CS 447: Networks and Data Communications

Project #03

Assigned Date	:	Monday, November 03, 2014
Due Date (Phase I)	:	**Monday, November 17, 2014 @ 02:59:59 p.m.**
Due Date (Phase II)	:	**Sunday, November 23, 2014 @ 02:59:59 p.m.**

1 Overview

For your final project assignment, you are going to implement a scaled down version of a **UDP-based** File Transfer Protocol (FTP) application. A shared directory named "`cs447-ftp`" is automatically mounted for you in `home.cs.siue.edu` to use as your file storage service.

1. Given the scope of this project (and the remaining amount of time in the semester), undergraduate students are **allowed** to form a **group of maximum 3** members.
2. Graduate students are allowed to work in a **group of maximum 2** members.
3. Forming a group is not mandatory; you are free to work on your own.
4. Start early. There will not be any time left for due date extensions.
5. The project has two phases.

1.1 Phase I: Design

In the first phase, you will develop and then submit a report containing:

(a) A listing of group members and the last 3-digits of their respective SIUE ID numbers. A group name.
(b) Answers (few sentences for each question is sufficient) to the questions listed in Appendix A
(c) A detailed plan of the rules and structure of your protocol
(d) A short example of transferring a small file using the plan you laid out above. The size of the file can be arbitrary but must be too large to sent in one datagram.
(e) The roughly equivalent breakdown of work for each member.
(f) Phase I is due on **Monday, November 17, 2014 @ 02:59:59 p.m.**

1.2 Phase II: Implementation and Verification

In this phase you will implement and test your design, and make making modifications if required. The second phase report should be appended to the original report and submitted at the same time as your final source code. This report should contain a postmortem with the following:

- Description of how to compile your program.
- Description of work accomplished by each student that is written independently by each individual group member.
- Description of any modifications you had to make to your original design (and why).
- Description of any difficulties you had with the project (that aren't mentioned above).
- Any requirements that were not met.
- Any additional commands added, their syntax, and description.
- Your overall opinion of the project and what you learned that is written independently by each group member.

This report and a folder containing your source code should be tarballed and submitted through Moodle by **Monday, November 17, 2014 @ 02:59:59 p.m.**

2 Requirements Solicitation: SIUE-FTP

For your sever and client to successfully communicate with each other you are to develop your own protocol with the following requirements:

- All network communication must be done using **UDP**.
- There must be **two connections**, a *control connection* and a *data connection*. The control connection should be maintained until the client terminates it, and the data connection should be created only when required. If the control connection is terminated, both connections should be terminated.
- The control connection should handle ftp commands.
- The data connection should handle transmission of data or meta-data about file transfers.
- The application **should handle control commands without waiting** for a file transfer to be completed.
- The control connection should use port 2XXXX, where XXXX is replaced by the last 4 digits of your 800 number. *(Properly document in Phase I)*
- The data connection should use port 3XXXX, where XXXX is replaced by the last 4 digits of your 800 number. *(Properly document in Phase I)*
- The data transfer mode should be **binary**, not ASCII or EBCDIC etc.
- The connection mode should be **passive**.
- All programs should terminate gracefully, not leaving corrupted, half-finished or with open files.
- At minimum, your implementation must support the commands in Table 1 below. If additional commands are required, or would make your life easier, you are welcome to implement them.

Table 1: SIUE-FTP Command Syntax and Description

Command	Description
`get [r_file] [l_file]`	`r_file`: File name to be transferred from a remote host (required). `l_file`: File name to be locally created (optional), default to `r_file`. This command will attempt to retrieve the specified file from a connected remote host and then create an identical file on the local host
`put [r_file] [l_file]`	`l_file`: File name on the local host to be transferred (required). `r_file`: File name to be created on remote host (optional), default to `r_file`. This command will attempt to send the specified file from the local remote host and then create an identical file on a connected remote host.
`cd [path]`	`path` - the path of a folder on the remote working directory. Changes the active working directory using to the one specified in the parameters.
`ls`	Retrieve a directory listing of the remote working direction
`quit`	Terminate the application

Deliverables

The due date for Phase I is **Monday, November 17, 2014 @ 02:59:59 p.m.**. The due date for Phase II is **Sunday, November 23, 2014 @ 02:59:59 p.m.**.

- Phase I does not have any digital submission requirements. Instead, you are required to hand over a **printed**, **stapled** report that meets the requirements outlined under Section 1.1 above. Your report should be prepared using a word processing software (*not handwritten*).
- For Phase II, a Moodle dropbox will be opened for your final submission that includes:
  - A short report **in PDF format** that meets the requirements outlined under Section 1.2 above.
  - A compressed tarball of the directory containing your source code. Do not include executables in this tarball. To create a compressed tarball of the directory source, use the following command: `tar -zcvf name-pr3.tar.gz source/`. Obviously, change the name to your group name.

The overall time allocation for the project is ~3 weeks; Phase I is due in ~2 weeks with ~one week left for final implementation. In reality, you are advised to finish Phase I as soon as possible and start working on Phase II. You may hand over your Phase I report before it's assigned due date to get feedback of your progress/plan sooner.

Note: You must convince the instructor that all team members are substantially and equally contributing to the project. *One member doing all the programming while the other member doing the documentation **is not** an equal breakdown*

of work. Failing to provide evidence of proper work breakdown may result in penalties for all team members, so choose your group wisely.

Related Info

File Transfer Protocol, or FTP, is a very commonly used protocol that facilitates the transfer of files between two network enabled hosts. FTP was first described in 1971 by Abhay Bhushan in RC 114 and has since been continuously modified and updated to improve on capabilities, robustness, and security of the protocol.

Today FTP operates in two modes, active or passive, and supports transmission of files in several data representations.

Active vs Passive

In active mode an FTP client will open an arbitrary port, `N` and establish a connection with port `21` of the FTP server. The client will then open port `N+1` and send a command over the already open connection with the command `PORT N+1` to the server. The server then opens the connection it will use to transfer data to the client's `N+1` port.

In passive mode, the client will follow the same starting procedure but instead of sending `PORT N+1` it will send `PASV` to let the server know that the connection type will be passive. The server responds by opening an arbitrary port, `P`, and then sending that port number back to the client. The client then opens the connection with port `P` to transfer data.

Data Representation and Transfer modes

When performing FTP transfers data can be sent in several different representations depending on the type of file that is being sent. The two most common types are ASCII mode, for text based files, and binary mode for exact byte for byte transfers.

When sending data FTP can use Stream mode, Block mode. Stream mode sends data as a continuous stream of packets where most of the work is handled by TCP. In block mode however, FTP, is responsible for decomposing the data into blocks, and then applying meta-data to those blocks to allow for reconstruction before passing data onward to the TCP layer.

Appendix A

1. What features does TCP provide than UDP doesn't? How does that impact your design and how do you propose to address those concerns.
2. Based on your answer above, if we suddenly started using UDP based FTP with no changes what would happen?
3. How do you mark the end of transmission of a file. Better yet, how do you ensure that the whole file was transfered as intended?

4. What is the difference between FTP's active and passive modes and why is passive mode important?
5. How can we handle multiple connections simultaneously with little to no user delay while maintaining efficient data transfers?
6. Can we use FTP's continuous transfer mode using UDP? Why or why not?