Serverless Distributed Application for Backing Up and File Sharing

Final Report

Mestrado Integrado em Engenharia Informática e Computação

Distributed Systems

Group 3, Class 4:
André Pires - ei12058
Filipe Batista - ei12068
Filipe Miranda - ei12021
João Bandeira - ei12022

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

June 5, 2015

Contents

1 Introduction

For this second assignment the group decided to develop a distributed application for backing up and file sharing, based on the first project developed in Distributed Systems.

The application presents a graphical user interface, requiring an authentication through login/register on the system or through "Login with Facebook".

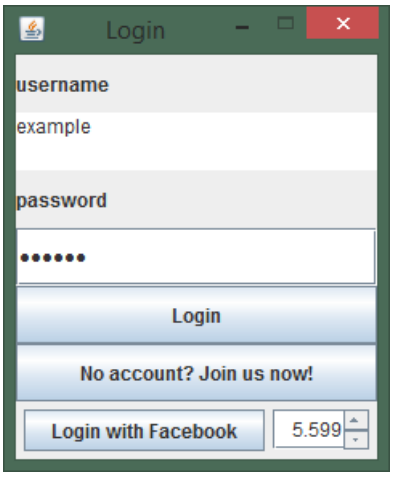

Figure 1: Login interface

After the authentication the user is capable of backing up files, restore, and share those with a group of friends.

In this report we will explore the project architecture and describe all of its functionalities. It will also address relevant issues to the project and present conclusions regarding the development of the project and the final project.

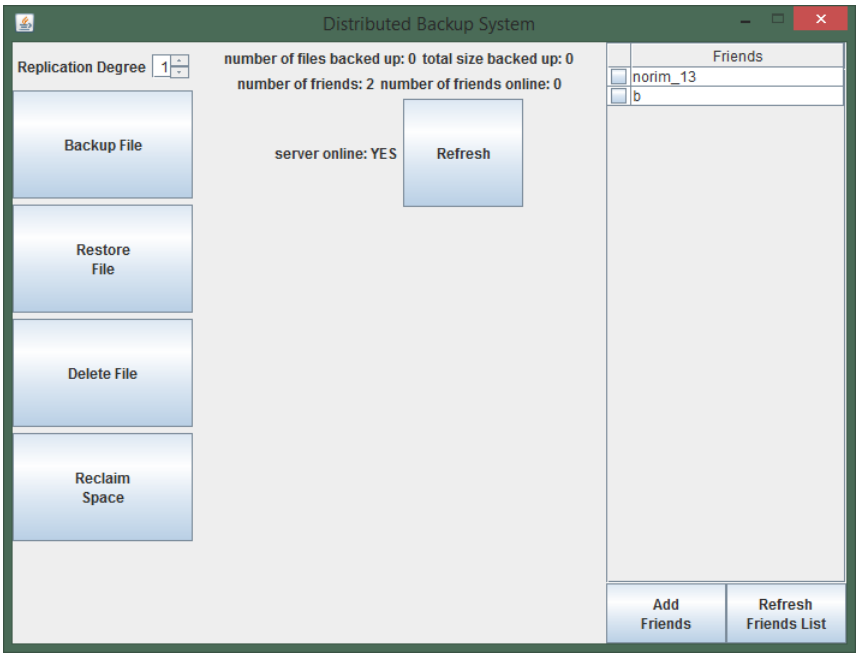

Figure 2: Main panel interface

2 Architecture

The application implemented is based on a server-peer and peer-peer interaction.

The server allows the authentication of the peers and provides a list of on-line peers, continuously checking, in a span of time, the users/peers on-line.

The user is required to log in into the application. The authentication can be done either using an application account or a Facebook Log In.

For an application account, the user must register, using an email and password. The information is then sent to the server where is stored in a database. After the registration, the user can Log In using the same credentials.

The user can also Log In into the application using a Facebook Account, where he/she must approve the application's access to Facebook's information like name, email and friends. Information of the user is then sent to the Server, where it is stored if there is no entry for the account on the database, or is validated.

The peer allows the user to Back Up a chosen file with a selected replication degree. The file is divided in chunks and sent to the on-line peers, provided by the Server. The peer then stores the peers' id who saved chunks of the Backed Up file.

The user can also Restore a previously Backed Up file. Using the list of peers' ids who saved chunks of the file, the peer sends an IP request of those peers, to the server and checks if they are on-line. Then, the peer requests, directly to each peer, a get chunk to recover the desired file.

For the reclaim function, the user first specifies the new folder size he/she wishes. The peer then selects chunks randomly to remove them until the new

folder size is reached. For each chunk removed the peer sends a REMOVED message to the original user.

The peer has an always running thread called *ConnectionListenerPeer* whose function is to continuously read input from the user's port. This thread is able to read several types of messages, execute an action if necessary, and respond to the message.

This thread can parse ISONLINE messages, sent by the server to check if a peer is online, and send an OK message in response.

A message PUTCHUNK is parsed by the thread, where it is checked if the chunk in the message is already stored. If it is not, the chunk is stored in the receiver. In the end, the thread sends a STORED message to the original transmitter.

In case of a DELETE message, the peer checks if it has chunks of the specified file to be deleted. If so, deletes them and sends an OK message, or a NOTOK message if it has not.

When a BACKUPFILE is received, the thread will try to send the file specified in the message to a peer.

For file sharing with friends the group implemented the following protocol:

1. The peer starts by requesting the last known IP of a friend
2. Sends a BACKUPFILE message to a friend using his IP. The message body contains the name and size of the file (in bytes). The BACKUPFILE message has the following format:

BACKUPFILE <Msize><MyID><CRLF><CRLF><Fname><Fsize>

3. The receiver (friend) after receiving the BACKUPFILE message, creates a thread (*FileShareReceiveThread*) in which it opens a socket, to receive the new file.
4. The receiver then sends an OK message, whose body contains the socket port opened in item 3.
5. Receives the data until the end of the file.
   (a) The file is sent line by line (each line starts by "LINE#").
   (b) When the peer receives a LINE, sends an OK response.
   (c) When the peer receives am "END#", it means that it received the hole file. and responds with an OK message.
   (d) In case of any error, the peer responds with a NOTOK message.

This thread can also parse GETCHUNK messages, where the peer responds to the transmitter a CHUNK message with the chunk specified on the original message.

Messages:

NOTRESPOND DELETE <FileID><UserID><CRLF><CRLF>
NOTRESPOND GETCHUNK 0 <FileID><ChunkNo><CRLF><CRLF>

PUTCHUNK <BodyLength><FileID><ChunkNum><RepDegree><CLRF><Body><CLRF>
STORED 0 <FileID><ChunkNo><CLRF><CRLF>
REMOVED 0 <FileID><ChunkNo><CLRF><CRLF>
DELETE 0 <FileID><CLRF><CRLF>
GETCHUNK 0 <FileID><ChunkNo><CLRF><CRLF>
GETALLUSERS 0 <CLRF><CRLF>
GETONLINEUSERS 0 <CLRF><CRLF>
OK 0 <CLRF><CRLF>
NOTOK 0 <CLRF><CRLF>
ADDFRIENDS <BodyLength><UserID><CLRF><Body><CLRF>
BACKUPFILE <Msize><MyID><CRLF><CRLF><Fname><Fsize>

3 Implementation

The project was developed in the Eclipse IDE for Java and using SVN for subversion control.

For the implementation of this project it was used largely the Java language and SQL. Java was used for the implementation of servers, peers, protocols and user interface while SQL was used for the management of databases.

For an easier implementation of the graphic user interface, the group used the SWING library.

It was used the Gson library for the encoding in Json format of information sent in some messages.

For an easier manipulation of arrays the group used the Apache Commons library.

4 Relevant issues

5 Conclusion

6 References