(ser_sfd, &buffer,nbytes,0,(struct sockaddr *)&clientaddr, addr_len);
                                         //Send uppercase message back to client
}
  return 0;
}
```

//Client program to implement UDP Chat service

```
int main(int argc, char *argv[])
{
   int cli_sfd, nbytes;
   socklen_t addr_size;
   char buffer[1024];
   struct sockaddr_in sa, ca;

   cli_sfd = socket(AF_INET, SOCK_DGRAM, 0);          //create a UDP socket
   sa.sin_family = AF_INET;                           //Fill the server address structure
   sa.sin_port = htons(atoi(argv[1]));
   sa.sin_addr.s_addr = inet_addr("172.16.0.6");
   memset(sa.sin_zero, '\0', sizeof sa.sin_zero);

   ca.sin_family = AF_INET;                           //Fill the client address structure
   ca.sin_port = 0;                                   //UDP will assign  a free port number
   ca.sin_addr.s_addr = htonl(INADDR_ANY);            //Machine's  IP address on which we run
   memset(ca.sin_zero, '\0', sizeof ca.sin_zero);
   bind(cli_sfd,(struct sockaddr*)&ca,sizeof(ca));    //client registration with UDP

   addr_size = sizeof sa;

    //Client server interaction using UDP protocol
   while(1){
       write(1,"CLIENT>  ", 8);                //prompt the user to enter a message
       fgets(buffer,1024,stdin);*Send message to server*/
       sendto(cli_sfd,&buffer,sizeof(buffer), 0, (struct sockaddr *) &sa, addr_size);
                                              //*Receive message from server
       nbytes=recvfrom(cli_sfd, &buffer,1024,0, (struct sockaddr *) &sa, &addr_size);
       buff[nbytes]='\0';
       printf("SERVER>  %s\n",buffer);
  }
  return 0;
}
```

TASKS to be performed

1. Modify the above the code to implement a simple chat service where both the client and server processes should be terminated when the user types the message '**bye**' or '**exit**'.
2. Take the snapshots of your experimentation results and prepare the lab record.
3. Submit the lab record for this experiment in the next lab session.

3.6: RESULTS AND DISCUSSIONS

..
.. take the results screenshot and write your comments on the results
..

3.7 CONCLUSIONS

- Successfully implemented the chat service using UDP protocol
- Tested by running the programs on two different machines with different messages

References:

1. http://www.tenouk.com/Module39a.html
2. http://pirate.shu.edu/~wachsmut/Teaching/CSAS2214/Virtual/Lectures/chat-client-server.html
3. http://www.tutorialspoint.com/unix_system_calls/socket.htm socket tutorials
4. Richard Stevens, "UNIX Network Programming", PHI