COMPUTER NETWORKS PRACTICE ASSIGNMENT 4

Name: Vinayak Sethi Roll No: COE18B061

File Transfer using TCP

The general idea of sending the file is opening the file in **'rb'** mode in client side and storing the content of it in sendbuffer using fread and then send the character bit by bit using **send() function** and then in server side opening the file in **'wb'** mode and and receiving the file using **recv() function** and write the contents in file using fwrite function.

Filename: TCP_Client_ft.c

~/Documents/Computer Networking/File_Transfer/

File Edit Selection Find View Goto Tools Project Preferences Help

TCP_Client_ft.c × TCP_Server_ft.c ×

```c
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<string.h>
#include<libgen.h>
#include<time.h>

#define buffsize 1024

int main()
{
    int client_socket,sin_size, check;
    char sendbuffer[buffsize];
    struct sockaddr_in server_address;

    //create a socket
    client_socket = socket(AF_INET,SOCK_STREAM,0);
    if(client_socket == -1)
    {
        printf("\nSocket Creation Failure\n");
        exit(EXIT_FAILURE);
    }

    //specify an address for the socket
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(9009);
    server_address.sin_addr.s_addr = INADDR_ANY;

    sin_size = sizeof(struct sockaddr_in);

    //connect to server
    if(connect(client_socket,(struct sockaddr *)&server_address, sin_size) == 0)
        printf("Connect Successful\n");

    char path[buffsize];
    printf("Enter the complete path of the filename you wish to send : ");
    scanf("%s",path);
    char *filename = basename(path); //return the last component of a pathname
    printf("%s\n", filename);

    send(client_socket, filename, strlen(filename), 0);

    FILE *fp = fopen(path,"rb");
    if (fp == NULL)
```

TCP_Client_ft.c | TCP_Server_ft.c

```
    {
        printf("Cannot open the file\n");
        exit(EXIT_FAILURE);
    }

    clock_t t;
    t = clock();

    while((check = fread(sendbuffer, 1, sizeof(sendbuffer), fp)) > 0)
    {
        send(client_socket, sendbuffer, check, 0);
    }

    fclose(fp);
    t = clock() - t;
    double time_taken = ((double)t)/CLOCKS_PER_SEC; //in seconds

    printf("File sent successfully ...\n");
    printf("File transfer took %f seconds to execute.\n", time_taken);

    close(client_socket);
    return 0;
}
```

Filename: TCP_Server_ft.c

~/Documents/Computer Networking/File_Transfer/TCP/

File Edit Selection Find View Goto Tools Project Preferences Help

TCP_Client_ft.c | TCP_Server_ft.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<string.h>
#include<time.h>

#define buffsize 1024

int main()
{
    int server_socket,client_socket,sin_size,check;
    char buffer[buffsize];
    struct sockaddr_in server_address, client_address;

    //create a socket
    server_socket = socket(AF_INET,SOCK_STREAM,0);
    if(server_socket == -1)
    {
        printf("\nSocket Creation Failure\n");
        exit(EXIT_FAILURE);
    }

    //specify an address for the socket
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(9009);
    server_address.sin_addr.s_addr = INADDR_ANY;

    //bind with the client
    if( bind(server_socket, (const struct sockaddr *)&server_address, sizeof(server_address)) < 0)
    {
        printf("Could not bind to Client\n");
        exit(EXIT_FAILURE);
    }

    //listen to the incoming client request
    if(listen(server_socket, 10) == 0)
        printf("Listen successful\n");
```

TCP_Client_ft.c | TCP_Server_ft.c

```c
    //accept a connection request from client
    sin_size = sizeof(struct sockaddr_in);
    if((client_socket = accept(server_socket, (struct sockaddr *)&client_address,&sin_size)) > 0)
        printf("Accept Successful\n");

    char filename[buffsize];
    memset (filename, '\0', sizeof(filename));
    recv(client_socket, filename, sizeof(filename), 0);
    printf("File received is : ");
    printf("%s\n", filename);


    clock_t t;
    t = clock();

    FILE *fp = fopen(filename, "wb");
    if(fp!=NULL)
    {
        while((check = recv(client_socket,buffer, sizeof(buffer), 0)) > 0)
        {
            fwrite(buffer, 1, check, fp);
        }
    fclose(fp);
    }

    t = clock() - t;
    double time_taken = ((double)t)/CLOCKS_PER_SEC; //in seconds

    printf("File received successfully...\n");
    printf("File received in %f seconds.\n", time_taken);

    //close the socket
    close(client_socket);
    close(server_socket);

    return 0;
}
```

Output:

vinayak@vinayak-Swift-SF315-52G: ~/Documents/Computer Networking/File_Transfer/TCP/Client

File Edit View Search Terminal Help

```
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Client$
make TCP_Client_ft
make: 'TCP_Client_ft' is up to date.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Client$
./TCP_Client_ft
Connect Successful
Enter the complete path of the filename you wish to send : /home/vinayak/Documents/india-f
lag.gif
india-flag.gif
File sent successfully ...
File transfer took 0.005138 seconds to execute.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Client$
./TCP_Client_ft
Connect Successful
Enter the complete path of the filename you wish to send : TAG.mp3
TAG.mp3
File sent successfully ...
File transfer took 0.010705 seconds to execute.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Client$
./TCP_Client_ft
Connect Successful
Enter the complete path of the filename you wish to send : rauf_faik.mp4
rauf_faik.mp4
File sent successfully ...
File transfer took 0.061589 seconds to execute.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Client$
```

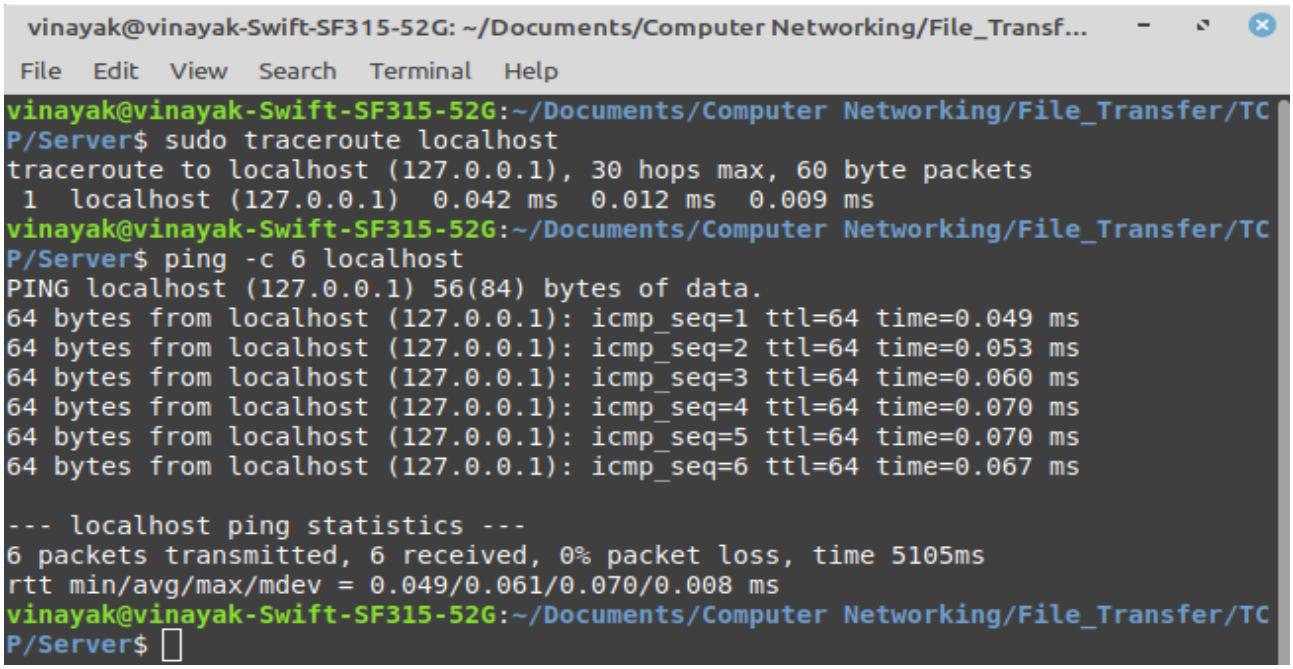
vinayak@vinayak-Swift-SF315-52G: ~/Documents/Computer Networking/File_Transfer/TCP/Ser...

File Edit View Search Terminal Help

```
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Server$
 make TCP_Server_ft
make: 'TCP_Server_ft' is up to date.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Server$
 ./TCP_Server_ft
Listen successful
Accept Successful
File received is : india-flag.gif
File received successfully...
File received in 0.008587 seconds.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Server$
 ./TCP_Server_ft
Listen successful
Accept Successful
File received is : TAG.mp3
File received successfully...
File received in 0.013118 seconds.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Server$
 ls
india-flag.gif  TAG.mp3  TCP_Server_ft  TCP_Server_ft.c
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Server$
 ./TCP_Server_ft
Listen successful
Accept Successful
File received is : rauf_faik.mp4
File received successfully...
File received in 0.063613 seconds.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Server$
 ls
india-flag.gif  rauf_faik.mp4  TAG.mp3  TCP_Server_ft  TCP_Server_ft.c
vinayak@vinayak-Swift-SF315-52G:~/Documents/Computer Networking/File_Transfer/TCP/Server$
```

- Using **ls** we can verify server has received the file.

Latency Test:



File Transfer using UDP

Same concept is used like TCP file transfer logic but in UDP there is chances of data loss, so to reduce data loss we have used sleep command because client keep sending data to server and server take time to write to file so to slow down this process sleep command is used. Also some functions are changed as per UDP protocol, like **sendto()** in client side and **recvfrom()** in server side is used.
Also **fgetc** command is used to read the content byte by byte and **fputc** command is used to write byte by byte to file in server side.

Filename: UDP_Client_ft.c

~/Documents/Computer Networking/File_Transfer/UDP/Client/UDP_Client_ft.c - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

UDP_Client_ft.c UDP_Server_ft.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<string.h>
#include<libgen.h>
#include<time.h>

#define buffsize 1024

int main()
{
    int client_socket,check;
    struct sockaddr_in server_address;

    //create a socket
    client_socket = socket(AF_INET,SOCK_DGRAM,0);
    if(client_socket == -1)
    {
        printf("\nSocket Creation Failure\n");
        exit(EXIT_FAILURE);
    }

    //specify an address for the socket
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(9009);
    server_address.sin_addr.s_addr = INADDR_ANY;

    socklen_t length = sizeof(server_address);

    sendto(client_socket,"Hello server", strlen("Hello server"),0,(struct sockaddr *)&server_address, sizeof(server_address));

    char path[buffsize];
    printf("Enter the complete path of the filename you wish to send : ");
    scanf("%s",path);
    char *filename = basename(path); //return the last component of a pathname
    printf("%s\n", filename);

    sendto(client_socket, filename, strlen(filename), 0,(struct sockaddr *)&server_address, sizeof(server_address));

    FILE *fp = fopen(path,"rb");
    if (fp == NULL)
    {
        printf("Cannot open the file\n");
        exit(EXIT_FAILURE);
    }

    clock_t t;
    t = clock();

    while((check = fgetc(fp)) != EOF)
    {
        sendto(client_socket, &check, sizeof(check), 0,(struct sockaddr *)&server_address, sizeof(server_address));
        sleep(0);
    }

    //send the EOF to signal file end
    sendto(client_socket, &check, sizeof(check), 0,(struct sockaddr *)&server_address, sizeof(server_address));

    fclose(fp);
    t = clock() - t;
    double time_taken = ((double)t)/CLOCKS_PER_SEC; //in seconds

    printf("File sent successfully ...\n");
    printf("File transfer took %f seconds to execute.\n", time_taken);

    close(client_socket);
    return 0;
}
```

Filename: UDP_Server_ft.c

~/Documents/Computer Networking/File_Transfer/UDP/Server/UDP_Server_ft.c - Sublime

File Edit Selection Find View Goto Tools Project Preferences Help

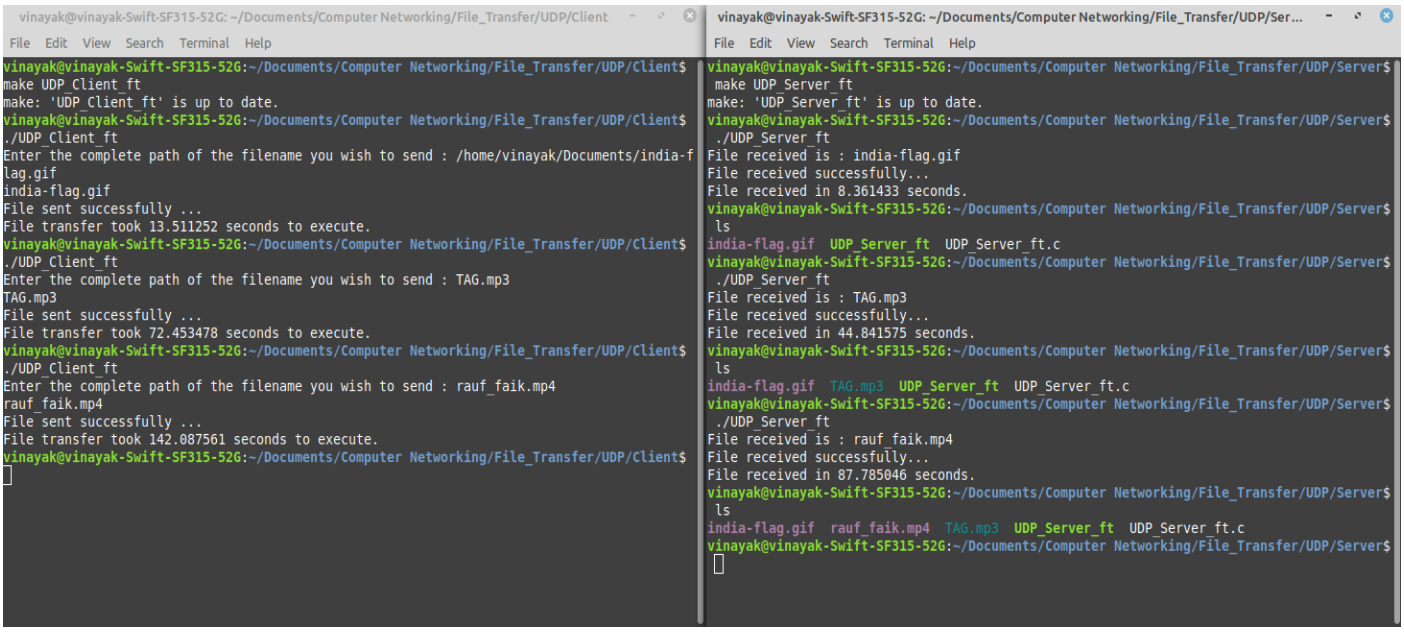
UDP_Client_ft.c | UDP_Server_ft.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<string.h>
#include<time.h>

#define buffsize 1024

int main()
{
    int server_socket,check,store;
    char buffer[buffsize];
    struct sockaddr_in server_address, client_address;

    //create a socket
    server_socket = socket(AF_INET,SOCK_DGRAM,0);
    if(server_socket == -1)
    {
        printf("\nSocket Creation Failure\n");
        exit(EXIT_FAILURE);
    }

    //specify an address for the socket
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(9009);
    server_address.sin_addr.s_addr = INADDR_ANY;

```

UDP_Client_ft.c | UDP_Server_ft.c

```c
    //bind with the client
    if( bind(server_socket, (const struct sockaddr *)&server_address, sizeof(server_address)) < 0)
    {
        printf("Could not bind to Client\n");
        exit(EXIT_FAILURE);
    }

    socklen_t length = sizeof(server_address);
    recvfrom(server_socket, buffer, sizeof(buffer),0,(struct sockaddr *)&client_address,&length);

    char filename[buffsize];
    memset (filename, '\0', sizeof(filename));
    recvfrom(server_socket, filename, sizeof(filename), 0,(struct sockaddr *)&client_address,&length);
    printf("File received is : ");
    printf("%s\n", filename);

    clock_t t;
    t = clock();

    FILE *fp = fopen(filename, "wb");
    if(fp!=NULL)
    {
        while(store != EOF)
        {
            recvfrom(server_socket, &store, sizeof(store), 0,(struct sockaddr *)&client_address,&length);
            fputc(store, fp);
        }
    }

    fclose(fp);
    t = clock() - t;
    double time_taken = ((double)t)/CLOCKS_PER_SEC; //in seconds

    printf("File received successfully...\n");
    printf("File received in %f seconds.\n", time_taken);

    //close the socket
    close(server_socket);
    return 0;
}
```

Output:



- Using **ls** we can verify server has received the file.

Latency Test: