Experiment 2

AIM: Socket Programming: Implementation of Connection-Oriented Service using standard ports.
i. Echo Service
ii. Date and Time Service
iii. Time of Day Service
iv. Character generation

THEORY:

Sockets: In Computer Networks, communication between server and client using TCP protocol is connection oriented (which buffers and bandwidth are reserved for client). Server will get so many hits from different clients, and then the server has to identify each client uniquely to reply to every request. To achieve this, we use "IP address of client (32 bit) + port number (16 bit) of the process". This is called Socket (48 bit). Any network communication should go through socket. Socket is a way of speaking to other programs using standard file descriptors.

In UDP, the client does not form a connection with the server like in TCP and instead, It just sends a datagram. Similarly, the server need not to accept a connection and just waits for datagrams to arrive. We can call a function called connect() in UDP but it does not result anything like it does in TCP. There is no 3 way handshake. It just checks for any immediate errors and store the peer's IP address and port number. connect() is storing peers address so no need to pass server address and server address length arguments in sendto().

ALGORITHM

Server Socket

1. Create a socket - Get the file descriptor
2. Bind to an address
3. Listen on a port, and wait for a connection to be established.
4. Accept the connection from a client.
5. Shutdown to end read/write.
6. Close to release data.

Client Socket

1. Create a socket.
2. Bind* -optional because you're the client, not the server.
3. Connect to a server.
4. For echo server, send a message to the server to be echoed using send() system call.
5. Receive the result of the request made to the server using recv() system call.
6. Shutdown to end read/write.
7. Close to release data.

SOURCE CODE:

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <time.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
int main()
{
 int sockfd, connfd, len;
 struct sockaddr_in server_addr, client_addr;
 if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
 {
 perror("socket creation failed");
 exit(EXIT_FAILURE);
 }
 else
 {
 printf("\nSocket created");
 }
 memset(&server_addr, 0, sizeof(server_addr));
 memset(&client_addr, 0, sizeof(client_addr));
 server_addr.sin_family = AF_INET;
 server_addr.sin_addr.s_addr = INADDR_ANY;
 server_addr.sin_port = htons(PORT);
 if (bind(sockfd, (const struct sockaddr *)&server_addr,
 sizeof(server_addr)) < 0)
 {
 perror("bind failed");
 exit(EXIT_FAILURE);
 }
 else
 {
 printf("\nSocket binded.\n");
```

```
 }
 while (1)
 {
 char response[256], buffer[1024], client_req[256];
 int n;
 strcpy(response, "");
 strcpy(client_req, "");
 int cli_len = sizeof(client_addr);
 n = recvfrom(sockfd, client_req, sizeof(client_req), 0, (struct sockaddr *)&client_addr,
&cli_len);
 client_req[n] = '\0';
 printf("%s\n", client_req);
 if (client_req[0] == '2')
 {
 time_t ticks;
 ticks = time(NULL);
 snprintf(response, sizeof(response), "%24s", ctime(&ticks));
 }
 else if (client_req[0] == '3')
 {
 time_t now;
 struct tm *now_tm;
 int hour;
 int mints, secs;
 now = time(NULL);
 now_tm = localtime(&now);
 hour = now_tm->tm_hour;
 mints = now_tm->tm_min;
 secs = now_tm->tm_sec;
 snprintf(response, sizeof(response), "Time: %02d:%02d:%02d", hour, mints, secs);
 }
 else if (client_req[0] == '4')
 {
 srand(time(0));
 snprintf(response, sizeof(response), "%c", (char)(rand() % 128));
 }
 else
 {
 snprintf(response, sizeof(response), "%s", client_req);
```

```
 sendto(sockfd, response, sizeof(response), 0, (struct sockaddr *)&client_addr,
sizeof(client_addr));
 }
 sendto(sockfd, response, sizeof(response), 0, (struct sockaddr *)&client_addr,
sizeof(client_addr));
 }
 close(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
int main()
{
 int sockfd;
 char buffer[1024];
 struct sockaddr_in server_addr;
 if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
 {
 perror("socket creation failed");
 exit(EXIT_FAILURE);
 }
 else
 {
 printf("\nSocket created");
 }
 memset(&server_addr, 0, sizeof(server_addr));
 server_addr.sin_family = AF_INET;
 server_addr.sin_port = htons(PORT);
 server_addr.sin_addr.s_addr = INADDR_ANY;
 if (connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0)
```

```
{
printf("\n Error : Connect Failed \n");
exit(0);
}
else
{
printf("\nConnection made before transmissions.");
}
printf("What task would you like to perform?\n1) Echo Service\n2) Day Time Service\n3)Time
of the day\n4)Character Generation\n");
int ch;
scanf("%d", &ch);
char message[256];
switch (ch)
{
case 1:
strcpy(message, "Echo this message\0");
break;
case 2:
strcpy(message, "2\0");
break;
case 3:
strcpy(message, "3\0");
break;
case 4:
strcpy(message, "4\0");
break;
default:
printf("Wrong choice");
exit(0);
}
int n, len, max_len = 1024;
sendto(sockfd, (const char *)message, strlen(message),
MSG_CONFIRM, (const struct sockaddr *)NULL,
sizeof(server_addr));
printf("\nRequest sent.\n");
n = recvfrom(sockfd, (char *)buffer, max_len,
MSG_WAITALL, (struct sockaddr *)NULL, NULL);
buffer[n] = '\0';
```

```
 printf("Server response : %s\n", buffer);
 close(sockfd);
 printf("\nClient socket closed.\n");
 return 0;
}
```

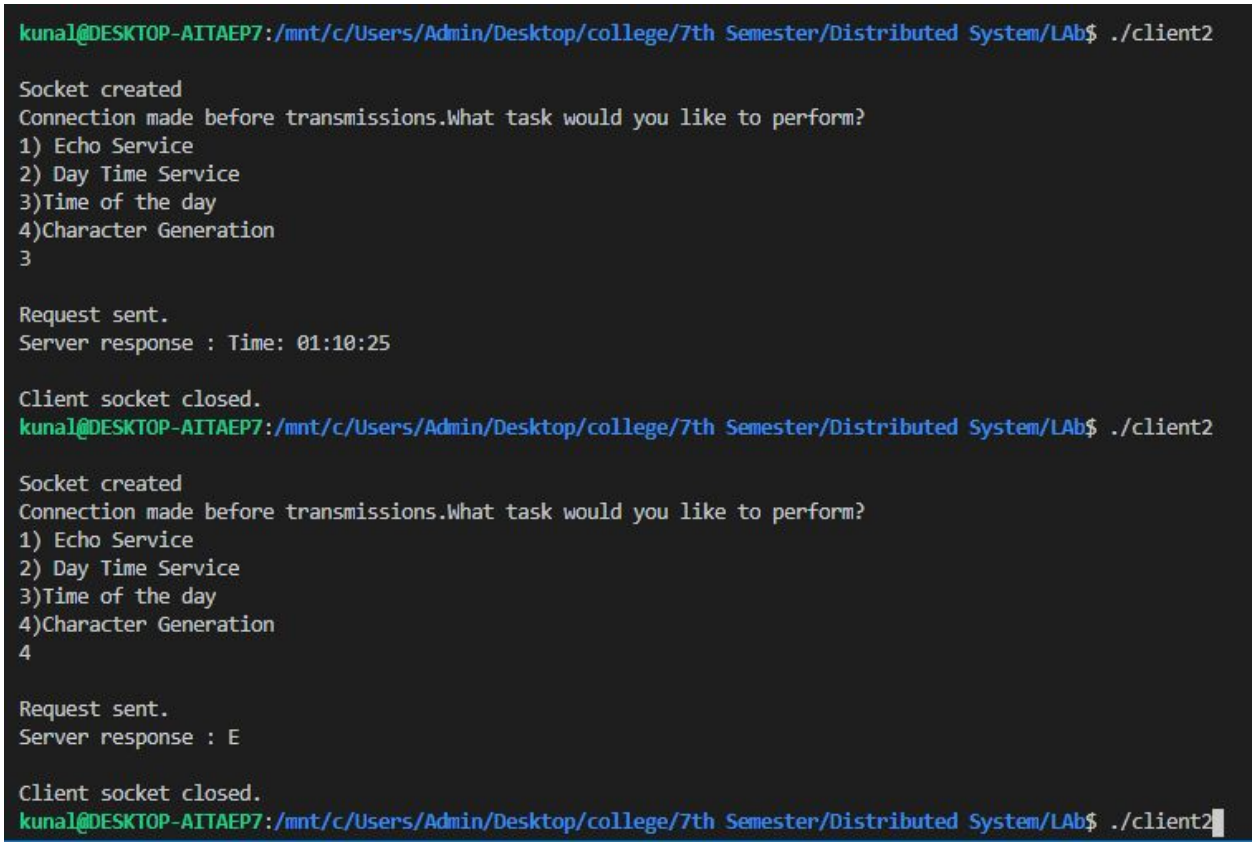
OUTPUT:

Server [Taken on WSL(windows subsystem for Linux)]

```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$ gcc program2server.c -o server2
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$ ./server2

Socket created
Socket binded.
Echo this message
2
3
4
```

Client [Taken on WSL(windows subsystem for Linux)]

```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$
gcc program2client.c -o client2
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$ ./client2

Socket created
Connection made before transmissions.What task would you like to perform?
1) Echo Service
2) Day Time Service
3)Time of the day
4)Character Generation
1

Request sent.
Server response : Echo this message

Client socket closed.
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/college/7th Semester/Distributed System/LAb$ ./client2

Socket created
Connection made before transmissions.What task would you like to perform?
1) Echo Service
2) Day Time Service
3)Time of the day
4)Character Generation
2

Request sent.
Server response : Mon Sep 14 01:09:26 2020
```



LEARNING OUTCOMES:

We learnt how to implement a Client-Server Communication using UDP protocol and also about socket programming. We also learned how using connect() is a much reliable way of communication as it secures a connection before any transmission happens.