



File Transfer Application

1. Introduction

The goal of this project is to create a file transfer application system. The file transfer application system will make use of the networking principles studied in the class e.g. protocols, application and transport layer, socket programming etc. You can imagine the components of the file transfer application system that we use in our daily lives e.g. FTP application to have an idea of basic functions/entities in a file transfer application system, however the project requires implementation of only some of these functions/entities so as to limit the scope.

Just like other file transfer application systems, your application should allow any user to send and receive different types of files via the file transfer client. The file transfer client will interact with the corresponding server and should be able to either receive/download or send/upload files to a desired repository on the server.

Note that although programming languages like Python may provide libraries that implement the required protocol i.e. FTP for you, this hides away the functionality of these protocols. Hence, you are **not allowed to use the standard libraries for these protocols** and should only use the basic socket methods. This would also imply that you would need to consult the RFC (Request for Comment) document for the FTP protocol in order to implement the protocol in a standard manner. Once the protocol is implemented as per the RFC, your file-transfer client must be able to communicate and exchange files with your own developed server and should also be able to perform these tasks with standard FTP servers as well.

2. Project Tasks

Developing the file-transfer application would require implementation of the following features. These are minimal features to be implemented. You can extend them based on your needs:

A. Features of the File-Transfer Client

1. A simple user interface allowing the user to perform required tasks e.g. login, upload/download files, enlist files in the server directory, status etc.
2. The implemented tasks should be performed as per the methodology specified in the FTP RFC document for that particular function/task.

3. The client should implement at least basic commands of the FTP protocol. Section 5.1 in the document of RFC 959 [1] discusses a set of commands for minimal implementation which can be considered as a reference for the minimum set of commands to be implemented.
You are welcome to implement more commands from the RFC, if needed.

B. Features of the File-Transfer Server

1. The file-transfer application server should maintain the records and repositories for different users. Please consult if RFC specifies anything in this regard.
2. Implement the server-side of the implemented commands e.g. the commands mentioned in section 5.1 of RFC 959 for a minimal implementation.
3. Note that many of the protocol commands from the client would solicit a reply from the server, see section 4.2 in RFC 959. You are not required to implement all the reply codes but should implement at least 5 reply codes, preferably from different reply code groups. Please specify the reply codes implemented in your final report as well as justify your choice of implemented reply codes.
4. Your file-transfer application server should not only be able to interact with your own implemented client, but also with a standard FTP client e.g. FileZilla etc.
5. A Multi-threaded version of the file transfer application server must also be created.

C. Other Features

1. Your file-transfer application must be able to deal with different types of files i.e. text, images, video clips etc.
2. The implemented file-transfer application server should be able to talk to the implemented client in both the cases: 1) client and the server are both hosted on the same PC/host and 2) client and server are on different hosts in the network.
3. The client and the server should be able to deal with common errors and generate appropriate responses as per RFC. Please specify the errors catered for in your final report.
4. You should be able to show that Wireshark can be used to monitor protocol messages between the client and the server.
5. The protocol implementations should strictly follow the respective RFC for the format of all the request and response messages and the order and types of messages exchanged. For this, you will have to go through the details of the RFC concerning your implemented commands and actions. This is important for communicating with the

standard clients and servers, for which it is also important to know the type and version of protocols implemented by the concerned standard clients and servers.

6. You are ONLY allowed to use low level socket class and associated methods for implementing the protocols needed for the file-transfer application, and the use of higher-level classes/libraries for such purposes is not allowed. However, for other aspects such as parsing, dealing with multimedia etc. you can make use of built-in libraries, if needed.

Finally, any suggestions to modify/improve the above system are welcome and can be discussed in class. Any cool additional features can get you bonus marks.

3. Project Assessment

This project contributes 70% to the total mark for the course. The assessment policy is as shown in the following table:

Outcomes	Checklist	Marks
Features Implemented as per section 'Project Tasks' above.	<ol style="list-style-type: none">1. FTP client basic features2. FTP Server basic features3. Interaction with standard FTP server and client4. Ability to deal with different types of files5. Multithreading6. Use of Wireshark	40%
Demonstration	- Presentation in PowerPoint (.ppt format) with embedded video clips or screenshots (recommended)	10%

Report	<ul style="list-style-type: none"> - Each student must submit a single report - Maximum 5 double-column pages excluding the appendices. <p>Should contain:</p> <ol style="list-style-type: none"> 1. Introduction 2. Description of the implemented system and the implemented features. What features could not be implemented and why. 3. Brief description of commands/replies for each protocol that were implemented. 4. Implementation of features in detail 5. Results (Wireshark Screenshots etc.) 6. Critical Analysis 7. Basic structure of the code (Classes, methods etc.) and how to use the code 8. References 9. Appendix 	40%
Source Code	<ul style="list-style-type: none"> – Should be well commented – Must have a readme file detailing how to use the code – Should be sent electronically 	10%

4. Other Details

- Your project demonstration should be presented in a PowerPoint presentation. Your presentation must include a demonstration for all the implemented features. You may include embedded video clips in your presentation, otherwise it is highly recommended that you include step-by-step screenshots of your implemented features. Be sure to include the demonstrations from both the perspective of the client and server side.
- You are also encouraged to give a brief explanation or reasoning for the choice of your designed implementation in the PowerPoint demonstration.
- I would recommend doing the implementation in Python language, but in case you prefer to use another language, kindly let me know beforehand. **Please specify your language, OS and IDE in your presentation. It would be your responsibility to make sure beforehand that your software is working on your system.**
- **External Contribution**
 - If you have received any assistance from an external source (friend, family member, online resources, etc), you are required to explicitly state what the external contribution was and how much is being contributed (in percentage) by the external source. Please

ensure that you also acknowledge who/where the external source is from.

- The final report must describe all work undertaken in the project
 - A good description would be “wrote function xyz() in server implementation.” A bad description would be “wrote some server code”.
 - You must discuss the implementation of your code in the report.
- Plagiarism in any form will be strictly dealt with as per School’s policy.

5. Submission Details

The submission deadline, as per deadlines given by School for 4th year projects, is at **8:00am on 12 June 2020**. You are required to submit the following project components by the given deadline via the Sakai site:

1. Project report
2. Project source code (zipped) - all files, functions, and project materials must be zipped as a single file
3. Project PowerPoint presentation

Late submission will be penalised according to guidelines specified in the “*Red Book*”.

References:

1. <https://tools.ietf.org/html/rfc959>
2. https://www.wired.com/2010/02/ftp_for_beginners/
3. <https://www.go4expert.com>
4. https://www.ncftp.com/libncftp/doc/ftp_overview.html