BLEKINGE INSTITUTE OF TECHNOLOGY

P2P FILE SHARING

‘The Octopus’ Group

Team Members:

AHMAD, FARHAN

ALIPOURSIMAKANI, KAMRAN

ANDERSSON EKSTRÖM, MAX

BERNTSSON, FREDRIK

CHADALAPAKA, GAYATRI

GHASEMI REZAEI, AMIN

IQBAL, NAYYAR

KUKKAPALLI, NAGA VYSHNAVI

NYHLÉN, JESPER

ROUTHU, VENKATA SAI KALYAN

SHAD MANFEAT, SEYEDEH MERSEDEH

ZAREI, KAMBIZ

Type of Document: Software Requirements Specification

Version 1.0

Publication Date: April 24th ,2016

BLEKINGE INSTITUTE OF TECHNOLOGY

1. PREFACE:

***Section 2*: Glossary and abbreviations**

Section 3: System architecture

Section 4: Requirements

Section 5: References

Release v1.0 on 2014-04-18

- Initial release

2 Glossary and abbrevations

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

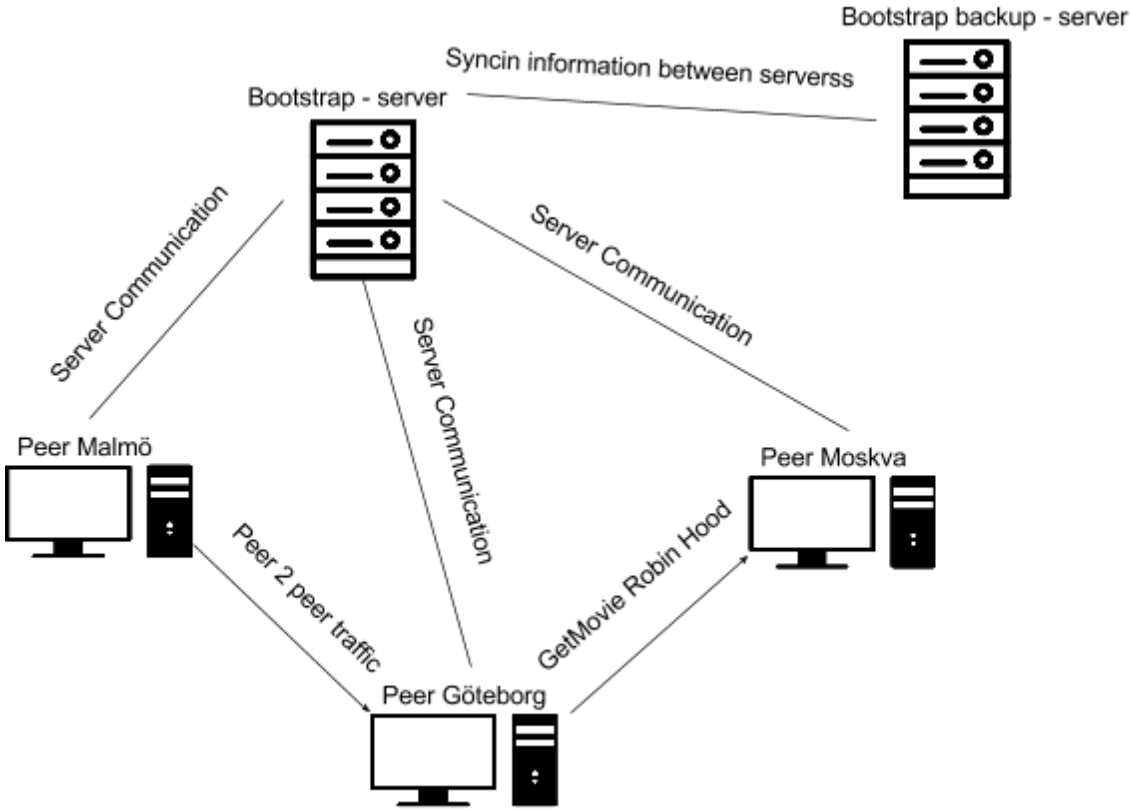
File Metadata - The set of headers together with the filename is the file metadata.

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

3: SYSTEM ARCHITECTURE:

The picture below is a high level description of what is going to be implemented. There will be a bootstrap server that contains swarm information , this information will the peers get through different REST calls, also when getting this information the peer is telling the Bootstrap server that it is alive and bootstrap will add it to its peers inventory which is a list of all peers it knows about.

The peers will now know about other peers and can connect to them. An example: *Peer Göteborg is bored and wants some fun, Göteborg search at the bootstrap server for different movies and decide to download Robin Hood. First the peer Göteborg will get the swarm data from Bootstrap server. In this information the client finds out that peer Moskva has this movie. Now Göteborg connects to Moskva and ask if he can download the movie. Peer Moskva answers yes and starts to send the movie to peer Göteborg. After some minutes Peer Göteborg has a copy of Robin hood. Moskva will update swarm information of the movie and add peer Göteborg to the list and send the list to Bootstrap server. Now if peer Malmö wants to download Robin Hood he can download it from both Moskva and Göteborg.*
This was a brief overview over how the system will work.



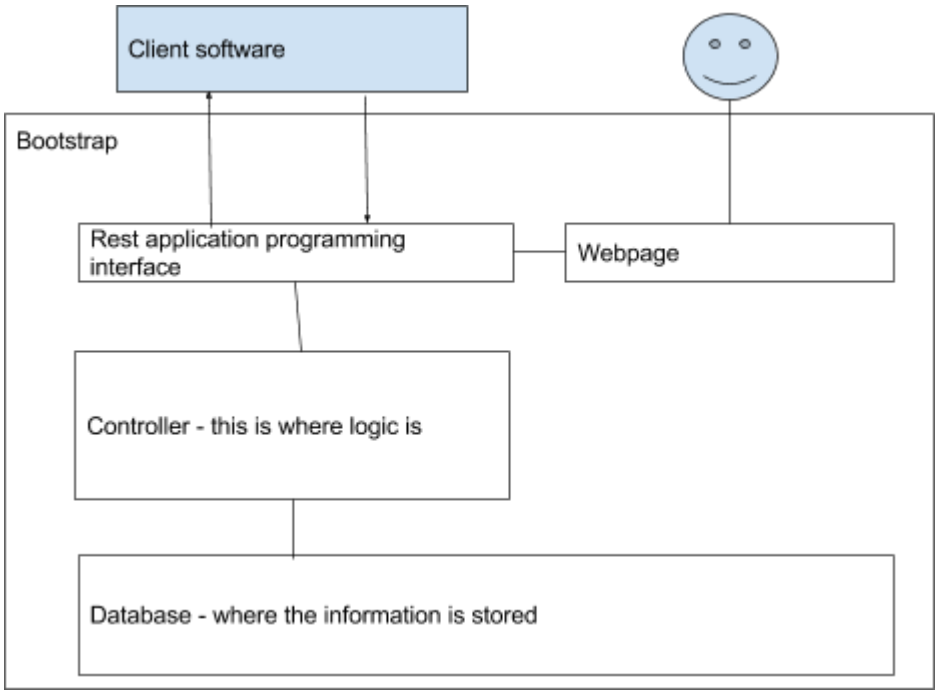
3.1 Bootstrap server

Below is a brief overview of the bootstrap server and what part it is containing of. The rest box is the software communication outwards to the world. It is not only letting clients connect to the bootstrap but will also provide a web page communication where one can see the information the clients can see. From the web page it will NOT be possible to download files. The data that bootstrap will send out will be formatted as JSON.

The controller is the “brain” of the bootstrap, this is where the programming logic will be placed. The controller will communicate with an database that will store the information that is needed on the server.

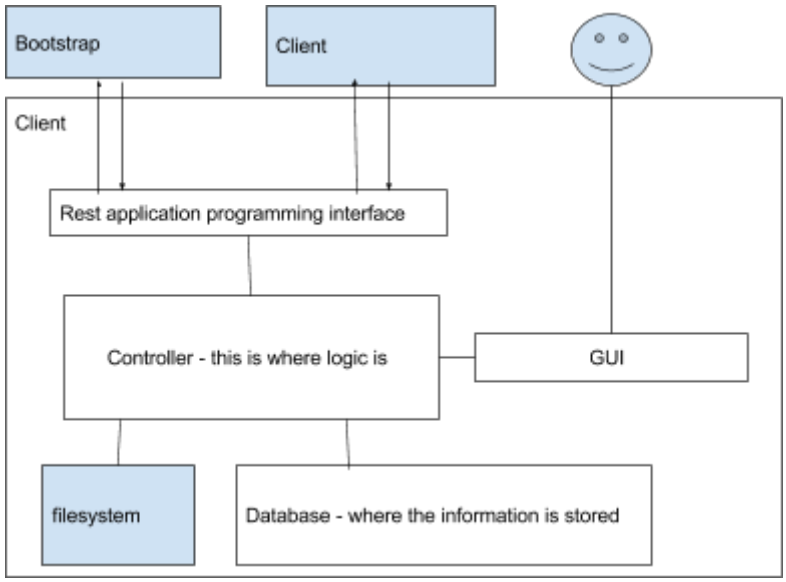
The database is similar to a text file but much more structured and easier to use when the data that is stored is of different kinds.

This means that the Bootstrap server will be controlled by the clients via the rest calls and not make decisions on its own.



3.2 Client architecture

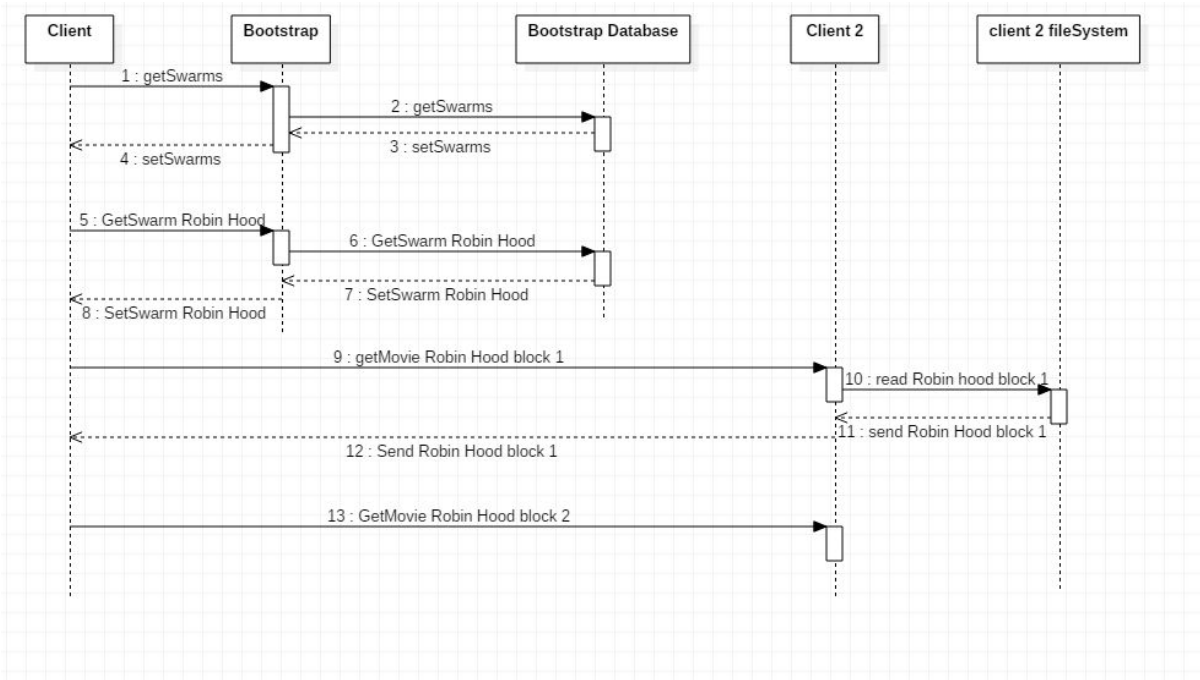
The client's architecture is similar to the bootstrap but with some changes:
The major changes are that it will connect both to the bootstrap and other client. It connects to the bootstrap to get swarms and search for files and to client to start download file content. A GUI will also be used for the user comfort which will communicate directly with the controller and it will decide which rest calls to make and what the GUI should present. Simultaneously the file information will be stored in a database. And the downloaded contents will be stored on the file system for ease of access by the user. Otherwise the building blocks will have the same functionality as bootstrap.



3.3 Sequence diagram for download

Below is a sequence diagram of what is needed in order to start download a movie. First the client will send a request to the bootstrap about target file (movie as an example). The Server replies with a list which contain the list of movie names, the client now choose one to get more information about and send a new request to the bootstrap server that respond with the information needed to start download the movie.

The client now sends a request to another client about getting part 1 of a specific movie and client 2 reads the necessary information from the the file system and sends back the first part of the movie. The client will request part 2 and this will continue until the movie is downloaded or that someone disconnects or an errors occur. It is not showed in the picture below but the bootstrap will get an update from client2 with updated swarm information about the specific file. Those pictures are simplified but shows the general view of how the program will be implemented and work.


Client
Bootstrap
Bootstrap Database
Client 2
client 2 fileSystem
1 : getSwarms
2 : getSwarms
3 : setSwarms
4 : setSwarms
5 : GetSwarm Robin Hood
6 : GetSwarm Robin Hood
7 : SetSwarm Robin Hood
8 : SetSwarm Robin Hood
9 : getMovie Robin Hood block 1
10 : read Robin hood block 1
11 : send Robin Hood block 1
12 : Send Robin Hood block 1
13 : GetMovie Robin Hood block 2

4.0 Requirements

The requirement will documented with the following formats:

- *ReqID, unique identifier with format Req-Module-unique number*
- *Functional/Non-Functional, lists whether the requirement is a functional or non-functional*
- *Description, describes the requirement*
- *Module, described which module will be mainly responsible for it*
- *Creation Date, date the requirement was created*
- *Change date, date the requirement was changed*
- *Dependencies, what other requirements it will depend on*
- *Test case, Id for the test case for this requirement*
- *Assignee, person responsible for the requirement*

Comments, notes regarding the requirement, for example changes that were made.

The full overview of the requirement are listed in a separate requirements sheet[1], due to easier editing and a better overview. The ReqID and description is listed below.

4.1 USER REQUIREMENTS

ReqID	Description
Req-Front_128	The system shall provide an option to make a set of files available for sharing in the GUI.
Req-Front_129	The system shall provide an option in the GUI to set a file as public to upload its metadata to the default bootstrap server.
Req-Front_130	The system shall provide an option to download a swarm in the GUI.
Req-Front_131	The system shall provide an option to create swarms in the GUI
Req-Front_132	The system shall provide an option to see a list of swarms in the GUI
Req-Front_133	The system shall provide an option to search for swarms in the GUI
Req-Front_134	The system shall provide an option to upload a swarm in the GUI.
Req-Front_135	The system shall provide an option to delete a swarm in the GUI.
Req-Front_136	The system shall provide an option to choose whether a swarm is dark or visible in the GUI.
Req-Front_137	The system shall show the progress of uploads and downloads.
Req-Front_138	The system shall show the estimated time to complete downloads
Req-Front_139	The system shall show each IP address associated with a swarm
Req-Front_140	The system shall show all IP addresses of connected peers.
Req-Front_141	The system shall show an error message in case a file digest fails and provide an option to redownload the file.

4.2 SYSTEM REQUIREMENTS

ReqID	Description
Req-Sys_101	The software shall be installed with the static ip address of the default bootstrap server.
Req-Back_102	All peers shall connect to the bootstrap server upon start.
Req-Back_103	All peers shall receive 3 randomly selected IP addresses of others peers upon start.
Req-Back_104	All peers shall receive a list of published swarms upon start.
Req-Back_105	All peers shall receive a blacklist of IP addresses that have been banned from the service upon start.
Req-Back_106	All peers shall block communication with the IP addresses that are banned.
Req-Back_107	All IP addresses except the default bootstrap server should have a validity time of 3 minutes.
Req-Back_108	All peers should refresh its validity time, this refresh should be done every 2 minutes and extend the validity time by a value of ? minutes.
Req-Back_109	All IP addresses that has exceeded their validity time shall be removed from the bootstrap.
Req-Back_110	All swarms without IP addresses shall be removed from the bootstrap server.
Req-Back_111	All peers and servers shall synchronize with a NTP-server and use the same timezone.
Req-Back_112	All peers should update their information from the bootstrap server every 5 minutes.
Req-Back_113	All peers which are not able to connect to the default bootstrap server, should connect to another bootstrap server. If that also fails, they should use the current data they have.
Req-Back_114	All peers which can not connect to a bootstrap server and does not have the necessary data should shut down.
Req-Back_115	When a peer receives the 3 IP addresses of other peers it should establish a HTTPS connection with them.
Req-Back_116	If a peer can´t connect to any of the IP addresses it receives it should receive 3 new ones from the bootstrap server.
Req-Back_117	A peer should at most have 3 connections to other

	peers at the same time.
Req-Back_118	The bootstrap servers should synchronize their data with each other every 5 minutes.
Req-Back_119	All swarms shall be divided into blocks with a sequence of 1014-bytes.
Req-Back_120	All swarms shall have a message digest which uses SHA-?.
Req-Back_121	All peers that connects to an IP address should get a list of swarms available on that IP address.
Req-Back_122	When a file has finished downloading the system shall recompute the message digest and verify the file is intact.
Req-Back_123	When a peer is uploading a swarm to another peer it should add their IP address to the metadata of the swarm.
Req-API_124	The system shall communicate between user-to-user and user-to-server with a RESTFUL API with JSON data encoding
Req-Sys_125	The system shall have an option to restart with encryption disabled.
Req-Sys_126	The system shall have encryption on all communication.
Req-Sys_127	The system shall support more than 2 peers.

5. REFERENCES:

[1] Requirements Document v1.0