CSC3002F Networks Assignment

# Max McGown-Withers

# Chelsea van Coller

# Michael Brough

Contents

[Introduction 3](#_Toc34165571)

[System Functionality 3](#_Toc34165572)

[Protocol 4](#_Toc34165573)

[Security 5](#_Toc34165574)

[Creativity 6](#_Toc34165575)

[Sequence Diagrams 7](#_Toc34165576)

## Introduction

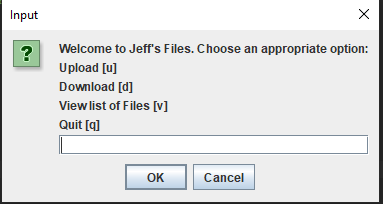
A network of has a vast array of uses, one of these uses being file sharing. Files can be shared over a network in different ways. One of these ways is between a client and a server. The server hosts and stores as well as manages the access to all the files. The clients then access the server and either upload files or download files.

In order for this to be possible, there needs to be a system in place that allows for the client to communicate to the server what processes and methods it wants to access. The client must be able to tell the server whether it wants to upload or download files or see what files it has access to download. These communications are called protocols. Protocols tell the client or server what is about to happen and what kind of data the client or server should expect next. How the protocols are made and how they work will be discussed in more detail later.

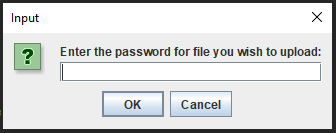
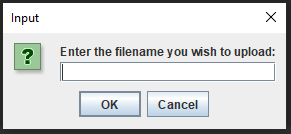
Our program/project was to build a client app and a server app that would be able to share files between the two. We had to implement sockets to send and receive the data as well as design protocols to manage the transferal of the files. Discussed below is the functionality of our two programs, the security measures we implemented and the protocols that we designed.

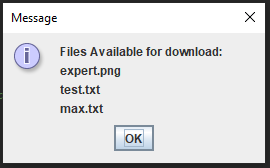
## System Functionality

In the system, the client can perform four options: upload or download files, view a list of available files and exit the program. In theory, the server will continuously be running as a client may need to connect to the server at any point. In order to allow for multiple connections to the server, the server implements threads, therefore this will reduce the overhead and speed up the process. For each client-server connection there will be a new socket created, therefore implying there shall be a socket for each connected client and an extra one to register if another client wishes to connect. However, one needs to run the server in order to begin the assignment. After the server is running, the client can commence and the following screen shall appear:



* Downloading a file
  + The client will prompt the user to enter the name of the file they wish to download. They will need to enter the exact name of the file and file type. They will also need to enter the security key with which the file is secured. The server will then send through the file and it shall be saved on the client side.



* Uploading a file
  + The client will prompt the user to enter the name of the file they wish to upload. They will need to enter the exact name of the file and file type. The user will also be required to enter the security key with which they wish to secure the file. After this the client will send through the file and it shall be saved on the server side
* View a list of files
  + The client sends their password to the server. The server then sends through all the names of the files that the client has access to and is able to download.  
    

## Protocol

Before any file can be sent between either the client or the server, a message, or protocol must be sent across to indicate to the client or server what is going to follow. The protocols will tell the receiving program how to process the data that follows the protocol. We developed separate protocols for client-server and server-client communication. Our protocols were built and then unpacked as follows.

Client to server protocol example:



Above is the upload file protocol. It is made up out of the name of the file to be uploaded, the size of the file, flag to indicate what method the server must call, and the password to use as security for the file. The .writeUTF sends one string to the output stream that is then sent through the socket to the server. Once the protocol is received  by the server socket, it is broken up into an array of all the elements in the protocol. The protocol is then decoded on the server side and the server begins to run the correct methods. The protocol breakdown is as follows.

***Table of protocols sent from the client to the server.***

|  |  |  |  |
| --- | --- | --- | --- |
| **Method of protocol origin** | **Protocol array position** | **Protocol** | **Meaning of protocol/use on server side** |
| Upload method | 2 | U | Indicates to the server to call the upload method |
| Download method | 2 | D | Indicates to the server to call the download method |
| View list method | 2 | V | Indicates to the server to call the download method |
| Upload method, download method | 0 | filename | This is parsed into the upload method so the method knows what to save the file as. It is also parsed into the download method so the program knows which file to send to the client. |
| Upload method, | 1 | filesize | This is parsed into the upload method so the method knows how many bytes to expect from the input stream from the client side. |
| Download method | 1 | password | This is parsed into the download file method so the program can check whether the user is allowed to access the requested file. |
| View file method | 3 | password | This is parsed into the view files method so that the program will know which files are available to display to the user. |

***Table of protocols sent from the server to the client.***

|  |  |  |  |
| --- | --- | --- | --- |
| **Method of protocol origin** | **Protocol array position** | **Protocol** | **Meaning of protocol / use on client side** |
| Send file | 0 | filename | This is used by the download file method. It is the name of the file to be saved on the client’s computer. |
| Send file | 1 | filesize | This is used by the download file method. It is needed so the program knows how many bites to expect from the server side. |

## Security

As demonstrated above in system functionality, the files are secured using a password. When a user uploads a file, they are required to enter a password/key. This password or key will be needed in order to download the file. They can choose to secure the file or they can make it publically accessible, in order to do this they simply press enter when asked to enter password to make the file public. In order to access these files, the user would need to enter the correct password. They will be able to create a shared file system, if multiple users use the same password. When accessing the list of files, the client will be able to see all the files listed under their password/key.

## Creativity

In order to enhance the user experience, we implemented a user interface, which facilitates a cleaner, fluid interaction. The GUI assists the user in selecting their options.

Some users may not know how to use the terminal; therefore the GUI assists them and reduces errors.

We also stress tested our programs. The tests that we performed were to have the user download a 1.5 Gb file as well as have multiple clients access the server at the same time. Both of these tests worked perfectly although the large file took some time to download.

## Sequence Diagrams

