Design Document

Table of Contents

Introduction and General Overview the xFx App:

Client-server communication, in its simplest form, refers to the exchange of data between a client process and a server process. In a client-server model, the client requests services, or resources from a server, which then processes the request and sends back the data. In this project, our objective is to create a file transfer system that leverages the client-server architecture to allow for the transfer of files between clients and the server. The system will prioritize efficiency and aim to minimize bandwidth usage while maintaining high transfer speeds.

xFx is composed of a client application, FxClient, and a server application, FxServer. FxClient is a Java-based application that enables clients to connect to a server running on the local network and perform file downloads. It operates by utilizing two command-line arguments: a command, “d" for download, followed by the name of the file to be downloaded. It implements a caching mechanism by storing downloaded files and reusing files that are unchanged between downloads to reduce server load and preserve bandwidth. The client uses socket programming to establish a connection with the server and perform the file transfer.

FxServer is also implemented in Java. It operates by passively opening a connection and listening for incoming requests from clients. The server supports various operations, such as downloading, uploading, and listing files.

Class Structure and the Socket API

The xFx program is implemented in Java and leverages the Socket API to establish a connection with the local server and perform file transfers.

The main method of the FxClient class takes two command-line arguments, the first specifying the operation to be performed (download, upload, or list), and the second argument is the file name. FxServer uses the ServerSocket class to bind to the specified port and the Socket class to

represent the communication channel. It opens the connection passively and waits for a header message from a client process.

The xFx App sets up input and output streams using the classes: InputStream, OutputStream, BufferedReader, BufferedWriter, DataInputStream, and DataOutputStream, BufferedReader, BufferedWriter, InputStreamReader, and OutputStreamWriter to read and write text data, StringTokenizer to extract information from a client request, and File, FileInputStream, and FileOutputStream to read and write files on the server's file system.

File Cache

The FxClient program's file cache system is designed to reduce the number of requests sent to the server with the goal of saving time and network bandwidth by not redownloading an already downloaded file, provided that its local copy has not become dirty/stale.

When a client requests a file download, the FxClient program first checks if the file is already in the cache by comparing the server's file's last modification time with the last modification time of the file stored in the cache.

If the file in the cache is perceived to be up to date, the program will copy it from the cache to the "ClientShare" directory without contacting the server. This saves time and reduces the number of requests sent to the server.

if the file in the cache is outdated or not found, however, the program will download the file from the server, store it in the cache, and then copy it to the "ClientShare" directory.

This system ensures that files are only downloaded when necessary, and previously downloaded files are reused if they have not changed. By caching files locally, the FxClient program reduces the load on the server and speeds up transfers for clients that frequently download the same files.

Protocol

A file detailing the protocol is included with this report, but here is a brief overview of the protocol:

1. List

If the client wants to get a list of files that are shareable by the server, it can send a header with the following format: "list -- [Line Feed]"

2. Upload

The application also implements a rather simple file uploading capability. To upload a file, the client sends a header in the following format: "upload [file name][one space][file size][Line Feed]" The client will then send the bytes of the file to the server.

3. Download

This protocol allows for simple and efficient file transfers between a client and a server. Protocol. The most extensive operation implemented by in the xFx App is the download operation:

Download Request:
The client sends a "download" request to the server using the following format:

download[one space][file name][Line Feed]

where [file name] is the name of the file that the client wants to download.

Download Response:
The server responds to the "download" request with either an "OK" or "NOT FOUND" header message, followed by the data of the requested file.

The “NOT FOUND” header message has the following format:

NOT[one space]FOUND[Line Feed]

The "OK" header message has the following format:

OK[one space][file size][one space][file last modified time][Line Feed]

where [file size] is the size of the requested file in bytes, and [file last modified time] is the time that the file was last modified.

Download Operation:
The client checks if the file is already stored in its cache. If the file is in the cache and has not been modified on the server, the client copies the file from the cache to a shared folder and sends a FIN header to the server to indicate that the download is complete. If the file is not in the cache or has been modified on the server, the client downloads the file from the server, stores it in the cache, and copies it to the shared folder.

The ”FIN” header message has the following format:

FIN[Line Feed]

Error Handling

The code implements some error handling mechanisms in place using header messages:

Client Side:

If there is an IOException or EOFException while downloading a file, the client sends a message indicating so.

An Exception is thrown if the client tries to upload a non-existing file to the server.

Server Side:
If the file requested for download is not found, the server sends a message 'NOT FOUND' to the client.

In case of an exception while trying to upload a file, the server sends an error message 'NOT FOUND' to the client.

If there is an IOException or FileNotFoundException while downloading a file, the server sends an error message 'ERROR' along with the exception message to the client.

Conclusion and Known Issues

The xFx App is a file transfer system that leverages the client-server architecture to allow for the transfer of files between clients and the server. The app uses socket programming to ensure communication ensure communication between client and server. The application is written in Java and uses the standard Java libraries for socket programming and file I/O. It supports downloading, uploading, and listing commands.

Possible flaws and future plans:

To determine whether an already cached file on client-side needs downloading, the file size and last modified dates of the files are compared. It is possible that this method may not be foolproof, which shall be revealed following thorough testing. A better implementation would have been to compare their checksums.

Bibliography:

Oiraqi. "P1-Communication." paradigms, GitHub, 12 Feb. 2023, github.com/oiraqi/paradigms/tree/main/P1-Communication.