Vikas Kumar 163059007
Satyaki Sen 16305R009

Project report phase 1
File backup system

Introduction

The project which we built is a file backup system program (eg: Google drive).The purpose of this file backup system is to store user files in a remote Backend Server for backup via a Middle Server.The program has multiple features like signup,login,upload file,download file,share file etc.

Architecture of the system

The system follows a 3 tier (Client ⇔ Middle Server ⇔ Backend Server) architecture model.

Basically our system has 3 major components.

1. Client side - There can be multiple Clients for our system. Each Client connects to the Middle Server using the ip address and port no. of the Middle Server.Then the Clients can multiple requests to the Middle Server. According to the requests the Middle Server will process the requests and will give back response to the Client accordingly. The Client will not have any direct connection to the Backend Server.

2. Middle Server - There is only one Middle Server in the system. The Middle Server will be connected with multiple Clients as well as the Backend Server. The Middle Server authenticates the Clients. The authentication data for every Client is stored at the Middle Server.If the authentication successful the Middle Server will process the Client requests and will connect to the Backend Server to get the response if necessary. The Middle Server works as a bridge between the Clients and the Backend Server. The Middle Server itself does not store the Client's backed up files.It only stores the data necessary for authentication and some data file for other operations like Client's uploaded or shared file information.

3. Backend Server - Currently we are using one Backend Server but it can be further extended to multiple Backend Servers. The Backend Server is the storage location for the Client's uploaded files.It maintains the every Client current backed up file system information. The Backend Server take 4 kinds of requests from the Server for each Client.

A.Receive file request

B.Send file request

C.Client's file system information request.

D.Delete a file request.

According to the request from the Middle Server the Backend Server sends response back to the Middle Server which in turn sends the response back to the Client.

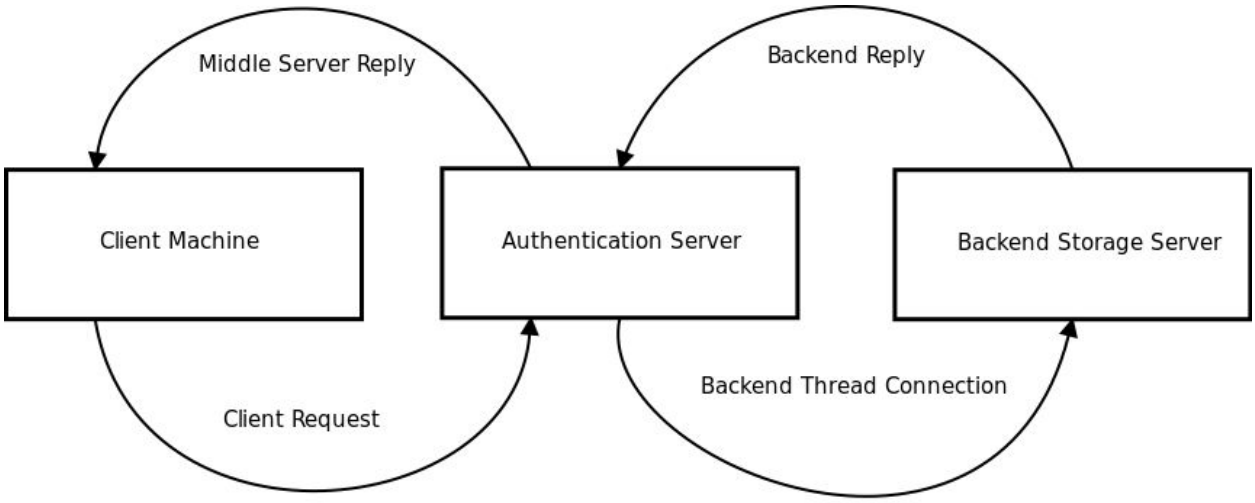
Multithreaded architecture model :

We have used a multithreaded architecture for our system.The multithreaded model for different components as follows :

Client side - The Client runs on a single thread.The Client does not need multiple threads because the Client does a single job only that is to send a request to the Middle Server and get a response from the Middle Server and then proceed to the next request.

Middle Server - The Middle has a master thread to listens to a given socket and waits for new connections from the Clients.Once a Client connects to the Server it accepts the the Client and creates a new worker thread to process it.Each worker thread will serve each Client.Now each of this worker thread will create a new thread whenever it does want to send a file to the Backend Server or to receive a file from the Backend Server.The new thread communicates with the Backend so the parent thread can take other requests from the Client.

Backend Server - The Backend Server also follows a master-worker multi thread model architecture.The master thread listens on a socket for different requests from the Middle Server. Whenever the Middle Server wants to communicate with the Backend it connects with the Backend Server.After a accepting a connection from the Middle Server for a new request the Backend Server creates a new worker thread to serve the request.



3 - Tier Architecture of System

Functionalities of File Backup System:

There are total 10 functionalities which are supported by this system currently as described below. These are also the different types of message requests made by the Client to Authentication Server which in turn serves these requests taking the help of the Backend Server.

1. **Login** : In this module, the Client authenticates himself to Middle Server (Authentication Server) using Username and Password. The authentication Server then checks if the given username and password matches in one of the entries of “new.txt” file, which is a file containing username and passwords of all the users of the system. If the authentication is successful then a random session ID is generated based on the time at which the login request was made and returned back to the user. This session remains valid until the user logs out.
2. **Sign Up** : Using this function a new user can signup into the system and later he/she can login to use the features of the system.

3. **Check File System** : This function allows the User to view its own Uploaded file which are available at the Backend Server.
4. **Upload Files** : In this part the Client uploads the files to store on the Backend Server. For this the Client first sends the session ID and the filename to upload to the Authentication Server which validates if the session ID is active or expired. If the session is active then the Client starts uploading to the Server and the Middle Server which stores the file temporarily with itself. Once the file is fully transferred to Middle Server from Client then the Server initiates a thread to connect to Backend and transfers the file to Backend Server to store there permanently. Once the transfer to Backend Server is complete then the Middle Server deletes the uploaded file from itself. While if the session check fails then the Client again initiates the process after login.
5. **Download File** : If the Client wants to download some of its private files then this function has to be used. The message from Client to Server contains the session ID and the filename to be downloaded. If the session match is passed then a check is performed to know if the requested file is previously uploaded by the Client or not. For this the Backend Server and Middle Server maintains the metadata file of the user which contains the list of all the files uploaded by the user till now. This metadata file is downloaded from the Backend once the session is generated for the user and updated on every upload so that the contents of this file remains same in both copies. If the requested filename is found in the metadata file of the user then a thread is created to fetch the file from Backend Server which is again temporarily stored at Middle Server before transferring to the Client.
6. **Share files** : Using this function the user can share its own uploaded file with other users so that they can access the same file. To implement share functionality we maintain a global file “share.txt” with both Backend and Authentication Server. It contains the file name and username who is owner of the file and has allowed the file to be shared with other users. Once the Client

initiates the request for a file to be shared a check is performed to know if the user is has already uploaded the file previously to the Server using the user's metadata file. If the file is found then an entry is made in "share.txt" file at both the Middle as well as Backend Server.

7. **Download a shared file** : This function allows the user to get the files shared by other users. For this the Clients send the filename to download along with session ID. Once session is validated the file is fetched from the Backend Server once it is verified that the requested file actually exists in the "share.txt" file available at the Middle Server. The Backend Server then transfers the file to Middle Server which in turn passes the requested file to the Client.
8. **Check shared files** : Using this function the user can check the contents of the "share.txt" file available with the Middle Server which contains the list of all shared files in the system.
9. **Delete file** : Using this function user can delete the files previously uploaded by him on the Server.
10. **Logout** : This module helps the Client to Logout from the system. Once the Client Logs out from the system the temporary metadata file is removed from the Client machine as well as the Middle Server.