EXPERIMENT - X
CONCURRENT TIME SERVER APPLICATION USING UDP

April 3, 2020

ADITHYA D RAJAGOPAL
ROLL NO : 9
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

AIM

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

THEORY

UDP (User Datagram Protocol) is primarily for establishing low-latency and loss-tolerating connections between applications on the internet. UDP sends messages, called datagrams, and is considered a best-effort mode of communications. It is considered a connectionless protocol because it does nott require a virtual circuit to be established before any data transfer occurs.

Server

The server here waits for the client's time request. When a request is received, the present system time of the server is sent to the client.

Client

The client sends the server a time request. The response from the server is received and provided as the output

ALGORITHM

Algorithm 1 Algorithm for the client

1: START
2: Create the socket using the function socket().
3: Configure socket details.
4: Connect the socket to server using function connect().
5: Receive the response from the server using recvfrom().
6: Send the acknowledge message to the server using sendto().
7: Print the response.
8: STOP

Algorithm 2 Algorithm for the server

1: START
2: We can use some time function to return the time.
3: Create the UDP socket.
4: Configure socket details.
5: Bind the address struct to the socket using bind().
6: Receive from client using recvfrom().
7: Print received message.
8: Send the time to the client.
9: STOP

SOURCE CODE

Client

```
#include<stdio.h>
#include<sys/socket.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
#define PORT 8080
#define MAXLINE 1024
#define SA struct sockaddr

void main()
{
 int sockfd,len,n;
 struct sockaddr_in servaddr;
 char *req="Client requested for time",buffer[MAXLINE];
 char *ack="Message recieved by client";
 sockfd=socket(AF_INET,SOCK_DGRAM,0);
 if(sockfd<0)
 {
  printf("Socket creation failed...\n");
  exit(0);
 }
 else
 printf("Socket successfully created...\n");
 memset(&servaddr,0,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_port=htons(PORT);
 servaddr.sin_addr.s_addr=INADDR_ANY;
 len=sizeof(servaddr);
 sendto(sockfd,(const char*)req,strlen(req),
        MSG_CONFIRM,(const SA*)&servaddr,len);
```

```
 printf("Time request send to server...\n");
 n=recvfrom(sockfd,(char *)buffer,MAXLINE,
            MSG_WAITALL,(SA*)&servaddr,&len);
 buffer[n]='\0';
 printf("Server : %s",buffer);
 sendto(sockfd,(const char*)ack,strlen(ack),
        MSG_CONFIRM,(const SA*)&servaddr,len);
 close(sockfd);
}
```

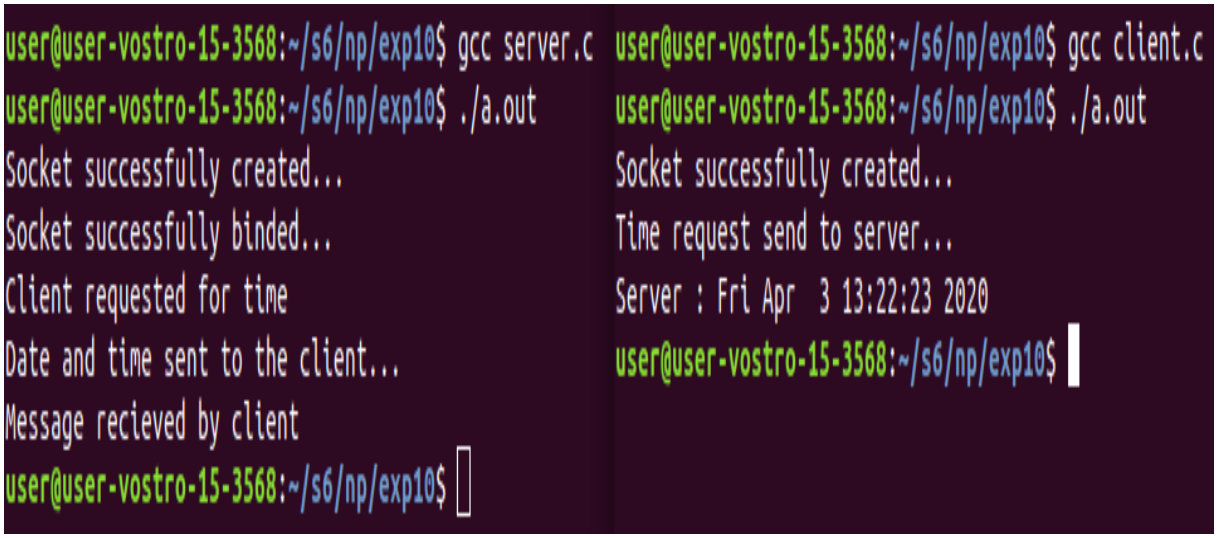
Server

```
#include<stdio.h>
#include<stdlib.h>
#include<arpa/inet.h>
#include<string.h>
#include<sys/socket.h>
#include<unistd.h>
#include<time.h>
#define PORT 8080
#define MAXLINE 1024
#define SA struct sockaddr

void main()
{
 int sockfd,bindfd,len,n;
 struct sockaddr_in servaddr,cliaddr;
 char buffer[MAXLINE];
 time_t t;
 time(&t);
 char *time=ctime(&t);
 sockfd=socket(AF_INET,SOCK_DGRAM,0);
 if(sockfd<0)
 {
  printf("Socket creation failed...\n");
  exit(0);
```

```
 }
 else
 printf("Socket successfully created...\n");
 memset(&servaddr,0,sizeof(servaddr));
 memset(&cliaddr,0,sizeof(cliaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=INADDR_ANY;
 servaddr.sin_port=htons(PORT);
 bindfd=bind(sockfd,(const SA*)&servaddr,sizeof(servaddr));
 if(bindfd<0)
 {
  printf("Socket bind failed...\n");
  exit(0);
 }
 else
 printf("Socket successfully binded...\n");
 len=sizeof(cliaddr);
 n=recvfrom(sockfd,(char *)buffer,MAXLINE,
             MSG_WAITALL,(SA*)&cliaddr,&len);
 buffer[n]='\0';
 printf("%s\n",buffer);
 sendto(sockfd,(char *)time,MAXLINE,
        MSG_CONFIRM,(const SA*)&cliaddr,len);
 printf("Date and time sent to the client...\n");
 n=recvfrom(sockfd,(char *)buffer,MAXLINE,
             MSG_WAITALL,(SA*)&cliaddr,&len);
 buffer[n]='\0';
 printf("%s\n",buffer);
 close(sockfd);
}
```

OUTPUT


user@user-vostro-15-3568:~/s6/np/exp10$ gcc server.c
user@user-vostro-15-3568:~/s6/np/exp10$ ./a.out
Socket successfully created...
Socket successfully binded...
Client requested for time
Date and time sent to the client...
Message recieved by client
user@user-vostro-15-3568:~/s6/np/exp10$
user@user-vostro-15-3568:~/s6/np/exp10$ gcc client.c
user@user-vostro-15-3568:~/s6/np/exp10$ ./a.out
Socket successfully created...
Time request send to server...
Server : Fri Apr 3 13:22:23 2020
user@user-vostro-15-3568:~/s6/np/exp10$

RESULT

Implemented a Concurrent Time Server application using UDP to execute the program at remote server using C.