8 Socket Programming : UDP

8.1 Aim

Implement Client-Server communication using Socket Programming and UDP as transport layer protocol.

8.2 Theory

8.2.1 UDP

User Datagram Protocol(UDP) is a data transmission protocol.It prioritises latency over reliability of connection and uses a connectionless communication model.It is unreliable.

8.2.2 Client,Server and Socket

- **Server**: A server is a program that processes requests from client programs and replies to the requests accordingly
- **Client**:A client is a program that requests services from the server.The client program sends a request in a predefined format to the server and the server replies accordingly to the request.
- **Socket**:A socket is a way to communicate in a connection using file descriptors.They act as endpoints in a connection.

8.3 Algorithm

Algorithm 1 Server

```
procedure Main Procedure
    if sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 then
        printf("socket creation failed");
        exit(0);
    end if
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);
    if bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0
then
        printf("bind failed");
        exit(0);
    end if
    for true do
        n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (
struct sockaddr *) &cliaddr, (socklen_t *)&len)
        scanf("%s",serverMessage)
        sendto(sockfd, (char *)&serverMessage, strlen(serverMessage),
MSG_CONFIRM, (const struct sockaddr *) &cliaddr, len);
    end for
end procedure
```

Algorithm 2 Client

```
procedure Main Procedure
    if sockfd = socket(AF_INET, SOCK_DGRAM, 0) < 0 then
        perror("socket creation failed")
        exit(EXIT_FAILURE)
    end if
    for true do
        scanf("%s",clientMessage);
        sendto(sockfd, (const char *)clientMessage, strlen(clientMessage),
MSG_CONFIRM, (const struct sockaddr *)&servaddr,sizeof(servaddr))
        n = recvfrom(sockfd, (char *)buffer, MAXLINE,MSG_WAITALL,
(struct sockaddr *) &servaddr,(socklen_t *)&len);
    end for
end procedure
```

8.4 Code

Server

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#define PORT    8080
#define MAXLINE 1024

void communicator(int sockfd){
    int len, n;
    struct sockaddr_in  cliaddr;
    memset(&cliaddr, 0, sizeof(cliaddr));
    char buffer[MAXLINE],serverMessage[MAXLINE];
    for(;;){
  memset(buffer,0,sizeof(buffer));
  n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (
    struct sockaddr *) &cliaddr, (socklen_t *)&len);
  printf("Client : %s\n", buffer);
  if(strcmp(buffer,"exit") == 0){
    printf("Server exiting \n");
    exit(0);
  }
  memset(serverMessage,0,sizeof(serverMessage));

    scanf("%s",serverMessage);
  sendto(sockfd, (char *)&serverMessage, strlen(serverMessage),
    MSG_CONFIRM, (const struct sockaddr *) &cliaddr, len);
  printf("Client message sent.\n");
    }
}

int main() {
  int sockfd;
  struct sockaddr_in servaddr;

  if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
    printf("socket creation failed");
    exit(0);
  }

  memset(&servaddr, 0, sizeof(servaddr));


  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = INADDR_ANY;
  servaddr.sin_port = htons(PORT);

  if ( bind(sockfd, (const struct sockaddr *)&servaddr,
      sizeof(servaddr)) < 0 )
  {
    printf("bind failed");
    exit(0);
  }

```

```
  communicator(sockfd);

  return 0;
}
```

Client

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#define PORT    8080
#define MAXLINE 1024

void communicator(int sockfd){
    int n, len;
  struct sockaddr_in    servaddr,cliaddr;
    char buffer[MAXLINE];
    char clientMessage[MAXLINE];
    memset(&servaddr, 0, sizeof(servaddr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_port = htons(PORT);
  servaddr.sin_addr.s_addr = INADDR_ANY;
    for(;;){
        scanf("%s",clientMessage);
      sendto(sockfd, (const char *)clientMessage, strlen(clientMessage), MSG_CONFIRM, (const struct sockaddr *) &servaddr,sizeof(servaddr));
      printf("Client message sent.\n");
    if(strcmp(clientMessage,"exit") == 0){
      printf("Client exiting \n");
      exit(0);
    }
      n = recvfrom(sockfd, (char *)buffer, MAXLINE,MSG_WAITALL, (struct sockaddr *) &servaddr,(socklen_t *)&len);
      buffer[n] = '\0';
      printf("Server : %s\n", buffer);

    }
}

int main() {
  int sockfd;
  if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
    perror("socket creation failed");
    exit(EXIT_FAILURE);
  }
    communicator(sockfd);
  close(sockfd);
  return 0;
```

```
46 }
```

8.5 Output

- **Server**

```
Client : hellowolrd
helo
Client message sent.
Client : timetoexit
okbye
Client message sent.
Client : exit
Server exiting
```

- **Client**

```
hellowolrd
Client message sent.
Server : helo
timetoexit
Client message sent.
Server : okbye
exit
Client message sent.
Client exiting
```

8.6 Result

Server and client was implemented in C++.A socket is created for the server and binded to a particular port.The client sends data to this port and the server services such request in a first come first serve basis.Thus the reliability of this system is low but the latency is low.