**Assignment No : A 3 (b)**

**UDP**

**SOURCE CODE:**

UDP SERVER:

// server program for udp connection

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include<netinet/in.h>

#define PORT 5000

#define MAXLINE 1000

// Driver code

int main()

{

char buffer[100];

char \*message = "Hello Client";

int listenfd, len;

struct sockaddr\_in servaddr, cliaddr;

bzero(&servaddr, sizeof(servaddr));

// Create a UDP Socket

listenfd = socket(AF\_INET, SOCK\_DGRAM, 0);

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

servaddr.sin\_family = AF\_INET;

// bind server address to socket descriptor

bind(listenfd, (struct sockaddr\*)&servaddr, sizeof(servaddr));

//receive the datagram

len = sizeof(cliaddr);

int n = recvfrom(listenfd, buffer, sizeof(buffer),

0, (struct sockaddr\*)&cliaddr,&len); //receive message from server

buffer[n] = '\0';

puts(buffer);

// send the response

sendto(listenfd, message, MAXLINE, 0,

(struct sockaddr\*)&cliaddr, sizeof(cliaddr));

}

UDP CLIENT:

// udp client driver program

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include<netinet/in.h>

#include<unistd.h>

#include<stdlib.h>

#define PORT 5000

#define MAXLINE 1000

// Driver code

int main()

{

char buffer[100];

char \*message = "Hello Server";

int sockfd, n;

struct sockaddr\_in servaddr;

// clear servaddr

bzero(&servaddr, sizeof(servaddr));

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

servaddr.sin\_port = htons(PORT);

servaddr.sin\_family = AF\_INET;

// create datagram socket

sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

// connect to server

if(connect(sockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0)

{

printf("\n Error : Connect Failed \n");

exit(0);

}

// request to send datagram

// no need to specify server address in sendto

// connect stores the peers IP and port

sendto(sockfd, message, MAXLINE, 0, (struct sockaddr\*)NULL, sizeof(servaddr));

// waiting for response

recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr\*)NULL, NULL);

puts(buffer);

// close the descriptor

close(sockfd);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

