ASSIGNMENT NO 09

Title:

Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each between two machines.

Theory:

Server-Client file transfer using UDP

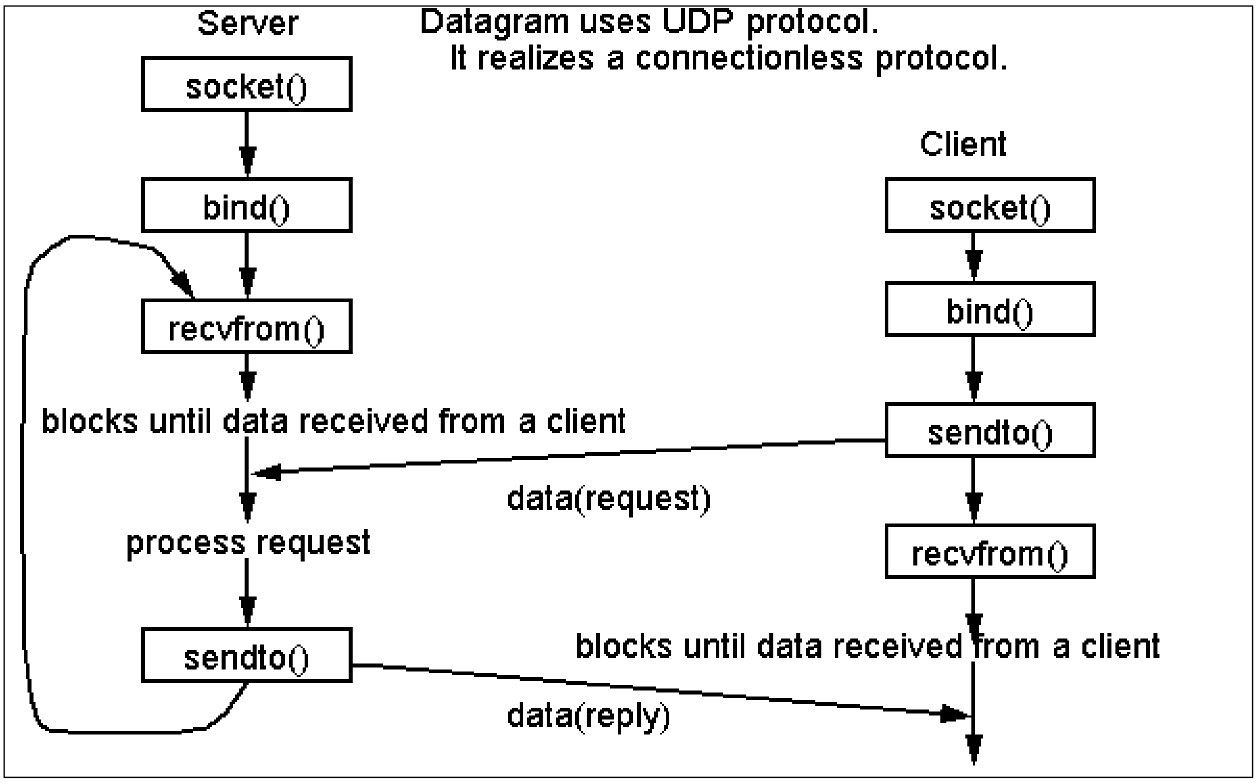
Server:

- Include appropriate header files.
- Create a UDP Socket using socket() system call.
- Fill in the socket address structure (with server information)
- Specify the port where the service will be defined to be used by client.
- Bind the address and port using bind() system call
- Receive a file name of text, audio or video from the Client using recvfrom() system call.
- Sends file to client using sendto() system call.
- Close the server socket
- Stop

Client:

- Include appropriate header files.
- Create a UDP Socket.
- Fill in the socket address structure (with server information)
- Specify the port of the Server, where it is providing service
- Send a file name to the server using sendto() system call.
- Receive a file from the Server using recvfrom() system call.
- Close the client socket
- Stop

Socket functions for UDP client/server in Connectionless Scenario


Server
Datagram uses UDP protocol.
It realizes a connectionless protocol.
socket()
bind()
recvfrom()
blocks until data received from a client
process request
sendto()
Client
socket()
bind()
sendto()
data(request)
recvfrom()
blocks until data received from a client
data(reply)

Testing

1. Run Wireshark tool

2. Run client and server program

3. Send and receive file

4. Capture UDP packets in Wireshark

Program:

Server.c

```
#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>
```

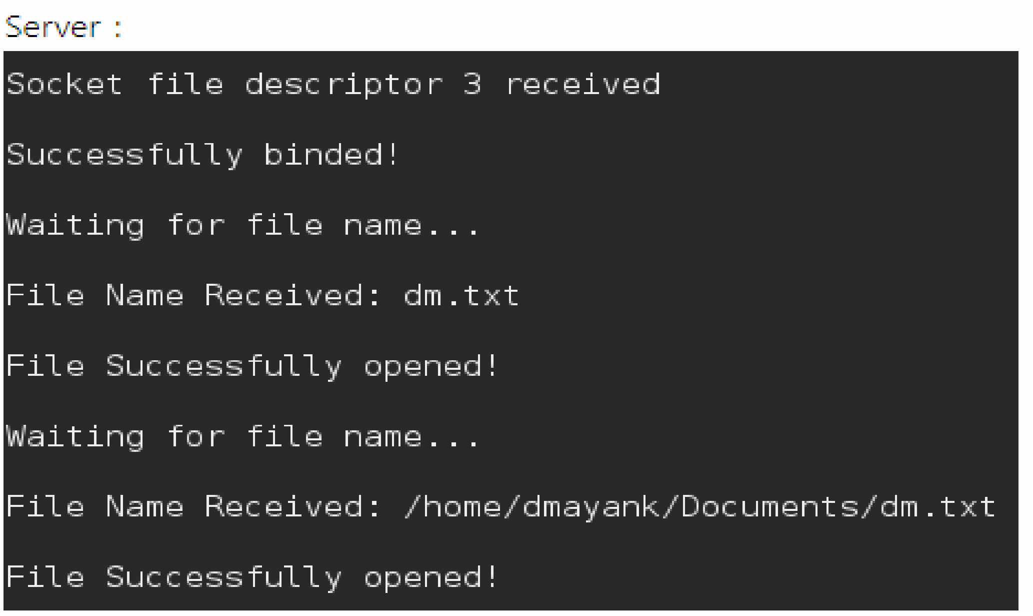
```
#include <sys/types.h>
#include <unistd.h>
#define IP_PROTOCOL 0
#define PORT_NO 15050
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0
#define nofile "File Not Found!"
// funtion to clear buffer
void clearBuf(char* b) {
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '';
}
char Cipher(char ch) {
    return ch ^ cipherKey;
}
int sendFile(FILE* fp, char* buf, int s)
{
    int i, len;
    if (fp == NULL) {
        strcpy(buf, nofile);
        len = strlen(nofile);
        buf[len] = EOF;
        for (i = 0; i <= len; i++)
            buf[i] = Cipher(buf[i]);
        return 1;
    }
    char ch, ch2;
    for (i = 0; i < s; i++) {
        ch = fgetc(fp);
        ch2 = Cipher(ch);
        buf[i] = ch2;
        if (ch == EOF)
            return 1;
    }
    return 0;
}
int main()
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    int addrlen = sizeof(addr_con);
    addr_con.sin_family = AF_INET;
    addr_con.sin_port = htons(PORT_NO);
    addr_con.sin_addr.s_addr = INADDR_ANY;
    char net_buf[NET_BUF_SIZE];
    FILE* fp;
    sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);
    if (sockfd < 0)
        printf("file descriptor not received!!");
    else
        printf("file descriptor %d received", sockfd);
    if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(addr_con)) == 0)
        printf("Successfully binded!");
```

```
else
    printf("Binding Failed!");
  while (1) {
    printf("Waiting for file name...");
    clearBuf(net_buf);
    nBytes = recvfrom(sockfd, net_buf,
                      NET_BUF_SIZE, sendrecvflag,
                      (struct sockaddr*)&addr_con, &addrlen);
    fp = fopen(net_buf, "r");
    printf("File Name Received: %s", net_buf);
    if (fp == NULL)
        printf("File open failed!");
    else
        printf("File Successfully opened!");
      while (1) {
        if (sendFile(fp, net_buf, NET_BUF_SIZE)) {
            sendto(sockfd, net_buf, NET_BUF_SIZE,
                   sendrecvflag,
                (struct sockaddr*)&addr_con, addrlen);
            break;
        }
        sendto(sockfd, net_buf, NET_BUF_SIZE,
               sendrecvflag,
            (struct sockaddr*)&addr_con, addrlen);
        clearBuf(net_buf);
    }
    if (fp != NULL)
        fclose(fp);
    }
    return 0;
}
```

Client.c

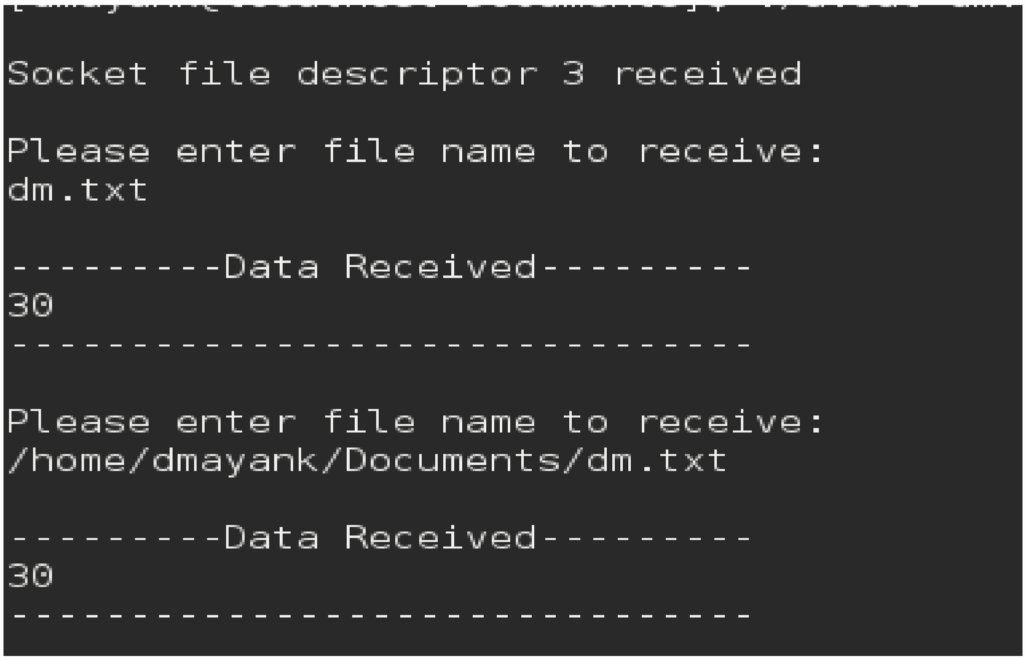
```
A#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#define IP_PROTOCOL 0
#define IP_ADDRESS "127.0.0.1" // localhost
#define PORT_NO 15050
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0
void clearBuf(char* b) {
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '';
}
char Cipher(char ch) {
    return ch ^ cipherKey;
}
int recvFile(char* buf, int s) {
    int i;
    char ch;
    for (i = 0; i < s; i++) {
        ch = buf[i];
        ch = Cipher(ch);
        if (ch == EOF)
            return 1;
        else
            printf("%c", ch);
    }
    return 0;
}
int main() {
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    int addrlen = sizeof(addr_con);
    addr_con.sin_family = AF_INET;
    addr_con.sin_port = htons(PORT_NO);
    addr_con.sin_addr.s_addr = inet_addr(IP_ADDRESS);
    char net_buf[NET_BUF_SIZE];
    FILE* fp;
    sockfd = socket(AF_INET, SOCK_DGRAM,
                    IP_PROTOCOL);
    if (sockfd < 0)
        printf("file descriptor not received!!");
    else
        printf("file descriptor %d received", sockfd);
    while (1) {
        printf("Please enter file name to receive:");
        scanf("%s", net_buf);

        sendto(sockfd, net_buf, NET_BUF_SIZE, sendrecvflag, (struct sockaddr*)&addr_con,
addrlen);

        printf("---------Data Received---------");

        while (1) {

            clearBuf(net_buf);

            nBytes = recvfrom(sockfd, net_buf, NET_BUF_SIZE, sendrecvflag, (structsockaddr*)&addr_con,
&addrlen);

            if (recvFile(net_buf, NET_BUF_SIZE)) {

                break;

            }

        }

        printf("-------------------------------");

    }

    return 0;

}
```

Output:

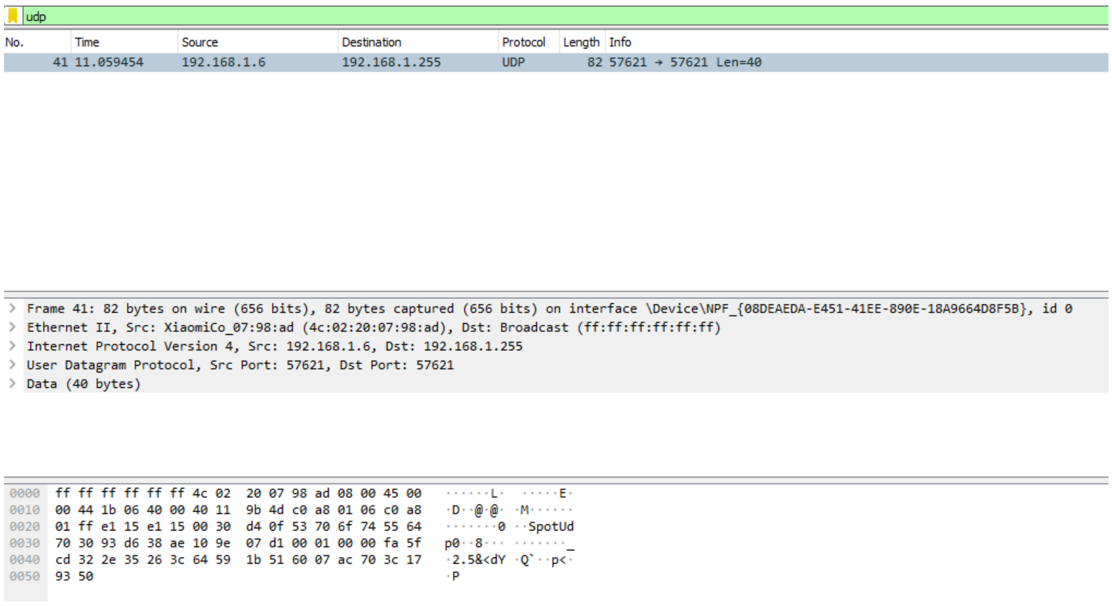


```
Server :
Socket file descriptor 3 received

Successfully binded!

Waiting for file name...

File Name Received: dm.txt

File Successfully opened!

Waiting for file name...

File Name Received: /home/dmayank/Documents/dm.txt

File Successfully opened!
```

Client :



```
Socket file descriptor 3 received

Please enter file name to receive:
dm.txt

---------Data Received---------
30
-------------------------------

Please enter file name to receive:
/home/dmayank/Documents/dm.txt

---------Data Received---------
30
-------------------------------
```



```
udp
No.   Time        Source        Destination     Protocol  Length  Info
   41 11.059454   192.168.1.6   192.168.1.255   UDP           82  57621 → 57621 Len=40
```

```
> Frame 41: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface \Device\NPF_{08DEAEDA-E451-41EE-890E-18A9664D8F5B}, id 0
> Ethernet II, Src: XiaomiCo_07:98:ad (4c:02:20:07:98:ad), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 192.168.1.6, Dst: 192.168.1.255
> User Datagram Protocol, Src Port: 57621, Dst Port: 57621
> Data (40 bytes)
```

```
0000  ff ff ff ff ff ff 4c 02  20 07 98 ad 08 00 45 00   ······L· ·····E·
0010  00 44 1b 06 40 00 40 11  9b 4d c0 a8 01 06 c0 a8   ·D··@·@· ·M······
0020  01 ff e1 15 e1 15 00 30  d4 0f 53 70 6f 74 55 64   ·······0 ··SpotUd
0030  70 30 93 d6 38 ae 10 9e  07 d1 00 01 00 00 fa 5f   p0··8··· ·······_
0040  cd 32 2e 35 26 3c 64 59  1b 51 60 07 ac 70 3c 17   ·2.5&<dY ·Q`··p<·
0050  93 50                                              ·P
```

Conclusion:

Hence we have studied UDP Socket Programming in C for file transmission of text, audio and video and captured UDP packets in Wireshark tool.