

Ali Abduelmula, Wesam Hussain
Faisal Bagelagel, Ahmad Alghizi
Idrees Syed

COMP-3670 Project Cover Report

Github Link: <https://github.com/Fossilila/Automated-Network-Backup-Tool>

Overview

When choosing what to implement for the final project, we were intrigued with the option of being able to create whatever network tool that reflects the concepts that we have learned throughout the semester. Originally, we had the initial idea of a Wi-Fi scanner, an application that would scan the nearby available Wi-Fi networks and display their SSID's, speed, MAC address, maximum and achievable data rate, and security details. However, upon further research of how to implement this application, we realized that it would be extremely difficult to do so as there were several concepts, namely that of scanning wireless Wi-Fi networks, that we had not still learned in this course. Upon this discovery, we needed to take a new direction for this project.

Throughout the semester, we had completed multiple assignments in which we had to implement a server and client program through which a client would communicate with a server and request specific services from it. Having had experience in implementing such programs, we drew upon these concepts to implement a problem that us students face: backing up our files on a server. We have several projects, assignments, and important documents that we would like to have backed up on some external system, and that is exactly what we decided to implement for our project. Since our entire group has extensive experience in Java, we decided to choose this as our primary programming language.

Objectives

As mentioned in the overview, this project will essentially be an Automated File Backup Server. The primary objective of this program is to allow the user to connect to a server that is connected to the same network and be able to send files for backup at regular intervals. This interval could be half an hour, a few hours, etc. Hence, this program would be required to always run on the client's system. Essentially, this program would consist of a client and a server, with the following characteristics for the client:

- Connects to a specific server through its IP address
- Asks the user to pick the folders they want to send for automatic backup
- Asks the user how often they want to back up their files
- Change the server
- Zips the files before the transfer
- Request a backup of a document from the server

For the server, it will be running on a device it will have the following characteristics:

Ali Abduelmula, Wesam Hussain
Faisal Bagelagel, Ahmad Alghizi
Idrees Syed

- Creates a folder for each device which client, this folder can be based on the IP address of the client
 - If the client has connected for the first time, it will create a new folder for it, otherwise, it will take the files and store it in the folder corresponding to the client
- Send the backups to the client if requested

The program will store the server's IP address, the length of the backup interval, and all the backup files in some form of persistent memory that will be read from whenever the program is opened.

Design

FileZipping.Java

This class focuses on zipping one file. What the program does is when a file is required to be zipped, we call this class from the controller class which contains the name of the file that we want to convert. The class gets the name of the file and its path using the File file object. and then use the ZipEntry and ZipOutputStream Libraries to convert the given file to a zip file. A FileInputStream object was created to be able to traverse the contents in the given file in a for loop and convert it to zip using the zos.Write() method. When this is complete, a FileOutputStream object was created to label the new file's extension as zip.

FolderZipping.Java

This class carries out the same task as the FileZipping class but does it for a folder containing multiple files. First Strings are created for the paths for the Source folder and for where the zip folder will be created. Then in the FolderZipping() method, an ArrayList is created to store all the files that will be in a folder. In the Zipit() method, it is the same as the FileZipping class and uses the same objects and functionalities in order to zip files. For the method emptyFilelist(), we call it to clear the ArrayList after the files have been generated so that if there's a new file to be zipped the old content is erased. We also set the source folder string called srcFolder equal to the directory and have to use the method generateFileList() to generate the files as Zip Entries and then add them to the ArrayList.

Controller.Java

This is the class that we used to create the UI and is also our class that controls initializing the server, determining the file/folders that the user would like to be backed up and zipped, and then displays the file for the user.

For the StartApplication() method, it is the initializer method that checks to see if a file is existing, which is called applicationInfo. It also saves the IP address of the server that will be initialized. If the file exists we call the displayMenu() method, if it doesn't then we call setApplicationEnvironment() method to determine the IP address of the server and also create the file.

DisplayMenu():

Ali Abduelmula, Wesam Hussain
Faisal Bagelagel, Ahmad Alghizi
Idrees Syed

Display menu displays the user interface with the options to use the different functions of the program, this is the options the menu gives you:

1. Add more files to back up on the server
2. Request back up from the server
3. Change server
4. Exit

For option 1, it allows the user to provide an absolute file path, and then it will ask the user to provide how often they want to back up that file (this is done with the `getIntervalUpInterval()` method).

For option 2, It allows the user to request a backup of a file they had in the `applicationInfo` file (this file will have a list of paths of files currently being backup up on the server). The controller will then call the `RetrieveFile()` method to carry out this task.

For option 3, it allows the user to input a server address, this will change the server the user's backups will go to.

For option 4, the Program will exit, and all created Client threads will also terminate.

`DisplayFiles()`:

This method displays the files that were sent for backup to the server, this is done by reading the files from the `applicationInfo` file.

`addFileDirectories()`:

This method asks the user to input the `filePath` to the file that the user wants to backup. Once the path is determined we then call the `folderZipping` class in order to then zip the file that we want to backup. This method will write all the absolute paths received from the user into the `applicationInfo` file so that they can be accessed even if the program is restarted.

`RetrieveFile()`:

This method retrieves the file that the user wants to access, the files will be presented in a number format (by calling the `displayFiles()` method) and based on the number that the user enters they will retrieve that particular file.

`SetApplicationEnviroment()`:

This method is run only when it is the first time the user has run the program (to check this, it checks if the `applicationInfo` file exists or not). It will first create the missing `applicationInfo` file then writes to the file the Server address that the user is asked to input.

Ali Abduelmula, Wesam Hussain
Faisal Bagelagel, Ahmad Alghizi
Idrees Syed

getBackUpInterval():

This method determines how often the user wants the backup to take place:

DD:HH:MM:SS

This is the format that the user is required to enter and based on what they enter the counter input string is split into 4 sections and is split by the “:” character. Then, based on what number each user enters it is multiplied by the different time frames which were converted in milliseconds (days, hours, minutes, seconds) and then they are all added. This info is written into the applicationInfo file, the counter is written on the line right after its corresponding absolute path.

Server.Java

This class initializes with the start() method and launches the server, we start the server by opening a server socket, when this is complete the program then waits to connect to a client and once that is connected the message header is then compiled. When split_message is called it either calls FILE_TRANSFER or REQUEST_BACKUP and if its FILE_TRANSFER from the client to the server. It then makes a specific file in the server with the date that was created in the Controller class and is sent from the client.

For the REQUEST_BACKUP if the user decides to request a backup from the controller class the server will send the latest backup of the file created to the client which will then be created where the user specifies where they want the file to be on their pc.

Client.java

This class deals with the client-side of the program, it is called by the Controller class to do all network communication tasks required, such as transferring files for a backup session and retrieving a specific file from the server. Here are the client methods with their specific functionalities:

Start(IP, line number):

This method is called to initialize the backup process method and makes sure that the backup process method is run on a new thread. It runs the runBackupProcess() method to run the backup process.

RunBackupProcess(line number):

This method is responsible for calling the sendFilepath() method, it exists so that the backup interval set by the user for the specific file from line number is working, it does this by putting the current thread to sleep for the set amount specified. It also deals with making sure the client properly closes the socket in case the client or server connection is closed unexpectedly.

SendFile(filepath):

This method deals with sending a file to the server, first it sends a message to the server which would look like this: “FILE_TRANSFER filename”. The filename would correspond to the file

Ali Abdueilmula, Wesam Hussain
 Faisal Bagelagel, Ahmad Alghizi
 Idrees Syed

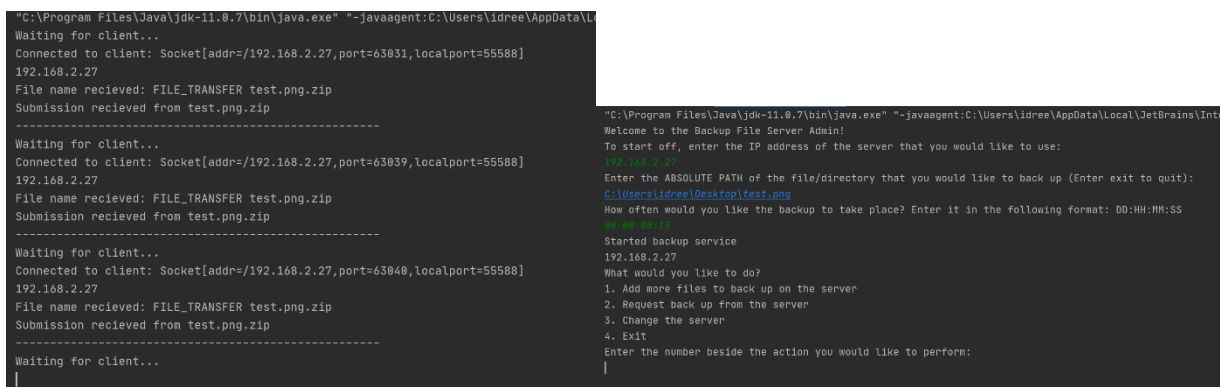
you want to send, it will then open the input and output streams to the server and begin sending the bytes of the file to the server. Once, the file is sent, the input and output streams will be closed.

ReceiveFile(filepath):

This method deals with receiving a file from the server, (which is called from the controller). The client would send a message to the server which would look like: "BACKUP_REQUEST filename". Once the server receives the message, it will send back the most recent backup of the file requested, and the Client will save the file in the Retrieved Client files folder (the client will create this folder).

Test Cases:

Test 1:



The screenshot shows two terminal windows. The left window displays the server's log, and the right window shows the client's interactive prompts.

```

"C:\Program Files\Java\jdk-11.0.7\bin\java.exe" "-javaagent:C:\Users\idree\AppData\Local\JetBrains\IntelliJ IDEA\bin\idea_rt.jar=63031:C:\Users\idree\AppData\Local\JetBrains\IntelliJ IDEA\bin\idea_rt.jar" 63031
Waiting for client...
Connected to client: Socket[addr=/192.168.2.27,port=63031,localport=55588]
192.168.2.27
File name recieved: FILE_TRANSFER test.png.zip
Submission recieved from test.png.zip
-----
Waiting for client...
Connected to client: Socket[addr=/192.168.2.27,port=63039,localport=55588]
192.168.2.27
File name recieved: FILE_TRANSFER test.png.zip
Submission recieved from test.png.zip
-----
Waiting for client...
Connected to client: Socket[addr=/192.168.2.27,port=63040,localport=55588]
192.168.2.27
File name recieved: FILE_TRANSFER test.png.zip
Submission recieved from test.png.zip
-----
Waiting for client...
|

"C:\Program Files\Java\jdk-11.0.7\bin\java.exe" "-javaagent:C:\Users\idree\AppData\Local\JetBrains\IntelliJ IDEA\bin\idea_rt.jar=63031:C:\Users\idree\AppData\Local\JetBrains\IntelliJ IDEA\bin\idea_rt.jar" 63031
Welcome to the Backup File Server Admin!
To start off, enter the IP address of the server that you would like to use:
192.168.2.27
Enter the ABSOLUTE PATH of the file/directory that you would like to back up (Enter exit to quit):
C:\Users\idree\Desktop\test.png
How often would you like the backup to take place? Enter it in the following format: DD:HH:MM:SS
00:00:00:00
Started backup service
192.168.2.27
What would you like to do?
1. Add more files to back up on the server
2. Request back up from the server
3. Change the server
4. Exit
Enter the number beside the action you would like to perform:
|
  
```

The first test case the server is running by running the BackupServerLauncher class. The client is running by running BackupClientLauncher. When BackupClientLauncher is initially run it will ask the user to connect to the server of their choice by entering the IP address. The user will then enter a file or folder path they'd like to back up and the time intervals they'd like the backups to occur. After they enter this information the backup process will begin, and the user can continue.

Test 2:

Ali Abdulmula, Wesam Hussain
Faisal Bagelagel, Ahmad Alghizi
Idrees Syed

```
Waiting for client...
Connected to client: Socket[addr=/192.168.2.27,port=63097,localport=55588]
192.168.2.27
File name recieved: FILE_TRANSFER Kernal.jpg.zip
Submission recieved from Kernal.jpg.zip
-----
Waiting for client...
Connected to client: Socket[addr=/192.168.2.27,port=63098,localport=55588]
192.168.2.27
File name recieved: FILE_TRANSFER test.png.zip
Submission recieved from test.png.zip
-----
Waiting for client...
Connected to client: Socket[addr=/192.168.2.27,port=63099,localport=55588]
192.168.2.27
File name recieved: FILE_TRANSFER Kernal.jpg.zip
Submission recieved from Kernal.jpg.zip
-----
Waiting for client...
Connected to client: Socket[addr=/192.168.2.27,port=63100,localport=55588]
192.168.2.27
File name recieved: FILE_TRANSFER Kernal.jpg.zip
Submission recieved from Kernal.jpg.zip
-----
```

```
"C:\Program Files\Java\jdk-11.0.7\bin\java.exe" "-javaagent:C:\Users\idree\AppData\Local\JetBrains\
Welcome to the Backup File Server Admin!
Started backup service
192.168.2.27
What would you like to do?
1. Add more files to back up on the server
2. Request back up from the server
3. Change the server
4. Exit
Enter the number beside the action you would like to perform:
1
Here are the folders/files that you have currently sent for backup:
1. C:\Users\idree\Desktop\test.png.zip
Enter the ABSOLUTE PATH of the file/directory that you would like to back up (Enter exit to quit):
C:\Users\idree\Desktop\Kernal.jpg
How often would you like the backup to take place? Enter it in the following format: DD:HH:MM:SS
00:00:00
Started backup service
192.168.2.27
What would you like to do?
1. Add more files to back up on the server
2. Request back up from the server
3. Change the server
4. Exit
Enter the number beside the action you would like to perform:
1
```

In this test case, we run our program again. When the client is run it will check if a previous backup was entered by looking into the applicationInfo file which contains the backup information. If a file is there it will start the backup process on that file with the corresponding server and time interval. The user then enters a second file they would like to have backed up with the time interval they'd like. Now multiple backups are happening in the background.

Test 3:

```
Waiting for client...
Connected to client: Socket[addr=/192.168.2.27,port=63140,localport=55588]
192.168.2.27
ServerFiles\192.168.2.27\test
ServerFiles\192.168.2.27\test\test_Tue Dec 22 19-42-50 EST 2020
File name requested: REQUEST_BACKUP test
File test.png.zip was sent.
-----
```

```
"C:\Program Files\Java\jdk-11.0.7\bin\java.exe" "-javaagent:C:\Users\idree\AppData\Local\JetBrains\
Welcome to the Backup File Server Admin!
Started backup service
192.168.2.27
What would you like to do?
1. Add more files to back up on the server
2. Request back up from the server
3. Change the server
4. Exit
Enter the number beside the action you would like to perform:
2
Here are the folders/files that you have currently sent for backup:
1. C:\Users\idree\Desktop\test.png.zip
2. C:\Users\idree\Desktop\Kernal.jpg.zip
Enter the number beside the file that you would like to retrieve (type -1 to exit):
1
Recieved latest backup of file: test.png.zip
What would you like to do?
1. Add more files to back up on the server
2. Request back up from the server
3. Change the server
4. Exit
Enter the number beside the action you would like to perform:
1
```

In this third test case, we are retrieving a backup from the server. The user selects option 2 to request a backup from the server. When the user selects this option, they are given a list of their files on that server they would like to retrieve. When the user selects the file, the server will send the file over to the client and it will be located in ReceivedClientFiles directory on the client computer.

Test 4:

```
What would you like to do?
1. Add more files to back up on the server
2. Request back up from the server
3. Change the server
4. Exit
Enter the number beside the action you would like to perform:
3
Enter the IP address of the new server that you would like to use:
192.168.2.27
Started backup service
192.168.2.27
What would you like to do?
1. Add more files to back up on the server
2. Request back up from the server
3. Change the server
4. Exit
Enter the number beside the action you would like to perform:
1
```

Ali Abduelmula, Wesam Hussain
Faisal Bagelagel, Ahmad Alghizi
Idrees Syed

In this case, we are changing the server on which we are backing up the file. The user will selection option 4 and enter in the IP address of the server. Once they enter the new server the files will automatically be backed up to the new server. And the user can continue to use the program.

Results:

After implementing the code and experimenting with different test cases, we examined our results. Our project resulted in 4 different test cases. These test cases resulted in different results each time, as they are shown above. Through this project, we were able to implement many various aspects related to networking that we learned throughout the course, from file transfers to TCP connections.

In the future, we would like to add a feature to able to find all the servers on the network and choose which one you'd like to connect to, instead of needing to enter a specific IP of the server. To do this we can implement UDP broadcasting to find all the servers and their IP address on the network. We would then terminate the UDP connection and connect the server and client via TCP start the backup process. We were almost able to implement this feature but due to time constraints, we were not able to fully complete it.