10 Concurrent Time Server application using UDP

10.1 Aim

Implement Concurrent Time Server application using UDP to execute the program at remoteserver. Client sends a time request to the server, server sends its system time back to the client. Client displays the result.

10.2 Theory

10.2.1 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.

10.2.2 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.

10.2.3 Time Server

Time server is a server which sends back the current system time of the server when the client sends a request to the server.

10.3 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 *)inp_buffer, MAXLINE,
MSG_WAITALL, ( struct sockaddr *) &cliaddr, (socklen_t *)&len)
        time (&rawtime)
        timeinfo = localtime(&rawtime)
        strftime(out_buffer,sizeof(out_buffer),"%d-%m-%Y
%H:%M:%S",timeinfo)
        if strcmp(inp_buffer,"now") == 0 then
            sendto(socketfd,out_buffer,strlen(out_buffer),MSG_CONFIRM,(sockaddr
*)&client,len)
            printf("Time sent to client ");
        end if
    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

10.4 Code

Server

```
#include<arpa/inet.h>
#include<sys/socket.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<unistd.h>
#include<sys/types.h>
#include<ctime>
using namespace std;

#define PORT 8080
#define MAXLINE 1024

void communicator(int socketfd){
    char inp_buffer[MAXLINE],out_buffer[MAXLINE];
    struct sockaddr_in client;
    int n = 0,len;
    for(;;){
    n = recvfrom(socketfd,inp_buffer,sizeof(inp_buffer),MSG_WAITALL
    ,(sockaddr *)&client,(socklen_t *)&len);
    inp_buffer[n] = '\0';
    printf("Client :%s\n",inp_buffer);

    time_t rawtime;
    struct tm * timeinfo;
    time (&rawtime);
    timeinfo = localtime(&rawtime);
    strftime(out_buffer,sizeof(out_buffer),"%d-%m-%Y %H:%M:%S",
    timeinfo);

    if(strcmp(inp_buffer,"now") == 0){
        sendto(socketfd,out_buffer,strlen(out_buffer),MSG_CONFIRM,(
    sockaddr *)&client,len);
```

```
        printf("Time sent to client \n");
    }
    }
}

int main(){
    int socketfd;
    struct sockaddr_in server;
    socketfd = socket(AF_INET,SOCK_DGRAM,0);
    if(socketfd < 0){
        printf("Socket creation failed !\n");
        exit(0);
    }
    printf("Socket creation successful !\n");
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(PORT);
    server.sin_family = AF_INET;
    if(bind(socketfd,(sockaddr *)&server,sizeof(server))<0){
        printf("Socket binding failed !\n");
        exit(0);
    }
    printf("Socket binding successful !\n");
    communicator(socketfd);

    return 0;
}
```

Client

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

#define PORT 8080
#define MAX 1024

void communicator(int socketfd){
    char inp_buffer[MAX],out_buffer[MAX];
    struct sockaddr_in server;

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

    strcpy(out_buffer,"now");
    sendto(socketfd,out_buffer,strlen(out_buffer),MSG_CONFIRM,(
    sockaddr *)&server,sizeof(server));
    int n,len;
    memset(inp_buffer,0,sizeof(inp_buffer));
    n = recvfrom(socketfd,inp_buffer,MAX,MSG_WAITALL,(sockaddr *)&
```

```
    server,(socklen_t *)&len);
    printf("Server: %s\n",inp_buffer);

}
int main(){
    int socketfd;
    socketfd = socket(AF_INET,SOCK_DGRAM,0);
    if(socketfd<0){
        printf("Socket creation failed !\n");
        exit(0);
    }
    printf("Socket creation successful !\n");
    communicator(socketfd);
    return 0;
}
```

10.5 Output

- **Server**

```
Socket creation successful !
Socket binding successful !
Client :now
Time sent to client
Client :now
Time sent to client
```

- **Client #1**

```
Socket creation successful !
Server: 14-03-2019 01:48:44
```

- **Client #2**

```
Socket creation successful !
Server: 14-03-2019 01:48:45
```

10.6 Result

Server and client was implemented in C++.A socket is created for the server and binded to a particular port.The client sends a request to the server.The server computes the current system time and sends the time back to the requesting client.The client displays the time and exits.