Introduction

The project was to implement a network file transfer application based on TCP. The server is concurrent indicating the server can handle multiple file transfer session. The client is able to download/upload files to/from the server. The client is able to request a change of directory on the server where it would perform the download/upload operation. The client is able to request a list of files in each directory.

Sockets are interior endpoints built for sending and receiving data. Sockets are a combination of an IP address and a Port number. Socket programming is a way to connect nodes/sockets of a network to communicate with each other. Servers nodes are devoted to manage network resources. Clients nodes will request information or services from the server. The clients sockets will initiate a connection with the server while the server socket will receive the connection request to establish a connect with the client before any operation can be performed. Once the connection is established, the client can request any of the operations authorized by the server. Below in **Figure 1** shows the implementation of the TCP interaction between the client and server.

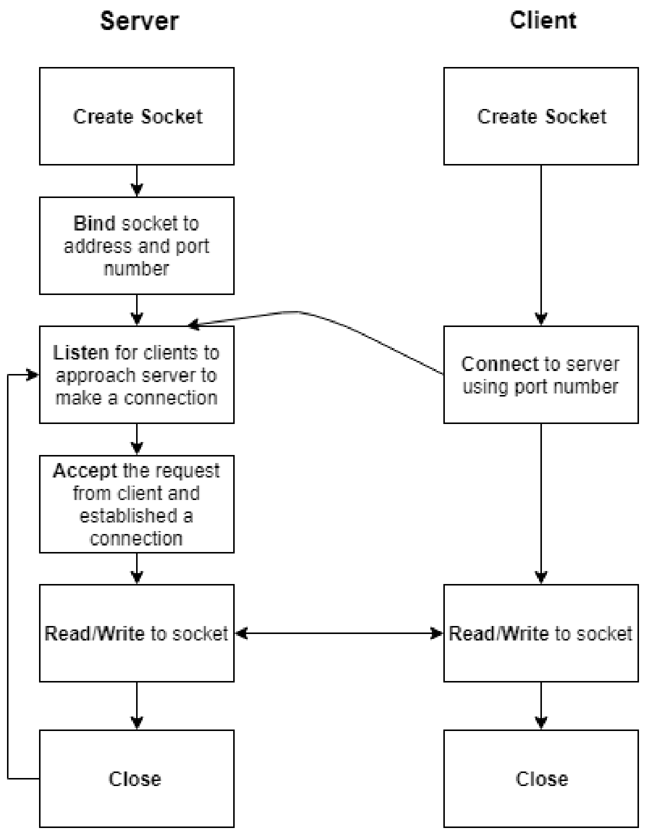

Server
Client
Create Socket
Create Socket
Bind socket to address and port number
Listen for clients to approach server to make a connection
Connect to server using port number
Accept the request from client and established a connection
Read/Write to socket
Read/Write to socket
Close
Close

Figure 1: TCP Client and Server

Description of Program

Process data unit(PDU) are the unit of information that are exchanged between the client and server during any operation. Below in **Figure 2** shows the format of the PDU used in this project.

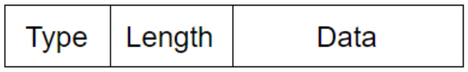


Figure 2: PDU Format

The type field of the PDU specifies the PDU type. The length field of the PDU indicates the length of the data field. The data field of the PDU contains the data to be transferred. The PDU is created using structure in C; the type and data fields is of type char and the length field is of type int.

Table 1: PDU Structure Table

Type	**Length**	**Data**
D - Download x File	Length of File name	File name
U - Upload x File	Length of File name	File name
R - Ready	N/A	N/A
F - File	Size of File	File Data
E - Error	Size of Error Message	Error message
P - Change Directory	Length of Path name	Path name
L - List of in x Directory	Length of Path name	Path name
I - List of File in x Directory	Length of all file names in directory	List of all file names

Listed above in the table are the types of PDUs that are exchanged between the client and server in this project.

File Download

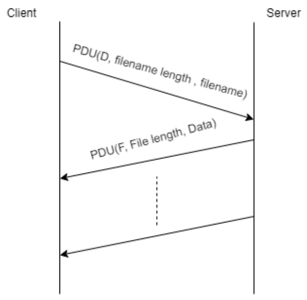


Figure 3: File Download PDU Exchanges

The client request a download operation by sending a PDU with type D to the server. The server then sends a type F PDU containing the file that will be downloaded onto the client. If the file name in the Type D PDU is not found in the download directory, the server will send a Type E PDU to indicate that an error has occurred.

File Upload

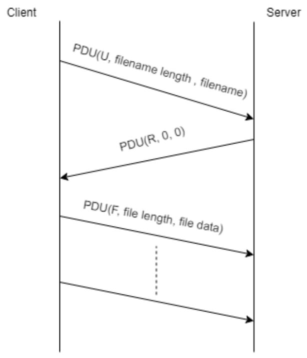

Client
Server
PDU(U, filename length , filename)
PDU(R, 0, 0)
PDU(F, file length, file data)

Figure 4: File Upload PDU Exchanges

The client request an upload operation by sending a PDU with Type U to the server. The server then sends a type R PDU indicating that the server is ready to receive the file that will be uploaded onto the current directory of the server. The client then sends a Type F PDU containing the file to be uploaded to the server. If the file name in the the Type U PDU is already on the server, the server will send a Type E PDU.

Error Reporting

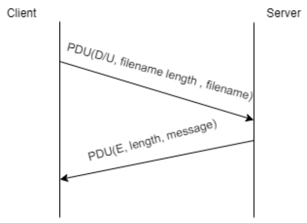


Figure 5: Error Reporting PDU Exchanges

The error reporting Type E PDU will only be sent by the server when the client request to download a file that does not exist on the current directory that the server or when the client request to upload a file that is already on the server.The type E PDU will send a different error message depending on the error uploading/downloading.

Change Directory

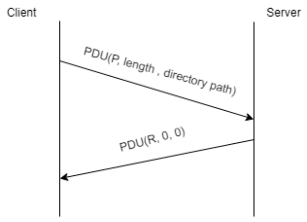


Figure 6: Directory Change PDU Exchanges

The client request a change of directory by sending a type P PDU with the directory path. The server will send a type R PDU when the directory is successfully changed.

List Directory

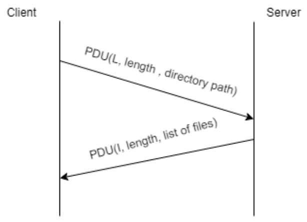
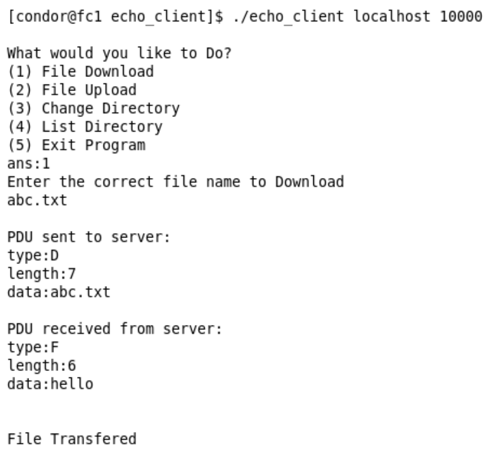

Client
Server
PDU(L, length , directory path)
PDU(I, length, list of files)

Figure 7: List Directory PDU Exchanges

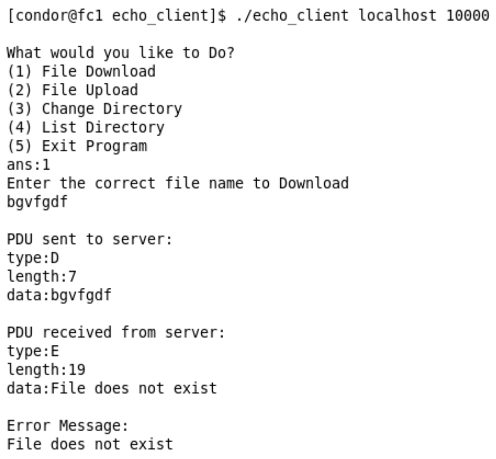
The client requests a list of files in a chosen directory by sending a type L PDU. The server responds with a type I PDU containing the list of files in the chosen directory.

Observations



```
[condor@fc1 echo_client]$ ./echo_client localhost 10000

What would you like to Do?
(1) File Download
(2) File Upload
(3) Change Directory
(4) List Directory
(5) Exit Program
ans:1
Enter the correct file name to Download
abc.txt

PDU sent to server:
type:D
length:7
data:abc.txt

PDU received from server:
type:F
length:6
data:hello


File Transfered
```

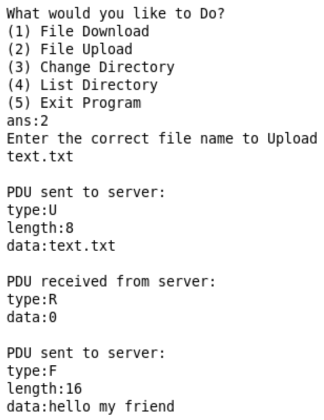
(a) Successful Client Download



```
[condor@fc1 echo_client]$ ./echo_client localhost 10000

What would you like to Do?
(1) File Download
(2) File Upload
(3) Change Directory
(4) List Directory
(5) Exit Program
ans:1
Enter the correct file name to Download
bgvfgdf

PDU sent to server:
type:D
length:7
data:bgvfgdf

PDU received from server:
type:E
length:19
data:File does not exist

Error Message:
File does not exist
```

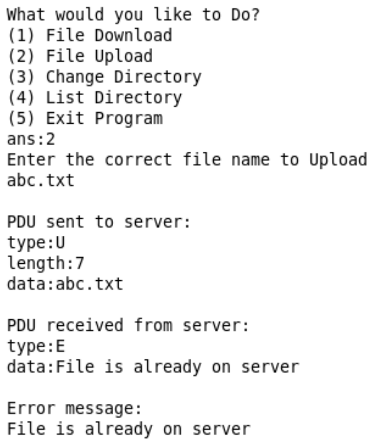
(b) Unsuccessful Client Download

Figure 8: File Download



```
What would you like to Do?
(1) File Download
(2) File Upload
(3) Change Directory
(4) List Directory
(5) Exit Program
ans:2
Enter the correct file name to Upload
text.txt

PDU sent to server:
type:U
length:8
data:text.txt

PDU received from server:
type:R
data:0

PDU sent to server:
type:F
length:16
data:hello my friend
```

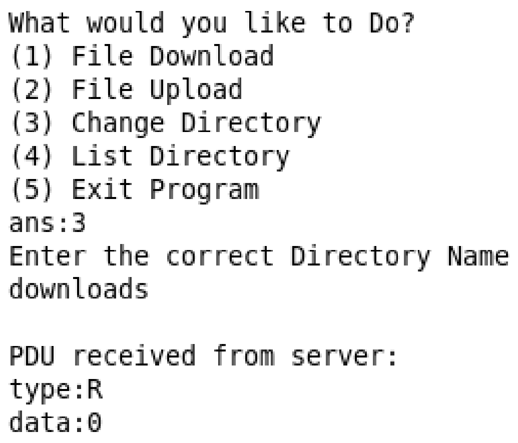
(a) Successful Client Upload



```
What would you like to Do?
(1) File Download
(2) File Upload
(3) Change Directory
(4) List Directory
(5) Exit Program
ans:2
Enter the correct file name to Upload
abc.txt

PDU sent to server:
type:U
length:7
data:abc.txt

PDU received from server:
type:E
data:File is already on server

Error message:
File is already on server
```

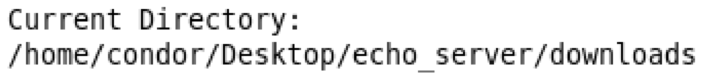
(b) Unsuccessful Client Upload

Figure 9: File Upload



```
What would you like to Do?
(1) File Download
(2) File Upload
(3) Change Directory
(4) List Directory
(5) Exit Program
ans:3
Enter the correct Directory Name
downloads

PDU received from server:
type:R
data:0
```

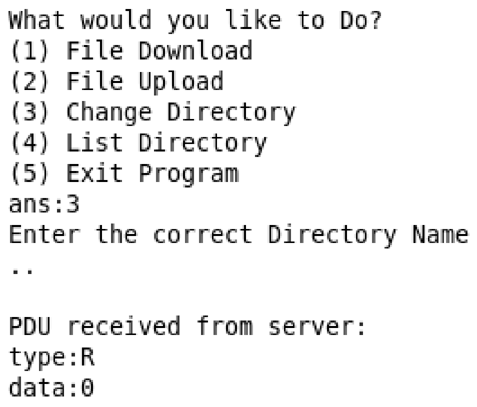
(a) Change Directory, Client input



```
Current Directory:
/home/condor/Desktop/echo_server/downloads
```

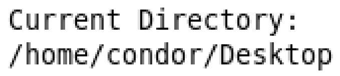
(b) Change Directory, Server Debugging

Figure 10: Change Directory to Child Folder



```
What would you like to Do?
(1) File Download
(2) File Upload
(3) Change Directory
(4) List Directory
(5) Exit Program
ans:3
Enter the correct Directory Name
..

PDU received from server:
type:R
data:0
```

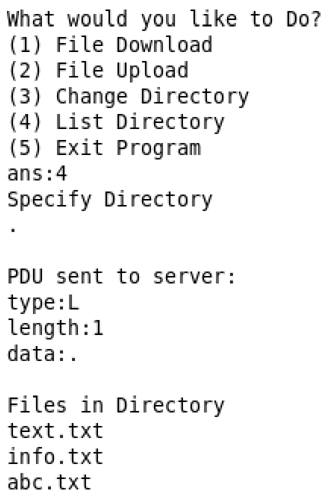
(a) Change Directory, Client input



```
Current Directory:
/home/condor/Desktop
```

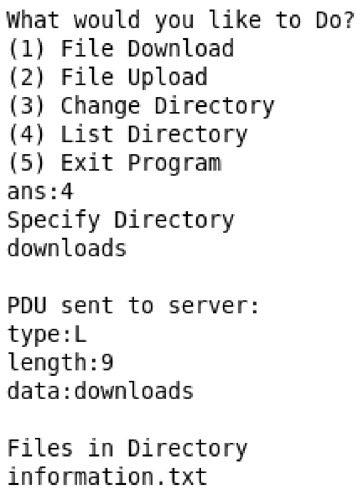
(b) Change Directory, Server Debugging

Figure 11: Change Directory to Parent Folder



```
What would you like to Do?
(1) File Download
(2) File Upload
(3) Change Directory
(4) List Directory
(5) Exit Program
ans:4
Specify Directory
.

PDU sent to server:
type:L
length:1
data:.

Files in Directory
text.txt
info.txt
abc.txt
```

(a) List Current Directory, Client input



```
What would you like to Do?
(1) File Download
(2) File Upload
(3) Change Directory
(4) List Directory
(5) Exit Program
ans:4
Specify Directory
downloads

PDU sent to server:
type:L
length:9
data:downloads

Files in Directory
information.txt
```

(b) Change Specified Directory, Client input

Figure 12: List Server Directory

Analysis

Figure 8(a) illustrates an successful download request to the client from the server. The client requests the abc.txt file with a PDU denoted with the 'D' type. It exists on the server therefore, the server sends the file's contents with a PDU denoted with a 'F' type. Figure 8(b) illustrates an unsuccessful download request. The client requests a file which does not exist on the server. The server sends an error PDU denoted with the 'E' type an error message as the data.

Figure 9(a) demonstrates a successful File upload request from the client to the server. The client requests to upload the text.txt file with a PDU denoted with the 'U' type. The file is not already on the server, therefore the server sends the ready PDU denoted with the 'R' type. The client then sends the the file data to the server with a PDU denoted with the 'F' type. Figure 9(b) demonstrates a unsuccessful File upload request. The client requests to upload a files which already exists on the server. The server sends an error PDU denoted with the 'E' type an error message as the data.

Figures 10(a) and 11(a) both demonstrate a change of directory from the clients perspective while figures 10(b) and 11(b) both show the change in directory of the server which the client requests. The client requests a directory with the PDU denoted with the 'P' type, then gets a PDU denoted with 'R' type as a response back from the server indicating that the server is ready.

Figures 12(a) and 12(b) both illustrate the listing of text files from a specified path which the client requests. First the client sends a PDU denoted with 'L' type that contains the specified path of that the server must create a list from. The server then complies a list and sends a PDU denoted with 'I' type containing the requested information.

Conclusion

The purpose of this project was to create a file transfer application using TCP protocol. The theoretical operation of the file transfer program matched our results attained through the implementation of the program. All of the cases and flow diagrams that needed to be incorporated in the file transfer application were included and behaved as expected in our project. The program had additional features, it was not only able to list files in it's own current directory but also able to list files in another directory without the need to change directories first.

The secondary goals of expanding our knowledge of C programming and the TCP protocol was also achieved through the development of the file transfer project.

Appendix

1. A. S. Tanenbaum and D. Wetherall, Computer Networks, Fifth Edition. Prentice Hall, 2010.

2. “COE768: Network File TransferApplication,” D2L. [Online]. Available: https://courses.ryerson.ca/d2l/le/content/299210/viewContent/2510697/View.