CS176B: Network Computing

FTPxpress

Project Deliverable B

Ben Patient
Danish Vaid
Jake Can

March 4, 2018

Project Deliverable #B

Network Computing

Ben Patient
Danish Vaid
Jake Can

Team Members:

- Ben Patient
- Danish Vaid
- Jake Can

Team Size Justification: Having been in the same capstone team for CS 189A/B the three of us established a good workflow and our similar schedules let us coordinate and work well together. We hope to use these advantages to build a solid project and have many ideas for features we could implement and networking challenges for us to overcome.

Project Overview

For our project, we are implementing our own version of a FTP server, client, and protocol. In doing this we are also implementing custom headers to tailor the protocol to our priorities and for the features we want to implement. These headers would include information on file, packet ordering, payload size, etc. Looking more into these headers, we decided that it would be better to have a custom handshake that exchanged meta-data, file priority, and other such data explained below. Upon the basics of FTP, we also plan to add network security features (for user privacy and ensuring data integrity) and support for parallelized multi-file upload/download for performance. We hope for our server program to server multiple clients, this feature would have us identify specific users to differentiate who is sending what packet to ensure there are no errors.

Project Goal Summary

At minimum we want to deliver a basic FTP protocol with our custom headers and a minimal terminal UI for usage. Having finished the features discussed above, we would like to support basic commands such as list files, delete file, move file, and such. Finally, our physical deliverable will be our source code, and a report that describes the project, its completion and usage, and performance analysis.

Custom File Handshake

This process will set metadata for each file transfer. The benefit of including a handshake is to reduce the number of headers, therefore reducing overhead on each packet sent so there is more data per packet. Our metadata would include file name, file type, FUID (file unique identifier), client ID, and security keys. The client ID field will pair the file with the client. Knowing this ensures which file is being transferred by who, and can also be used to prevent collisions of file names. Lastly, the security key sent over will be used for asymmetric encryption, to make sure the connection is confidential.

Custom Headers

Note: Many of our custom headers became a part of the custom handshake to improve performance and efficiency.

Sequence Number

This is will be a 4 hex-decimal character describing the sequence number of this packet... Finish explaining.

File Unique Identifier (FUID)

This is will be a 4 digit number assigned to each file transfer. The server will assign an ID to each file that is being uploaded or downloaded. This ID will pair with the file name and type so the header will only include the ID, and not the file name and type, thus reducing overhead.

Packet Type

This is will be 1 bit information that tells whether the packet contains command data or file data. Finish explaining.

Performance Analysis

Stubbed out, still have to do this. Ideas: encrypted vs unencrypted, serialized vs parallelized file transfer, performance compares to popular options (i.e. filezilla, cyberduck, etc.), load testing.

Timeline

Week 3: Design our headers and protocol schematics
Week 4: Successfully send over a file from 1 user to another
Week 5: Implement basic terminal UI and command support
Week 6: Implement parallelized upload/download
Week 7: Implement network security
Week 8: Run performance testing methods
Week 9: Refine code and finalize project/report.

We have fallen behind on our timeline slightly and are current in the week 6 phase of implementing parallel transfer. We spent the past week focused on other commitments, since those have ended we should have more time to catch up in the following weeks.