P2P FILE SHARING

‘The Octopus’ Group

Team Members:

AHMAD, FARHAN

ALIPOURSIMAKANI, KAMRAN

ANDERSSON EKSTRÖM, MAX

BERNTSSON, FREDRIK

CHADALAPAKA, GAYATRI

GHASEMI REZAEE, AMIN

IQBAL, NAYYAR

KUKKAPALLI, NAGA VYSHNAVI

NYHLÉN, JESPER

ROUTHU, VENKATA SAI KALYAN

SHAD MANFEAT, SEYEDEH MERSEDEH

ZAREI, KAMBIZ

Type of Document: **Project Specification**

Version Number: **1.0**

Publication Date: **April 17th ,2016**

1. Preface

Section 2: Glossary and abbreviations

Section 3: Background

Section 4: Proposed Solution

Section 5: Limitations

Section 6: Time Plan

Section 7: Project Organization

Section 8: Configuration Management

Section 9: Progress Tracking

Section 10: Quality Control

Section 11: Risk Management

Section 12: System Release Plan

Section 13: References

Release V 1.0 on 2016-04-17

- Initial Release

2. Glossary and abbreviations

Java - A general purpose object oriented programming language

Apache Maven - automated build tool for java code. Build the source code to a working ". Jar" file.

JUnit - Small explicit test to see that the code's functionality does what you think it does. For example, 3+3 should generate 6 if it does it pass, if it generates another number it fails.

Git - A tool that keeps track of changes in the source code and lets people collaborate.

Gitlab - A program that has the Git functionality for the users and then uses a website with additional features. In order to let people, collaborate with Git it is needed to be run online on a server.

Dark Peer - The peers who first connected to the bootstrap server and get a list of peers from the bootstrap server. After its validity time, it doesn't talk to the bootstrap server anymore.

Dark Content - The file contents which exists on the dark peers

Swarm - The main file which each peer gets from the server at its first connection which includes the shared files, the list of peers who shares the file and the swarm metadata

Swarm Metadata - Includes file names, file message digest

File Metadata - The set of headers together with the filename is the file metadata.

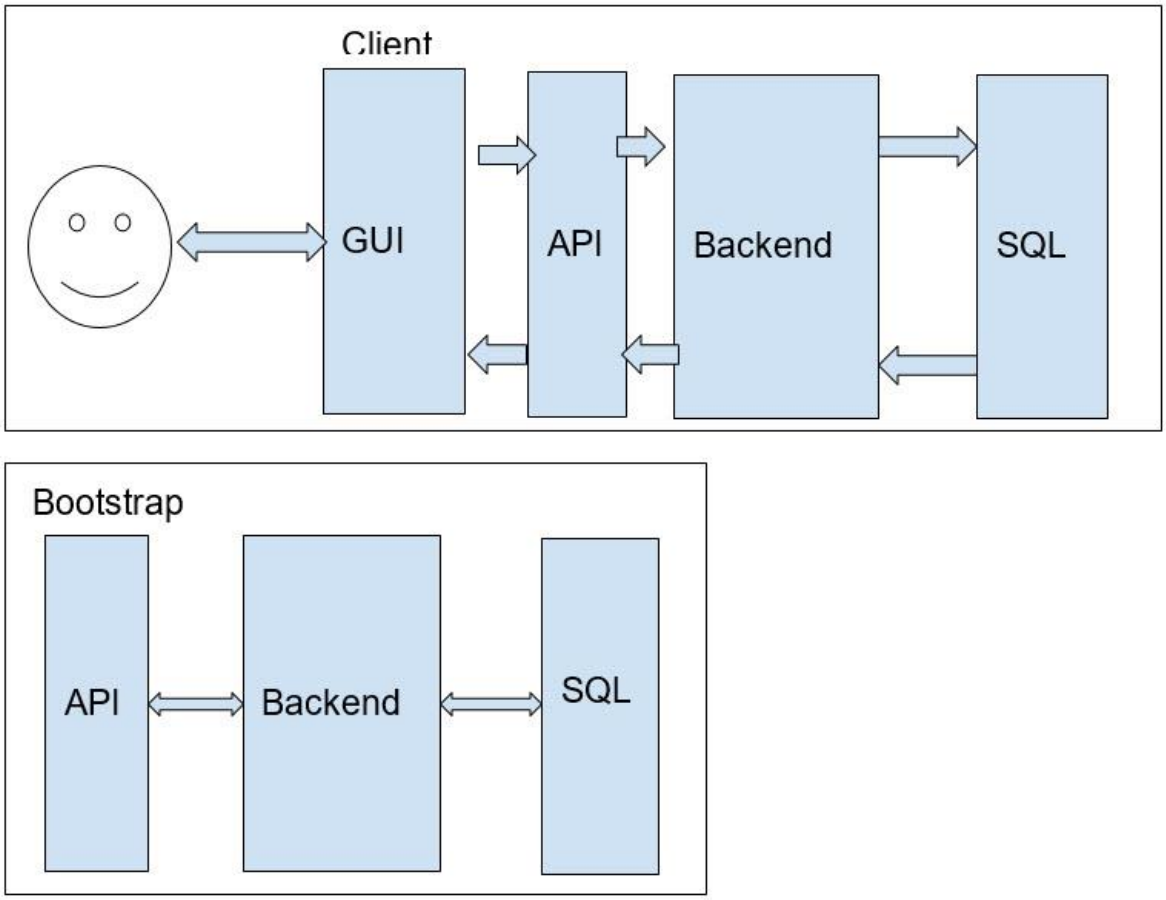
Bootstrap Server - the main server which new incoming peers connected to and get their swarm metadata from.

3. Background

The customer P2P4FUN requests to design a simple yet efficient P2P file sharing application which should satisfy its needs as follows:
Upon start, the peers connect to the bootstrap server and they receive a list of published swarms and a list whit banned IP addresses are which they should refuse to communicate with.
Each peer is responsible to refresh the validity time for its IP address; IP addresses that exceed the expiration date are removed from the bootstrap server.
There will be at least one backup bootstrap server. If no bootstrap server is available, the peers use the current data they have and will receive a message telling them about no bootstrap server where available.
Each peer must maintain a maximum of 3 connections to other peers.
the P2P application must have a friendly graphical user interface (GUI) providing all necessities to create, list, and delete swarms, and to search, upload and download files. In addition, there should be a way to select if a peer goes dark or becomes visible again. Progress of uploads and downloads must be shown together with estimated time to complete. The GUI must also show the IP addresses associated with a swarm and the IP addresses of the connected peers.
The application must also keep track of upload and download bandwidth and show this as live time-series plots.
P2P4fun insists that all communication must be encrypted. Plus, It should be possible to restart the system with encryption disabled for debugging purposes.

Finally, P2P4fun requests that all interaction user-to-user and user-to-server be based on a RESTful API with JSON data encoding.
your product supports at least 3 peers.

4. Proposed solution



In order to make the software flexibility for changes it will have loose couplings between frontend and backend. The frontend is the GUI (Graphical user interface) and the backend is where all the calculations and program logic is. The communication between the backend and the front end will be through a simple API which will make it as Lego bricks, easy to change and replace as long as the format is followed.

Communication between the client and the bootstrap server will be through REST - calls over HTTPS where the data will be encoded in JSON. Also the communication between client and client will be with REST and encoded in JSON (peer to peer traffic). The communication will be from the API on the client to the API on the bootstrap, the “API” is the face outward for the program.

Java programming languages will be used for the client and bootstrap server.

We connected our bootstrap to our API, to define swarm metadata or file metadata like our peer lists (as whitelist or blacklist) and other functionalities which can be managed easily from the API.

A database will be used to store the swarms created by peers, the file metadata and the servers' metadata.

5. Limitations

The bootstrap server won't allow peers to download the file from the server just the metadata that is needed for downloading from peers.

The client will be platform dependent and run on windows and if possible also Ubuntu.

The client will support 3 peers using the service and if possible even more.

There will only be one Bootstrap server as backup.
The service will not be available all the time.

The client and Bootstrap server will be in English only.

6. Time plan

We will be having 5 sprints which are exactly based on the deadlines of our assignments. First sprint will be this Sunday and we are going to submit our project specification and software requirement specification. It will be exactly the same for the next sprints. We are going to break down our project to smaller tasks and by using ProjectLibre (or at the same time using a Kanban board) we will choose tasks, analyze, develop and test them accordingly!

7. Project organization

WORK	TEAM MEMBER
1. Programming:	
A. Frontend:	ALIPOURSIMAKANI, KAMRAN NYHLÉN, JESPER ROUTHU, VENKATA SAI KALYAN KUKKAPALLI, NAGA VYSHNAVI IQBAL, NAYYAR
B. Database:	AHMAD, FARHAN CHADALAPAKA, GAYATRI IQBAL, NAYYAR
C. Backend:	BERNTSSON, FREDRIK ZAREI, KAMBIZ ANDERSSON EKSTRÖM, MAX SHAD MANFEAT, SEYEDEH MERSEDEH
2. Management:	BERNTSSON, FREDRIK ALIPOURSIMAKANI, KAMRAN ZAREI, KAMBIZ NYHLÉN, JESPER
3. Testing:	BERNTSSON, FREDRIK ANDERSSON EKSTRÖM, MAX ALIPOURSIMAKANI, KAMRAN NYHLÉN, JESPER ZAREI, KAMBIZ

4. Documentation:	ROUTHU, VENKATA SAI KALYAN NYHLÉN, JESPER GHASEMI REZAEE, AMIN KUKKAPALLI, NAGA VYSHNAVI

8. Configuration management

Version management Git will be used on a Gitlab server. The reason that we choose Git is in order for everyone in the team to be able to access and share the latest code easy. Also Git will keep track of the code history and who has written which part of the code.

System building Apache Maven will be used; the reasons for this is that the code will be in Java and Maven will automate the building process and with a POM file it will fix all the dependencies. A big pro with Maven is that in order to build the project it has to pass the JUnit test (See section code quality), this will make sure that the code uploaded to the git is not broken.

Release management

Our release management begins in the development cycle.If our customer request us for some new features or changes for instance. If the request is approved, the new release is planned and designed. The new design enters the testing or quality assurance phase, in which the release is built, reviewed, tested and tweaked until it is ultimately accepted as a release candidate. The release then enters the deployment phase.

9. Progress tracking

After creating the baseline schedule, it's important to track the progress of work on the project. Since different sub-groups will be created and tasks will be assigned to each group. So for tracking purpose, all members use some sort of tracking tool which is ProjectLibre. We show our progress using it. In order to ensure that our project delivery meets the deadline. First step would be to agree on a fair deadline.

10. Quality control

In order to have a high quality of the code there will be regularly code reviews especially on complex functions. Commenting will be in English and common code standards will be followed. There will be task for each user requirement due to milestones that will be tested. Junit testing will be made where it is possible to make sure that functionality is working as expected.

11. Risk management

Risk	Likelihood (1-3)	Significance (1-3)
1. Management	3	3
2. Group room	3	2
3. Communication	2	2
4. Lack of knowledge	2	2
5. Sickness	2	1
5. Moral	1	3

1. **Management** is the key to success, we are twelve persons so if not everyone has something to do all the time we will lose lots of time. In order to avoid losing time we will use Kanban board with tasks that each member can grab.

2. **Group room** will be a big problem because it will make the communication harder. But also that it's hard to have a real schedule for work and working home is not as efficient. We will minimize the risk with using task tracking tool (Taiga) and time tracking tool and use a centralized repository for the code, Gitlab.

3. **Communication**, the team has many nationalities and not the same mother tongue. The communication will be done in English and it is easy for misunderstandings. To minimize the risk, we will have many face to face meetings.

4. **Lack of knowledge**, the area we are working in is new for many in the team and not everyone is comfortable with the practical part of programming. The security part is new for almost everyone and also working in this large groups.

5. **Sickness** with team will make us lose valuable time and knowledge. To minimize the risk, we will have meetings to keep track of what each person does and two people that should know about the areas.
6. **Moral** will be a problem if the team get stuck. In order to minimize the moral troubles, we will bring Fika if the moral goes down and talk about it.

12. System release plan

12.1 Testing plan

- User acceptance Testing: The first testing phase is 15 May
- System Testing: The first testing phase is 15 May
- Performance Testing: The first testing phase is 15 May

Testing plan:

After developing the complete product, tests are performed. Few members of group will run tests to ensure efficiency of the product and to get easy accessibility of tool to the customer. Tests are mainly performed to check whether the tool runs in the work environment and also ensure that the tool satisfy customer requirements.

Packaging plan:

A compressed tar.gz archive is available to the user which includes of the alpha code, library files, related documentation and tools. The release plan details are as follows

Documentation plan:

Installation Documentation:

PDF format is preferable to release installation document. It includes the installation procedures of the configuration settings for various components.

Time Schedule:

Preparing installation document's time schedule is grouped to three phases with respect to the time plan.

12.3.2 User Documentation:

The PDF format is used to release User documentation. It includes the scope of the tool, tool functionality, linking different modules in the tool, process of supplying inputs to the tool, generation of output and various scripts.

12.3.3 Developer Documentation:

This Developer document includes about the scope of the project in future. This helps to develop the sources code and necessary information about the GUI's used in this project.

13. References

http://junit.org/junit4/

https://maven.apache.org/

http://www.oracle.com/technetwork/java/index.html

https://about.gitlab.com/

https://git-scm.com/