Project Part 1

CS 4504 01

Gavin Frey, Cameron Lowry, Shaun Teague, Tanner Velzy, John Sheffield

Abstract

Sending and downloading files from one system to another is integral to data transfer in our modern world. In order to do this a client and a server must interact to share and download files. Our Goal for this was to modify 4 modules in order to allow files such as mp4 or videos files to be transferred and downloaded to another system. We accomplished this by changing key parts of the code to convert the file to byte array. The client then sends the bytes individually to the server for it to convert back to the file.

In our runs, we found that the size of the file matters greatly as well as the time it takes for the server and client to connect through the serverRouter. While the connection time for the server and client will become less crucial, the file size will always play a massive part in the time it takes for the files to transmit no matter what the file type is.

Introduction

For this part of the project, we used TCP modules to allow a client to request a file from a server using a server router. We were supplied with the starting code of 4 modules SThread, TCPClient, TCPServer, and TCPServerRouter. At first we exchanged text files before changing parts of the code to allow bigger files like mp4 or video files across the network. This was done by the client transforming the file into bytes, sending them to the server and then the server transforming them back to a file.

Design Approach

The first part of the project has us implement a distributed Client-Server file sharing system. Subnetworks containing devices using wired and wireless connections exist, but we exclusively relied on wireless connections. Devices, or nodes, located in the system can act as clients and servers. Each node communicates through a CS-Router, which is responsible for all connections over the network, and a server router, a node that connects clients to servers. The server router implements a routing table to keep track of the addresses of clients and servers.

The transfer process begins with a request from a sender to the CS-Router to begin a new transfer. On the sender side, files are converted into a byte array to allow for any type of system to be able to convert back and read the file. Each byte is sent individually inside of a TCP packet. After locating the device running the server router code, the CS-Router asks the device for the location of the intended recipient for the transfer. After consulting its routing table, the server router returns the address of the server to receive the file if its address exists in the table and opens a thread for the file transfer. The sender and receiver perform a handshake and the CS-Router transfers the file to the server over TCP. After each packet delivery, an acknowledgment of receipt is sent back to the sender following the same route as the initial transfer, and the next packet transfer begins. Once all packets of the file are transferred, the sender and receiver exchange messages to end the transfer, and the connection is terminated.

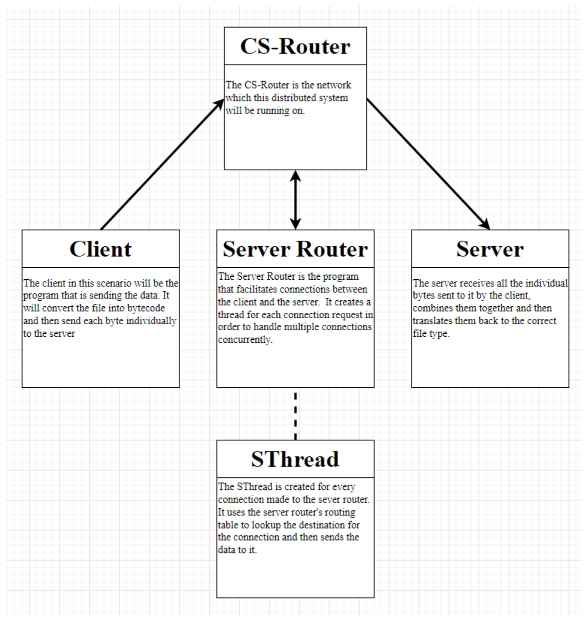

CS-Router
The CS-Router is the network which this distributed system will be running on.
Client
The client in this scenario will be the program that is sending the data. It will convert the file into bytecode and then send each byte individually to the server
Server Router
The Server Router is the program that facilitates connections between the client and the server. It creates a thread for each connection request in order to handle multiple connections concurrently.
Server
The server receives all the individual bytes sent to it by the client, combines them together and then translates them back to the correct file type.
SThread
The SThread is created for every connection made to the sever router. It uses the server router's routing table to lookup the destination for the connection and then sends the data to it.

Fig.1: Block Diagram of Design

Implementation of the Modules

The four modules used in this project are: Client, Server, ServerRouter, and SThread. To start, the Client and Server are the nodes communicating with one another with the Client module sending a request to the Server module. Both modules must first connect to the ServerRouter before making or receiving any requests with one another and must be on the same network or router in order to work; however, they cannot be operated on the same computer and same drive. They must also have the address of the other as well as the router name in order to properly connect.

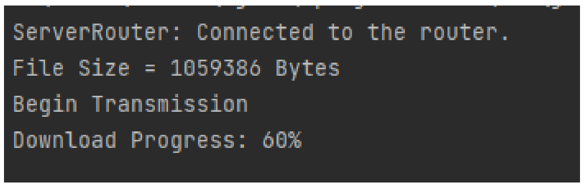
As for specifics, the Client will have a file in its directory that could be a text file to a video file. The Client will then convert the file into bytes which is sent through the network and ServerRouter to the addressed Server. The Server will then take those bytes and convert it back into a readable format.

The ServerRouter connects the Client and Server modules together so that they may send and receive requests. As mentioned before with the Client and Server modules, the ServerRouter must also be connected to the same network or router to connect with the nodes on that network. The ServerRouter will then, upon being run, will listen to port 5555 for any incoming connections on the network.

In order to connect multiple clients and servers, the ServerRouter must also use the fourth module SThread which is placed in the same directory as the ServerRouter. The SThread provides any client/server that connects to the ServerRouter a thread so that the ServerRouter can keep multiple connections going at the same time.

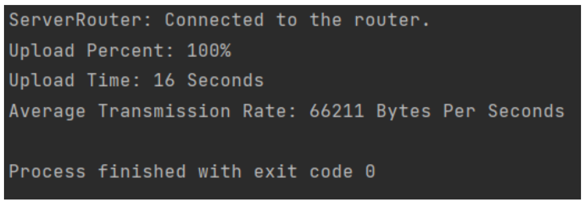
Simulation Discussion

The simulation allows for transfer of files from one system to another through a simulator router. This is done by converting the file to be sent into a byte array and then converting the byte array back to its file form at the receiving end. An advantage of this approach is that it allows for virtually any file type to be sent as long as the receiving end knows what file extension to convert the byte array into. The receiver starts by displaying the length of the file being sent, then followed by a message saying the download starts. While the download / upload is in progress a message displaying what percentage of the file has been transferred up to that point. Once the download / upload is completed the sender then displays the time it took for the file to be transferred and the average byte per second.



```
ServerRouter: Connected to the router.
File Size = 1059386 Bytes
Begin Transmission
Download Progress: 60%
```

Fig.2: Screenshot of the receiving side mid transmission showing the file size and the download progress.



```
ServerRouter: Connected to the router.
Upload Percent: 100%
Upload Time: 16 Seconds
Average Transmission Rate: 66211 Bytes Per Seconds

Process finished with exit code 0
```

Fig.3: Screenshot of the sender side after the program had run to completion and displays the upload time and average transmission rate in bytes per second.

File Type	Size (Bytes)	Transmission Time (Seconds)	Average bytes/second
.mp3	1,059,386	16	66,211
.mp3	5,319,693	72	73,884
.mp4	5,485,935	68	80,675
.mp4	9,840,497	111	88,653
.pdf	3,833,674	46	83,340
.png	2,060,826	30	68,694

Fig.4: Table showing test files we sent from one system to another and data about the transmission. Such as the file's extension, size, the time the transmission took, and the average transmission rate in bytes / second.

Data Discussion

In our timing, the size of the file matters greatly in terms of the transmission time. What also matters is the time it takes for the server and client to start up. With these modules, the server and client can be started at any time after the serverRouter is run. When the server and client start, they begin counting and connect to the serverRouter. Fortunately, instead of rejecting the connection, if there's no receiver or sender yet, the serverRouter will instead block the connection until it can properly connect it to the appropriate process. This means that the average time for transmitting files is based on the size of the file and the time it takes to connect to the sender/receiver. This relates to interprocess communication as the serverRouter will block processes until it can be connected and used.

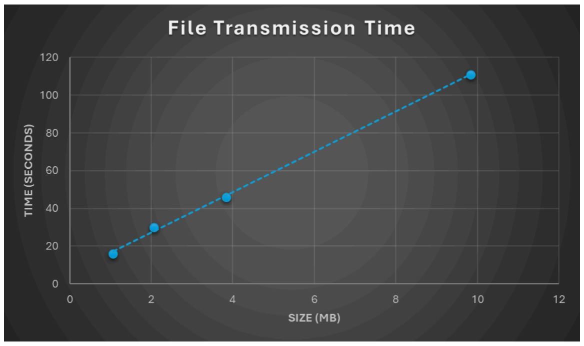


Fig.5: Line Graph of file transmission time with relation to file size

As shown here in this line graph, the time it takes for file transmission is congruent to the data size without taking into account the time it takes for the client and server to connect. This makes sense as the more bytes there are, the more the router and serverRouter have to process and transmit.

Conclusion

In this part of the project, we learned about the modules that make up TCP networking and the code behind it. In Client-Server networking, it is much more than just the router taking information and giving it to the receiver. It involves having the client and server know where they are sending/receiving files and it is up to the server-router to facilitate the file transfers. This system also allows multiple clients and servers to transfer files through the server-router as each is given its own thread.

In our testing of various files, we found that mp4 files take the longest to transmit while mp3 take the least amount of time. This is most likely because mp4 files constitute both video and audio while mp3 contains just the audio. It also seems that mp3 files are usually smaller in terms of byte size due to having less information.