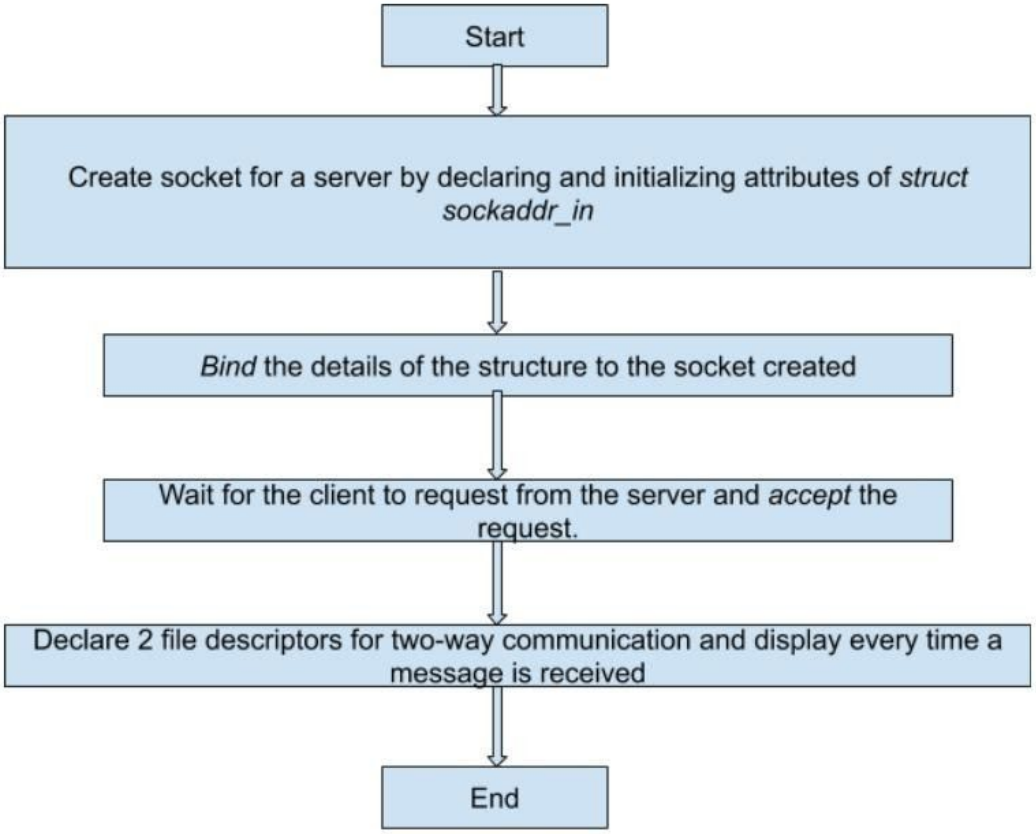
CN Assignment - 2

Samarth Mittal, IIT2018184.

Answer 1]



<u>Steps:</u>

Step 1 - Create a socket using *socket()* and initializing the values of the structure *sockaddr_in.*

Step 2 - *Bind()* assigns the details specified in *sockaddr_in* to the socket created using *socket().*

Step 3 - *accept()* lets the server wait for the client to give a request and respond to it.

Step 4 - 2 file descriptors are declared and every time a message is received is displayed both on the client as well as on the server.

~/Desktop/IIT2018184_server.c - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

IIT2018184_server.c

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

int main(){
    int fd1, fd2, t;
    socklen_t len;
    char buf[100];
    fd_set n;
    struct sockaddr_in s_address, c_address;
    memset(&s_address,0,sizeof(s_address));
    memset(&c_address,0,sizeof(c_address));
    fd1 = socket(AF_INET,SOCK_STREAM,0);
    s_address.sin_family=AF_INET;
    s_address.sin_addr.s_addr=inet_addr("127.0.0.1");
    s_address.sin_port=PORT;
    t = bind(fd1,(struct sockaddr *)&s_address,sizeof(s_address));
    t = listen(fd1,20);
    len = sizeof(c_address);
    fd2 = accept(fd1,(struct sockaddr *)&c_address, &len);
    printf("Enter Message\n");
    while (1){
        FD_ZERO(&n);
        FD_SET(0,&n);
        FD_SET(fd2,&n);
        int ret=select(10,&n,0,0,0);

        if (FD_ISSET(0,&n)){
            fgets(buf,100,stdin);

            t = send(fd2,buf,100,0);

            if (strncmp(buf,"bye",3)==0){
                close(fd2);
                close(fd1);
                return 0;
            }
        }
        if (FD_ISSET(fd2,&n)){
            t = recv(fd2,buf,100,0);
            if (strncmp(buf,"bye",3)==0){
                close(fd2);
                close(fd1);
```

Line 18, Column 44

~/Desktop/IIT2018184_client.c - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

IIT2018184_client.c

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

int main(){
    char buffer[100];
    int fd1, t;
    struct sockaddr_in cl_addr;
    fd_set n;
    memset(&cl_addr,0,sizeof(cl_addr));
    fd1 = socket(AF_INET,SOCK_STREAM,0);

    cl_addr.sin_family = AF_INET;
    cl_addr.sin_addr.s_addr=INADDR_ANY;
    cl_addr.sin_port=PORT;

    t = connect(fd1,(struct sockaddr *)&cl_addr,sizeof(cl_addr));

    printf("Enter Message\n");
    while (1){
        FD_ZERO(&n);
        FD_SET(0,&n);
        FD_SET(fd1,&n);
        int ret=select(10,&n,0,0,0);

        if (FD_ISSET(0,&n)){
            fgets(buffer,100,stdin);
            t = send(fd1,buffer,100,0);

            if (strncmp(buffer,"bye",3)==0){
                close(fd1);
                return 0;
            }
        }
        if (FD_ISSET(fd1,&n))       {
            t = recv(fd1,buffer,100,0);

            if (strncmp(buffer,"bye",3)==0){
                close(fd1);
                return 0;
            }
            printf("\nServer msg: %s\n",buffer);
        }
```

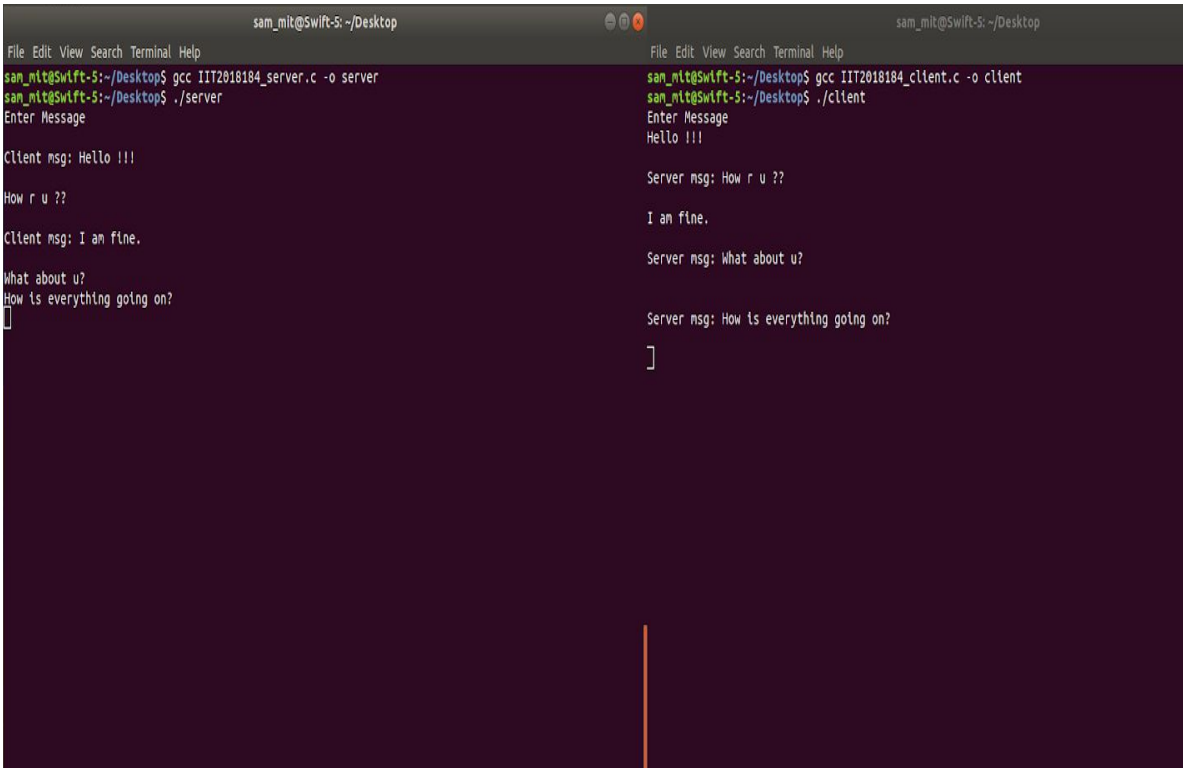
Line 22, Column 26

Functions:

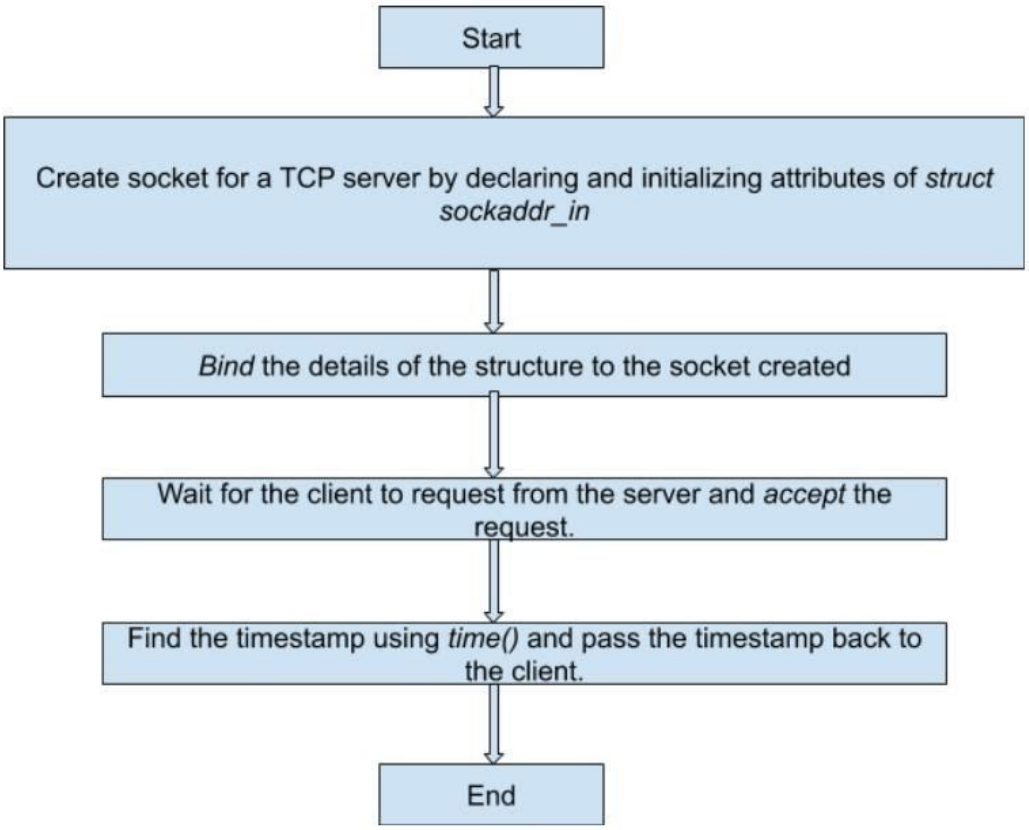
- *socket()* - to create an unnamed socket in the kernel and returns the socket descriptor.
- *bind()* - to bind the details specified in the structure *sockaddr_in* to the socket descriptor created by the *socket()* function above.
- *listen()* - to specify the maximum number of clients that can be queued by the server to come.
- *accept()* - to wait for the client to request and then respond to it.

Algorithm:

Create sockets for both the server and the client. The server is *bind()* to the client socket whereas the client socket is *connect()* connected to the socket of the server. Both the client and the socket wait to create a three way handshake. When it is created, the *accept()* call wakes up the server and responds to it. The received messages are thus displayed on both the server and the client respectively.



Answer 2]



Steps:

Step 1 - Create a TCP socket using *socket()* and initializing the values of the structure *sockaddr_in.*

Step 2 - *Bind()* assigns the details specified in *sockaddr_in* to the socket created using *socket().*

Step 3 - *accept()* lets the server wait for the client to give a request and respond to it.

Step 4 - *time()* is used by the server to know the timestamp of the address given to it by the client. It writes to the client socket using the descriptor obtained by *accept().*

~/Desktop/IIT2018184_time_server.c - Sublime Text (UNREGISTERED)

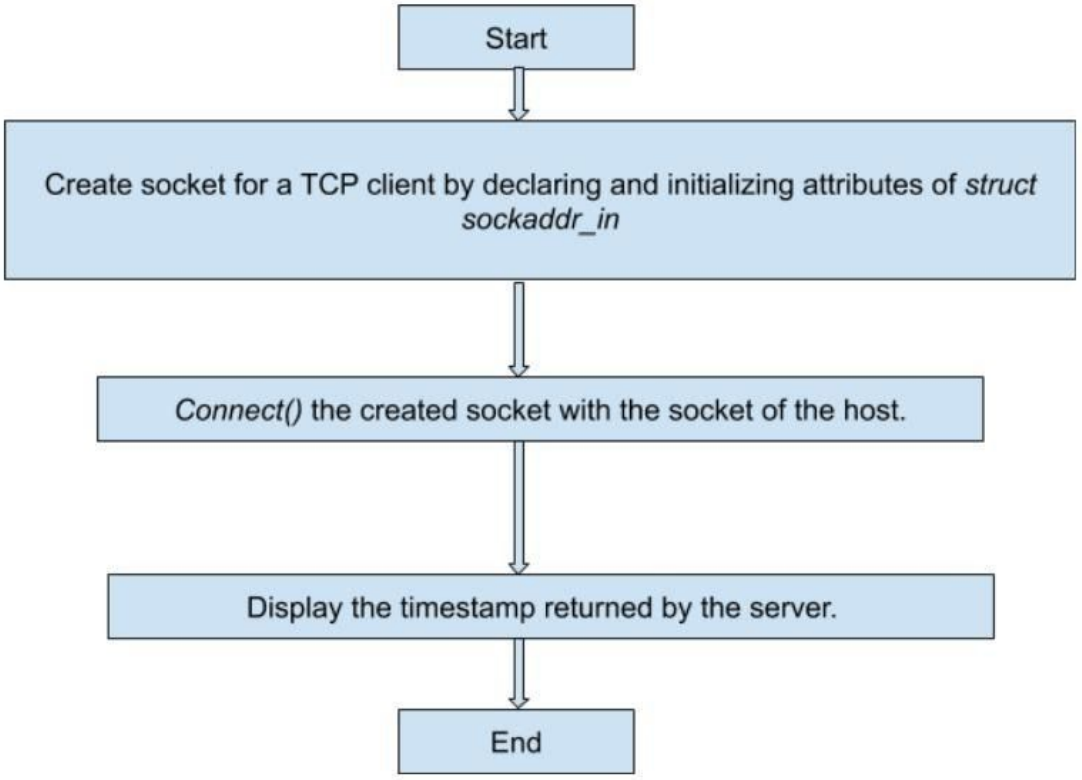
File Edit Selection Find View Goto Tools Project Preferences Help

IIT2018184_time_server.c

```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <time.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#define PORT 8180

int main(int argc, char *argv[])
{
    int fd1 = 0, fd2 = 0;
    struct sockaddr_in serv_addr;
    char sbuffer[1025];
    time_t ticks;
    fd1 = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sbuffer, '0', sizeof(sbuffer));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(PORT);
    bind(fd1, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
    listen(fd1, 10);
    while(1){
        fd2 = accept(fd1, (struct sockaddr*)NULL, NULL);
        ticks = time(NULL);
        snprintf(sbuffer, sizeof(sbuffer), "%.24s\r\n", ctime(&ticks));
        write(fd2, sbuffer, strlen(sbuffer));
        close(fd2);
        sleep(1);
    }
}
```

Functions:

- *socket()* - to create an unnamed socket in the kernel and returns the socket descriptor.
- *bind()* - to bind the details specified in the structure *sockaddr_in* to the socket descriptor created by the *socket()* function above.
- *listen()* - to specify the maximum number of clients that can be queued by the server to come.
- *accept()* - to wait for the client to request and then respond to it.
- *time()* - to prepare the timestamp as per the request given by the client.



Steps:

Step 1 - Create a TCP socket using *socket()* and initializing the values of the structure *sockaddr_in.*

Step 2 - *Connect()* to the socket of the host(server) using the socket created above.

Step 3 - Display the timestamp returned by the server.

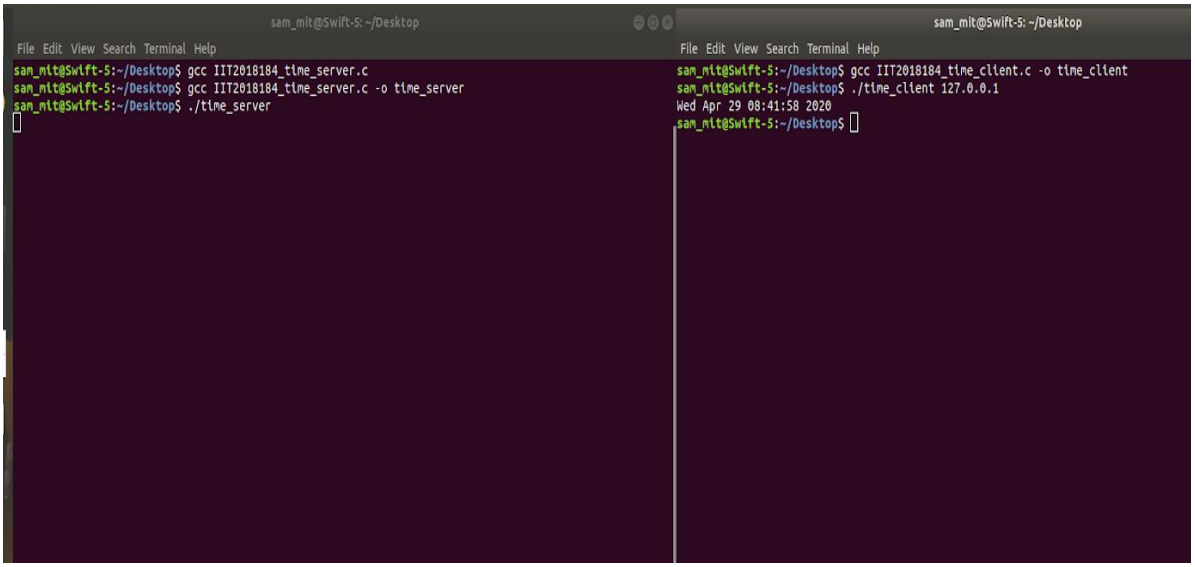
File Edit Selection Find View Goto Tools Project Preferences Help

IIT2018184_time_client.c

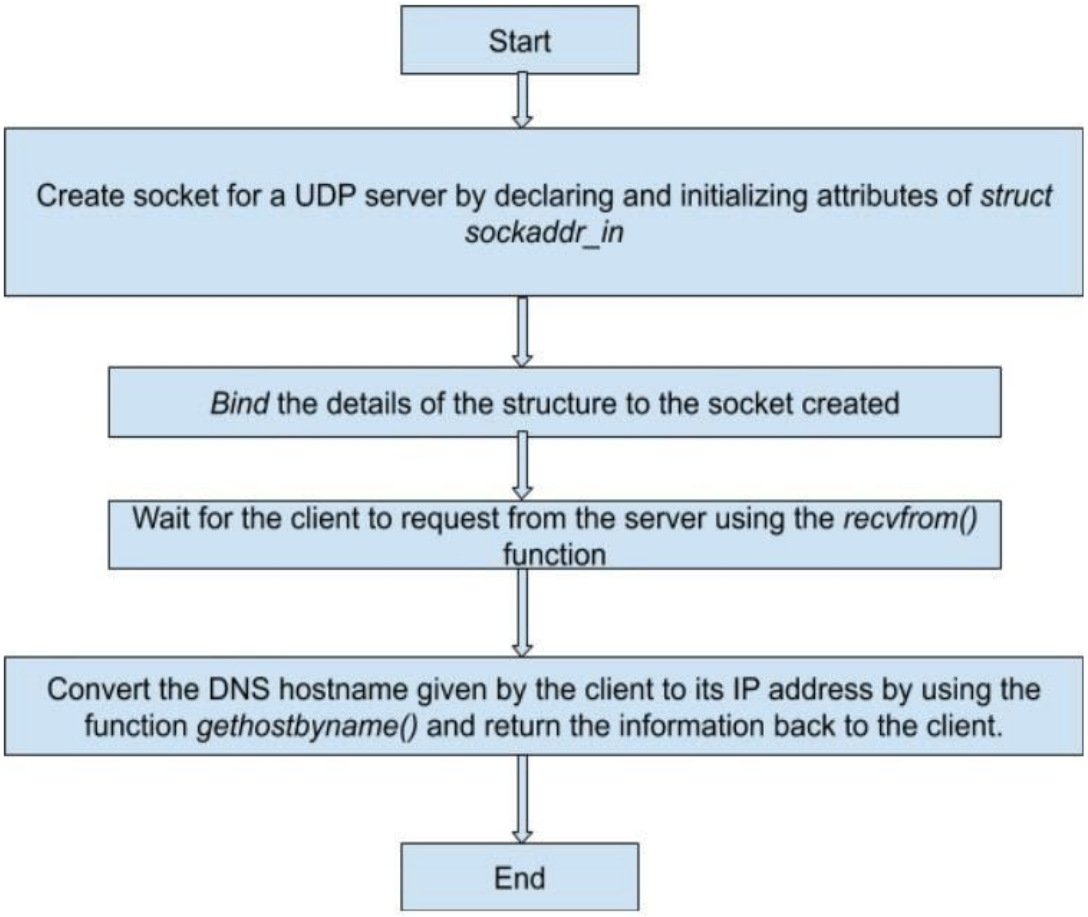
```c
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>

#define PORT 8180

int main(int argc, char *argv[])
{
    int fd1 = 0, n = 0;
    char buffer[1024];
    struct sockaddr_in s_address;
    if(argc != 2){
        printf("\n Usage: Enter the IP address also as second argument\n");
        return 1;
    }
    memset(buffer, '0',sizeof(buffer));
    if((fd1 = socket(AF_INET, SOCK_STREAM, 0)) < 0){
        printf("\n Error\n");
        return 1;
    }
    memset(&s_address, '0', sizeof(s_address));
    s_address.sin_family = AF_INET;
    s_address.sin_port = htons(PORT);
    if(inet_pton(AF_INET, argv[1], &s_address.sin_addr)<=0){
        printf("\n Error\n");
        return 1;
    }
    if( connect(fd1, (struct sockaddr *)&s_address, sizeof(s_address)) < 0){
       printf("\n Error\n");
       return 1;
    }
    while ( (n = read(fd1, buffer, sizeof(buffer)-1)) > 0){
        buffer[n] = 0;
        if(fputs(buffer, stdout) == EOF){
            printf("\n Error\n");
        }
    }
    return 0;
}
```

Algorithm:

Create sockets for both the server and the client. The server is *bind()* to the client socket whereas the client socket is *connect()* connected to the socket of the server. Both the client and the socket wait to create a three way TCP handshake. When it is created, the *accept()* call wakes up the server and responds to it by finding the timestamp using *time()* function call. This is finally returned to the client which requested for it.



Answer 3]



<u>Steps:</u>

Step 1 - Create a UDP socket using *socket()* and initializing the values of the structure *sockaddr_in.*

Step 2 - *Bind()* assigns the details specified in *sockaddr_in* to the socket created using *socket().*

Step 3 - *recvfrom()* lets the server wait for the client to give a request and receive it.

Step 4 - *gethostbyname()* is used by the server to convert the DNS hostname into the IP address and then send the information back to the client which requested it.

~/Desktop/IIT2018184_dns_server.c - Sublime Text (UNREGISTERED)

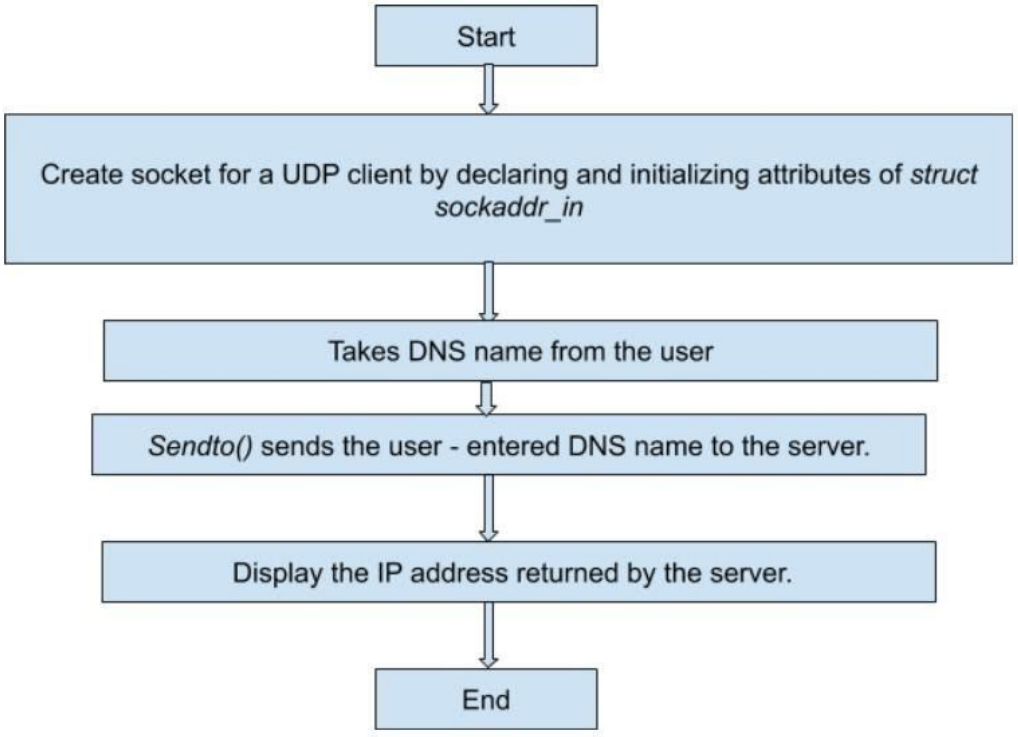
File Edit Selection Find View Goto Tools Project Preferences Help

IIT2018184_dns_server.c

```c
#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/select.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <math.h>
#include <ctype.h>
#include <time.h>
#include <stdio.h>

#include <arpa/inet.h>
#define PORT     8181
#define MAXLINE  1024

int func(char * hostname , char* ip){
  struct hostent *he;
  struct in_addr **addr_list;
  int i;
  if ( (he = gethostbyname(hostname) ) == NULL){
    herror("gethostbyname");
    return 1;
  }
  addr_list = (struct in_addr **) he->h_addr_list;
  for(i = 0; addr_list[i] != NULL; i++){
    strcpy(ip , inet_ntoa(*addr_list[i]));
    return 0;
  }
  return 1;
}

int main(int argc, char *argv[]) {
    char buffer[MAXLINE];
    int sock = socket(AF_INET, SOCK_DGRAM, 0);
    struct sockaddr_in s_address, cliaddr;
    memset(&s_address, 0, sizeof(s_address));
    s_address.sin_family = AF_INET;
    s_address.sin_port = htons(PORT);
    s_address.sin_addr.s_addr = INADDR_ANY;
    bind(sock, (const struct sockaddr *)&s_address, sizeof(s_address));
    socklen_t len = sizeof(cliaddr);
    int n = recvfrom(sock, (char *)buffer, MAXLINE, MSG_WAITALL, ( struct sockaddr *) &cliaddr, &len);
    buffer[n] = '\0';
    char ip[100];
    func(buffer , ip);
    printf("Server received DNS name: %s\n", buffer);
    printf("%s resolved to %s" , buffer , ip);
    printf("\n");
    close(sock);
}
```

Functions:

- *socket()* - to create an unnamed socket in the kernel and return the socket descriptor.
- *bind()* - to bind the details specified in the structure *sockaddr_in* to the socket descriptor created by the *socket()* function above.
- *recvfrom()* - to wait for the client to request and receive it.
- *gethostbyname()* - to convert the given DNS hostname to its IP address.



Steps:

Step 1 - Create a UDP socket using *socket()* and initializing the values of the structure *sockaddr_in.*

Step 2 - Inputs DNS hostname from the user.

Step 3 - *sendto()* sends the DNS hostname to the server and waits for the server to respond.

Step 4 -Displays the IP address returned by the server.

~/Desktop/IIT2018184_dns_client.c - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

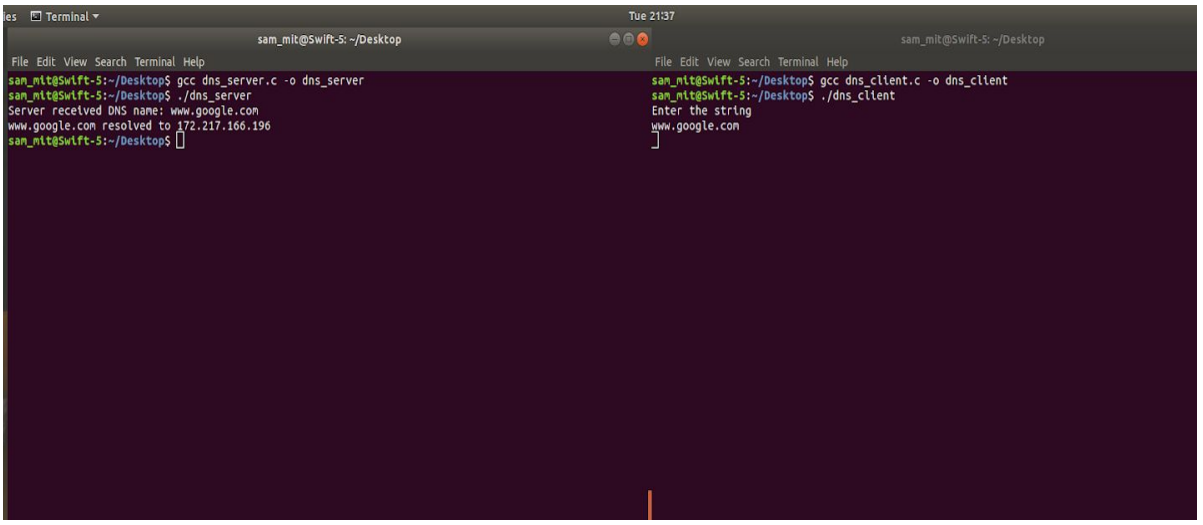
IIT2018184_dns_client.c

```c
#include <netdb.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/select.h>
#include <sys/socket.h>
#include <math.h>
#include <ctype.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define PORT  8181
#define MAXLINE 1024

int main(int argc, char *argv[]) {
  printf("Enter the string\n");
  char ch[100];
  char buffer[MAXLINE];
  scanf("%s", ch);
  int sock = socket(AF_INET, SOCK_DGRAM, 0);
  struct sockaddr_in out, serv;
  memset(&out, 0, sizeof(out));
  out.sin_family = AF_INET;
  out.sin_port = htons(PORT);
  out.sin_addr.s_addr = INADDR_ANY;
  sendto(sock, (const char *)ch, strlen(ch), MSG_CONFIRM, (const struct sockaddr *)&out, sizeof(out));
  socklen_t len = sizeof(serv);
  int n = recvfrom(sock, (char *)buffer, MAXLINE, MSG_WAITALL, ( struct sockaddr *) &serv, &len);
  buffer[n] = '\0';
  printf("Message from server: %s\n", buffer);

  close(sock);
}
```

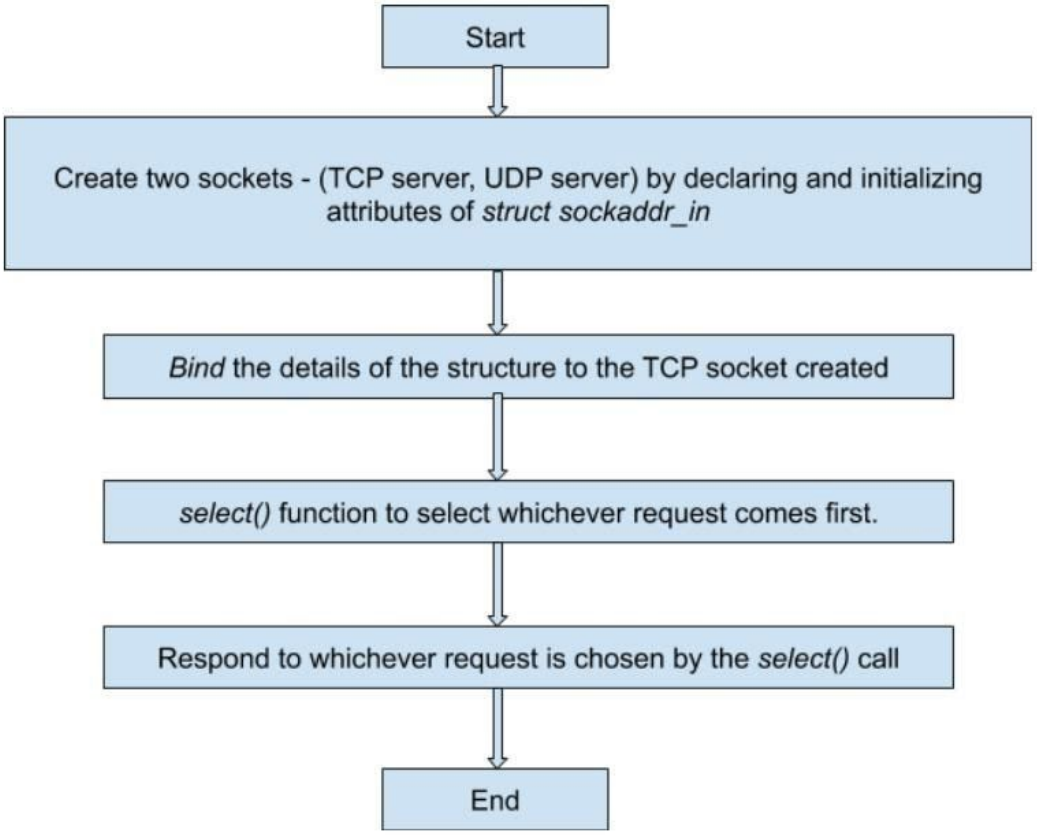
Algorithm:

Create sockets for both the server and the client. The client then requests the user to enter the DNS hostname and then send the entered information to the server.

The server then processes and using *gethostbyname()* function call returns the IP address of the given DNS hostname. The server then sends this information back to the client which displays it on the screen.



Answer 4]



Steps:

Step 1 - Create two sockets - (both UDP and TCP) using *socket()* and initializing the values of the structure *sockaddr_in.*

Step 2 - *Bind()* assigns the details specified in *sockaddr_in* to the socket created using *socket().*

Step 3 - *select()* lets the server to select whichever request comes first.

Step 4 -The server finally responds to the request selected by the *select()* function call.

Sublime Text Wed 09:37

~/Desktop/IIT2018184_combined_server.c - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

IIT2018184_combined_server.c

```c
#define MAXLINE 1024

int func(char * hostname , char* ip){
  struct hostent *he;
  struct in_addr **addr_list;
  int i;
  if ( (he = gethostbyname( hostname ) ) == NULL) {
    herror("gethostbyname");
    return 1;
  }
  addr_list = (struct in_addr **) he->h_addr_list;
  for(i = 0; addr_list[i] != NULL; i++) {
    strcpy(ip , inet_ntoa(*addr_list[i]) );
    return 0;
  }
  return 1;
}
int max(int x, int y){if(x > y)return x;else return y;}
int main(){
    int listenfd, connfd, udpfd, nready, maxfdp1;
    char buffer[MAXLINE];
    pid_t childpid;
    fd_set rset;
    ssize_t n;
    time_t ticks;
    socklen_t len;
    const int on = 1;
    struct sockaddr_in cliaddr, servaddr;
    void sig_chld(int);
    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
    listen(listenfd, 10);
    udpfd = socket(AF_INET, SOCK_DGRAM, 0);
    bind(udpfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
    FD_ZERO(&rset);
    maxfdp1 = max(listenfd, udpfd) + 1;
    for (;;) {
        FD_SET(listenfd, &rset);
        FD_SET(udpfd, &rset);
        nready = select(maxfdp1, &rset, NULL, NULL, NULL);
        char sendBuff[1025];
        if (FD_ISSET(listenfd, &rset)) {
            len = sizeof(cliaddr);
            connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);
            if ((childpid = fork()) == 0) {
                close(listenfd);
```

Line 1, Column 2 master (413) Spaces: 4 C

Algorithm:

Create sockets for the two servers and the clients. One of the clients then requests the user to enter the DNS hostname and then sends the entered information to the server. Other client request the timestamp from the server.

The server then selects the request whichever comes first using the *select()* function call. According to the selected request it serves first the selected request using the algorithm similar to those of Answer 1 and Answer 2 according to the selection. And finally respond to the client which displays it.

```
sam_mit@Swift-5: ~/Desktop
File  Edit  View  Search  Terminal  Help
sam_mit@Swift-5:~/Desktop$ gcc combined_server.c -o combined_server
sam_mit@Swift-5:~/Desktop$ ./combined_server

Message from UDP client: www.google.com
Server received DNS name: www.google.com
www.google.com resolved to 172.217.166.196
Message From TCP client:
```

```
sam_mit@Swift-5: ~/Desktop
File  Edit  View  Search  Terminal  Help
sam_mit@Swift-5:~/Desktop$ gcc time_client.c -o time_client
sam_mit@Swift-5:~/Desktop$ ./time_client

 Usage: ./time_client <ip of server>
sam_mit@Swift-5:~/Desktop$ ./time_client 127.0.0.1
Tue Apr 28 21:19:07 2020
sam_mit@Swift-5:~/Desktop$
```

```
sam_mit@Swift-5: ~/Desktop
File  Edit  View  Search  Terminal  Help
sam_mit@Swift-5:~/Desktop$ gcc dns_client.c -o dns_client
sam_mit@Swift-5:~/Desktop$ ./dns_client
Enter the string
www.google.com
sam_mit@Swift-5:~/Desktop$
```