# Public testing of a peer-to-peer based social network application

by

Florian Schulte



UNIVERSITÄT PADERBORN
*Die Universität der Informationsgesellschaft*

Fakultät für Elektrotechnik, Informatik und Mathematik
Heinz Nixdorf Institut und Institut für Informatik
Fachgebiet Softwaretechnik
Warburger Straße 100
33098 Paderborn

# Public testing of a peer-to-peer based social network application

Master's Thesis

Submitted to the Software Engineering Research Group
in Partial Fulfillment of the Requirements for the
Degree of

Master of Science

by
Florian Schulte
An der Kapelle 35
33104 Paderborn

Thesis Supervisor:
Jun.-Prof. Dr. Kalman Graffi
and
Prof. Dr. rer. nat. Christian Scheideler

Paderborn, May 2015

# Declaration

(Translation from German)

I hereby declare that I prepared this thesis entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them. This thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

## Original Declaration Text in German:

## Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

City, Date

Signature

# Contents


**1 Introduction** **1**
- 1.1 Motivation . . . 1
- 1.2 Solution Idea . . . 2
- 1.3 Overview . . . 2

**2 Current state of the software** **5**
- 2.1 Overview Structure . . . 5
- 2.2 Overlay . . . 6
  - 2.2.1 PAST . . . 7
- 2.3 P2PFramework . . . 9
  - 2.3.1 Configuration Layer for Overlay . . . 10
  - 2.3.2 Data structures . . . 10
  - 2.3.3 Access Control for Data structures . . . 10
  - 2.3.4 Monitoring . . . 11
  - 2.3.5 Test Simulations . . . 13
- 2.4 Plugins . . . 13
- 2.5 Webfrontend . . . 21
- 2.6 General changes . . . 27
  - 2.6.1 Name and Logo . . . 27
  - 2.6.2 Installation and Setup . . . 28
  - 2.6.3 Improvements for unattended operation . . . 29

**3 Testsetup** **31**
- 3.1 What do we want to achieve . . . 31
- 3.2 Virtual Machines . . . 31
- 3.3 Shell script for automated setup of nodes . . . 31
  - 3.3.1 Monitoring . . . 32
  - 3.3.2 Configuration File . . . 34
  - 3.3.3 Feedback . . . 37
- 3.4 Distribution . . . 38
- 3.5 User task . . . 39

**4 Evaluation** **41**
- 4.1 First test: PhD students . . . 41
  - 4.1.1 Monitoring Analysis . . . 42
  - 4.1.2 Reported Bugs . . . 52

4.1.3 Evaluation Results / Outcome of the first test . . . . . 55
4.2 Second test: Students groups . . . . . 55
4.2.1 First Try . . . . . 55
4.2.2 Second Try . . . . . 56
4.2.3 Monitoring Results . . . . . 56
4.2.4 Survey Results . . . . . 64
4.2.5 Outcome . . . . . 74

**5 Summary and Future Work** **75**
5.1 Achievements . . . . . 75
5.2 Future Work . . . . . 76

**Appendix**

**A My Appendix** **77**
A.1 List of reported Bugs . . . . . 77
A.2 Survey Sheet . . . . . 80
A.3 Monitoring Data Measurement 1 : Phd Students . . . . . 80
A.4 Monitoring Data Measurement 2 : Open Test . . . . . 80

**Bibliography** **81**

# List of Figures


2.1 LibreSocial Architecture !!!ADD SOURCE REFERENCE!!! . . . . 6
2.2 LibreSocial Network Overview !!!ADD SOURCE REFERENCE!!! 7
2.3 SkyEye.KOM Domain Subdivision !!!ADD REFERENCE!!! . . . . 12
2.4 Example of conversion from SQLite format to PDF files . . . . . 13
2.5 Data Structure of Photo plugin . . . . . 18
2.6 WebProvider . . . . . 20
2.7 Feedback Panel . . . . . 25
2.8 Screenshot editor with highlighted and blackened elements . . . . 26
2.9 LibreSocial Logo . . . . . 28

3.1 Flow chart of launcher script . . . . . 35
3.2 LifeSocial parameters (1) . . . . . 36
3.3 LifeSocial parameters (2) . . . . . 37
3.4 Screenshot of the Website . . . . . 39

4.1 Node count . . . . . 42
4.2 sent data . . . . . 43
4.3 received data . . . . . 43
4.4 Sent data rate . . . . . 44
4.5 received data rate . . . . . 44
4.6 Messages sent amount . . . . . 44
4.7 Message sent rate . . . . . 44
4.8 Pastry data read amount . . . . . 45
4.9 Pastry data read rate . . . . . 45
4.10 JVM Total Memory . . . . . 45
4.11 JVM Memory Usage . . . . . 45
4.12 Time needed to join network . . . . . 46
4.13 Number of wall views . . . . . 47
4.14 Number of Wall Entries . . . . . 47
4.15 Number of Wall Comments . . . . . 47
4.16 Number of Stored Files . . . . . 47
4.17 Number of File Downloads . . . . . 47
4.18 Number of friend requests . . . . . 48
4.19 Number of accepted friend requests . . . . . 48
4.20 Count of user blockings . . . . . 48
4.21 Maximum size of friend list over all logged in users . . . . . 48
4.22 Number of created albums . . . . . 49

4.23 Number of uploaded photos . . . . . 49
4.24 Number of views of the album list . . . . . 49
4.25 Number of sent live chat messages . . . . . 50
4.26 Messages sent by Messagedispatcher . . . . . 50
4.27 Messagedispatcher send rate . . . . . 50
4.28 Messages received by Messagedispatcher . . . . . 50
4.29 Messagedispatcher received rate . . . . . 50
4.30 Message lost count . . . . . 51
4.31 Global Storagedispatcher Stored Items Count . . . . . 52
4.32 Node Count . . . . . 56
4.33 Network Sent Data . . . . . 57
4.34 Network Received Data . . . . . 57
4.35 Network Sent Data Rate . . . . . 57
4.36 Network Received Data Rate . . . . . 57
4.37 Global Network Messages Sent . . . . . 58
4.38 Global Network Messages Sent Rate . . . . . 58
4.39 Global Network Memory Total Size . . . . . 58
4.40 Global Network Memory Usage Size . . . . . 58
4.41 Global Network Node Joinnetwork Time . . . . . 59
4.42 Global Plugin Wall Number of View Wall . . . . . 59
4.43 Global Plugin Wall Number of Wall Entries . . . . . 59
4.44 Global Plugin Files Number of File Downloads . . . . . 60
4.45 Global Plugin Files Number of Stored Files . . . . . 60
4.46 Number of friend requests . . . . . 61
4.47 Accepted friends requests . . . . . 61
4.48 Size of friend list . . . . . 61
4.49 Number of created albums . . . . . 61
4.50 Number of uploaded photos . . . . . 61
4.51 Number of album views . . . . . 62
4.52 Number of live chat messages . . . . . 62
4.53 Messagedispatcher messages sent count . . . . . 62
4.54 Messagedispatcher messages sent rate . . . . . 62
4.55 Messagedispatcher messages received . . . . . 63
4.56 Messagedispatcher messages received rate . . . . . 63
4.57 Messagedispatcher messages lost . . . . . 63
4.58 Storagedispatcher stored items count . . . . . 63

# List of Tables


4.1 Connecting to a network - answers of the users . . . . . 64
4.2 Wall feature - answers of the users . . . . . 65
4.3 Search feature - answers of the users . . . . . 66
4.4 Videochat Feature - answers of the users . . . . . 66
4.5 Text chat feature - answers of the users . . . . . 67
4.6 Photo gallery feature - answers of the users . . . . . 67
4.7 Messaging feature - answers of the users . . . . . 68
4.8 File upload feature - answers of the users . . . . . 68
4.9 Group / Forum feature - answers of the users . . . . . 69
4.10 Voting feature - answers of the users . . . . . 69
4.11 Calendar feature - answers of the users . . . . . 70
4.12 What do you like most in LibreSocial? - answers of the users . . . 70
4.13 What would you use LibreSocial for? - answers of the users . . . . 71
4.14 What features do you miss most in LibreSocial? - answers of the users . . . . . 71
4.15 Which features are unneccessary? - answers of the users . . . . . 72
4.16 How do you like the GUI of LibreSocial? . . . . . 72
4.17 What do you think about the GUI? - answers of the users . . . . 72
4.18 Relation between mobile and stationary usage . . . . . 73
4.19 Raspberry PI Version - answers of the users . . . . . 73

A.1 List of reported Bugs . . . . . 78
A.2 List of reported Bugs (cont.) . . . . . 79

# 1 Introduction

In the last years social networks became a very popular mean to communicate with other people. While most operators of social networks offer the usage to the users at no cost, they have to pay for their infrastructure like data centres and network capacity. To cover these costs, they earn money with their user base by placing advertisements and use the information the users produce passively or actively. This inherently conflicts with the desire of privacy of the users. Using a P2P approach, every user of the social network can share a small amount of his computation and bandwidth resources. This could either make a central infrastructure completely superfluous or at least drastically reduces the operating costs. It also means that there is not a single instance that has access to the users private data. Instead the data is stored at computers in the peer-to-peer network protected properly by an encryption. To make this approach fault tolerant and secure, a more complicated and therefore prone technical structure is required. With LibreSocial an implementation of a distributed social network exists that was developed at several German universities. It was developed in the last years by groups and individual students. In the project group of summer semester 2013 and winter semester 2013/2014 many things were improved and stabilized so that it makes sense to test it with a larger scale of users at this point. We expect that we get more information about the use of the application in a more realistic field. Users might behave completely different than the simulations and therefore they could also claim the resources of the network in another way. Furthermore a larger group of users also increases the number and the heterogeneity of system configurations which helps to identify technical issues.

## 1.1 Motivation

With the master thesis of [weifeng/dheeraj] a documentation exists that dealt with the systematic testing of LibreSocial in an artificial and controlled setting. This gives us information about the stability of the software under laboratory conditions as homogeneous computer equipment was used and a stable network connection was available. In real situations this can not be guaranteed. The users will use different hardware and internet connections. Also the connections might be terminated abruptly and larger parts of the network may become unreachable. Furthermore the tests that were made so far, were not very load intensive and peak usage, which may occur during special events, was not really simulated. To analyse the problems that arise in such cases, a more realistic investigation is

necessary. Therefore we plan to ask a group of people to use the software. Our goal is to find out how the software handles larger groups of users. We want to observe whether performance problems arise and if the software can be considered stable so that it can be opened for a wider group of people. Moreover we are interested in problems that the users have when utilizing the software. Our program tends to be somewhat more complex to set up than operator driven social networks due to the additional configuration options. On top of that it is interesting for us which functionality is actually used and if some functionality is utilized in another way apart from the intended use case because the proper options are not available. This gives us a hint in which direction further development has to go.

## 1.2 Solution Idea

To satisfy these intents, our plan comprises the following tasks:
At first we need to analyse the current status of the software to see where problems occur. As we want people to test the software that are not involved in the development the usage must be understandable and steps that need explanation should be simplified. All planned changes should be collected and categorized to decide what is possible and reasonable to change.
The next step is to do some internal testing to see whether the modifications work as intended and to identify difficulties that would lead to distortions of the test results, such as general bugs that prevent further testing of certain features.
The final task is the actual test with the participants. The usage has to be observed, emergent issues have to resolved and the results have to be evaluated. The tests should be performed in several stages. At first only a few technically experienced users will be asked to try the application and after fixing the problems more users are invited gradually if possible.

## 1.3 Overview

### Chapter 1

This chapter consists of the introduction and the motivation of this thesis.

### Chapter 2

In the second chapter of this thesis we see how the application is organized and what changes are made during the elaboration of this thesis.

### Chapter 3

Here we describe the test setup and the preparations that are not directly related to the application's source code.

## Chapter 4

This chapter contains the observations during the tests and the analysis of the results.

## Chapter 5

In the last chapter the whole thesis is summarized and a conclusion is drawn from the results. Additionally some ideas are presented that could be realized in the future.

# 2 Current state of the software

LibreSocial originally is a eclipse Rich Client Platform[1] application and written in Java. In the beginning of the development process this was a reasonable decision as it was developed using the eclipse IDE. Due to the fact that the graphical user interface was not very aesthetic and inflexible, the interface was rewritten in 2013 as a web application which now is used with a browser. So currently only a few specific parts of the software still rely on the SWT graphic library[2] which are not mandatory.
The application is organized in multiple Java projects that are managed via OSGi[3]. OSGi is based on Java and allows to dynamically load and unload the modules of the software based on dependencies that can be specified for all the individual projects. With this information the framework is able to organize the starting order automatically. As LibreSocial is based on Java, the software is platform independent and therefore available for multiple systems. Binaries are currently available for Windows, Mac OS X and Linux Operating Systems.
In this chapter we explain the structure of the whole application a bit closer and we also give an overview on the changes that are made to simplify the use for uninvolved users or to resolve different problems. We explicitly combine this two steps in one chapter so that the reader does not have move between multiple chapters to get an overview about the current state of certain features.

## 2.1 Overview Structure

The application is organized in layers that are more or less independent from each other. This cohesive approach theoretically allows to exchange single parts of the software. At the lowest layer the network functionality is implemented which is responsible for the connection management and routing of data packets. It organizes the mesh network which is used for communication.
The framework layer is the nexus between the technical network layer and the actual social network functionality. It is used to set the parameters for the overlay and also provides the abstracted communication and storage functions for the upper layers.
The third layer is organized as a loose collection of plugins which typically rely on the functionality of the framework layer but also can have dependencies between

[1] https://eclipse.org/
[2] The Standard Widget Toolkit; https://www.eclipse.org/swt/
[3] Open Services Gateway initiative; http://www.osgi.org/

each other.
The topmost layer is the graphical user interfaces which runs as a web interface in a browser. The interface between the application and the front-end is established using a special web provider plugin which translates and forwards the GUI requests to the plugins.

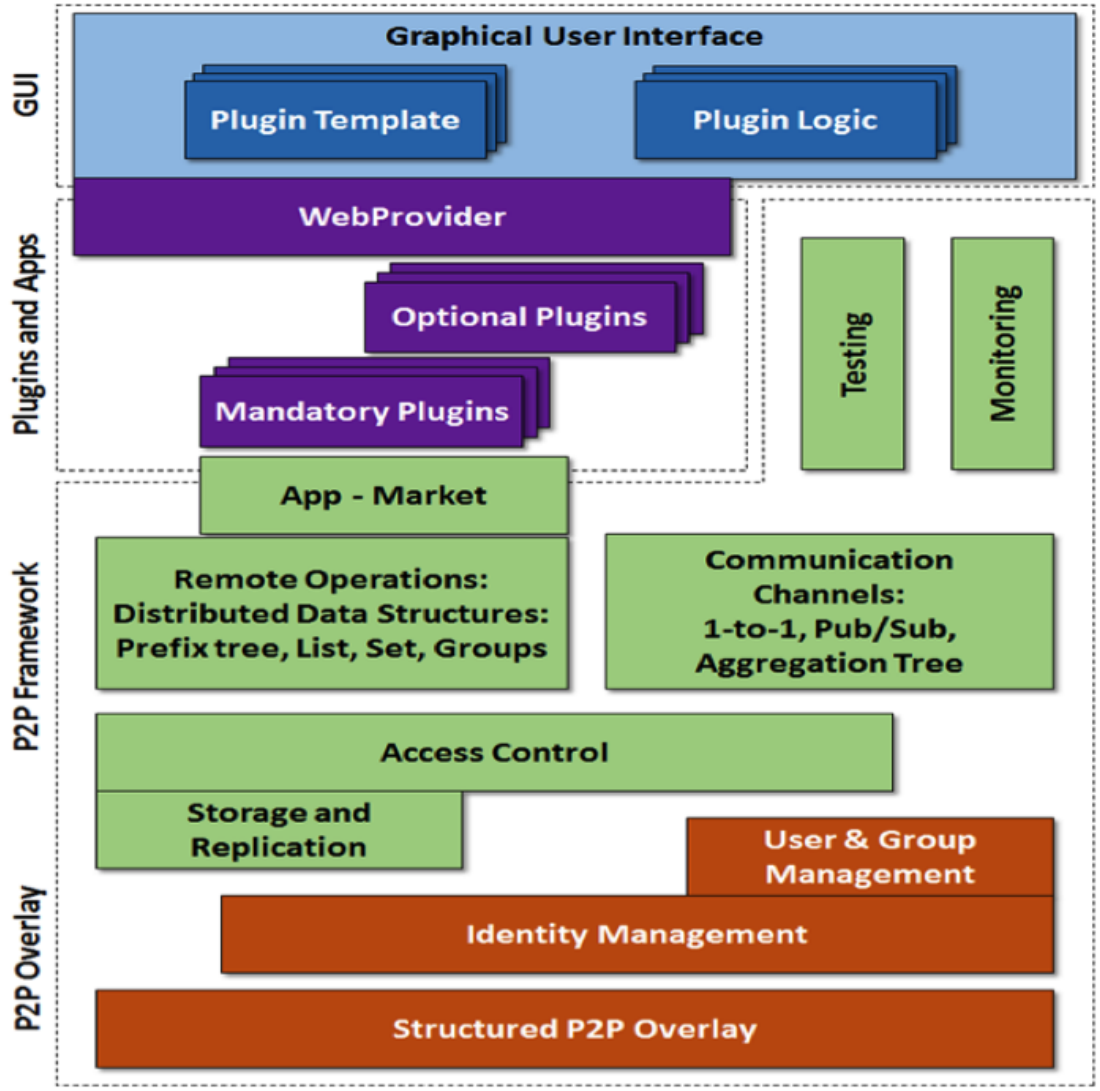



Figure 2.1: LibreSocial Architecture !!!ADD SOURCE REFERENCE!!!

## 2.2 Overlay

For the underlying network an already existing library named FreePastry[4] is used. FreePastry is an implementation of Pastry REFERENCEwritten in Java. It can

[4]http://www.freepastry.org/

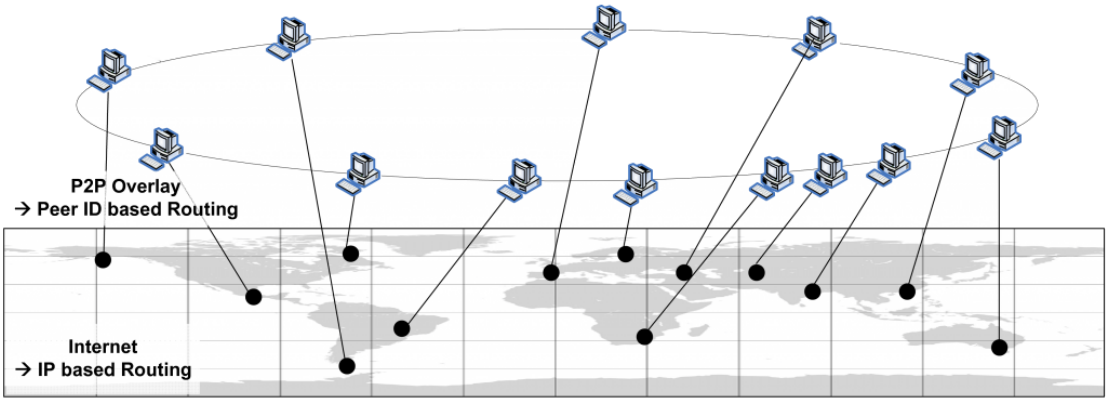



Figure 2.2: LibreSocial Network Overview !!!ADD SOURCE REFERENCE!!!

be used to combine multiple computers to a peer-to-peer network and allows to write an application that implements an actual application for it. FreePastry provides mechanisms to send messages between the nodes of the network and allows to store data in the network. For all computers that are associated to a FreePastry network, an id out of an id space is assigned which simultaneously serves as the unique node address. This id space has a size of $2^32$ Elements, so that it is rather unlikely that collisions occur. Also the objects that are stored in the network get an identifier. In this context the ids of the stored objects also determine which computer is responsible for storing the object by using the lowest numerical distance between the ids, whereby the id space is considered circular.

### 2.2.1 PAST

PAST is an application that is build on top of FreePastry just like the LibreSocial stack. The difference is that it is included in the FreePastry library and that it is not designed to be used as a standalone application. Instead it can be used by other applications as a persistence layer. In LibreSocial it is responsible for storing and distributing the actual user data across the computers in the network. PAST basically implements the methods that are characteristic for Key-Value-Stores:

- insert
  An item is stored in the network under the given id.

- lookup
  An item is retrieved from the network.

- lookupHandle
  Yields a handle for an object that is stored for the given id.

- lookupHandles
  Resolves handles for all replicas of an object for a certain id.

- fetch
  An item that is associated to a handle is transferred to the requesting host. So in contrast to the lookup method a handle is used as a reference.

- exists
  Checks whether an object is stored for a given id at the local client.

- existsInOverlay
  Checks whether an object with a given id is stored in the network.

With PAST the data objects are uniformly distributed across the nodes in the network. The identifiers of the data objects are either derived hashes from a specified key or randomly generated. The latter ones are typically objects that are somehow referenced used by intermediate objects that are accessible using a computable key. For instance the message list consists of an object that can be accessed directly (with a derived hash of "MessageInbox_MyUsername") but it references to other objects with random keys which hold the individual messages. The actual storage node in the network is then determined by using the node with the lowest numerical distance between the object and node id. As the nodes in the network may join or leave at any time, PAST also keeps track of file replication in such cases.

### Caching

To improve the response time of the application and the network, PAST uses a caching mechanism that holds data items that were retrieved recently. This allows to serve the content faster when subsequent requests arrive at the node, no matter if they are initiated by the node itself or if they come from different nodes. This helps to absorb peak demands for popular data items.

As the memory on the nodes is finite, the cache can only accommodate a limited number of objects. When an object is being fetched from the network, the algorithm checks whether there is enough space left in the cache memory. If this is the case, the object is stored in the cached to answer subsequent requests faster. If there is not enough place left, an old object has to be removed. Here the subject is picked that was not accessed for the longest time. To calculate the filling level of the cache and the size of the new object, the object must be known. As Java does not offer a simple call to get the object size due to the nature of the object references, the objects have to be serialized into a string representation first to get their length.

In the relevant serialization mechanism there is a deficiency. The object properties are converted into XML Data. As some the objects themselves contain serialized representations of the encapsulated encrypted storage objects, very long byte arrays are serialized twice. But here the XML serialization transforms each byte of the array into an own XML tag instance. This increases the size of the object by more than a magnitude. To counter this to a certain extend, the outcome is compressed. The problem with this is, that the compressed data is still much bigger than the actual object and a relatively high number of cpu cycles is needed to convert build the XML format. When the application is actually used, a larger number of objects is cached and inspected. Because the procedure is so slow, the CPU load of the whole LibreSocial process proliferates and the work can not be done in time.

To bypass this inefficiency, we are using an own algorithm that replaces the relevant part with the much more efficient kryo[5] serialization library. Experiments show that with the new implementation an object with a original size of 512 kilobytes can be processed in 2 ms. In contrast the other implementation requires about 500 milliseconds at the same computer and produces a much larger file. The JVM Serializers Project[6] also confirms our measurements.

### Persistence layer

As the data objects in the network exist as java objects and need to be saved to the hard disk to persist them, a serialization mechanism is used. In the old versions the XML Serializer is used just like it is for the caching, so also the same problems exists. Possibly the storage objects cannot be serialized fast enough so that the waiting queues spill over and messages have to be dropped. To avoid this we use the same strategy here and change the serialization to kryo which is significantly faster. The only difference here is, that the serialized data is actually used and stored to the nodes hard drive and loaded when a participant requests it.

## 2.3 P2PFramework

The framework forms the second layer of the application and is a not use case related part of the software. So this part would remain even if the foundation of LibreSocial would be used for other purposes than a social network. In any case it provides higher level functionality for storage and communication such as more advanced data structures, monitoring and testing services. On the other hand it controls the overlay layer. It is responsible for loading the application and network configuration and initiates the connection process.

[5]https://github.com/EsotericSoftware/kryo
[6]https://github.com/eishay/jvm-serializers/wiki, accessed 25.04.2015

### 2.3.1 Configuration Layer for Overlay

A central aspect of LibreSocial is to connect the users to each other. While the overlay manages the network once it is connected, the essential parts of the software to initiate the connection to other nodes is part of the framework. The network configurations are managed by the framework and are handed over to the overlay when the applications establishes a connection. Because the data is stored across the whole network and the nodes must be able to access the data objects, the clients must be able to establish connections to the nodes where the data is. This can be a problem when one of the clients is operated behind a firewall where incoming connections are blocked. To alleviate this problems, a relaying functionality is used to establish indirect connections but these still need a reasonable ratio of available computers to nodes that are protected by a firewall. To increase the amount of directly accessible nodes, in the new version UPNP[7] is now enabled, a functionality to automatically configure routers to open the corresponding firewall ports. Although it does not work all the time and is also disabled by the users sometimes for security reasons, it should help a bit to have more users in the network that can be contacted directly.

### 2.3.2 Data structures

Furthermore the framework provides data container classes for the parts of the software that build upon the framework. It acts as a abstraction layer for the different data structures and unifies them so that the data can be stored as an item in the PAST module of the overlay layer.

### 2.3.3 Access Control for Data structures

Obviously the stored data is user centric and therefore only predefined users are allowed to edit the data elements. In a distributed network every user could manipulate the software client to bypass simple access controls. In LibreSocial we use a public-private key pairs in combination with symmetric keys to encrypt and sign the data objects to guarantee privacy and integrity. Using the signature information the owner of an object can be identified and verified. One of the extensions of LibreSocial to the FreePastry library is the support for alterably content. When changed objects are about to be stored at one of the network nodes which already hold a copy of an older version, the owner of the original content is compared to the owner of the new content. Only if these two match, the object is stored.

[7] Universal Plug and Play

#### Distributed Data Structures

The low-level PAST library only supports single item objects. At many points it is necessary to store data in form of collections to map the relations between multiple objects. By using references the framework supports multiple data types of this kind. Although it is possible to use a storage item for each data unit of a data set along with a reference to the next item like a linked list, this approach would be rather slow. For each element of the set a new network transfer is necessary. To minimize the number of requests not only one, but multiple items of a larger set can be stored as a bucket in one storage item along with a reference to the next bucket. As there are less storage items used for the whole set, the individual buckets are necessarily bigger so it takes longer until the first data is completely transferred. Therefore a trade-off has to be found between these two metrics which heavily depends on the use case. By using a different structure of references also other data structures than linked lists like trees or sets are supported by the framework.

### 2.3.4 Monitoring

In LibreSocial not only the actual application forms a network of users, but also at the technical layer a mesh of nodes without central instances is created. This makes it more complicated to gain statistical information about the performance of the whole interconnected system like node count or network throughput. For this reason, a modules called SkyEye.KOM acts as an additional layer which collects the different metrics on the network hosts and brings them together. To efficiently aggregate the data, a tree structure is being built on top of the circular id space of the overlay. REFERENCEHERETheThe ring of node ids is considered as a linear id space which is recursively split into multiple parts in each iteration. Theses parts are called domains and form a tree with parent-child-relation. For each of these domains in every layer a node is responsible to aggregate the metrics that are computed by its child nodes. The available aggregation functions are SUM, MEAN, MIN and MAX. At the lowest level there are only single nodes left that only propagate their own values. At the topmost level of the tree only one root node exists which calculates the metrics for the whole network. These aggregated metrics are then sent to the lower levels again so that they are available in the whole network. The metrics cover a wide variety of performance indicators of the network and include network performance, storage statistics and user interactions for the individual plugins.

The implementation of SkyEye.KOM allowes to write the data in different file formats to be able to use this data in statistical evaluations. Thins includes the comma-separated-value format (CSV) and a gnuplot[8] data file format. Basically all of the formats were text based so that they are also human-readable. The problem with this was that many new metrics were added for the systematic test

[8]Graphing utility for multiple platfroms; Available at http://www.gnuplot.info/

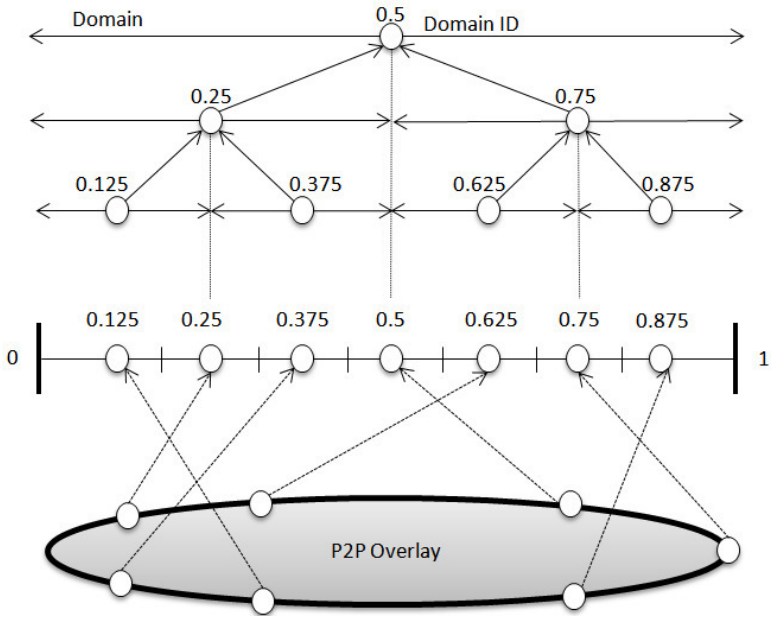



Figure 2.3: SkyEye.KOM Domain Subdivision !!!ADD REFERENCE!!!

phase. For the text formats the whole data file has to be read to the memory to be able to add the updated metrics due to the way the metrics were organized. This causes very high load after some time because the file is getting bigger over time. Since new data is generated every few seconds, after a certain period of time, the data file cannot be written anymore until the next data update arrives. This again causes that the available heap memory of the system fills up until the system becomes unresponsive. To circumvent this problem a new file format modules was integrated that writes the data to a SQLite database. This ensures that for each data update only the new data has to be written. Because the number of different metrics does not increase very much anymore after some time and due to the fact that databases are designed for such cases, the complexity for writing new updates is almost constant. The SQLite format even allows to read the data when the monitoring is still in progress, so that statistic graphs can be derived at run time without interrupting the measurement. Because the SQL output writer implementation is based on the JBDC API[9], the SQLite database driver can be effortlessly exchanged by other database systems. Especially for debugging a larger amount of computers, it could be helpful to use a single mysql database where all nodes in the network write their data to. Although the monitoring itself should keep track of synchronizing the statistics between all nodes, maybe some anomalies can be detected easier. The drawback of this method is, that the database has to be post-processed afterwards as the database format cannot be read by other tools. For this purpose a converter tool is supplied with the source code of the software that writes the data to a gnuplot data file. An example usage

[9]http://www.oracle.com/technetwork/java/javase/jdbc/index.html, accessed 23.04.2015

is shown in 2.4 along with the necessary gnuplot command that generates PDF files for the single metrics graphs.

```
SQLiteToGnuplot /path/to/monitoring_file.sqlite
gnuplot /path/to/monitoring_file/monitoring.plt
```

Figure 2.4: Example of conversion from SQLite format to PDF files

### 2.3.5 Test Simulations

The framework also includes a mechanism for automated tests. For these tests multiple computers are used to form create a network. Then predefined test plans are executed at the participating hosts. The test plan files either contain an exact schedule for the individual actions or a list of events without any time specifications that are started randomly. The operation is typically used in conjunction with the monitoring instruments to see the impacts of the tests.

## 2.4 Plugins

The plugins represent the third layer of the application. They basically provide the actual functionality of the application and form the use case. By using the generic API for storage and network access that are provided by the framework, use case specific features are implemented. For example the distributed data structures are internally used to store the wall entries of the users. The application comes with a number of plugins which are attached to the application as OSGi-Bundles. OSGi allows to define dependencies between the different plugins and dynamic loading during runtime. While all plugins presented here are always loaded when the application starts, this would be very convenient for optional plugins. The idea is that the users are able to install and use own plugins to extend the capabilities for a more specific scope such as business or learning.

### Login Plugin

The login plugin is the point of entry for the network. Its main task is to provide access to the registration and login process of the underlying framework. The login method is called directly by the user action and initiates the connection progress. When this has finished, the user is logged in. In contrast to that, the registration method is only a part of the user creation progress. The user's registration call is directed to the profile plugin which creates the user object and stores the necessary data. For this the registration method of the login plugin

is used which also establishes a connection to the network or creates it if necessary.

For the new version of the software we introduce a finer differentiation for the different cases of failure, so that the user can be informed whether the network is not available, the connection times out or the local port is in use instead of using a generic error message. This gives the user a hint if some settings have to be modified.

Moreover the login plugin contains a auto login mechanism which is used for unattended nodes that are connected of the network, but do not represent an actual user. This is helpful to add some computational power to the network by using dedicated computers. The mechanism was improved so that the application's tcp and udp port can be determined automatically.

## Profile Plugin

The profile plugin is used to create a data item that contains the users personal data. This item is stored in the network storage. For the id a unique identifier is used that is computed from the name of the user so that every user in the network can build the unique identifier if the username is known. To allow a granular adjustment which personal data is available for other users or user groups, the data for the user data fields (e.g. place of residence, hobbies, ...) can be encrypted.

## Search Plugin

Due to the face that the freepastry network in its present form is not designed to search items by its properties, a reference to the user is stored in a list of users for each property. So for instance for the country of residence an identifier is generated that consists of the field name and the field value. The object for the identifier key is requested then. If a corresponding object already exists, another user with the same combination must exist and the reference to the current user is added. Otherwise a new object with the reference is created and stored in the network. When a user runs a search, the identifier is built and the corresponding object is requested so that the list of matching users can be obtained. The problem with this approach is that different notations of the field values also result in different identifiers and only exact search matches can be found. To allow deviated search queries, a normalized form of the field is generated and used both for storing the reference object and the subsequent search queries. For the normalization the soundex algorithm[10] is used which yields same identifiers for similarly pronounced words. Because the algorithm is not very exact it happens that also dissimilar search results are found. For this reason the actual results are double checked using an algorithm that analyses the actual distance for each search result and

[10]http://www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm, accessed: 13.04.2015

the query. The results are accepted only when the value is lower than a certain threshold which is also depending on the word length.

### Friends Plugin

The friends plugin is used to store the relations between the users. It basically forms the social interconnection aspect of the network which can be absolutely different from the technical network structure. For each user a list object exists that contains references to other users. When one user signals somebody else that he wants to add him as a friend, a notification is send so that the other one has the option to accept or decline the request. Being a friend of another user, also includes additional privileges. While the plugin itself only exchanges the current online status between the clients periodically, the privacy settings of the other plugins typically allow to specify that either all users in the networks, the friends or specific users from the friend list can access the respective content. In the friends plugin also a blocking list exists. Users that are enlisted there are unable to contact the owner of the list.

### Notification Plugin

The notification plugin is used to inform users about new events. These events can be incoming messages, calls or friend requests. They are stored in a distributed list as encrypted notifications objects so that the user can always access them even if he was not online during the occurrence. As the notifications have to be written by other users the list itself cannot be write-protected and encrypted entirely. So malicious users could delete or overwrite elements from the list, but at least the notification objects themselves are encrypted so that the privacy of the user is preserved.

### Files Plugin

This files plugin is used to store and distribute larger amounts of data in the network. When the selects user selects a file from the local hard drive it is fragmented in chunks and encrypted by the application. For each fragment a predefined number of copies created which is transferred to other nodes in the network and stored under a random id. This ensures that the data is evenly distributed and no nodes are excessively burdened in terms of bandwidth and storage utilization. When all chunks are transferred, an additional StorageItem is generated and stored that contains all ids of the file chunks. Using the reference of this item and the decryption key all parts of the files can be retrieved and combined. This way it is easy to share big files with other users.

In addition to the direct file upload that is performed by the user, the files plugin is used as a data storage for other parts of the application. For instance

the messaging, multichat and group forum systems use the file uploads. The files the user is about to as message attachment, are uploaded and only a reference and the decryption key that belongs to that files are sent along with the message. This ensures that data is not duplicated for each recipient of the message.

For the new version a maximum file size limit is introduced to prevent single users from uploading masses of data which take up large parts of the limited amount of storage space.

## Appstore Plugin

As some functionality is not of interest for all users but for a very specific audience, it makes sense to allow the users to add certain features when required. For this reason an app store function exists which allows to dynamically load optional plugins from external resources. It makes use of the OSGi-Framework which allows to load, to start and to stop optional bundles after the application is started. The included implementation of the app store is was not adapted for the new graphical interface and therefore does not work. But a new version is being developed in the context of a parallel master thesis.

In the current version not usable as it was never adapted for the new graphical user interface.
A new version is being developed in a parallel master thesis.
Uses and will use OSGI bundle loading
Some changes for the webprovider plugin are necessary, so that plugins can be loaded after startup
This also applies to the browser part

## Calendar Plugin

This plugin allows the user to store the appointments and events in a chronological view. In the current version a single distributed data list is used for the agenda of each individual user. The present implementation does not allow other users to access the calendar and therefore it cannot be used as a collaboration tool. For the future this is an important aspect to improve as many other plugins could interact with the plugin. One thinkable feature are invitations that can be sent using the messaging or chat plugin or events could be managed with this plugin.

## Forum / Group Plugin

This plugin is used to give users, that share the same interest, a common platform where they can discuss their topics. Also each group has a virtual drive where they can upload files that are available for all users in the group.

## Message Plugin

The message plugin covers the mailing feature of the application. It is organized like known email clients such as Mozilla Thunderbird or Microsoft Outlook. The view is splitted into a folder pane and a message view. In contrast to the multichat plugin the contents are arranged by individual messages. It is also possible to send attachments to other participants of the network. The files that are sent are not actually sent along with the message. Instead the files is uploaded to the users individual file storage and a link is attached to the message. This way some bandwidth and storage space is saved.

messages can be sent to different users
can be sent to both on- and offline users
is stored as an storage object
online users are also notified via push notification (notification plugin)

## Multichat Plugin

The so-called mulitchat plugin allows to send instant messages to other users of the network. The chat messages here have to be distinguished from the messages which are sent with the messaging plugin. The live chat plugin lists the whole communication between a users and the conversation partner. Therefore the messages are typically shorter and less formal than in the other form of communication. The Multichat plugin is also capable of conversations between multiple users.

## Video chat Plugin

The Videochat plugin currently is completely separated from the multi chat plugin. It uses browser functionality to establish a video and audio channel between two users of the network. It is currently not possible to let more users join a video chat as this would raise the bandwidth requirement and complexity. This plugin only uses LibreSocial to notify the other conversation partner who is informed by a notification along with a ringing sound. When the connection is initiated once, a direct connection between the browsers of the two users is establish using WebRTC[11] for the video and audio data. As this technique is not implemented in all browsers, in the new version a warning is shown if the users tries to use it with an unsupported browser.

## Voting Plugin

This plugin can be used to carry out surveys among a group of predefined users or all users of a network. The initiator typically defines a question and some choices

[11] http://www.webrtc.org/, accessed: 21.05.2015

and then invites users to it. With the new version it is possible now to share the votings to the users wall so that the friends of the user can see it there.

## Photo Plugin

The photo plugin offers the user the possibility to upload photo galleries. Each album contains an arbitrary number of photos.

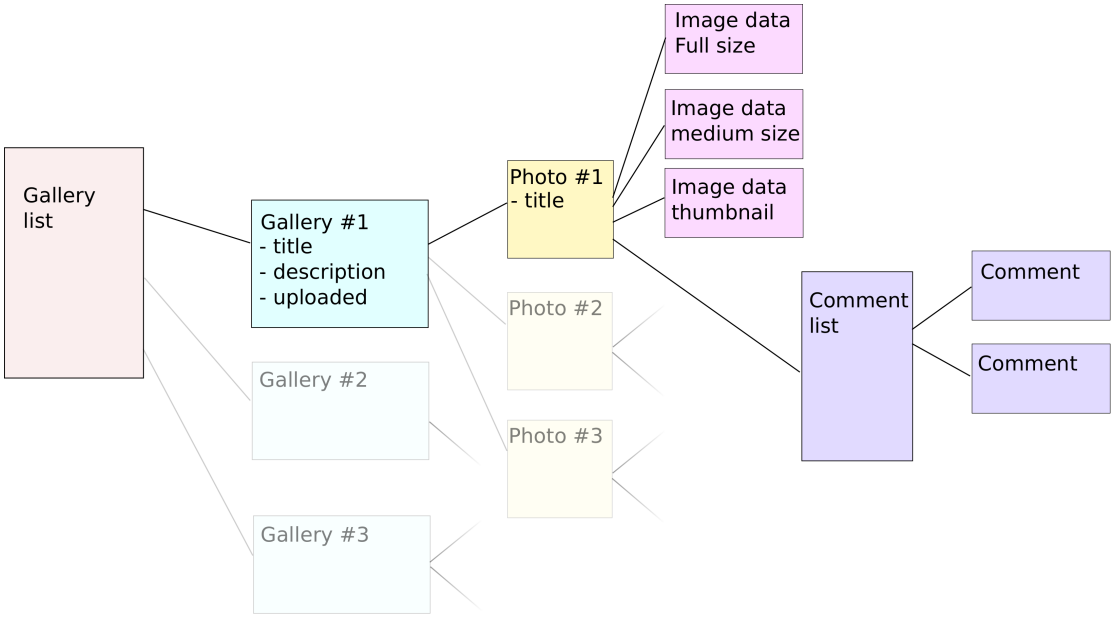



Figure 2.5: Data Structure of Photo plugin

Internally for each user a list of photo albums is stored as a distributed link list. The list is publically retrievable and contains references to the actual photo album elements. Theses elements contain some metadata like a name and a subtitle as well as a list of references to the photo elements that belong to the album. In order to safeguard the privacy LibreSocial allows to select for whom the gallery is visible. Unless it is stored in public mode, the album is encrypted can be decrypted only by the persons that are authorized to do so, by using their public keys. The photo elements consist of a title and a reference to a list which is used to store comments for the photos. The actual image data is stored in multiple sizes and variants. Theses include full format pictures but also cropped quadratic versions of the images to avoid the need for image resampling after download. This ensures that the thumbnail overview can be presented in acceptable time by downloading the miniaturized images only. Additional sizes for the photos exist, that are displayed when the user clicks on one of the thumbnails to see the bigger image and an even larger variant is used when the user wants to download the image.

### Performance optimizations

For the test version we reduce the size of the largest image variant. In the previous variant the image is saved in full resolution. This results in very large files which do not only occupy a lot of storage space but also result in longer upload times because the data is replicated to multiple nodes. By decreasing the maximum resolution to 1600 pixel per edge and setting the compression factor to 70 percent, the resulting image data has one tenth of the size of the original image file for typical camera shots.

To optimize display performance we implemented a preload strategy that downloads the adjacent image files when an picture is opened in the detail view. For users who flick through the photo galleries this makes the transitions much more fluent. Admittedly it is not as effective when the images are not accessed consecutively.

### Bug fixes

Furthermore a bug was fixed, that displayed the incomplete album when the upload is still in progress. To avoid this, we add an additional flag to the gallery entity which is set when all images are stored in the network.

## Wall Plugin

The wall plugin provides one-to-many-communication. Each user sees a wall page after the start of the application where he or she can post entries. This newsfeed does not only show the own entries, but also the friends things that are explicitly made accessible. The access control is managed by cryptography so that the texts are encrypted with a random symmetric key which in turn is asymmetrically encrypted for each user who is to have access. For the new version a new field in the data structure is added that is used to store attachment data as a serialized string. This field can be used for images or links that can be displayed along with the post by the GUI.

## WebProvider Plugin

The webprovider plugin has a special role in LibreSocial. It functions as the interface between the plugins and the grapical user interface. The webprovider uses the embeddable webserver jetty[12] to communicate with the users browser which acts a frontend. Although the jetty module uses a single network port, it satisfies three different tasks (so-called "contexts") which are distinguished by the request URLs. Figure 2.6 gives an overview of the structure.

[12]http://eclipse.org/jetty/

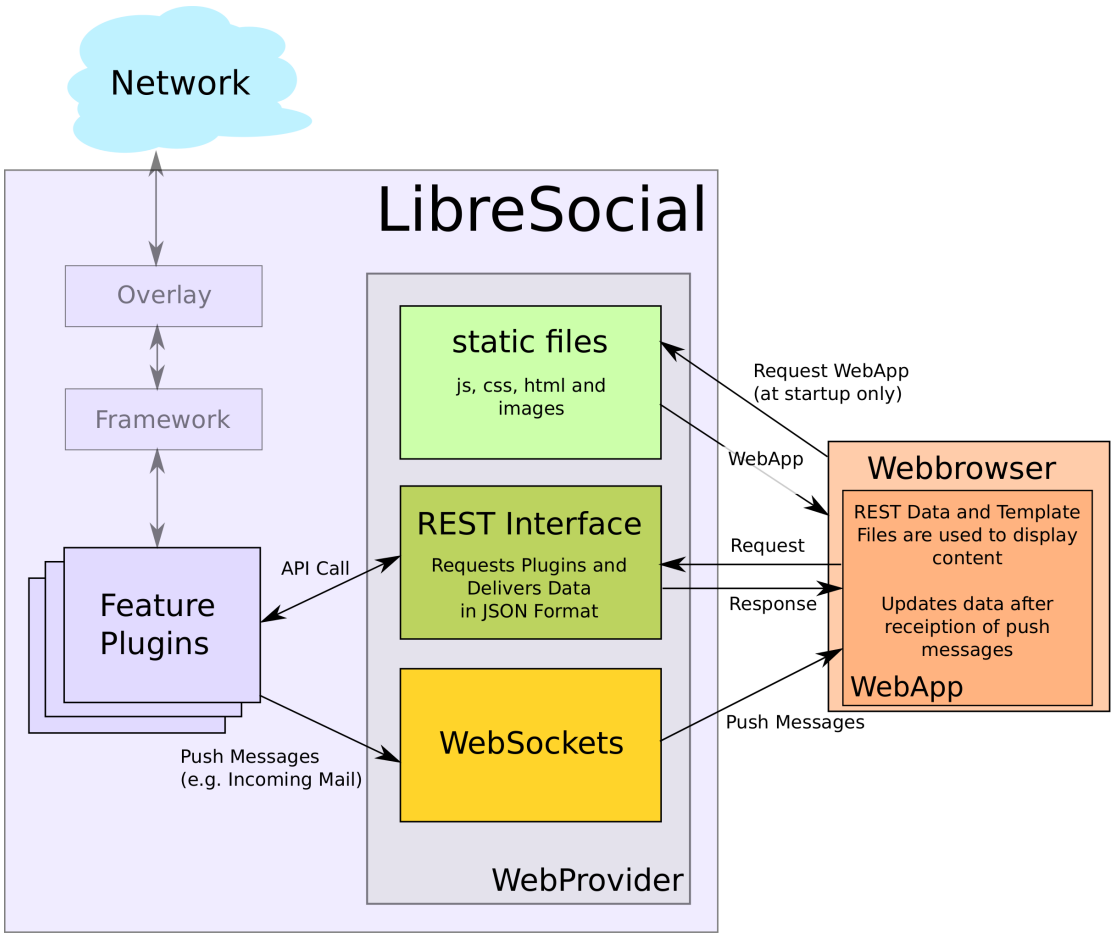



Figure 2.6: WebProvider

### Static File Server

As the graphical user interface is a single page web page based on javascript, the data and the presentation layer are combined in the browser only. So the graphical user interface theoretically is a more or less independent web application that retrieves the data from the backend and displays it to the user. When the LibreSocial is started, a installed web browser starts and retrieves the application data from the static webserver context. When the user interface is loaded at most some template files for the plugins and images are demanded afterwards.

### REST Interface

The actual data is retrieved through the REST[13] interface. This jetty context uses special handler classes for the individual plugins that access the API ex-

[13]Representational State Transfer, HTTP based data access

posed by the feature plugins and transform the response data in a serialized format. In our case we use JSON[14] format because it is bandwidth-efficient and can be processed faster than markup languages like XML, especially in browser environments which contain an javascript interpreter. The communication with the REST interface always consists of a request/reply cycle which is initiated by the gui application running in the browser. To display existing data, typically a GET-Request is sent from the browser to the REST interface which transforms the request into a API call to the corresponding plugin. The plugin serves the data from its local storage or initiates a network request to fulfil the inquiry. The data is then transported back to the gui which uses the corresponding template file to composite the view. When the user wants to create, delete or manipulate data, a POST-Request is sent to the LibreSocial main application. The requested actions is again performed using the plugins API and only a success or failure status is sent back to the frontend.

#### Websocket Server

As in social networks all participants can initiate communication with somebody else, the front-end initiated requests are not suitable for a user-friendly operation. With the technique explained above, data that is induced by other people or even the back-end application, the only way for the front-end to get notified about the data are periodical requests. These intervals are typically either very large so that the user has to wait for a relatively long time or cause a high load. To avoid this, a third web server context is used that uses persisting WebSocket connections, which can be used to send data from the back-end to the front-end immediately when new data becomes available. Normally only a notification message is send that informs the corresponding plugin about new data. The actual retrieval is still done using the REST-Interface.

## 2.5 Webfrontend

Whereas the last section dealt with the backend part of the graphical user interface, we show the structure and the changes of the web application the user is in direct contact with. In contrast to many usual website, the browser does not simply present the information coming from a server. Instead it runs as a JavaScript web application that manages the exchange of data with the actual p2p application and constructs the views itself by bringing together the data of the feature plugins and the corresponding templates. Apart from a few exceptions (e.g. for the login screen), the browser page is never reloaded entirely. Only parts of the screen are updated or exchanged by other modules. This approach has some advantages for an application like LibreSocial where communication takes place in real time. Multiple interaction objects can be kept open and updated

[14] JavaScript Object Notation

individually which is helpful when users are communicating with each other and doing other things in parallel. For example a user can have a video chat window opened while he is uploading photos. With the traditional way, either the upload would have to be cancelled or the chat has to be established again after page load. Because LibreSocial offers many different features and is designed for extendibility, a set of frameworks is used to provide standardized interfaces for plugin development. In Particular Knockback.js[15], Knockout.js[16] and Backbone.js[17] in combination with some extension libraries and their dependencies like JQuery[18], underscore.js[19] are used for this reason. Backbone.js basically is a software that manages the exchange of data between the two parts of the application. The data coming from the REST interface is stored locally that can be attached to HTML Objects. When the data model is updated (e.g. when a new set of messages was fetched), the corresponding view is updated automatically. Knockout.js in principle is used for similar things but has a more powerful templating engine then Backbone.js while it lacks some of the data sync functionality that Backbone.js offers. Therefore Knockback.js combines the best parts of both libraries so that the Backbone-Models and syncing functions are used in combination with the view and view logic of Knockout.js. While it is impossible to install the LibreSocial application on mobile phones due to the limited bandwidth and availability, it is at least possible to use the graphical user interface on such devices as there is no need to have the backend running on the same machine as the frontend. As users tend to use social networks on the go, we at least try to provide a user interface that works as good as possible on mobile devices. For this reason bootstrap[20] is used, which provides gui elements as building blocks that are arranged automatically in a way that is optimal for the used screen size. It also ensures that interaction controls like buttons or input fields are large enough for touch screen operation.

## Changes

Although the software was designed with end users in mind, this thesis is the first step to actually involve these users. With a closer consideration of the usability some aspects arise where optimization is needed. These are some visual tweaks and simplifications on the one hand, but on the other hand also some additional functions that encourage the users to interact.

[15] http://kmalakoff.github.io/knockback/, accessed 23.04.2015
[16] http://knockoutjs.com/, accessed 23.04.2015
[17] http://backbonejs.org/, accessed 23.04.2015
[18] https://jquery.com/
[19] http://underscorejs.org/
[20] http://getbootstrap.com/

### Login screen

The login screen is the first part of the application the user sees. As the concept of LibreSocial is to have a unlimited number of distinct networks, a bit more configuration is necessary than for traditional social networks. Because of this the login screen is split into two parts. In the first part either a network has to be created or a connection to an existing network can be established. In both cases the network is listed in a list containing all known networks. One of the usability improvements here is the automated selection for the local network interface. The application tries to determine the most suitable network interface for the network by shortly contacting one of the remote counterparts or simply using the interface that has access to the internet. In previous versions the user explicitly has to select the correct network device. This could confuse users that are not familiar with these terms and is also an error source for laptop users that frequently change from a wireless to a wired connection. In addition to this, a functionality was added to optionally show popup windows at the login screen. At the first time the application is used a tutorial appears which explains how to participate in a LibreSocial network. Another popup is used to optionally show a message that can be defined by the LibreSocial web site administrators to easily provide information about new versions or to inform about problems.

### Wall GUI

The wall plugin is the first plugin that is the part of the application the user sees when the application is opened. The reason for this is that it is expected that the most activity takes place here. The wall can be used to publish content which can be commented by other users. During the project group 2013/2014 the way how it works was modified so that the wall works like a news stream which displays all the posts from the friends for which this was enabled. As we think that the wall is a central point of contact for the end users during the public tests, the wall functionality is extended with post attachments. The post attachments allow to add a some media content that is then displayed along with the wall post directly in the wall view, whereas in the old version only text content is possible. The media data itself is generated by the other plugins, e.g. the photos plugin. A share button is available in these plugins that transfers some data to the wall plugin along with a template that defines the appearance. This is then visible in the posting form as a preview and in the actual post when it is published. This functionality is currently available for the photos plugin where a few thumbnails of an album are shown, for the voting plugin where a link to the voting page is presented. It is also possible to link files from the files plugin. Here either a corresponding file icon is used, or in case the file is an image, the image is presented directly.

### Photo Plugin GUI

The photo plugin allows to upload, share and view photo albums. If one user clicks on an album, all photos are shown as a thumbnail preview. When the users again click on one of these small images, a large version of the image is shown in a showcase view along with an optional image title and comments. In the new version some optimizations are included. The first thing is an improved preload functionality for the showcase view. As the users often click through a gallery image by image, it makes sense to load or respectively keep the next and the previous image to memory. This always happens when an image is loaded so that the user does not have to wait for the application that possible has to load the image from other nodes in the network. It also does not overstress the network as typically only a few additional images are loaded. The second improvement refers to the photo upload screen. When the user selects a number of files from the local drive to upload them to an album, a preview is shown which can be used to exclude images. The problem with this is in the old versions is that cameras often only add a special tag in the photos to indicate whether they are in landscape or portrait format. This means that the actual image data is always stored in the orientation and the viewing software is responsible for a correction. While this is already implemented in the back-end image store of LibreSocial, the front-end in the old versions is unable of this which leads to confusion of the users. With the new version the desired behaviour is also implemented in the javascript front-end part. The third improvement for the photoplugin is the sharing function that is already described in the Wall Plugin section.

### Search Plugin GUI

Worth noting for the usage of LibreSocial is the search plugin as it is the starting point for the users when they want to connect with others. The GUI includes a search form that basically accesses the search plugin using the REST interface. A number of different criteria can be specified to find the matching users. As the networks that will be created during the tests tend to be rather small-scale, an automated search is now performed when the user registers a new account. This search request suggests some users from the same country to improve the social cross-linkage between the participants. A further improvement is the selection field for the countries that is used instead of the standard text input field which is used for the country field in the earlier versions of the profile and search forms. The drawback with the old solution is that many countries have an international designation that differs from the local name (e.g. "Germany" and "Deutschland"). The old version of the software does not combine them to the same category. To unify the search requests and profile information, the software now always uses the international names internally, while it displays the variant that matches the selected GUI language.

### Feedback Tool

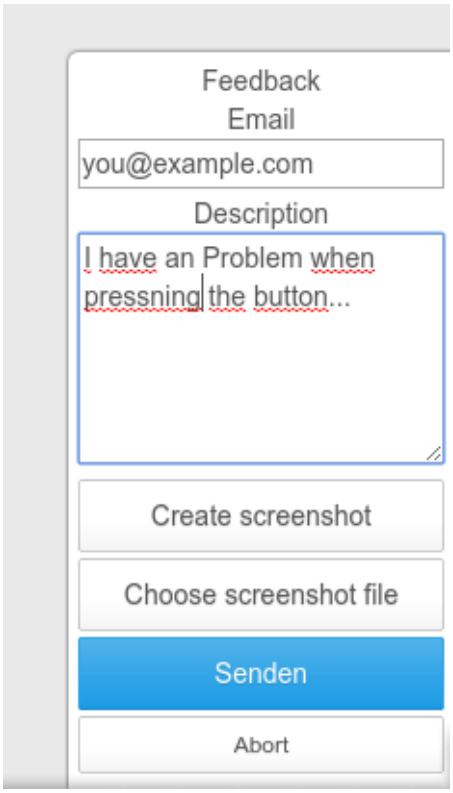



Figure 2.7: Feedback Panel

As the application is tested by volunteers that are not directly supervised during their tests, it is necessary to provide a simple way of giving feedback. It is especially interesting to let some users participate that are not particularly involved in the domain of software development because they can identify problems that developers do not see because of their routine-blindness. But for these users it is often harder to explain what the problems are, because they do not know the internal processes. In such cases as a developer it often helps to have a screen shot and therefore a function we provide a feedback tool which can automatically capture the screen. It is available as a small button in to bottom right corner and opens a small panel where the user can enter a description for the occurring problem and an email contact address for further questions. The feedback panel also allows to add an image or to take a screen shot. These images can be edited rudimentary to highlight certain aspects and to blacken private data. The form itself is submitted to the official LibreSocial web server which keeps track of the reports.

For the implementation of the screen shot feature some investigation became necessary. For security reasons the web browsers do not provide simple screen capturing mechanisms as private data could be exposed. There exist some approaches

like html2canvas[21] which try to emulate a browser engine and draw the page elements to a canvas. The problem with this approach is, that it is not perfectly accurate. As an alternative there are services like Usersnap[22] or Bugmuncher[23] which provide solutions for exactly this concern. Unfortunately they both have in common that they cannot work with web servers that are only locally accessible like the integrated gui web server of LibreSocial. To circumvent this problem we take advantage of the fact that in most cases we have an application running on the local machine in addition to the web application. Therefore an additional REST call is used which takes a screen shot from within the Java application that is then sent to the browser. To avoid capturing as little private information as possible an algorithm is used that cuts the screen shot to the size of the LibreSocial browser window. For this some markers are placed in the web application view during the capturing process that are used by the algorithm.

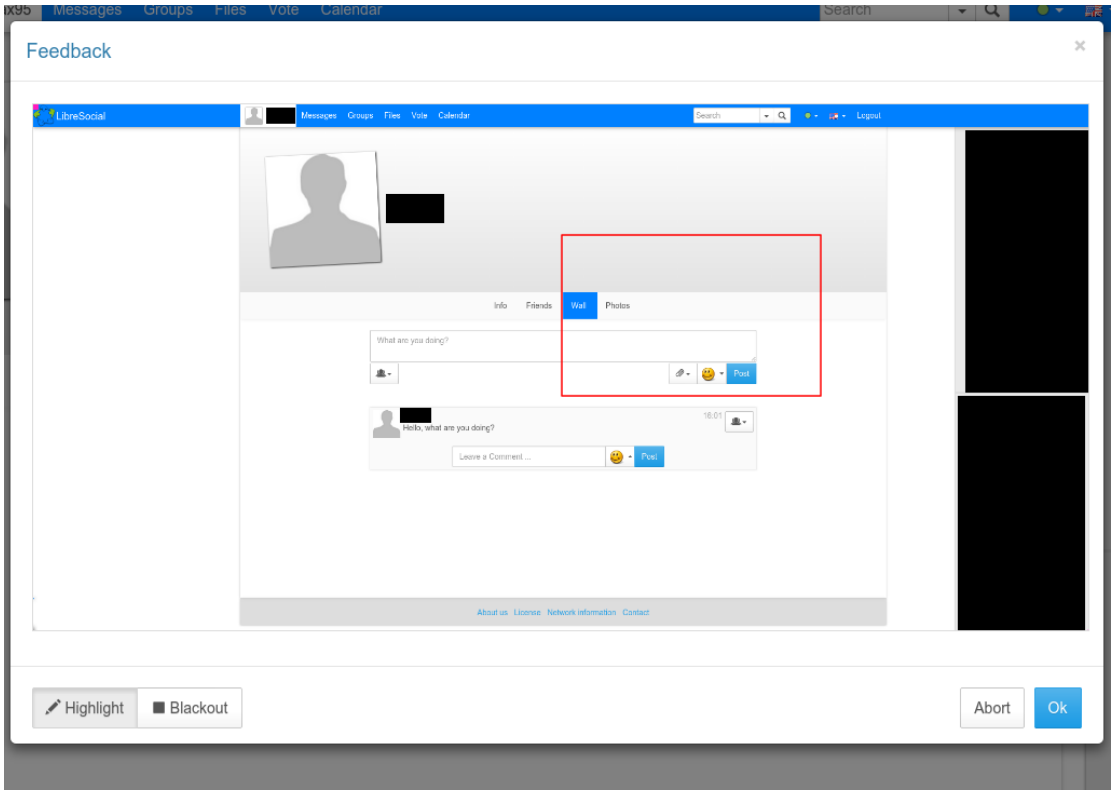



Figure 2.8: Screenshot editor with highlighted and blackened elements

[21] http://html2canvas.hertzen.com/
[22] https://usersnap.com/
[23] http://bugmuncher.com/

#### Security

The concept of LibreSocial is to store the user data at random nodes in the network. To ensure that nobody else can access this data, it is encrypted. With the change to the splitted front-end / back-end architecture this however induces a new problem. As all the cryptography logic is still remained in the back-end application and the connection between the two parts of the application was introduced, we have an insecure channel here. Even when both parts of the application run on the same computer, it may be a security risk here. To avoid this, the connection between the two parts is again transport encrypted using normal TLS/SSL[24] based HTTPS instead of HTTP. Also the browser vendors state that they consider unencrypted HTTP as deprecated in the future [25] and [26]. The problem with this again is that the browsers give out a warning when the application is used without a matching SSL-Certificate which are not even available for local ip addresses. This could confuse the users as it is shown by before they can see the first screen of the application. Therefore to circumvent this problem, a new domain is now used that points to the localhost ip address and has a valid certificate. It is worth noting that this approach is still not optimal because the private key of the certificate has to be shipped with the application and can therefore be extracted and used to decrypt the traffic.

#### License Information

As the software released for a public audience for the first time, it is important that all license information of the included libraries and dependencies are included. For this reason we now provide an overview page that lists the external components separately for each architecture layer of LibreSocial along with a link to the libraries origin and the corresponding software license.

## 2.6 General changes

### 2.6.1 Name and Logo

In the past the application was developed under the name LifeSocial. Due to legal reasons it was necessary to rename the software. So at the Hannover Messe in 2014 the application was presented as P2PSocial. To achieve an optimum acceptance for the name, we decided to reconsider the decision and ask the users themselves what name they prefer. For this we preselected some variants of names and logos

[24] Transport Layer Security/Secure Sockets Layer

[25] https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/, accessed: 20.05.2015

[26] https://www.chromium.org/Home/chromium-security/marking-http-as-non-secure, accessed: 20.05.2015

which were safe in terms of legal issues and started a survey.

For the name the following options were available:

- P2PSocial
- P2P Social
- LibreSocial
- Libre Social
- userdefined

In the end the the name "LibreSocial" received the most votes and for the logo the variant "Unity and Harmony" (Figure 2.9) was the most popular one.



Figure 2.9: LibreSocial Logo

The renaming of the application also required that many occurrences of the old name in the application are now changed. Also some internal identifiers that prefigure to the old name and even other internal names in the earlier versions are adapted in the new version so that the whole source code is more consistent now. To avoid problems however, some internal names that have some ingrained dependencies, are kept for the moment.

### 2.6.2 Installation and Setup

To ensure that the users are not overtaxed, the installation of the application has to be simplified. The program is currently deployed to the most important desktop operating systems, Windows, Linux and Mac OS X. Although it is a Java-based program, which allows platform independent development, some peculiarities have to be taken into consideration. For software projects that are written in Java and are exported for deployment, the Java Runtime Environment (JRE) is required. Additionally our software uses the JCE unlimited strength extension[27] of the runtime environment, which, however, are not part of the standard installation of it due to legal restrictions. As the software and the security extensions for the strong

[27] Java Cryptography Extension Unlimited Strength

algorithms are not installed on the testers computers with a high probability, we are supplying both along with our software.

As the usual way of software installation differ for the platforms, an individual package is created for each system. Because the process of generating this software bundles requires many steps, some bash[28] and batch[29] scripts are used to build these files. For Linux and Windows the archive files that are created by the eclipse export wizard the contents are reorganized and enriched with the previously mentioned JCE files. Additionally the currently unnecessary update mechanism "p2" is removed to save some bandwidth and storage space. The installation package for Mac OS X is created as a DMG-File[30]. The process is more complicated than for the other operating systems but allows to install the software via simple drag and drop.

### 2.6.3 Improvements for unattended operation

Since LibreSocial is using a frontend-backend architecture, it is possible to run the backend software on a computer that has no input devices or monitors attached and use a different device as a client. We also use this approach on the servers we offer as bootstrap nodes for our test networks. For this case of application it is very interesting to run the software on linux hosts without any dependencies on a graphical software stack. To achieve this, the tray icon, which is the only graphical element that is still initialized by the backend application, is now only started if needed. In combination with the reduced memory usage for the application itself the application can be used on much weaker systems.

[28]Bourne Again Shell, a scripting language primarily used on unixoid operating systems
[29]Scripting Language for Microsoft Windows
[30]Apple Disk Image Format, often used to deploy software because

# 3 Testsetup

## 3.1 What do we want to achieve

In the tests we basically want to find out, how the network performs in real world situations on the one hand and what insufficiencies the users see in the application on the other hand.

We assume that the users find a number of bugs in the software, some of which may result in instabilities. Some of the problems could keep the users away from other functionality that is therefore not tested then. So it will be necessary to do the tests in multiple stages so that problems can be eliminated in the meantime.

## 3.2 Virtual Machines

For the tests we set up a network instance to allow all participants to join it. Although it was possible for them to create their own networks, we encouraged them to use our network, so that we have a larger user base for the performance tests. For the network we used ten virtualized computer systems that were run in the universitie's data center.

The virtual machines had the following soft- and hardware configuration:

- Processor: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz (Only one cpu core is assinged per virtual machine)
- RAM: 4056 MB
- Storage Space: 62 Gigabyte (virtual HDD)
- Network Interface: Gigabit Ethernet
- Operating System: openSUSE 13.2 (x86_64)
- Java VM: OpenJDK JRE 1.8.0

## 3.3 Shell script for automated setup of nodes

To be able to manage the initial nodes without having to manually start the software individually the virtual machines can be controlled with a shell script that handles the most important steps. The script was created during the project group

2013 / 2014 but we made some changes for the users tests. The basic functionality is presented in Figure 3.1. The script typically is started in a screen session on the desired test nodes. At the beginning some initializing steps are performed. For instance it creates some folders for the statistic data if they do not exist. As also the test network needs an initial bootstrap node, one node has to be started in this mode while the others connect to this node. From then on, they all have the same capabilities and all of them can be used as initiator nodes by everyone else. The script however has to know in which mode the instance should be running to control the autologin mechanism of LibreSocial. This can be specified by a parameter that is passed to the script or can be determined from the auto login configuration file, which is downloaded from the server. In this file a master host has to be specified and the script checks if the current hostname matches the entry in the file. If this is the case, the current node is configured to create a new network after start-up. After the initialization phase the scripts executes a series of commands in a loop. The first step here is to retrieve a list of commands from the server. Here only a few keywords are possible such as "new script", "reset all" or "exec". If the commands file contains, which is also downloaded from the central server, contains one of them, an action is performed. In case of a new version of script, this is downloaded and restarted. The reset command deletes all data from the LibreSocial directory except the monitoring data and restarts the current script. In contrast to that the exec command downloads an extra bash script, executes it and continues to run the start script. This is very helpful sometimes as it still works when SSH login does not work anymore (e.g. when there is too much load on the server because of LibreSocial itself, the java process can be killed). In case no new commands are available, the script checks whether there is a new version of the program, downloads it if necessary and stores this information by setting the *LAUNCH* flag. After that the application checks if the current host is the master node and checks whether a new network should be created. If that is the case, the *LAUNCH* flag is also set. In the next step the script checks if the application is not running for some reason or should be updated. If that is the case, the steps are performed. After that the script waits for 5 minutes and begins with the next iteration of the loop.

### 3.3.1 Monitoring

The monitoring module of LibreSocial writes the statistical information that is gathered from the network to a local sql database. The statistics are available on all nodes that have the monitoring enabled, because the data of all nodes is aggregated and made available to all nodes. So it is sufficient to use the results from a single node for the evaluation. Nonetheless the nodes are writing a copy on all of the virtual machines to have a backup in case of data loss or corruption. For the further processing and evaluation the database files have to be converted to the gnuplot data format. For this the conversion tool is used that writes all entries from the database to a text files and generates a corresponding gnuplot script file

which is used to generate graphs for all metrics of the whole measurement.

Because for the tests the application produces a large number of metrics, we we analyse the metrics in which we are particularly interested in:

- NETWORK_DATA_RECEIVED_COUNT
- NETWORK_DATA_RECEIVED_RATE_COUNT_bytes_per
- NETWORK_DATA_RECEIVED_RATE_MAX_COUNT_bytes_per
- NETWORK_DATA_RECEIVED_RATE_MEAN_COUNT_bytes_per
- NETWORK_DATA_SENT_RATE_COUNT_bytes_per
- NETWORK_DATA_SENT_COUNT
- NETWORK_MEMORY_TOTAL_SIZE_COUNT
- NETWORK_MEMORY_USAGE_SIZE_COUNT
- NETWORK_MESSAGES_SENT_COUNT
- NETWORK_MESSAGES_SENT_RATE_COUNT_messages_per !!
- NETWORK_NODE_JOINNETWORK_TIME_COUNT
- NETWORK_NODES_COUNT
- NETWORK_PASTRY_DATA_READ_COUNT
- NETWORK_PASTRY_DATA_READ_RATE_COUNT_kbytes_per
- PLUGIN_FILESTORAGE_GETFILE_COUNT
- PLUGIN_FRIENDS_FRIENDSHIPREQUEST_MEAN_COUNT
- PLUGIN_FRIENDS_FRIENDSHIPREQUESTACCEPTED_COUNT
- PLUGIN_FRIENDS_FRIENDSHIPREQUESTBLOCKED_COUNT
- PLUGIN_FRIENDS_FRIENDSLISTSIZE_MAX_COUNT
- PLUGIN_LIVECHAT_NUMBEROFCHATMESSAGESSENT_COUNT
- PLUGIN_PHOTOS_UPLOADEDPHOTO_COUNT
- PLUGIN_WALL_NUMBEROFVIEWWALL_COUNT
- STORAGEDISPATCHER_STOREDITEM_COUNT
- STORAGEDISPATCHER_STORAGE_STORED_PLUGIN_WALL_MEAN_COUNT
- MESSAGEDISPATCHER_MESSAGES_LOST_COUNT

### 3.3.2 Configuration File

LibreSocial and especially the overlay layer allow some adjustment of the network parameters. For the user tests the settings differ a bit from the configuration for the automated simulations as described by [WF/Dheeraj]. Also some parameters were added. Here we describe which parameters were changed for the tests and why.

#### heterogeneitycache

We use an increased cache size as this will improve the performance of the network.

#### replicationFactor

The replication factor defines how many duplicates of a data object are stored in the network. A higher number reduces the probability of data loss when many nodes are leaving the network. On the other hand each replica consumes storage at the network participants' computers. Also with the current implementation all duplicates have to be transferred to all destinations by the original poster, so bandwidth is split up. As a good compromise a factor we choose a factor of 4 here.

#### autoFriendRequestAccept

This option is typically useful for automated tests which do not involve any users. As we want to let the users try as much functionality as possible, the automated accepting of friend requests is disabled.

#### monitoringToGnuplot, monitoringToSQLite, monitoringToDat and monitoringToCSV

These settings were newly added to control which output formats are written by the monitoring plugin. As the plain text formats cannot be written fast enough anymore after some time, only the SQLite-output is enabled.

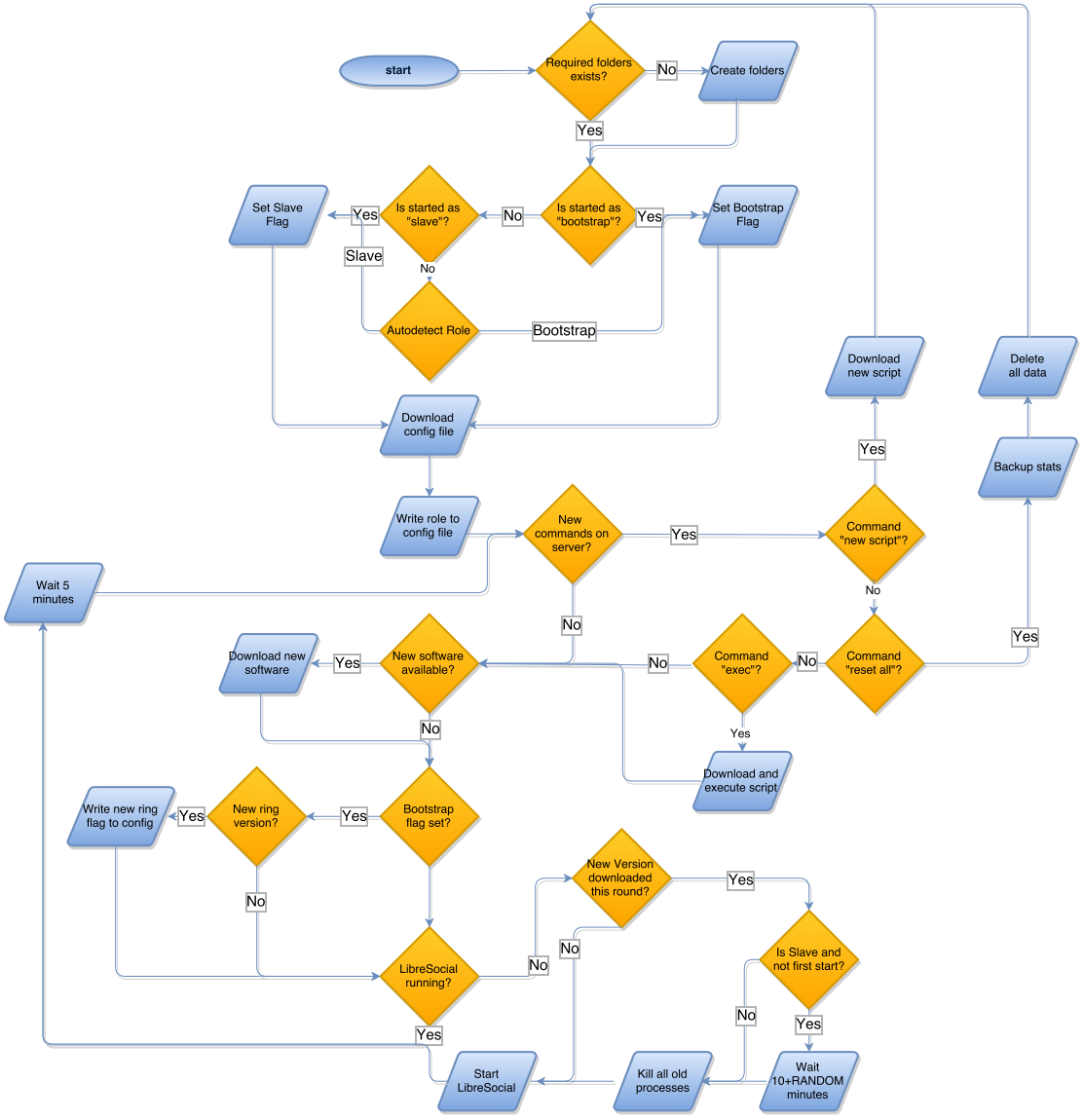



Figure 3.1: Flow chart of launcher script

```
# auto storage deletion before accessing into network (1 = enabled,
# 0 = disabled)
autoDeleteStorage = 0

# enables the heterogeneity awareness (1 = enabled, 0 = disabled)
enableHeterogeneityAwareness = 0

# size of the heterogeneitycache in MB
heterogeneityCacheSize = 24

# timeout for a canyouprovidemessage in ms
CanYouProvideTimeOut = 5000

# maximal number of providers for each data object
providerListLength = 1

# number of replications of a data object in the system
replicationFactor = 4

# enables the object size calculation for the monitoring
# (1 = enabled, 0 = disabled)
objectSizeCalculationEnabled = 0

# enables the random selection mode (1 = enabled, 0 = disabled)
randomSelectionModeEnabled = 0

# enables the measuring data function, the data measured located in Plugin,
# Framework levels (1 = enabled, 0 = disabled)
measuringData = 1

# enables the collecting of social metrics, such as friend requests
# (1 = enabled, 0 = disabled)
socialMetricCollection = 1

# auto start monitoring (1 = enabled, 0 = disabled)
autoStartMonitoring = 1

# enables the auto friend request accepted (1 = enabled, 0 = disabled)
autoFriendRequestAccept = 0
```

Figure 3.2: LifeSocial parameters (1)

```
# enables the auto live chat invitation accepted (1 = enabled, 0 = disabled)
#unused
autoLiveChatInvitationAccept = 1

# enables the auto voting user invitation accepted (1 = enabled, 0 = disabled)
#unused
autoVotingInvitationAccept = 1

# define monitoring output formats (1 = enabled, 0 = disabled)
monitoringToGnuplot = 0
monitoringToSQLite = 1
monitoringToDat = 0
monitoringToCSV = 0

# mobile nodes, if enabled, the odd port number stands for strong node,
# the even port number stands for weak node (1 = enabled, 0 = disabled)
# now weak node is not used, so we comment it.
#unused
#mobileNodes = 0

# visualization of monitored data in runtime, this function will keep the
# memory usage increasing! (1 = enabled, 0 = disabled)
#unused, only relevant for old gui statistics visualization
monitorDataVisual = 0

# url to the bulletin board
#unused
bulletinURL = a url
```

Figure 3.3: LifeSocial parameters (2)

### 3.3.3 Feedback

During the tests the participants have several possibilities to submit their feedback about the application. It is planned that the users send us an email for their general resonance. For more feature specific feedback they can use the integrated feedback tool. This is an utility that is directly integrated into the graphical web interface. When it is activated, a text input field is shown along with an upload selector and highlighting tool for screenshots. This way the users can report ideas and problems without leaving the application. When they submit the feedback report, the data is transferred to the official LibreSocial Webserver which generates an email from the data.

To manage the reports, we use a bug tracking system to manage the reports that is being sent to us via the various communication channels. It is located

at the LibreSocial website[1]. The bugs and feature requests are categorized by importance, time requirement and difficulty. By this classification we derive a priority and decide whether the problem has to be resolved before doing any further testing.

## 3.4 Distribution

For the distribution of the application a new website[2] is created, which informs the users about the concepts of LibreSocial and allows to download the software. For the website choose a modern layout in which we incorporate the new logo and name to hopefully attract some visitors to test the application. Also it is designed to be used as the official web presence later on. We added an email registration formular that had to be filled out by the testers before they could download the program files. This way we got the email addresses to send roundmails in case of problems or version update notifications. We also counted the number of downloads to see how many people we reached with our call for testers. In addition to that we generated an anonymous but unique identification number that was sent from within the running application to the webserver.

[1]The bugtracker is available at https://libresocial.com/bugs
[2]The web presence can be accessed via https://libresocial.com/




Figure 3.4: Screenshot of the Website

## 3.5 User task

To animate the test participants to become more active, we gave them tasks. This way we wanted to ensure that the users try the most important functionalities.

The task we gave them was:

> Until Thursday at noon you can upload and share some photos in an albums to the network. You can upload whatever you want, personal photos, your meme collection, whatsoever. Just make sure you post it in public mode so that everyone in the network can see it. Then send a friend request to the user "Florian" and post the album (again publically) on your wall using the share button in the photo overview. In the second phase a voting will be created with all the available albums and you can decide who uploaded the best set of photos.

# 4 Evaluation

In this chapter we describe our observations during the testing and analyse users' feedback. We are particularly interested in the users a few aspects. First we want to know what problems the users encounter when they are installing and starting the software because their system configurations and network connectivity may differ from those which are present in the synthetic tests. Especially the login process is a bit more complex than it is for traditional web based social networks where the users simply register on a website. In our case the users additionally have to connect to an existing network or create an own. We also have to see what impact it has when many nodes that are protected by a firewall join the network. The second aspect we are interested in are the actual bugs in the software the users come across. Although there are some weaknesses already known, the versatile operating system and browsers combinations will reveal some more. The third aspect we want to examine are the monitoring statistics which could help to identify reasons for occurring problems on the one hand and also show how cost-effective the network operates on the other.

## 4.1 First test: PhD students

For the first task a test group of PhD students is asked to try the software. The first step is to download and install the software. After that it has to be started and the users are asked to connect to the test network so that we are able to collect the monitoring results.

When the users are logged in, they are free to try out all features of the software. To increase the interaction between the testers, we advise them to upload some pictures to the photo gallery.

As we cannot supervise the testers during the tests, we have to rely on monitoring, the feedback of the users and the log output of the bootstrap nodes which give us information about completely crashed test instances. These problems however do not occur during this test. Instead we notice some problems after some time regarding delayed logins and missing walls or photo albums. These problems also have arisen during internal test during the development phase.

After some time we get emails from the testers that for the most part contain a list of bugs and feedback. Beside many minor bugs, we also receive several reports about problems with the login and logout process, which are no longer working correctly. We can confirm this behaviour when we try to connect to the network ourselves.

In addition to this some of the testers are intentionally or unintentionally testing some performance metrics and security features by rapidly clicking on buttons or using well-known techniques of attacking to reveal typical problems of HTTP interfaces.

### 4.1.1 Monitoring Analysis

In this part we analyse the statistics data written during the first test. We see how several different metrics develop over time and how this

#### Node Count

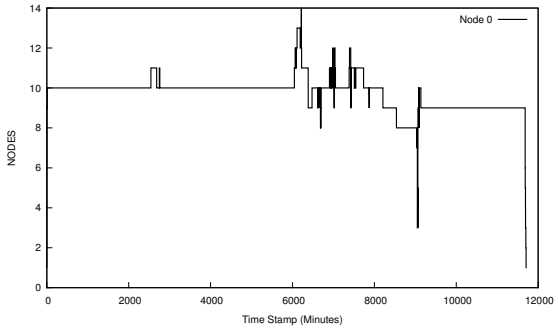



Figure 4.1: Node count

In figure 4.1 we can see the number of nodes that were connected to the network over the time. As we are using ten nodes as initial bootstrap nodes, it is not remarkable that the diagram shows this number of nodes in the first part. At about 2600 minutes to 3000 minutes one additional node connects to the network. As the work instructions were not sent at that point, this local maxima only indicates a test run.

After 6000 minutes of operation time some more nodes connected to the network with a maximum of 14 nodes at 6400 minutes which approximately equals 4 days and 10 hours. This value drops to 11 which shows that the users are disconnecting the clients after a while.

Interestingly the number falls below the number of initial bootstrap nodes for a certain period of time. It might be reasonably assumed that the network churn, which occurs due to wanted or unwanted disconnects, leads to reduced values. Nodes that previously delivered values may be unavailable now and therefore the overall values can decrease. Another explanation is might be that some nodes are unable to provide the monitoring data on time.

In the following time we observe multiple smaller peaks with subsequent small drops which can be tracked back to connections of users and the described reorganization of monitoring nodes.

At about 8300 minutes the node count decreases for a longer time which could indicate that some of the initial nodes begin to have problems. After 6 days and 4 hours an all-time low of 3 nodes is reached.

Although the value increases again to 9 nodes after some oscillations, considering the other diagrams it seems that some greater problems seem are occurring.

### Data Transfer

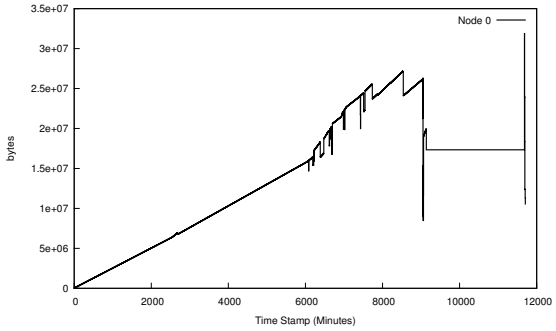



Figure 4.2: sent data

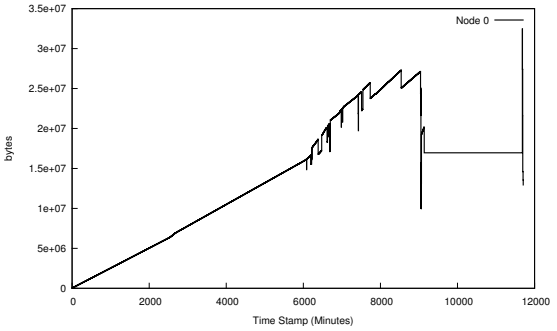



Figure 4.3: received data

The figures 4.5 and 4.2 show the number of bytes that were sent and received by all nodes in the network. The data is cumulated over time so that the series of measurements normally must be monotonous increasing and either the drops or the preceding peaks either must be measuring errors because of the previously given reasons or are also related to the way the monitoring system works.

As the data of all nodes in the network are combined, all sent data is received by another node and it is not surprising that the shapes of both diagrams are almost congruent.

A particularly conspicuous aspect are the spikes beginning at 7500 minutes. They might point out, that single nodes are not sending data anymore at this point.

After 6 days and 4 hours a major drop to $1e^7$ bytes occurs simultaneously to the drop of nodes in the node count diagram. After that the value becomes constant at a $2e^7$ bytes. This shows that some nodes are not providing monitoring data at all anymore and that no more monitoring information is transferred.

The next two diagrams show the actual data rate for all nodes. The sending rate which is presented in figure 4.4 in general is a bit higher than the receiving rate. The reason for this is that obviously no messages data can be received that was not sent previously. On the other hand not all data that was sent must be necessarily received by another node. It could have been lost on the way. Also in this diagrams the time axis can be subdivided in three parts. From 0 to 6000 minutes no specific incidents occur. From 6000 to 9000 minutes the data rates are higher because of the connected users that are publishing their content in the

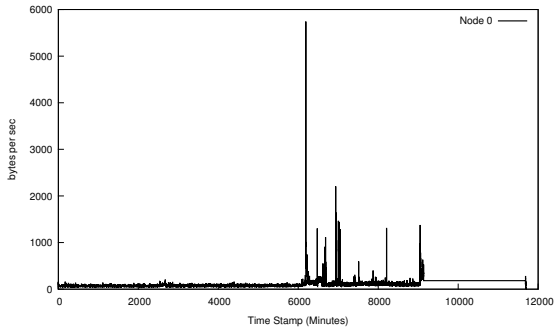



Figure 4.4: Sent data rate

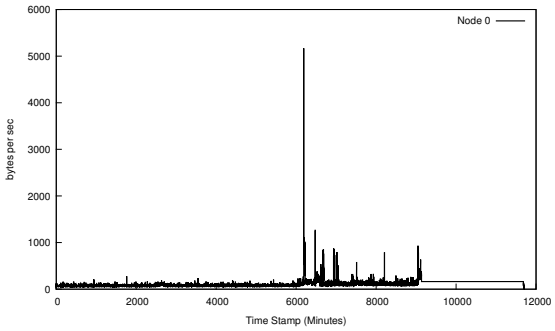



Figure 4.5: received data rate

network. After that the rate is constant as in the preceding diagrams this again indicates that there is something wrong with the measurement.

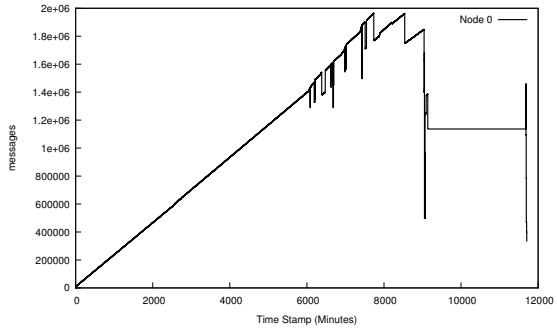



Figure 4.6: Messages sent amount

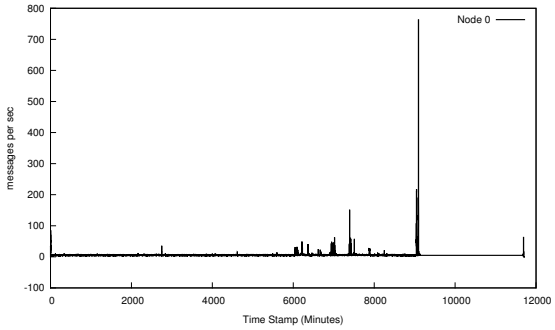



Figure 4.7: Message sent rate

The chart that represents the number of messages looks similar to the data diagrams 4.5 and 4.2. Figure 4.7 however shows a major deviation at the moment when the other diagrams have a drop. Here we can see that a large number of messages is generated which may even lead to the problems we observe in the other graphs. The reason could be congested message queues which also occurred during development and lead to message droppings there.

### Pastry Statistics

The statistics of the pastry module in figure 4.8 have a similar form as the message statistics. Again we see some oscillations beginning at 6000 minutes and a drop at about 9000 minutes. The transfer rate diagram (4.9) however shows that data was transferred the whole time with only a small increase at the time the users join the network.

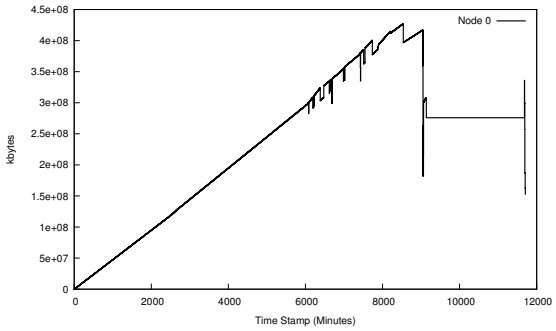



Figure 4.8: Pastry data read amount

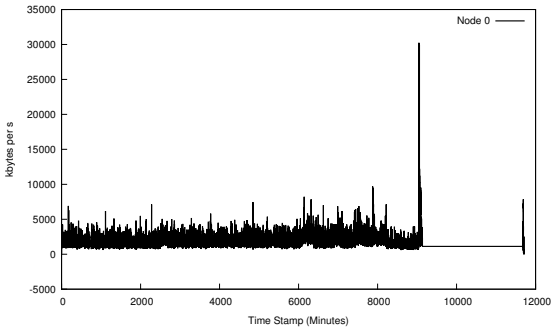



Figure 4.9: Pastry data read rate

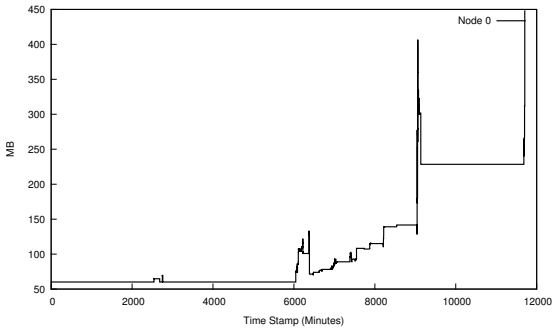



Figure 4.10: JVM Total Memory

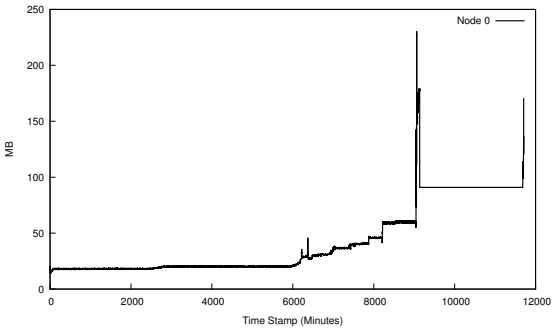



Figure 4.11: JVM Memory Usage

### Memory Statistics

The memory statistics in Figures 4.10 and 4.11 show the available and actually used memory by the Jva Virtual Machines for the nodes in the network. The value represents the average over all participating nodes.

In contrast to the results of [Weifeng/Dheeraj], s389 (pdf page), the values for the actually used memory are fairly constant at a level of 20 megabytes for the interval from the beginning until 6000 minutes. Also the graph in theirs thesis shows more volatility. We assume that this can be explained by the garbage collector activity that is necessary for the large amount of statistic data that was loaded and deleted over and over again.

Even in the 6000 to 9000 minutes time frame, the value is comparatively low and does not exceed the 60 megabytes mark. Because this value shows an average over the participants in the network, we can assume that the value is still more or less realistic as failed nodes are simply not taken into account and missing values do not change the computed values. At 9000 minutes however we see a spike in the curve and the value reaches 330 megabytes. As we are experiencing inconsistencies also for the other measurements at this point, we cannot be sure if the value is correct or inaccurate.

We assume that the reason for the more efficient use of memory is the improved monitoring method. During development we already saw that saving the measure-

ment statistics to a SQLite database instead of plain text was time and memory consuming. The problem with the old implementation is that it loads the whole statistics file to memory for each new data record and after a certain period of time the file gets too big.

### Plugin Statistics

The following statistics show the metrics that are related to the plugins. Most of the monitoring triggers here are bound to user actions so we can conclude from the values what the participants are doing.

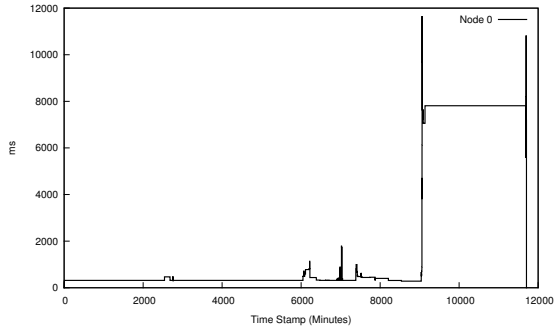



Figure 4.12: Time needed to join network

**Login Plugin** The statistics related to the login plugin primarily show the time that the nodes need to connect to the network as shown in 4.12. Because the participants need to retrieve the corresponding login token element from the network, the time depends on the routing and retrieval of the overlay layer of LibreSocial. Only in the section from 6000 to 9000 we can see that the time increases a bit, even though users report and we see ourselves that the login process takes much longer and even reaches the timeout limits.

**Wall Plugin** The statistics for the wall plugin basically include three different metrics, the number of wall views (Fig. 4.13) and the count of entries (Fig. 4.14) and comments (Fig. 4.15). We can see that the a the wall plugin was used first at about 2400 to 2500 minutes, even though at that time no wall entries were created. This is also the point in time where we can see a change in the other statistics so that we can assume that the first real user connected to the user at that time. From the 6000 minute mark on we can see a sharp increase of requests along with the first wall entries which shows us that the users interaction.

The first comments that belong to the wall entries were made at 7000 minutes. As this is the same point in time where we see irregular drops in the other two graphs, it could be useful to further analyse the wall plugin comments as a error source for problems.

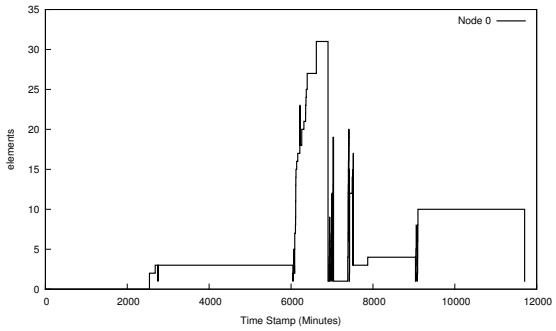



Figure 4.13: Number of wall views

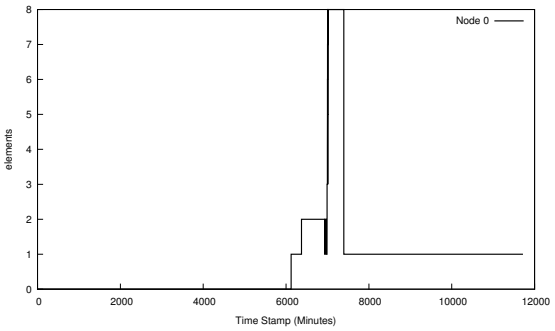



Figure 4.14: Number of Wall Entries

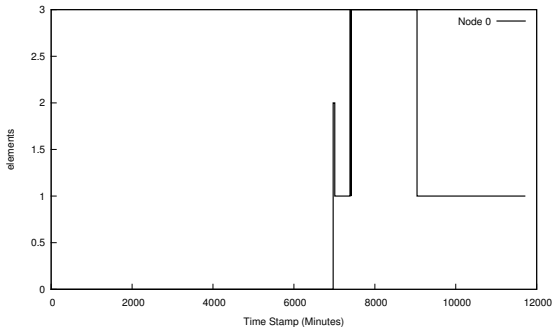



Figure 4.15: Number of Wall Comments

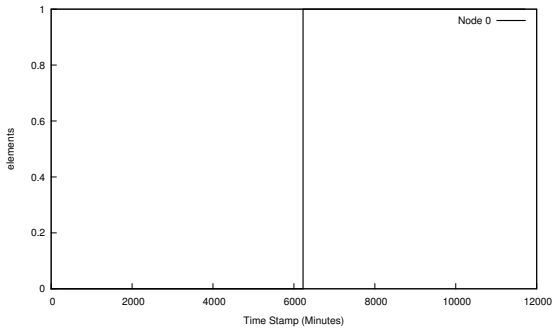



Figure 4.16: Number of Stored Files

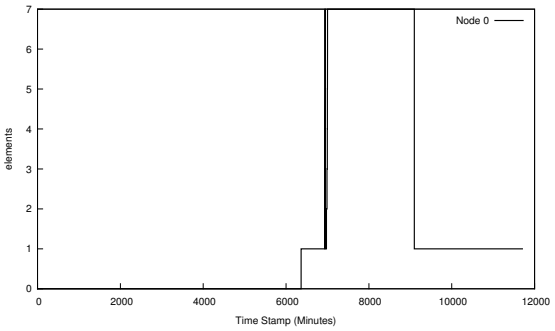



Figure 4.17: Number of File Downloads

**Files Plugin** The files plugin allows to store files in the network. As the files the users try to upload may be rather big, they are split up in chunks which are then distributed across to the network. The plugin is not only used for file uploads the users initiate directly but also as a storage mechanism for the file attachments of the wall, message, chat and group plugin. The diagram 4.16 shows us the number of files that were uploaded by the users. We can see that only one file was uploaded during the whole test after 6100 minutes which was then downloaded shortly after and accessed again multiple times again at about 7000 minutes (Fig. 4.17).

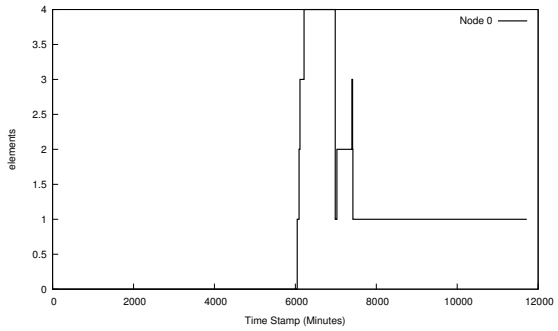



Figure 4.18: Number of friend requests

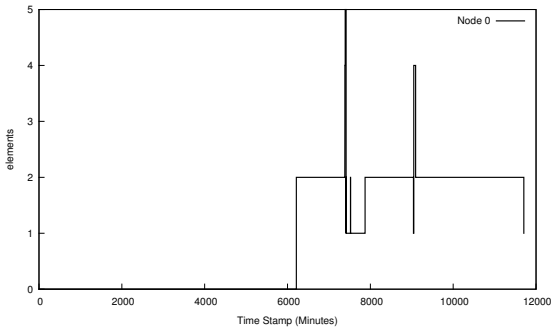



Figure 4.19: Number of accepted friend requests

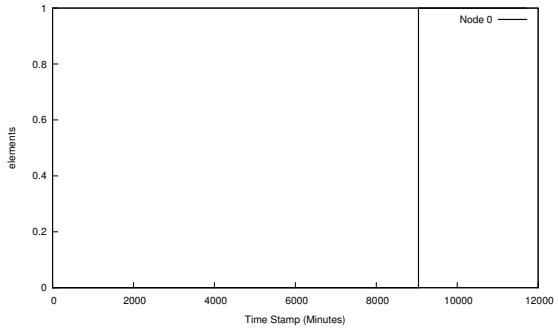



Figure 4.20: Count of user blockings

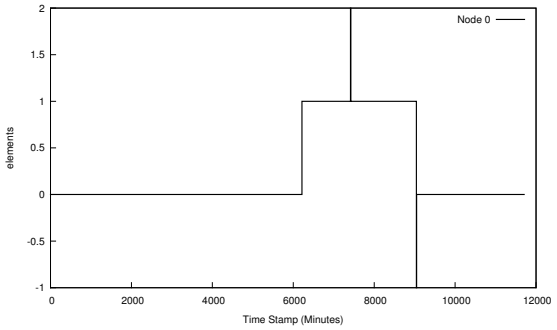



Figure 4.21: Maximum size of friend list over all logged in users

**Friends Plugin** At first glance it is clear that the metrics of the friends plugin show some contradictions. While the graph that shows the accepted friend requests (Figure 4.19) indicates a number of 5 accepted requests, according to Figure 4.18 only 4 requests were sent during the test. Here we can see again, that the monitoring somehow yields inaccurate results. The reason here can be the fact that the friend requests are not always answered immediately on the one hand and are typically accepted by users on different computers on the other hand. When nodes leave the network that previously contributed a non-zero value to the number of accepted friend requests, the quantity is no longer taken into account by the monitoring.

In contrast to that, the number of blocked friend request seems to be more realistic. We can see only one single action at about 7000 minutes in Figure 4.20 that stays at the same level afterwards.

The fourth metric of the friends plugin shows the maximum size of the friends lists (see Fig. 4.21). Here we can probably see the same effect as for the number of requests because we can see a maximum of 2 for the maximum number of friends in a single list for a short time. We assume that we have a one or more users that have single friend in their list, while there is one user who has two friends. As this

user logged out after a short time, the value decreases again due to a shortcoming in the monitoring which does not take old values into account.

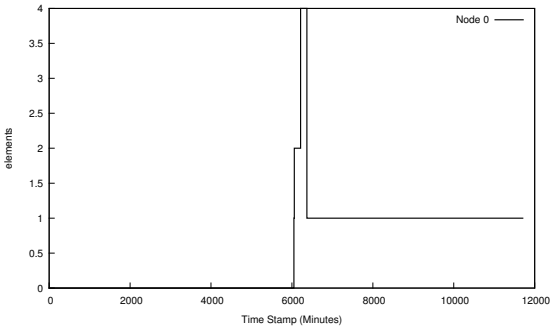



Figure 4.22: Number of created albums

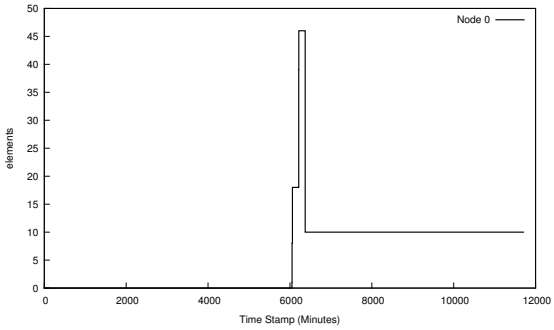



Figure 4.23: Number of uploaded photos

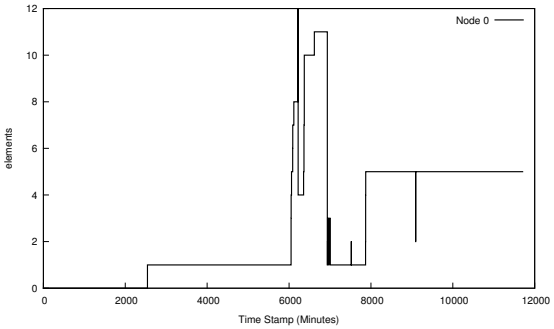



Figure 4.24: Number of views of the album list

**Photos Plugin** The Figures 4.22 to 4.24 give us an overview of the photo plugin usage. While a user accessed albums list once at about 2500 minutes, the first albums are being created after 6000 minutes. In total we can see that there were four albums with 45 photos stored in the network. These albums were viewed at by at least 12 times. We cannot see the actual number of accesses here because we can see a drop in the curve that was most probably caused by a leaving node on which some of the views took place. After 7000 minutes the value evens out for some time at the value of 1. This could indicates that only the node is taken into account here that was already logged in before the first album was uploaded.

**Text-Chat Plugin** Figure 4.25 shows the number of live chat messages that were sent through the network. We can see that a few messages were sent after 6000 minutes and again after about 7200 minutes. We see the effect of decreasing values after a short time which can probably be attributed to the monitoring again.

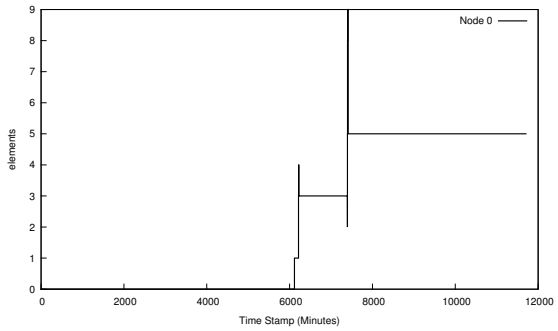



Figure 4.25: Number of sent live chat messages

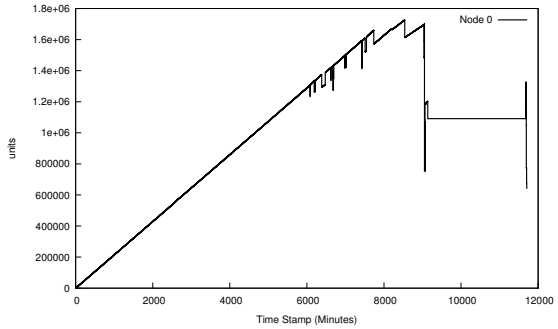



Figure 4.26: Messages sent by Messagedispatcher

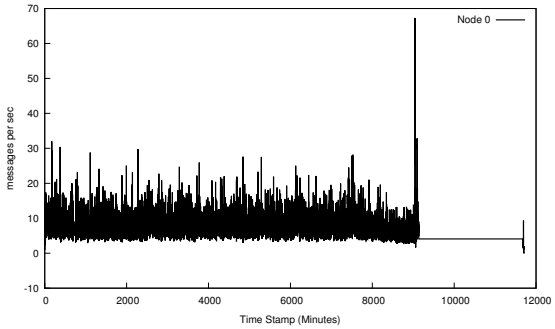



Figure 4.27: Messagedispatcher send rate

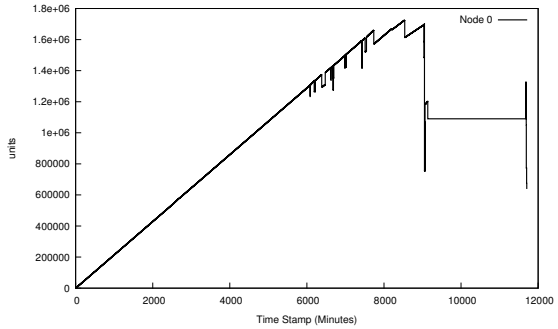



Figure 4.28: Messages received by Messagedispatcher

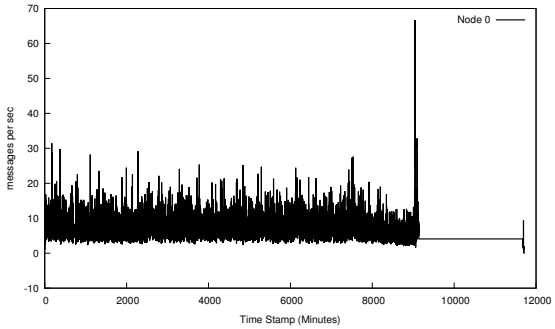



Figure 4.29: Messagedispatcher received rate

### Messagedispatcher

The Messagedispatcher is part of the framework and is responsible for sending messages between the nodes. The messages counted here have to be differentiated from the messages that are sent by the overlay layer of the application to coordinate the network. The first graph shows the number of messages which are

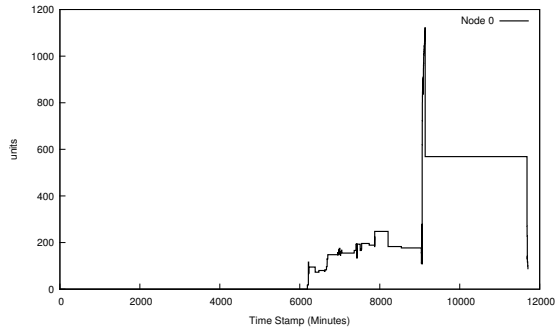



Figure 4.30: Message lost count

being sent over the network. It shows an constantly increasing straight line for the section from 0 to 6000 minutes. After that we can see the typical drops that we already saw in the previous diagrams. From 9000 minutes on we see a constant value. Interestingly we do not see the effect here, which [Weifeng/Dheeraj] observed during their tests. They found out that only about 9 to 12 percent of the messages sent through the overlay layer are actual payload messages. In our case we can see a lower overhead if we compare the messages that are sent by the network layer as shown in Figure **??** to the messages that are sent by the MessageDispatcher class (Figure 4.26). On the other hand our statistics here are less reliable here as we have a much higher network churn which reduces the validity of the monitoring tests.

We can also see that the the message rate they measured increased over time and was drastically higher [Weifeng/Dheeraj], Appendix, page 793 and 888 than in our measurements as presented in 4.27. This may be related to the changes of the serialization engine which drastically reduces the effort of storing objects to the hard drive and prevents an overflow of the data queues. This in turn reduces the number of messages that have to be sent multiple times because they are rejected.

Diagram 4.28 that shows the number messages that are received by the nodes does not differ from the sending statistics at first glance. This is also true for the receiving rate in 4.29 and sending rate. The statistics of lost messages in 4.30 however show that there is a slight difference between the curve development of the two diagrams.

### Storagedispatcher

Figure 4.31 shows the number of data items that are stored by the storage dispatcher. The storage dispatcher handles all items that are persisted by the framework or the associated plugins. In the beginning roughly 100 to 120 items are stored. This values does not change until the LibreSocial runs for 6000 minutes. After that the value is alternating from approximately 380 to 1000 Items. Again we see the effect of decreasing trends that are most probably caused by the in-

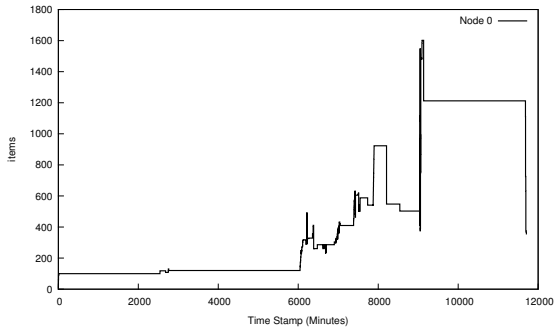



Figure 4.31: Global Storagedispatcher Stored Items Count

ability of the monitoring module to include historical values.

### Summary of Monitoring Information for the first user test

The monitoring statistics for the first test show us several things. First we can definitely see an impact on the memory usage of the application. As tools like JProfiler and VisualVM already indicated a significant decrease of resource consumption during the development phase, we assume with certainty that the changes in the monitoring and data storage engines have an effect here. Furthermore we observe that that the monitoring itself has the inability to use historical values for nodes that leave the network during the measurement. Because of this also cumulative metrics can decrease during the runtime when nodes that contribute to the value become absent at some point. In addition to that we also recognize a peak in the most of the graphs at a certain point in time following by an absolutely constant development of the value that indicates some inconsistency.

## 4.1.2 Reported Bugs

During the user tests the users also send a large number of bug reports via mail or by using the feedback tool. In this section we present some of these bugs and explain how they are eliminated if the problem is resolved already. The complete list of issues can be found in the appendix.

#### Security Issues (XSS, CSRF)

**Description** One of the test user reports that it is possible to exploit some weaknesses in the REST-interface of the back-end and in the browser GUI itself. The first shortcoming is a Cross Site Request Forgery attack. This kind of attacks reuse the users login context so that an arbitrary website is able to retrieve data or perform actions in LibreSocial. The application works in the way that the user identifies himself to the application and a token is generated and transferred to

the browser. This token is stored in a cookie and therefore sent to the REST-interface each time the user wants to retrieve some data or executes an action, so that the application can check whether the user logged in before. The problem is that also other websites that are opened in parallel can try to send these requests and retrieve a valid answer as the browser automatically send the cookie with the token to the back-end. This way a malicious website for instance is able to post new wall entries or to retrieve the username without the knowledge of the user. Especially the username is interesting here as it allows the de-anonymize the current user when the data is combined with some other source which can lead to serious consequences when LibreSocial is used in oppressed countries. Another problem are Cross-Site-Scripting attacks that inject JavaScript Code into user editable content. The code is then executed when one of these content strings in then displayed again. In general the application is protected against this kind of attack, but the tester reports that it is possible to add some malicious code to the username during the registration process which is then executed as soon as another user receives a friend request from the attacker as the notification contains the username. Also the two attacks can be combined so that an external website can inject code to the username field.

**Solution** To overcome the first type of attack, it is necessary that the front- and the back-end share an additional secret that other websites cannot read out. So a new random string is generated and stored at the back-end upon the login process and transferred to the front-end. At the front-end the token is stored to a local key-value storage that is accessible for the current website only. All subsequent requests to the back-end must contain the secret token, otherwise the connection is terminated. As the token is stored locally, it has to be included with every request manually, otherwise it would suffer the same problems as the cookie based token for this use case. While this technique is able to protect the application against these kind of attacks, the current implementation still has some problems with the association of the token for some requests. Therefore some calls fail which should be accepted. The problems here can be circumvented by transferring the token using a HTTP header, but the function is currently disabled. The Cross-Site-Scripting issues are resolved by filtering the content for code fragments and by explicitly denoting all content as plain text which is then not executed by the browser.

### Problems regarding the network

**Description** During the test our impressions are confirmed that the problems with the connectivity still exist. The users report a variety of issues that indicate that there is something wrong in the lower layers of the software. They are either completely unable to log in or get error messages during logout.

**Analysis** We do not have a solution for this category of problems but assume that they are related to the stability issues and Message-Lost-problems we still see when the application runs for some time. As far as our experience goes, the nodes in the network do not react in time when the problems occur and the login process takes longer and longer until the timeout limits are exceeded and the connecting client refuses to connect.

### Default network interface

**Description** Multiple testers report that a problem with the preselected network exists. On the login screen for each stored network a selector exists that allows to choose which local network interface on the computer should be used for the network. Normally the operating systems picks a suitable interface if none is specified but the FreePastry library expects a concrete interface configuration due to its nature. Especially Notebooks have a separate network adapter for wireless lan and also a wired interface. The first problem here is that users that initiated a connection to a network using their wired network card, and now use the other one, have to keep in mind that they have to change the selection here which is very untypical. The second problem is that most end users are completely unaware of how connections are initiated and have no idea what they have to select. The actual bug however was that even the automatic preselection algorithm in the old version yields a wrong interface and the users are unable to connect to the network.

**Solution** To resolve this issue, we now use a more advanced autoconfiguration mechanism. When a user tries to establish a connection to an existing network, a tcp connection is opened to this endpoint for a short time. During this test, the local interface that was selected by the operating system is determined and used as a default. If the user creates an own network instead, the application at tries to determine a good interface by using the libresocial.com web server as an endpoint. This way the interface is selected that has access to the internet which should be the right choice in most cases. Since, however some users still want to select the interface themselves, this strategy is only used, when "auto" is selected as an interface which is also the default. In all other cases, the selected interface is used.

### Graphical bugs and usability flaws in the user interface

**Description** Beside the technical problems with the back-end, the test participants also report a number of problems regarding the user interface. For instance some of the testers complain about missing descriptions on the action buttons and that the purpose of them is unclear. Other reports refer to problems with the modal windows in the UI which cannot be moved around correctly with some browsers. Also some of the windows have a minimization icon that does not change

it's look when the window is maximized or minimized. In addition to that the wall posts show an incorrect time when when a different time zone is used.

**Solution** To make it more clear, which function the buttons provide without changing the whole layout, a tooltip text is shown now, when a user moves the mouse over them. As the texts have to be localized here, also some new translations are added to the language file. During the analysis of the problems regarding window movement, we notice that in the test version not only the windows cannot be moved in some browsers, but also that in some cases the windows are displaced after they are moved. This problems are not fixed in the newer version of LibreSocial now, while the problem with the incorrect posting time still persists.

### 4.1.3 Evaluation Results / Outcome of the first test

Summarizing we can say that we have a enormous number of helpful bug reports and feedback from the PhD students which are now categorized in the bug tracking systems. We see ourselves that the storage layer of the application also causes problems when LibreSocial is used under real conditions.

## 4.2 Second test: Students groups

For the second test stage a new software version is used where many of the reported bugs are resolved. For this test not only the Phd students are invited to join, but also members of previous project groups. Additionally all limitation on the website are removed, so that anybody who visits the website, can join the test. To motivate the users we again ask them to publicly upload photo galleries of any kind until a certain date. We also tell them that we will create a voting subsequently and let the users decide which album is the best.

### 4.2.1 First Try

After the announcement email was sent, we see normal behaviour by the testers. After some time however, we see that that our test client get very slow. The CPU utilization is extraordinary high and also all network operations are very slow, no matter if they are related to LibreSocial or not. The log output of the clients offers a number of Exceptions regarding incorrect signatures of storage objects. We can see that they occur extremely frequently so that the computer is unable to handle them. A deeper analysis shows that the client refuses to store objects of other nodes but the data is sent again as a reaction of this which causes an infinite loop and leads to the increased load.

### 4.2.2 Second Try

To resolve these problems we decide to disable the signature checking of the data objects. This basically undermines large parts of the security concept but makes the software usable at least. The problem with this approach is that any object that was previously created and signed by a certain user, can now be overwritten by other users. Normally this is not possible as the signature of the existing object is compared to the new object and only accepted if they match. Because we are unable to recover the running instances from their malfunction, we decide to tell the testers to download the revised version and start over again.

### 4.2.3 Monitoring Results

This section deals with the monitoring results of the second user test where a larger group of participants is invited. We only show the monitoring results of the second run with the disabled signature checks, because no useful monitoring results exist for the first round.

#### Node Count

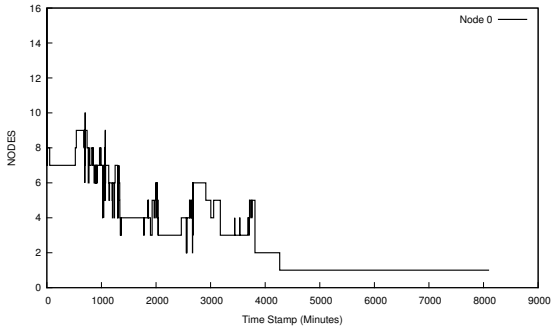



Figure 4.32: Node Count

Although we also use all ten virtual machines as bootstrap nodes like in the first test, we see a significantly different picture in the figure that represents the number of active nodes (see Fig. 4.32). Even at the beginning when all test nodes should be available, the value reaches at most nine nodes except for a short time after approximately 750 minutes. After that time the value is often varying but in general decreasing more and more over time. At about 4250 minutes the number of actives nodes reaches a value of one, which means that no other node is accessible anymore for the instance that is writing the statistics file. This means that not only all test participant left the network but also that the other bootstrap nodes we provide, are not available anymore. Because we do not actively shut them down, they must be failing.

### Data Transfer

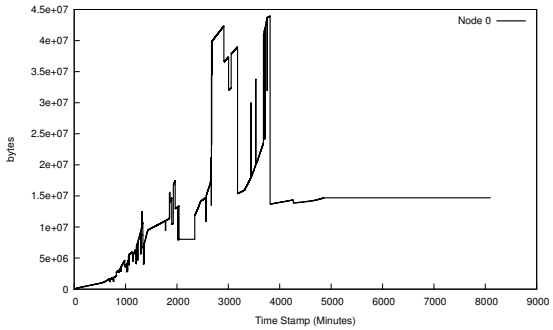



Figure 4.33: Network Sent Data

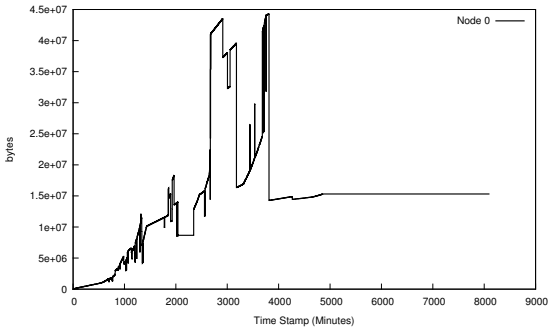



Figure 4.34: Network Received Data

The figures for the sent and received data (4.36 and 4.33) look very similar. Nevertheless on close inspection we can see that some amplitudes differ between the two diagram. For instance there are two peaks in the first figure in the area between 3200 and 3800 minutes that are higher than in the second one.

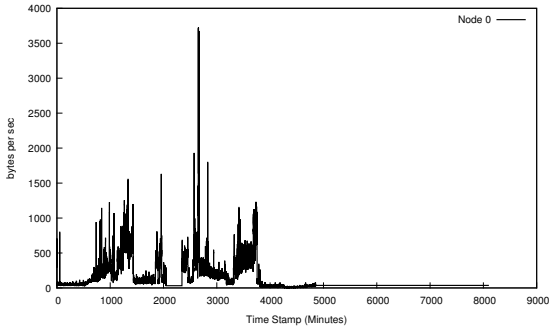



Figure 4.35: Network Sent Data Rate

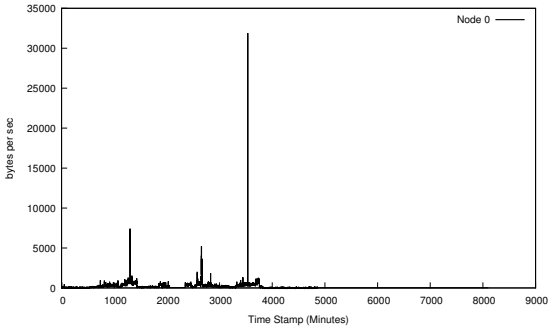



Figure 4.36: Network Received Data Rate

The curves for the data rates in 4.35 and 4.36 are not that congruent however. First of all it is worth noting that the diagrams are presented in a different scale. While the maximum of y-axis of the first diagram is 4000 bytes per second, the second diagram has peaks that exceed the 30000 bytes/s mark. The reason for this effect could be that there is some kind of message congestion at the receiver message queues. In that case some nodes must be sending data to the network over a longer time which is processed at once at the receiving instances.

The number of messages that are sent by the application (Fig. 4.37) reaches a high level after 800 minutes. After that the value is strongly alternating most likely due to the fact that the number of nodes is also decreasing at that point as shown in Figure 4.32. We can assume that only a few participants are generating a high number of messages here. After approximately 4500 minutes we see that the curve only has constant values.

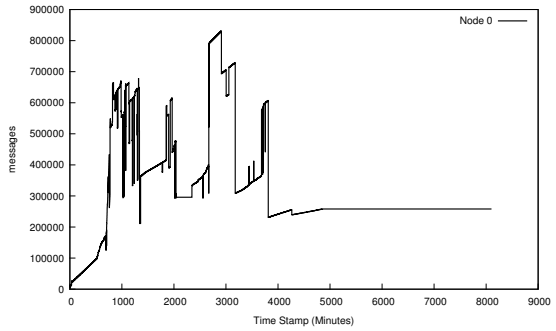



Figure 4.37: Global Network Messages Sent

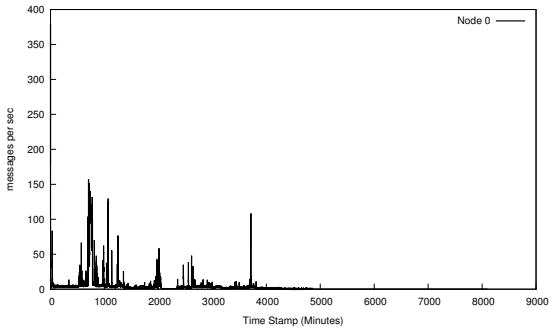



Figure 4.38: Global Network Messages Sent Rate

The diagram of the messages sent per second (4.38) shows similar activity like the figure representing the data rate in 4.35 at least in the first beginning. However the subsequent part of the graph show a lower development which indicate that larger messages are sent there.

### Memory Statistics

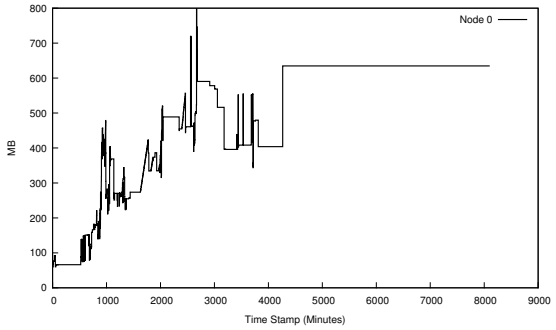



Figure 4.39: Global Network Memory Total Size

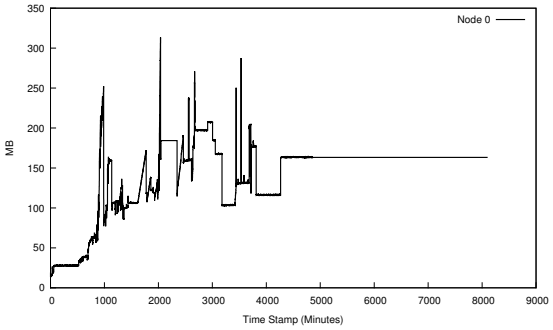



Figure 4.40: Global Network Memory Usage Size

The memory statistics show a different image than the observations in the first test. The second graph (Figure 4.40) is more interesting to us as it shows the actual consumption of the application in contrast to the memory that is being allocated by the Java Virtual Machine (Figure 4.39). We see that the usage is a whole lot higher but still does not really exceed the 300 megabyte mark and therefore it should still be possible to use low power computers to run LibreSocial. As in the other graphs we experience that the value levels out at approx. 160 MB after 4500 minutes.

### Plugin Statistics

The plugin statistics again mostly show the user actions performed during the test.

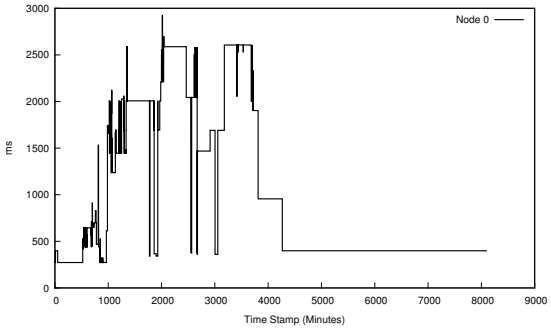



Figure 4.41: Global Network Node Join-network Time

**Login Plugin** The network join time presented in Figure 4.41 does not really show any anomalies. The join times range from 400 to almost 3000ms which would be an acceptable time for the users. During the tests however we again experience timeouts when we try to connect to the network at a later time. We assume that the connection attempts are not taken into account as this metric actually only measures the time that is needed to request and retrieve the login token object. When it is not possible to establish a connection, the login process is terminated before this operation is initiated.

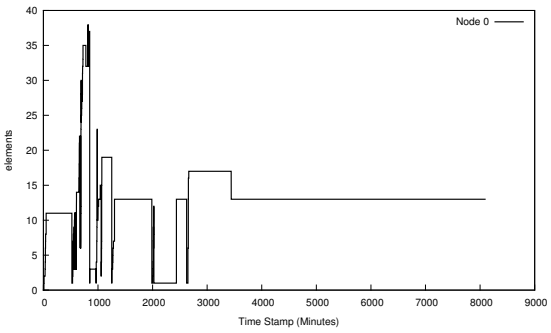



Figure 4.42: Global Plugin Wall Number of View Wall

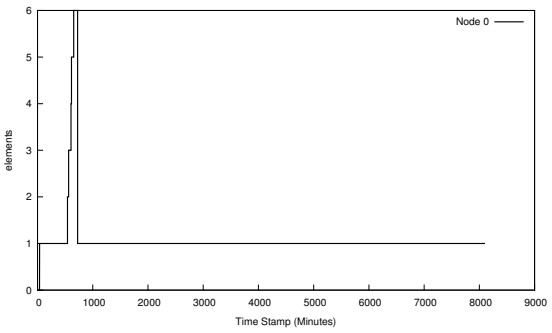



Figure 4.43: Global Plugin Wall Number of Wall Entries

**Wall Plugin** The retrieval of the wall data structure is shown in Figure 4.42. We see a number of views very early indicating that users join very early after the beginning of the test. We can see that at least one node accessed the wall

elements more than 35 times. As the value drops drastically then, we can assume that the corresponding user either logs out at this point or the instance fails here. The curve than again strongly alternates which most likely shows users that log in and out. The number of entries posted on the walls on the other side is rather low. We can only see a peak of six posts in Fig. 4.43. Also there are no comments to the wall entry so that the monitoring service does not even write the metric to the statistic output.

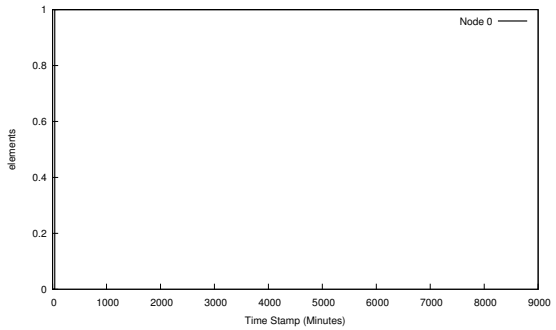



Figure 4.44: Global Plugin Files Number of File Downloads

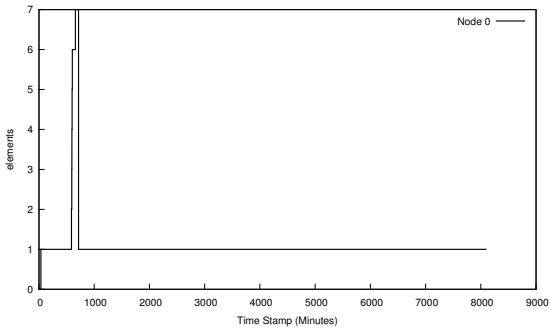



Figure 4.45: Global Plugin Files Number of Stored Files

**Files Plugin** The file plugin is only very little used as we can see in Figures 4.44 and 4.45. At the beginning of the test, a user uploads a single file and after 700 minutes some additional files are added. The graph here again has a major drop most probably because the node that is used for the upload and is therefore responsible for keeping the statistics leaves the network at this point. Also only one of the files is downloaded.

**Friends Plugin** The friends plugin presents a more interesting picture. We can see a number of friend requests in diagram 4.46. Over the whole time at which the network is actively used, requests are sent. As in the other graphs we see downturns due to the way how the monitoring works. If we only take into account the increases and assume that the drops occur due to the monitoring, we can see 13 friend requests.

Fro these 13 requests we see can see that seven are accepted if we use the same counting method. The most action takes place here at 500 to 900 minutes and again in the time span between 2500 and 3000 minutes as shown in 4.47.

In addition the diagram 4.48 shows the totalized size of the friend list for all users. Here we can see that the variations occur at the same positions as in the corresponding graphs.

**Photos Plugin** Figures 4.49 to 4.51 show the usage of the photo plugin for the second test run. As the users have the task to upload photos to the network,

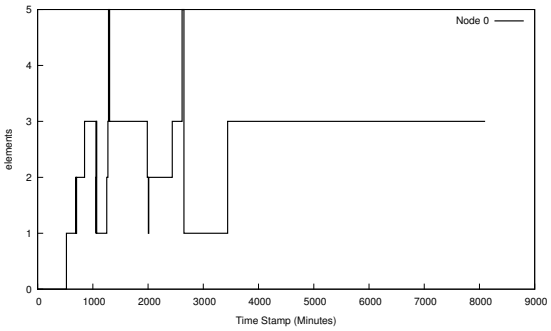



Figure 4.46: Number of friend requests

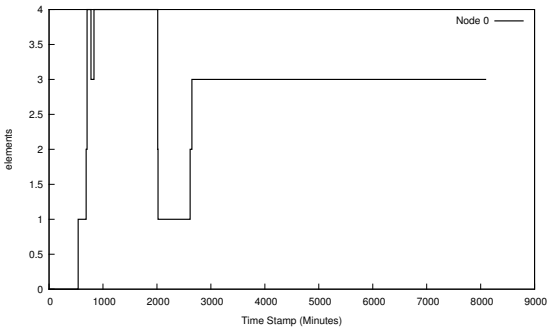



Figure 4.47: Accepted friends requests

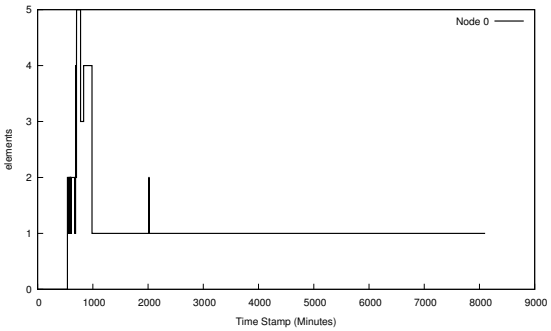



Figure 4.48: Size of friend list

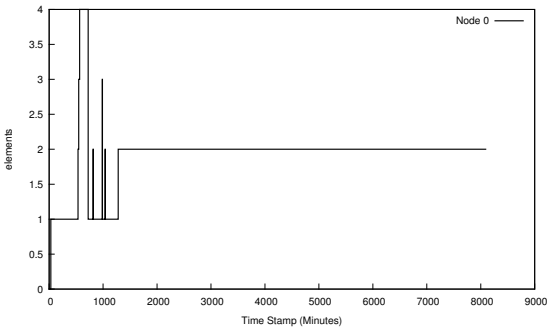



Figure 4.49: Number of created albums

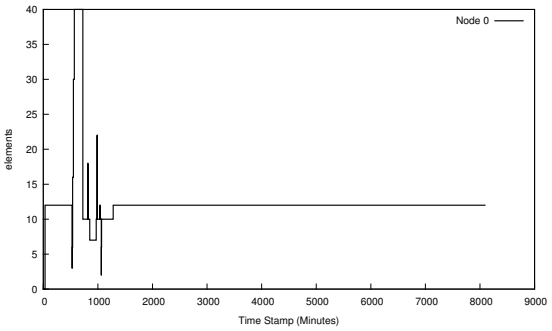



Figure 4.50: Number of uploaded photos

as expected we see some more activity here. Omitting the drops in the first two graph and assuming all increasings denote a creation of an album or upload of a photo, 9 photo galleries with 81 photos in total are created during the test. Also the viewing statistics show a regular use of the photo gallery.

**Text-Chat Plugin** In the second test the live chat plugin is more actively used as in the first test, as we can see when we compare Figures 4.52 and 4.25. Because of this we assume that there are no issues with the plugin.

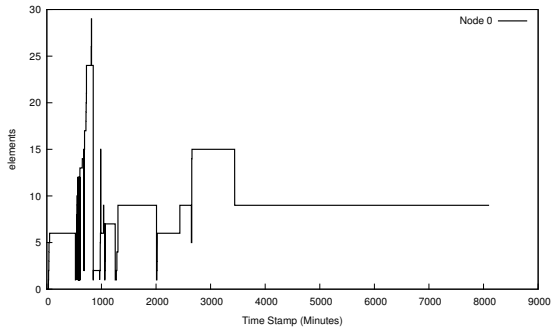



Figure 4.51: Number of album views

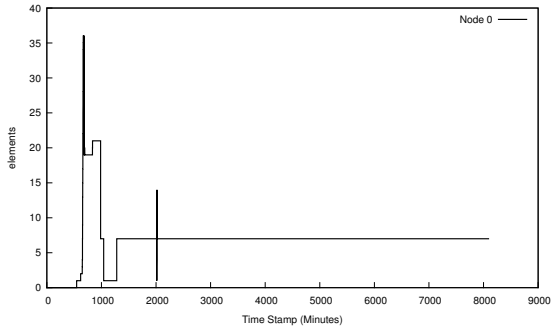



Figure 4.52: Number of live chat messages

### Messagedispatcher

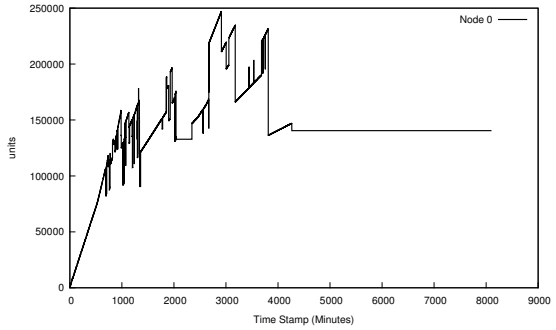



Figure 4.53: Messagedispatcher messages sent count

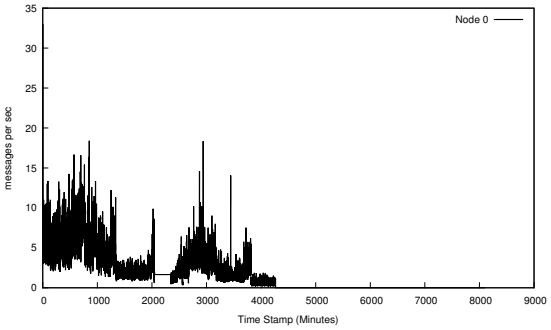



Figure 4.54: Messagedispatcher messages sent rate

The message dispatcher statistics in for the second test show an active usage. Interestingly we can see some anomaly in the diagrams from 2000 to 2300 minutes after the beginning. In the Figures showing the message rate (4.54 and 4.56) however, wee can see this most visibly as they normally oscillate. We can also

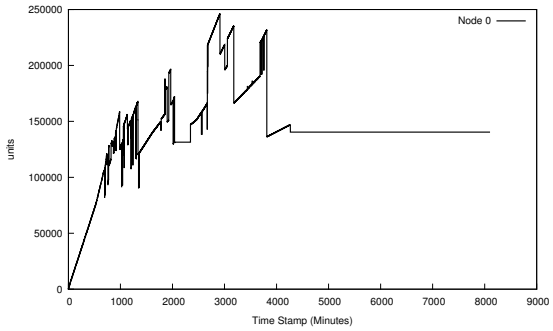



Figure 4.55: Messagedispatcher messages received

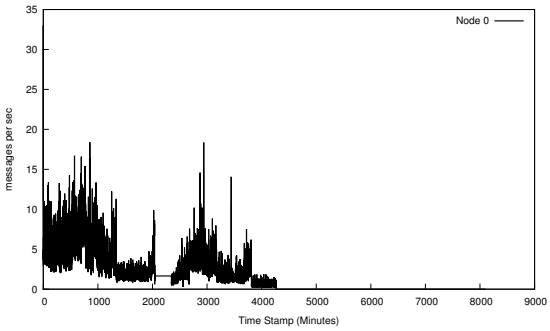



Figure 4.56: Messagedispatcher messages received rate

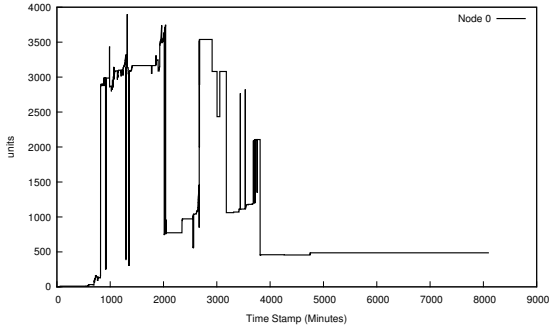



Figure 4.57: Messagedispatcher messages lost

observe that we have a much higher percentage of lost messages in contrast to the first user test.

### Storagedispatcher

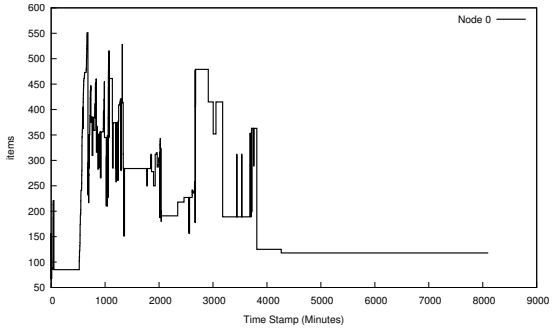



Figure 4.58: Storagedispatcher stored items count

The last diagram shows the number of data objects that are stored in the

network. We can see that much more items are stored than in the first test (Digram 4.58). If we take into account that the leaving nodes spuriously reduce the values because of the monitoring, that normally would have to be included, we see that the number is at least a magnitude higher here. But all in all we can see active usage over nearly the whole time until the 4000 minutes mark where the measurement stops.

#### Summary of Monitoring Information for the second user test

In general we can see that the second user shows some deviations to the first one. This firstly is due to the long inactivity phase in the beginning of the first test, but we also see much more activity from the users.

### 4.2.4 Survey Results

By carrying out a survey after the actual user tests, we want to find out what the test participants think about our program and what happened when they were using it. This way we also get information from more users and not only problem-related feedback. Due to the stability problems, the user tests were not extended to a larger audience. We only have 9 testers who answered the questionnaire out of 20 to 25 persons that downloaded and potentially tried the software. So at least the statistical answers are not of particular significance, however the individual opinions and the bug reports may be helpful for the further development.

The full questionnaire is available in the appendix.

#### Connecting to a network

Table 4.1: Connecting to a network - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I was not able to connect to a network (technical reasons) | 4 |
| It was not clear how i can join or create a network at first | 1 |
| It was not clear how i can join or create a network at all | 0 |
| I had to open and forward a port in the firewall | 0 |
| I created my own network | 3 |
| Other statements: | |
| It worked fine (2x) | |
| I was able to connect to a network | |

The first question aimed at the connection problems that we noticed during the tests. By analysing the answers we can see that our impressions are confirmed here. Several users were unable to connect to the network but created their own networks at least so that they could see the program. Also in our experience this always worked and the program could be used.

### Wall feature

Table 4.2: Wall feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 2 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 0 |
| I had problems to see the wall entries directly after registration | 0 |
| I had problems to see the wall entries after some time | 1 |
| The GUI elements were somehow deformed | 0 |
| Other statements: | |
| It worked fine (4x) | |

For the wall plugin we see that it worked for the majority that tested it although we also see an answer where one user states that problems occurred when the application was used for some time. Probably this happened due to the problems we experience with the data storage layer, which resulted in data object that get lost or delayed in transmission when the application runs for a longer time.

### Search feature

According to the answers the search functionality was not used by many of the users. This may be due to the fact that some of the users were unable to connect to the publicly available test network and tested the application with a single user. The rest of the users state that the search worked on the one hand but also had problems after some time the results were not showing up at all or in reasonable time. As the search plugin also depends on storage objects, we assume that this is also related to the known problems.

### Videochat feature

The video chat plugin however should work without any data that has to be stored in the data layer. Here only notifications messages are that tell the recipient that another user wants to have a conversation with him or her. The actual video

Table 4.3: Search feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 4 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 0 |
| I had problems to see the search results directly after registration | 0 |
| I had problems to see the search results after some time | 2 |
| The GUI elements were somehow deformed | 0 |
| Other statements: | |
| It worked fine | |

Table 4.4: Videochat Feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 5 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 0 |
| I was not able to see my own preview video | 1 |
| I was not able to see my chat partners video | 1 |
| The GUI elements were somehow deformed | 1 |
| The application / browser crashed | 1 |
| Other statements: | |
| It worked fine (3x) | |

data is transferred directly between the browsers as soon as the connection is established. Looking at the results, we can see that mainly problems with the front-end part of the plugin are mentioned whereas it also worked for many of the users who tried it.

### Text chat feature

For normal text chat or multi chat we received only one negative answer was given reporting that there must be some bug in the feature. With regard of the program code and the description we assume that this is also related to the front-end part where normally some filters are used to display the relevant messages only. Probably there is something wrong with it.

Table 4.5: Text chat feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 1 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 0 |
| The GUI elements were somehow deformed | 0 |
| Other statements: | |
| It worked fine (3x) | |
| After I closed the chat window and reopen it again, the chats were not in order. All the chats from me together and all chats from my friend were together | |

### Photo gallery feature

Table 4.6: Photo gallery feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 0 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 0 |
| I was not able to see my own photo albums | 0 |
| I was not able to upload photo galleries | 0 |
| The GUI elements were somehow deformed | 0 |
| The application / browser crashed | 0 |
| Other statements: | |
| It worked fine (5x) | |

According to the table, all test participants tried the photo gallery and it worked without any problems. The first point is not very surprising as the users were explicitly asked to use the plugin. However we expected that more problems occur when the photos are displayed after some time as we had the problem when we connected to the network.

### Messaging feature

In conjunction with the messaging feature no technical problems were reported. Two users explicitly say that it worked fine for them. Only a single user said that the usage was unclear. Unfortunately we don't have any further information about this.

Table 4.7: Messaging feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 2 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 1 |
| The GUI elements were somehow deformed | 0 |
| Other statements: | |
| It worked (2x) | |

### File upload feature

Table 4.8: File upload feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 3 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 0 |
| I was unable to upload files | 0 |
| I was unable to download files | 0 |
| Other statements: | |
| It worked fine (3x) | |

Also for the files plugin we only see positive answers from the users that tried it. We even do not see any reports about problems during the download. This may be due to the fact that the users tried to download their files only once or at least not after the network problems we experienced.

### Group / Forum feature

For the groups and forum plugin we see that many users didn't even try the feature. However we do not have any negative feedback here and it seems that no problems occurred.

### Voting feature

According to the result tables the results for the voting feature are identical with the answers given for the group plugin.

Table 4.9: Group / Forum feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 5 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 0 |
| I was unable to upload files | 0 |
| I was unable to download files | 0 |
| I was unable to join a group | 0 |
| I was unable to create a group | 0 |
| Other statements: | |
| It worked fine (2x) | |

Table 4.10: Voting feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 5 |
| It didn't work at all | 0 |
| It was not clear to me how it works | 0 |
| I was unable to create a new voting | 0 |
| I was unable to participate in a voting | 0 |
| Other statements: | |
| It worked fine (2x) | |

### Calendar feature

The survey results for the calendar yield similar results but two users are complaining here that they had problems when they tried to use it. For one of them it was somehow unclear how to use it and the second one apparently had some technical issues. Unfortunately both did not explain precisely what was the problem here.

### What do you like most in LibreSocial?

This questions aims on the things that LibreSocial is already good at and what the testers think about it. It gives an interesting insight of the benefits they see. In this question we can definitely see that the users are comparing the application to commercial services and emphasize the advantages of the concept like independence, security and that it is free from advertising. Two participants also state that the features are useful, which indicates that we have put the focus on the

Table 4.11: Calendar feature - answers of the users

| **Statement** | **Number of Mentions** |
|---|---|
| I did not try this feature | 5 |
| It didn't work at all | 1 |
| It was not clear to me how it works | 1 |
| Other statements: | |
| It worked fine (2x) | |

Table 4.12: What do you like most in LibreSocial? - answers of the users

| |
|---|
| "Simplicity and the distributed security within friend boundaries." |
| "It provides some very good functionalities like video chat, group adn voting which made social networking more interesting." |
| "No advertisement" |
| "The philosophy." |
| "independence from proprietary services" |
| "Its security mechanisms and its decentrality" |
| "LibreSocial is very fancy"<br>"it offers nice features that other social networks do not feature, such as the video chat or file upload" |
| "Clear Gui layout and easy to use once connected" |

right things.

### What would you use LibreSocial for?

Here the most of the users say that they want to use LibreSocial for their personal communication with their friends and families. Some of them explicitly refer to facebook which is typically also used for this reasons. It seems that the users are somehow convinced of the concept and would actually use it if more users would use the software. Also two of them mention business usage in but cover different aspects with their answers. On the one hand the application could be optimized for functionality that can be used to form relations between companies or individual persons so that it can be used as some kind of job and contact portal like the centralized counterparts LinkedIn[1] or Xing[2]. On the other hand it could be extended for an internal company as it securely stores the data and does not give up the control to others.

[1]https://linkedin.com, accessed: 12.05.2015
[2]https://www.xing.com/, accessed: 12.05.2015

Table 4.13: What would you use LibreSocial for? - answers of the users

| |
|---|
| "Personal networking." |
| "Business networking if features are enhanced and companies are able to present their business." |
| "I would like to use it for charting with friends, join group and share files with other users, creating a poll and invite friends to vote and so on. All facebook-like functionalities would be attractive to me to use it." |
| "Communicating with friends." |
| "Instead of Facebook!" |
| "I would use it in a data-sensitive environment (e.g. in a company)" |
| "For everything i use facebook now. The main problem is that no one uses libresocial instead" |
| "Working environment: Send/share working documents wir working partners." |
| "Secure communication with family and friends." |
| "Internal groups for info exchange not open to the public" |

**What features do you miss most in LibreSocial?**

Table 4.14: What features do you miss most in LibreSocial? - answers of the users

| |
|---|
| "A current network status information. I never knew if the problem was on my site or the network's." |
| "Adding friends or users to the calendar event." |
| "Adressbock and android app" |
| "maybe a browser displaying available open networks wich are free-to-join" |
| "Schared calendar" "File Synchronisation across friend's PCs" |
| "A list of public networks that I can join" |

With this question we explicitly wanted to know what ideas the users have for social networks and which existing functionality is the most important to concentrate on in the future. Multiple reports refer to essential networking and monitoring functions. In two answers some kind of network browser is wished that shows all publicly available networks a user can connect to. This would be relatively easy to achieve. The only drawback is, that some central instance has to be used to coordinate this kind of directory service, which however would not be critical for operating the networks. In another requests a users asks for more detailed about the network he or she wants to connect to, so that problems can be identified easier, especially in case login failures. Also the already existing calendar is mentioned, because it currently lacks the actual networking functionality. This

is a point we are aware of and that we already have on our agenda. Interestingly some user also asked for a file synchronisation tool. This is actually already in the making as a topic of a parallel master thesis and will be integrated.

**Which features are unneccessary?**

Table 4.15: Which features are unneccessary? - answers of the users

| |
|---|
| "I think the Files plugin is not necessary because we could directly send files as attachment in message plugin. User can select files from there disks. There is no need to upload it first to network then from there send them to other users. The files occupy storage space of framework also." |
| "I do not think at this moment there are any." |

The answer about the files plugin given by one of the users cannot really be taken into consideration because the files plugin is actually not only used for the file upload view but also for the virtual drive of the group plugin and even for the messaging attachments the respondent refers to. So also the storage consumption would not be reduced.

**What do you think about the GUI? What can be improved?**

Table 4.16: How do you like the GUI of LibreSocial?

| **Value (1: Very good, 8: Very bad)** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **Number of Mentions** | 1 | 2 | 3 | 1 | 1 | 0 | 0 | 0 |

In table 4.16 we can see that most of the feedback for the graphical user interface is rather positive and users are satisfied with it. If we combine the rating to an average score, we get an 2.875 out of 8.

Table 4.17: What do you think about the GUI? - answers of the users

| |
|---|
| "The GUI is great!" |
| "It looks like facebook" |
| "Some buttons and labels should be having tool-tip." |
| "Draft and spam could be added as new functionality to message plugin. Then the message plugin would be good enough to replace the email function."<br>"The current applications can meet the basic requirement from users. Some amusing functionalities like Tag Friends on Photo or Location Detection may also attract more users." |

In addition to the numerical rating we also asked the survey participants for their personal opinion. Basically they are proposing only smaller extensions to existing parts of the application but in general they do not miss core functionality. One of them also states that the LibreSocial GUI looks like facebook, leaving open if this is positive or negative. At least we can say that the arrangement and the layout are somehow similar to other social networks because these concepts are proven and we know that the approach of a distributed setup and multiple networks already associates some additional complexity for the usage. The other ideas like photo tagging and the extended messaging functionality however are worth thinking about.

#### On which kind of devices do you use social networks?

Table 4.18: Relation between mobile and stationary usage

| **Value (1: Desktop only, 8: Mobile only)** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **Number of Mentions** | 5 | 0 | 0 | 0 | 2 | 1 | 1 | 0 |

With this question we wanted to find out, how and where other social networks are used, be it on mobile devices or on stationary computers. As LibreSocial currently is limited to a certain degree, and mobile devices can only be used if the back-end application runs on a separate computer. The results for this question reveal that most of the responders use social networks at their computers only, but we also see some exceptions, so that we have to keep in mind this kind of users.

#### Are you interested in a Raspberry Pi Version of LibreSocial?

Table 4.19: Raspberry PI Version - answers of the users

| **Answer** | **Number of Mentions** |
|---|---|
| Yes | 3 |
| No | 5 |
| Undecided | 1 |

This question is associated with the previous one. As the graphical user interface is independent from the actual application, a browser on a mobile phone can be used to display the GUI while a computer at home is used as the counterpart. Thus LibreSocial at least offers a compromise for mobile usage. The new version is much more memory and cpu efficient so that it also runs an low-end computers like the raspberry pi which in turn allows to run LibreSocial permanently without

consuming too much energy. As a second benefit the continued availability improves the stability of the network. The results in 4.19 demonstrates that there is an interest for this setup.

#### Conclusion of Survey

Although there were not so many users that participated in the survey, we can recognize some tendencies. We can conclude that there is interest for our program in principle and most of the testers can think of scenarios where it can be very useful. With regard to the problems we experience with the software, we can see that also the users at least partly noticed some problems. It becomes clear that the plugins that mainly store data have more problems then the plugins that normally send messages to other nodes and do not need to persist information. In addition to this, the survey participants also share some problem-related feedback, feature requests and ideas.

### 4.2.5 Outcome

After the test we can say that the application works quite good to a certain degree. We can see that the users are able to use the software in general and smaller problems regarding usability are already resolved. However we definitely have to acknowledge that we also have some larger problems with the stability of the software. While experienced users may be able to overcome them, it would be very hard for other that are unfamiliar with technical things to connect to the network when this step fails over and over again. This is also the reason for us to refrain from further tests with completely uninvolved people although this would most likely yield some interesting details and reveal some rough edges in the usability concept.

# 5 Summary and Future Work

In this chapter we recapitulate the work of the whole thesis and see what aspects are left open for the future.

## 5.1 Achievements

During the making of this thesis we have seen that it is a challenging task to transform an application towards a product that can be utilized by end users. In the first half of this elaboration we described the changes that were made to the software to make it more robust and to improve the usability. The most important issues that were resolved are the performance regressions in the monitoring service and the data storage. In the first case the inefficient storage mechanism that relied on plain text files was replaced by an SQLite storage. The persistence layer of FreePastry which uses XML serialization on the other hand was changed to use the much more efficient kryo library. With these two improvements the application needs significantly less resources and even can be used on much slower devices. In addition to the performance optimizations we also removed many usability flaws. On the one hand the number of unnecessary input controls were reduced that would confuse users. On the other hand the features of the application were extended to encourage the users to interact. For instance the media attachments for the wall posts now allow to use visual media instead of simple text.

In the second half of the thesis we see the entire process of the test. We use three different aspects to analyse the behaviour of the program and the testers. The monitoring as a permanently recording of the internal performance indicators, the bug reports and feedback of the users during the test and the survey as a third pillar which gives us an insight on the users thoughts and opinions. While the monitoring did not always provide derivable results due to the high churn in the network, the bug reports from the PhD students have a high quality. The survey also states that the testers are interested in the concept and like to keep the control over their data. On the other hand it turned out during the test phases that there are still some major problems in the code base. Although some of them could be identified and ironed out, still some critical bugs exist, that kept us away from extending our test to a larger group of users. At least we found out that they are somehow related to the storage engine. When the issued are resolved and the applications runs stable enough, new user tests can be performed. For a possible second iteration of this topical field also large parts of the infrastructure can be reused. So in addition to the actual software tools like the automated start-up

and update script were improved and a new website was created in the same style as the application that can even be used once LibreSocial leaves the test stages.

## 5.2 Future Work

If we try to take a more critical look at the project, we can still identify a number of issues that should be addressed.

**Lost data** The most obvious problems that have to be resolved are the instabilities and problems regarding lost data that the application show when it runs for several days. Otherwise it does not really make sense to carry out more user test as the participants may get frustrated when such essential functionality is not working. However, it may not easy to overcome these problems as it is difficult to find the root cause. In our opinion it would be the best strategy to setup a proper test environment to identify the anomalies. For this the existing unit tests and the test plugin should be used. It also turned out, that a virtualization software can be very helpful as it is capable of making and reverting snapshots of the the virtual systems on which the application run on. This helps to get to the point in time where the problems occur without waiting several days. If this is still not expedient, one could consider exchanging the overlay layer of the application with a different implementation. This however would involve many changes and should only be seen as a last resort.

**Firewall users** The performance of the application still suffers from users that cannot be accessed from the public internet because the nodes are running behind an address translating network gateway somehow. Although it the relaying mechanism helps to circumvent the problem, it also introduces the problem that single nodes may become responsible for larger parts of the networks. Here a more advanced NAT traversal technique should be used.

**Version for low power computers** Although less than half of the participants of the survey voted for it, it may be an interesting project to use low power computers to run LibreSocial. As the back-end application does not run on mobile devices, this would enable a way to run the software itself at home without consuming too much energy, while the graphical user interface can be used on mobile devices.

# Appendix A

# My Appendix

## A.1 List of reported Bugs

Table A.1: List of reported Bugs

| |
|---|
| Automatically preselected network interface is incorrect |
| Animation not allways shown during search |
| README File |
| Send search results to browser before all results are found |
| Make it possible to download software packages from the command line |
| German umlauts are not displayed properly in city selector |
| Title- and mouse hover texts for buttons |
| Quotationmarks in user names make the gui crash |
| CSRF-Waekness Wall |
| Deanonymization of user name |
| XSS-Weakness if code is injected into user name and friend request is sent |
| Time of Wall posts |
| Notifications at Login |
| LibreSocial after System Hibernation/Suspend |
| Login fails right after logout |
| Repeatedly sending friend requests |
| loading animation for different profiles |
| Chat: minimize / maximize icon |
| Memory is almostly exhausted-Warning |
| Warning: No Bootstraps reachable |
| Netzwork name is duplicated for each click on Discover |
| Responsive Design / telescoping of elemente |
| rice.p2p.past.PastException occurs |
| Search does not find all results |
| Popup windows cannot be moved |
| Video chat: black window only |
| Global privacy selection |
| More detailed Tutorial |
| Feedback for Logout process |

Table A.2: List of reported Bugs (cont.)

| |
|---|
| Telefony feature does not work correctly... |
| Firefox Videochat: alsa-devices only |
| NodeIsFaultyException when Logout fails |
| Serverinstallation / LibreSocial: Cannot open display |
| DoS possible if actions are repeated |
| Use selector element for country in search formular (like at the login page) |
| Editor for Background/Profile images |
| Backgroundimage on Friendlist |
| Friendslist (Mainview) in multiple colums |
| cached profile image is not refreshed on wall after upload |
| Uploads start in parallel |
| Modal window movement |
| Disable text selection |
| Abort image preload for albums when modal window was closed |
| Upload the whole album even when the preload did not finish |
| Voting Results can not be displayed if opened a few times |
| Content Security Policy |
| Gallery Title and Subtitle slide over Submenu-Bar |
| File download with incomplete names |
| Different default name for Chat Groups |
| Group Messages not sent |
| Remove text from group chat input field when text is sent |
| Resizable Movable Windows |
| External address seems to be identified/presented wrong. |
| Changing the monitoring selection does not have any effect |
| Cannot leave Group Chat |
| Tooltips for Group Chat Buttons |
| OS X 10.10.3 Libresocial still not working |

## A.2 Survey Sheet

## A.3 Monitoring Data Measurement 1 : Phd Students

Due to the size the statistics are only part of the digital PDF version of this thesis.

## A.4 Monitoring Data Measurement 2 : Open Test

Due to the size the statistics are only part of the digital PDF version of this thesis.

# Bibliography