

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Fakultät für Elektrotechnik, Informatik und Mathematik

Masterthesis

Testing Of A Peer-To-Peer Framework-based Social Network

Siddharth Kaneriya
Matriculation number : 6633319
E-Mail: kaneriya@uni-paderborn.de

Paderborn, 22nd November 2015

Supervised by
Jun.-Prof.Dr.-Ing. Kalman Graffi
and
Prof. Dr. rer. nat. Christian Scheideler

Declaration

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.

Paderborn, 22nd November 2015

Siddharth Kaneriya

Contents

1. Intoduction **1**

1.1. Motivation . . . 2
1.2. Solution strategy . . . 2
1.3. Structure . . . 3

2. LibreSocial: P2P Framework Based Social Network **5**

2.1. Background . . . 5
2.1.1. Peer-to-Peer Systems . . . 5
2.1.2. OSGi Service Platform . . . 6
2.2. LibreSocial Architecture . . . 7
2.3. P2P overlay . . . 9
2.3.1. PAST . . . 10
2.4. P2P Framework . . . 13
2.4.1. Access Control . . . 13
2.4.2. Data Structure . . . 13
2.4.3. Storage and Replication . . . 14
2.4.4. Testing . . . 15
2.4.5. Monitoring . . . 16
2.5. Plugins And Apps . . . 18
2.5.1. Photo Plugin . . . 19
2.5.2. Notification plugin . . . 20
2.5.3. Messaging plugin . . . 20
2.5.4. Wall plugin . . . 20
2.5.5. Chat plugin . . . 21
2.5.6. Group Plugin . . . 21
2.5.7. Files Plugin . . . 21
2.5.8. Friends Plugin . . . 22
2.5.9. Calendar Plugin . . . 22
2.5.10 Search Plugin . . . 22
2.5.11 Profile Plugin . . . 22
2.5.12 Vote Plugin . . . 23
2.5.13 Audio/Video Chat Plugin . . . 23

2.5.14 Apps Store . . . 23
2.5.15 Login/Logout Plugin . . . 23
2.5.16 WebProvider Plugin . . . 24
2.6. Graphical User Interface . . . 25
2.7. Improvements and bug fixing . . . 25
2.8. Chapter Summary . . . 26

3. Testing **27**
3.1. Test Plugin . . . 27
3.1.1. Prerequisites . . . 28
3.2. Test Cases . . . 32
3.2.1. Test Case Structure . . . 32
3.2.2. Available Test Cases . . . 32
3.3. LibreSocial Configuration Parameters . . . 39
3.4. Chapter summary . . . 41

4. GUI based test **42**
4.1. Test Environment . . . 42
4.2. Observations . . . 43
4.3. Outcome . . . 45

5. Long Term Test **46**
5.1. Test Environment . . . 46
5.2. Test Deployment . . . 47
5.2.1. Setting Up Development Environment . . . 47
5.2.2. Test configuration parameters . . . 47
5.2.3. Secure connections . . . 47
5.2.4. GUI access to Virtual Machines . . . 48
5.2.5. Test Execution . . . 49
5.3. Test Script . . . 50
5.4. Monitoring . . . 53
5.5. Test Evaluation . . . 54
5.5.1. First Round Results Evaluation . . . 54
5.5.2. Outcome Of First Round . . . 61
5.5.3. Second Round Results Evaluation . . . 61
5.5.4. Outcome Of Second Round . . . 65

6. Conclusion **66**
6.1. Achievement . . . 66
6.2. Future Woke and Suggestions . . . 67
6.2.1. Length And Size Of Message queue . . . 68
6.2.2. Node handling . . . 68
6.2.3. Error handling . . . 69

List of Figures **71**

List of Tables **73**

Bibliography **74**

1. Intoduction

Social networking is becoming increasingly important in all the aspects of life. Usage of social networks is not just limited to chatting with friends and making new friends but it has relatively wide range of functionality to offer. Such functionality includes sharing your day by day activities among the known people, sharing photographs, videos, blogs, discussion on forums etc... Because of high potential of social network to offer numerous functionality, their use is not just limited to normal users. Social networks are equally famous in private industries and other organizations as they offer high level of collaboration and information sharing among all the users and responsible parties. As a common phenomenon, majority of the social networks are based on Client-Server architecture and to run such social networks, companies need to centralize all the services. Centralization of social network required huge servers and storage space to process and store all the services of users. As a result of this, companies need to spend huge amount of financial and human resources just to maintain such social networks. Social networks have also given a rise to modern critical issues like a threat to privacy and identity of the users as users tend to share their information on social networks which are being stored in the centralized servers owned by particular social network company.

Peer to Peer (P2P) technology seems to be the answer in order to overcome all the problems described above caused by social networks as it enables peers to communicate and share data with each other without any centralized server. Moreover, it is really cost effective, fault tolerant and very much secure compare to server client architecture as data and information is not centralized. Basically, all the peers in P2P network act dually as server and client. Due to this nature all the nodes belong to P2P network are capable of sharing processing loads and storage, which improves the utilization of resources to in P2P networks. Examples of such P2P networks are Kademlia, BitTorrent and Gnutella which are mostly used for file sharing purpose. P2P techniques can also be useful in oppressed countries where social networks have to face many restrictions and strict laws as it is hard to detect and shut down such networks. P2P networks

facilitate end to end encryption and privacy options which enables secure and decentralized communication. So even though your data is stored at some other nodes in the P2P network, it will be encrypted and protected against unknown and unauthorized access. These ideas are main building blocks for the development of P2P framework based social network called "LibreSocial".

LibreSocial is relatively new social networking application compare to other P2P based applications. Considering the functionality provided by LibreSocial, makes it really unique in the group of all P2P framework based applications. LibreSocial provides all the functionality which are provided by ideal social networks, for example, chatting, video calling, information sharing, friends graph, access control etc... It was also possible to include various security technologies like encryption, access control to LibreSocial due to it’s diverse nature. More detailed information about LibreSocial can be found in the chapter below.

1.1. Motivation

Earlier testing of LibreSocial has shown that the product works fine in the starting period of activity but eventually it seems to be getting unstable and unbelievably slow due to various exceptions and lost messages between peers. As a result of it, for the initial time LibreSocial becomes really slow due to the slow retrieval of data(files,photos,groups etc...) and sometimes it also results into the crashing down of whole network. It is really essential to find out the root cause of this problem in order to make this product more reliable and usable among the users. Product needs to be go through various testing passes and feedback surveys so that more and more problems can be found and solved.

1.2. Solution strategy

Due to the various masters thesis being conducted on LibreSocial, product was constantly under the improvement phase. Such phases include an improvement of monitoring and improved usability of LibreSocial. Despite of improved testing and monitoring, there are still critical bugs exists in the P2P overlays causing slow behavior and multiple errors in the LibreSocial during run time. Main goal of this thesis is to identify bugs exist in main framework and overlay layer and make products more stable and usable in the

real environment by the real users. All the goals can be summarized as below,

- Systematic analysis and testing of the communication layers in FreeP-astry.
- Fix existing bugs and improve error handling in order to reduce the amount of different exceptions caused by unexpected activity with the LibreSocial.
- Identification of protocol design and implementation issues that lead to a lack of stability.
- Setup of a large-scale test environment in order to produce critical errors if exists.
- Conduction of several tests and evaluation of the stable and proper functionality of the P2P Overlay and Framework.

1.3. Structure

Chapter 1 (introduction) contains the introduction about the thesis topic and motivation behind the making of thesis with solution strategy in precise.

Chapter 2 (LibreSocial:P2P Framework Based Social Network) provides brief overview of P2P systems and technology and OSGi service plateform which were used to develop LibreSocial application. It also explains the underlying architecture of LibreSocial in details with brief introduction of plugin development.

Chapter 2 also consists all the changes and improvements which were made by the author as a part of the solution strategies.

Chapter 3 (Testing) gives detailed information about the test plugin and it’s usage in order to conduct a successful test on LibreSocial.

Chapter 4 provides the information about the GUI based test and all the observations made by the author in this area.

Chapter 5 (Long term test) is a core part of the making of this thesis and explains the test setup in order to conduct long term testing of LibreSocial. It also provides an detailed evaluation of all the results.

Last chapter summarizes all the work which was done as a part of this thesis and draws useful conclusions followed by future work and author suggestions in order to improve the reliability and stability of LibreSocial.

2. LibreSocial: P2P Framework Based Social Network

2.1. Background

This section describes all the underlying technology used, features and architecture of "LibreSocial" in details. Current chapter also presents the detailed view of current status of software and basic definitions used in order to put reader into the right context.

2.1.1. Peer-to-Peer Systems

"*The term Peer-to-Peer (P2P) refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized manner*" [1]. Peer-to-Peer (P2P) technology offers direct communication between all the participants/nodes, hence enables decentralization of data and information. Unlike traditional centralized system where central system(server) has to bear majority of the cost with huge resources, P2P offers effective cost sharing and reduction as well as equal utilization of resources due to it's decentralized nature. P2P technology has wide range of advantages to offer like Autonomy as P2P system require that local node perform all the necessary operation on the behalf of user, improved scalability and reliability, privacy, cost reduction etc...

On the other hand, the trend of social networking is growing tremendously day by day. There are some margins exist in the current social networking systems as they use client-server approach for data sharing. The major issues are expanding with network traffic, server maintenance cost, data privacy, single point of failure etc. These limitations can be avoided using P2P approach.

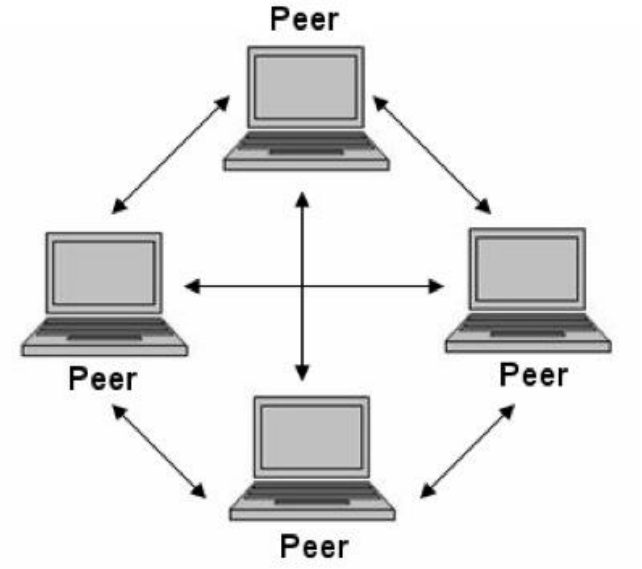


Figure 2.1.: Peer-to-Peer Network

Despite of having potential amount of positive effects, P2P systems have introduced many challenges over the time. One of the main challenge is acceptance and use as peers are dependent on one another in order to provide service. It is important that majority of the peer be available for the service to be useful. P2P systems also requires huge bandwidth in order to perform various operations and load balancing is very important aspect of P2P systems.

2.1.2. OSGi Service Platform

Since LibreSocial plugins are written using OSGi service platform, it is necessary to understand how OSGi service plateform helps a java application to achieve the modularity which allows applications to be developed in small parts. This section provide basic overview of OSGi service platform and how it has been used to develop "LibreSocial" plugins.

Java suffers with lack of modularity resulting into different drawbacks like error-prone class path concepts, limited deployment and management support and low level code visibility control [2]. For more information please refer [2]. OSGi service platform enhance the usefulness of java platform by providing modularity support which enables lose coupling between components and services of huge systems. Modularity provides separation of

concerns by providing separation of massive structure into minor element where each part represents unique logic.

OSGi service platform is made up of two parts: the OSGi framework and OSGi standard services. The OSGi framework is responsible for the implementation of basic functionality and it is an application's execution environment. Modularity is provided OSGi framework through three conceptual layers called Module layer, Lifecycle layer and Service layer. Similar to layered architecture, each layer is dependent on other layers. Below are the characteristic details of each layer of the structural design.

Module Layer

The module layer contains bundle which contains meta data. Bundles are nothing but a JAR file with extra metadata which includes class files, resource files (.xml„jpg etc...) and manifest file.

Lifecycle Layer

Lifecycle layer is responsible for manipulation of bundles. It provides services to the application internally by providing support for the interaction of bundle with execution environment and OSGi framework end externally by providing management support to the bundles.

Service Layer

The service layer works on the principle of service oriented architecture.

2.2. LibreSocial Architecture

LibreSocial is layered architecture based eclipse Rich Client Platform application [3]. LibreSocial uses the phenomenon of Peer-to-Peer Framework as it is developed on the top of FreePastry [4]. FreePastry is an open-source implementation of Pastry developed by Rice university in the cooperation with other partners. Pastry is a substrate for P2P application and provides numerous capabilities secure routing in the network, load balancing, caching, mapping etc... in order to support P2P application which could be built on the top of it. LibreSocial application is using OSGi platform, so it can be organized into smaller bundles where OSGi framework provides support for the modularization of application into those smaller bundles [2]. Due to

the fact that whole LibreSocial application is based on java, it is platform independent and is available for different operation systems. Currently it is available for Windows Os, Mac OS X, and Linux operating system [5].

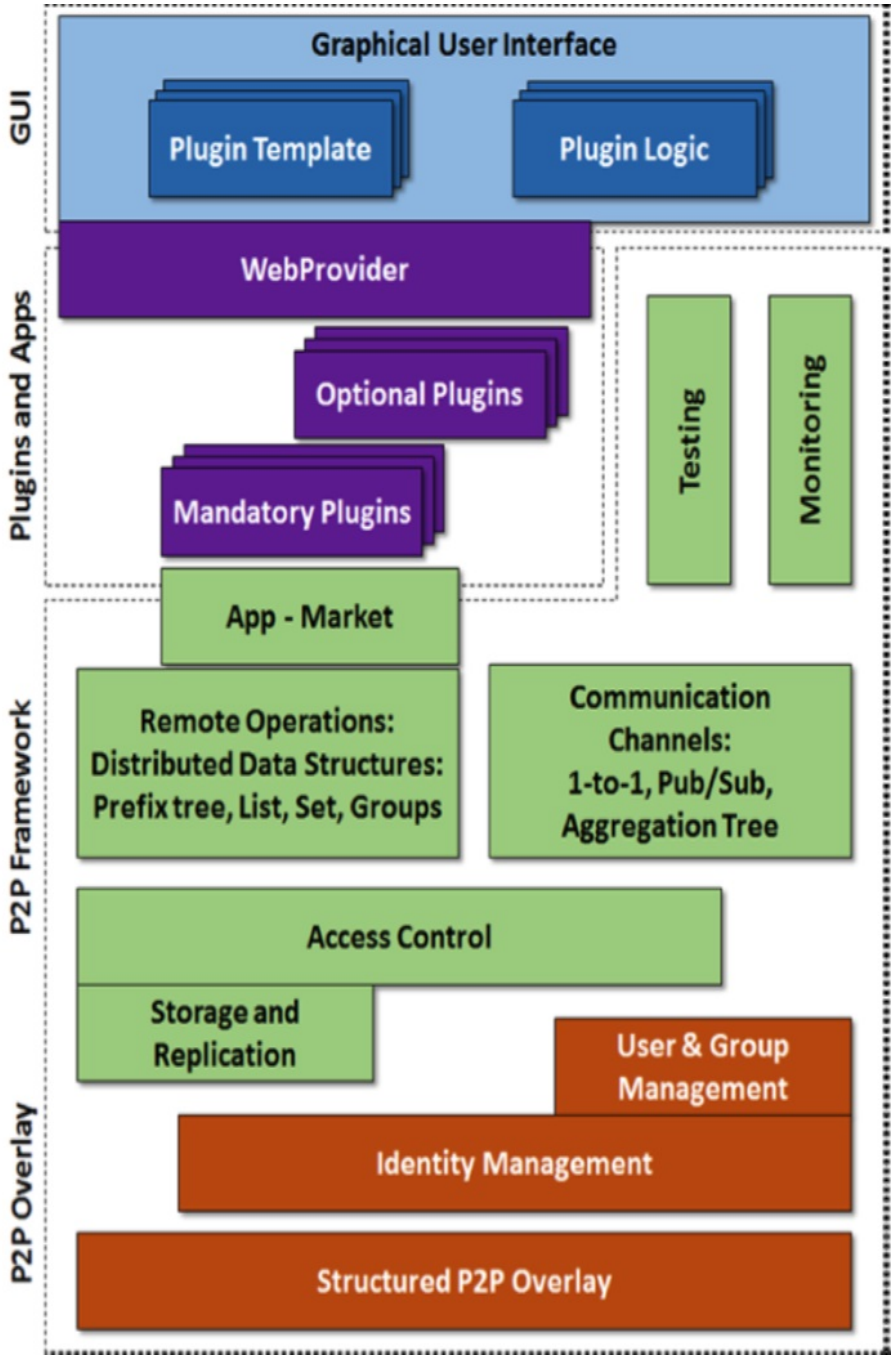


Figure 2.2.: LibreSocial:Architecture [6]

It can be seen from the figure 2.2 that LibreSocial contains three different

layers named overlay,framework, Plugins and apps receptively from bottom to the top. Each layer communicates with each other using well defined interfaces. More detailed information about layers and their applications will be discussed in this chapter and will be followed by the modifications made as a part of this thesis.

2.3. P2P overlay

Structured P2P overlay networks provide very important substrate for building distributed, scalable large-scale P2P applications but there exists some challenges like security, secure routing, which need to be addressed in order to achieve reliability of such networks. Structured P2P overlay is placed at the bottom of LibreSocial architecture. It is responsible for enabling secure routing among the peers through the formed network as shown in figure 2.3. Pastry is based on such structured P2P overlay. Pastry [4] assigns 128-bit unique identifier to each node belongs to that Pastry network. Whenever, any pastry node is presented a message with 128-bit key, it will route the message to the node which is numerically closet to the presented key among all the node in the Pastry network. This way it take maximum of O(logN) in order to transmit the message to responsible node with given N number of live nodes in Pastry network. At each Pastry node a Routing table will be maintained of maximum size of O(logN).

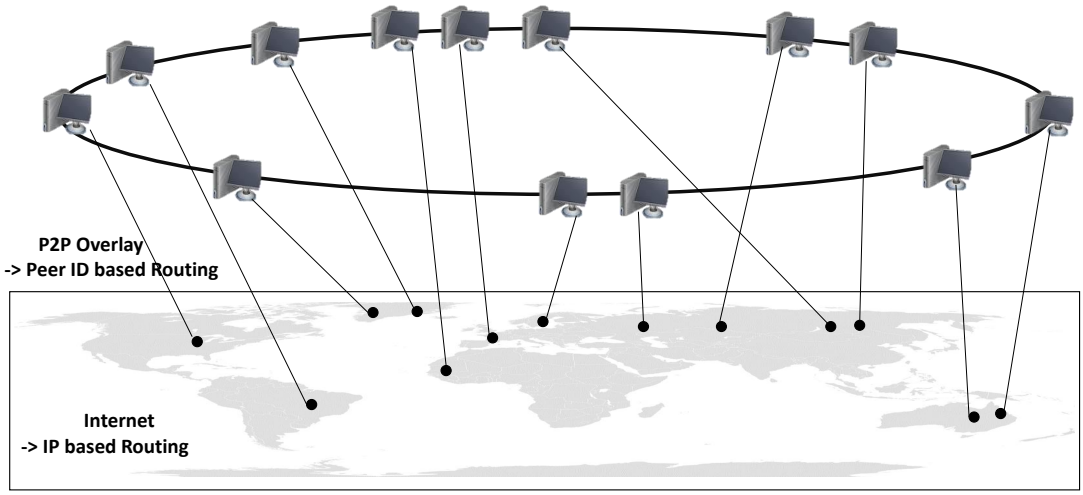


Figure 2.3.: LibreSocial:Overlay Network [6]

Also, each node in Pastry network keeps the track of immediate neighbors, this way they can be aware of failure or new arrival of nodes which makes Pastry network more reliable and self-organizing. Pastry also provide mechanisms for accessing Objects though routing of Pastry messages, so the application built upon Pastry can own such mechanisms. PAST is an ideal example of such application built on Pastry. PAST provide various mechanisms like Storage management, Replication management inherited from Pastry.

2.3.1. PAST

PAST is an internet based global storage utility built on Pastry with an aim to provide mechanisms like storage, replication and security to LibreSocial. PAST [7] is a normal Peer-to-Peer application comes inbuilt with the Pastry which can be used as a persistence storage for other P2P applications built on top of FreePastry. PAST basically forms the network of PAST nodes which are able to store, retrieve and replicate objects into the PAST network. Objects stored in PAST node can be accessed only by using it's unique ObjectID. Each PAST node in the network act at least as an access point for user or can be a part of storage. PAST provides number of operations in order to access Objects distributed among other PAST nodes.

- **Insert(name,Userkey,k,file) :** Stores file with k copies across the network. This operation produces 160-bit unique fileId. Any ambiguity in fileID will be detected and it will results into the rejection of later inserted file with a fact that this happens very rare.

- **Lookup(fileId) :** Lookup operation is used to retrieve the copy of the file from the network of PAST nodes with a condition that user must know the unique Id of file which is going to be retrieved.

- **Reclaim(fileID,UserKey) :** This operation is used to retrieve the storage of file with the matching fileId with no guarantee that lookup operation will produce the file.

Storage Management In PAST

PAST distributes objects in uniform manner across the PAST nodes in the network overlay. However, in the real world the file sizes as well as the capacity of storage PAST nodes is not uniform in a way that overlay network comprises of strong and weak node with different capacities. In such cases PAST requires some explicit storage management in order to achieve high degree of global storage utilization. There are two ways that PAST can achieve the expected level of storage management: File diversion and Replica diversion. File diversion is performed in order to compensate the difference in the storage capacity and utilization of nodes within leaf set. While, replica diversion is performed in order to achieve the gloab load balancing among the large potion of nodeId space.

Caching In PAST

On the other hand, PAST also provides certain mechanism for the faster retrieval of the files by minimizing the fetch distance. For this purpose, PAST stores K replicas of file into K nodes with adjacent nodeIds, So whenever the client lookup request comes, Pastry will likely rout the message to the replica closet to the client. In the LibreSocial, Caching does the same thing by storing most recent retrieved data items into the cach. The mechanism improves the response time between network and application by allowing speedy retrieval of popular data items. In order to control caching, limited storage space will be dedicated for caching. Whenever a node stores a new copy or replica of an object/file, it certainly delete the older cached file in order to make more room for the newly cached copies or replicas. caching

performance can be really poor at the highest level of increased storage utilization.

Modifications in Overlay

During the project group 2012, some modifications were done to FreePastry in order to provide support of heterogeneity and security. While during project group 2013, Overlay was changed provide/add more functionalists like providing support for NAT (Network Address Translation), providing support for distinguishing different pastry nodes and to make the user aware of the bootstrap nodes.

In project group 2013, the FreePastry has been modified and a new Transport layer has been added to it. This Ring Id Layer contains two methods, discover() and checkMessageHeader(). The method discover() can be used by the user to ask another computers that in which ring they currently are. This helps a user to discover networks. The method, checkMessageHeader() is used to check if the incoming message is really intended for that particular ring. It gives an advantage that no two rings can be joined accidentally as each message also contains a Ring Id number so it is very easy to verify that which message is for which ring. This diminishes the possibility of sending message intended for one ring to another. One another major change in the Free Pastry is new Ring Manager which is responsible for storing all the rings that has been discovered and further it is also responsible for caching the addresses of all the online nodes. At the enf of the project group, we were able to provide functionality listed below.

- **Create a local ring:** The user has the freedom to create his/ her own private local network and share its details to ask others to join the same, hence creating their own private social networking.

- **Discover a ring:** The user can discover a ring based on its IP address and can user that as the initial bootstrap.This ring has to be saved after successful discovery providing a user desired name. After that, the user will be a node of that ring.

- **Select/ change a ring:** the saved rings that the user has discovered or/ and created can be used anytime as long as the user does not delete that manually. Hence the user know exactly to which ring he

is connected and moreover, has the complete command on the ring selection.

2.4. P2P Framework

P2P Framework layer is very important layer as it mediates between application and overlay. It is located on the top of overlay. P2P framework layer provides numerous mechanisms for data structures that allow to store data in distributed linked lists, Sets, or in a Prefix Hash Tree, access control, testing and monitoring. Some additional functionality for streaming and publishing/subscribing is also provided by framework. This section briefly describes all the functionality provided by P2P framework.

2.4.1. Access Control

In the distributed networks, the manipulation of stored data items is always possible as data items are stored in a distributed manner. Access control enables only authorized users to read/write data items present in the network. In LibreSocial, Combination of Public-private key pairs with symmetric keys is used to encrypt and sign the data objects to guarantee privacy and integrity through out the application.
To access the data item for reading, the symmetric key is decrypted and is then used to decrypt the encrypted data item. The symmetric key can be decrypted by using the user's private key or the groupâ€™s private key. In a hierarchical group, the symmetric key is retrieved by a depth first traversal through the group hierarchy [6].
In order to overwrite the data item, the public-private key combination of the user will be used. Public key of the user/group is always stored together with the data item. Whenever the write request comes for the particulate data item, Algorithm will check whether the requested data item is signed with corresponding private key belong to the specified public key. This often verified by the node where corresponding data item is stored [6].

2.4.2. Data Structure

From the above discussion of PAST, we have an idea that PAST supports different methods to access only single object at a time.

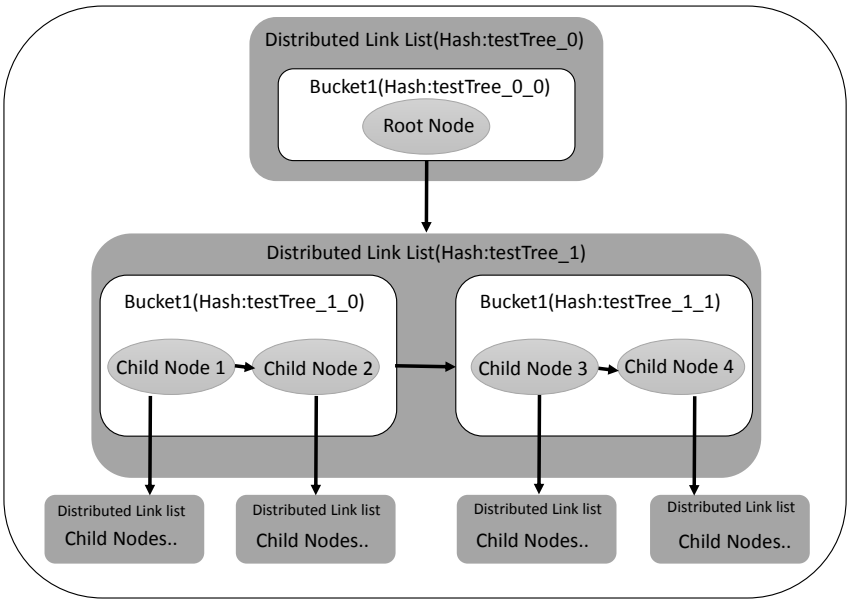


Figure 2.4.: Distributed Linked List [6]

Sometimes, in a real world it is mandatory to store data in the form of collection or group in order to show certain relation between those data items. For this purpose, we can use traditional distributed linked list where we use a storage item for each individual data item with a reference to next item.

In the case of really big application where numbers of thousands of data items are being stored, this approach sounds relatively slow and results into very high response time. To avoid this problem we use buckets instead of single storage item. Bucket contains multiple data items to choose from, with a reference to next bucket. In this way we can improve storage process. Each buckets refers to the distributed linked list where its child nodes are stored [6]. The tree structure can be seen in Figure 2.4 Framework also supports the other data structures than linked lists like trees or sets, Prefix Hash Tree etc...

2.4.3. Storage and Replication

LibreSocial is using main characteristics of P2P systems. Replication is one of the most important characteristic of P2P systems and very important for the efficient retrieval of files/objects stored in P2P networks.

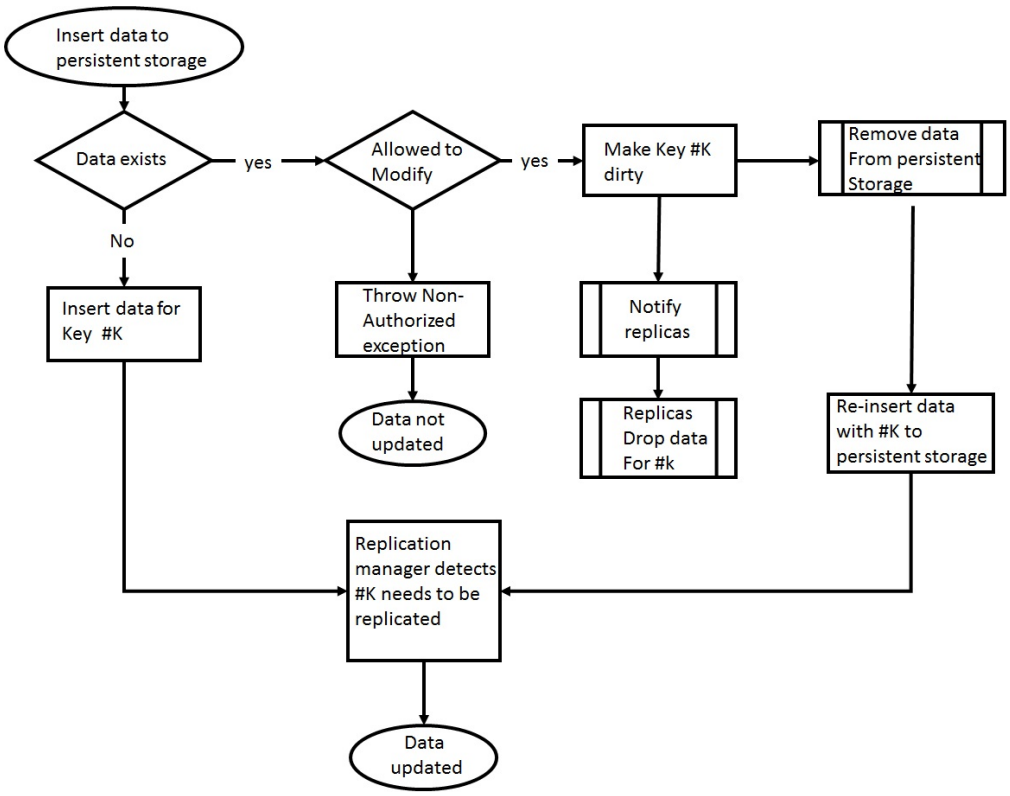


Figure 2.5.: Replication work flow in LibreSocial [6]

In LibreSocial, we use replication to store the data, so that data is available quickly on request even when the node responsible for the storage of data goes offline or failed due to unexpected circumstances. As we discussed in PAST, it uses (K,r) replication scheme in order to store the replica in the network. Object witha unique identifier "K" will be stored as many as "r" replicas in the network. Figure 5.22 describes the exact work flow of how replication takes place into LibreSocial [6].

2.4.4. Testing

Testing is very important phase in any kind of software development at any point in software development life cycle. Framework layer of LibreSocial application provides certain mechanisms which supports automated testing. For this purpose, predefined test plans are used in order to test application running on multiple virtual machines. Testing is normally used with combination of monitoring commands which produces the over all result of global statistics of LibreSocial network [6]. More details about testing and instruction in order to conduct a successful test can be found in the next

chapter.

2.4.5. Monitoring

From the above discussion, we know that Peer-to-Peer systems must utilize the global storage in order to achieve the high performance. In order to improve quality of service (QoS), we must address the issues like heterogeneity awareness of the peers, load balancing between peers due to the fact that different peers have different storage capacity and available bandwidth. All these issues can be addressed by keeping the P2P systems under constant monitoring. The simple way to monitor P2P systems is to implement a layer on the top of overlay which gathers information globally from all the peers and provides a global view of P2P systems in terms of matrix and values.

SkyEye.KOM [8] is used in order to monitor LibreSocial and to provide a statistical overview of the performance of LibreSocial during the testing. SkyEye.KOM is an information management layer which is applicable on the DHT (Distributed Hash Tables)and enables capacity based peer search. It gathers information from all the peers in the overlay network and provide statistic on the system.

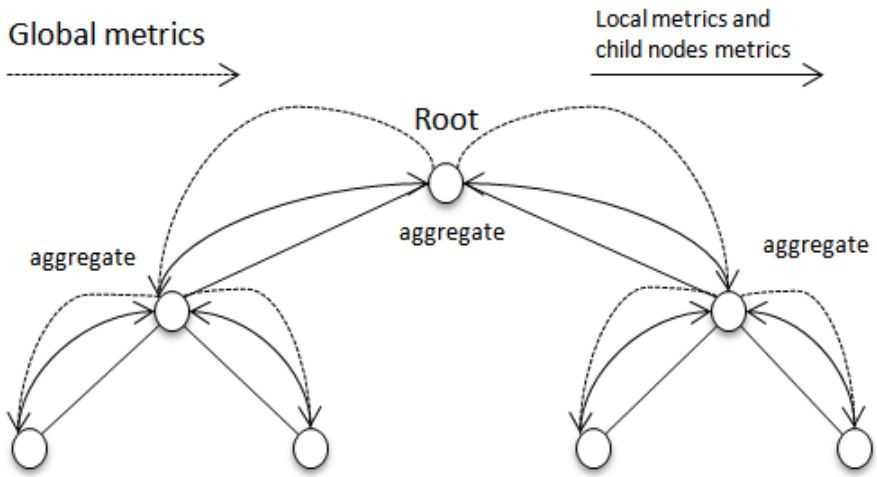


Figure 2.6.: Message routing in SkyEye.KOM [6]

Figure 2.6 describes how message routing takes place inside SkyEye.KOM in order to gather statistical information from all the peers and report to responsible peer. SkyeEye.KOM uses a tree based structure for the purpose of gathering data from the peer nodes.

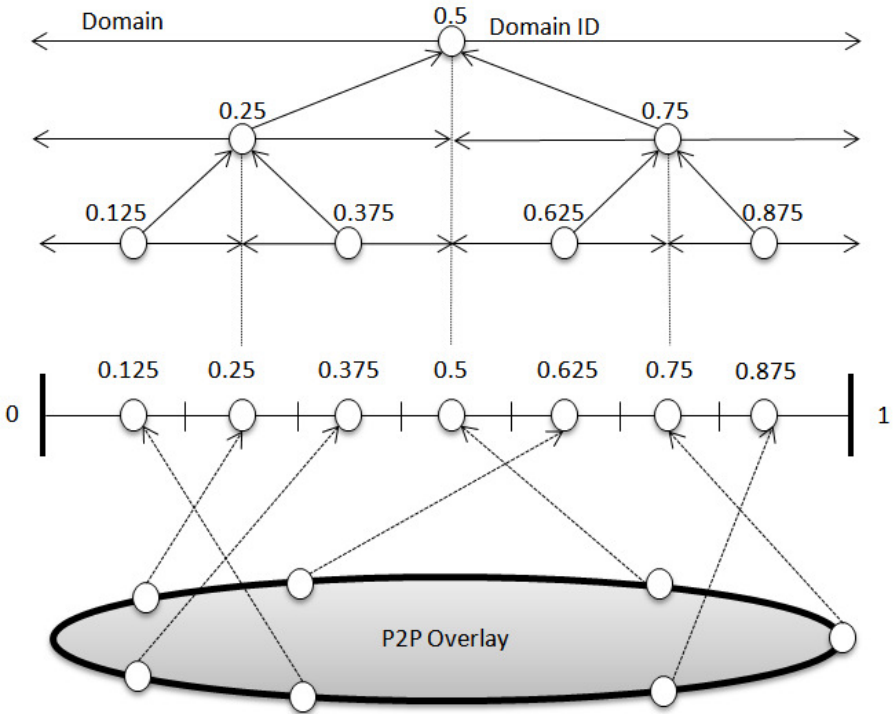


Figure 2.7.: SkyEye.KOM Tree topology [6]

Each node under SkyEye.KOM periodically calculates aggregate of own local view and received from child nodes and send it to the parent node(bottom to top). At the end, global data will be calculated by the root node and it will be re-distributed to the children in the same way(top to down). So at the end of this procedure each node in P2P network would have statistical data which can be used to analyzed the system performance.

It can be seen from the figure 2.7 that a tree topology is built on the top of the overlay to provide the efficient aggregation of the data. The ring of node IDs will be split into multiple parts correspond to the level of the tree with dynamic assignment of each peers to the domains where the largest domain size is [0,1] at level 0. So for each domain in layer, a node is responsible for gathering information from it's children, make aggregation and passes it on to it's parent. For example, in figure 2.7, the node "0.75" is responsible for aggregating information received from node "0.625" and node "0.875" and passes the aggregated information to the parent node "0.5"(root node in this case). Available aggregation strategies are: SUM, MIN, MAX, AVG, COUNT, MEAN.

On the other hand SkyEye.KOM also enables various mechanism to write the gathered monitoring data into various file format. Earlier, it used to write

data to the CSV files which is text based human readable format. Accroing to the master's thesis of Florian [9], the problem with CSV file was that the whole file must be read before executing any operations to update or add new data to that file. This causes really high load once the file gets bigger in the size. In a complex P2P application like LibreSocial, new data is generated with every single second due to the constant monitoring. So it is really time consuming and memory consuming to write new data into CSV file until the next update arrives which results into increased heap size and unresponsiveness of systems after certain period of time. In order to overcome this problem, Generation of CSV files were replaced with SQLite files during Florian's thesis [9]. I/O operations like read, write and update of SQLite files are much more faster as they can take place in parallel. i.e it is possible to write SQLite file witout reading whole file to the memory. The only thing which is not so impressive with SQLite files is, that the files need to be post-processed in order to be in the human readable form. More information can be found in Florian's master thesis.

As a result of this change, LibreSocial require very less memory for the monitoring but SQLite files cover more storage space than traditional CSV files.

2.5. Plugins And Apps

Topmost layer of LibreSocial application is Plugins and apps layer which provides application specific functionality.LibreSocial Plugins are written in OSGi framework. Each plugin provides OSGi commands for specific functionality which are used in the test plugin as an input in the form of test scripts. Plugins are developed in a way that they are using GUI based interactions as well as implemented functionality of framework layer in order to store and retrieve data for corresponding user actions. Figure 2.8 provides the names of all available plugins in LibreSocial application.

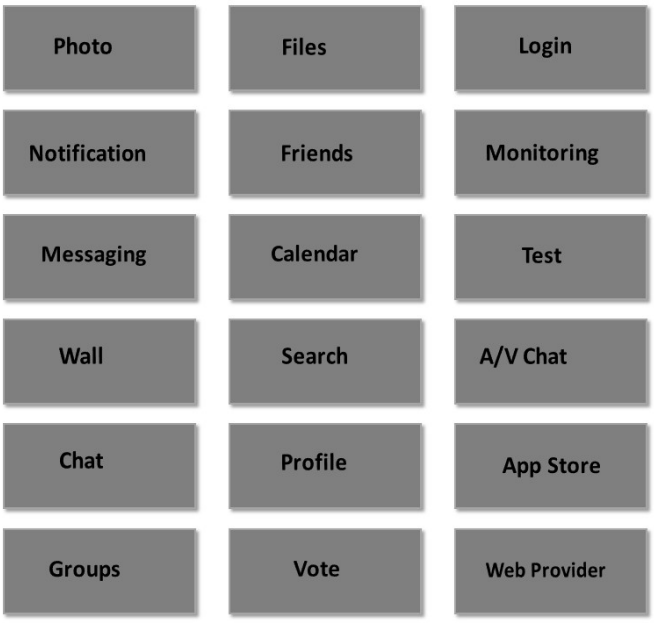


Figure 2.8.: Plugins in LibreSocial [6]

2.5.1. Photo Plugin

Photo plugin in the LibreSocial provides a functionality to create and update albums in order to share photographs. User of LibreSocial can share and upload photographs and album regardless of it's size and numbers. Apart from that, Photo plugin also provides security mechanisms to protect the rights of user's security and privacy. Photo plugin enables user to decide the with whom they want to share their photographs. Other users can also comment on the share/uploaded photographs. Photo plugin is encoded with a very nice image holder which enables the view action of photographs in a very attractive image container where user can simply change between the photo graphs by just using the arrow keys. Moreover, Photo plugin provides features like slide-slow, emoticons in the comment, caching and realizable image container for the smoother viewing operations.

Photo plugin is an essential part of LibreSocial application and it is really important that photo plugin should work as expected any given circumstances. Photo plugin provides number of OSGi commands which can be used to during the testing of photo plugins. These OSGi command includes AddPhoto, AlbumLoaded, CreateNewAlbum, DeleteAlbum, DeletePhoto, EditAlbum, EditPhoto.

2.5.2. Notification plugin

Notification plugin is important in a way that it notifies LibreSocial user during the occurrence of any event related to his/her user account. At the moment, Notification plugin is able to notify users about login, incoming calls/messages, new friend requests, comments on photographs, wall entries and comments on wall entries, group messages, voting invitation. Notification are being stored in the distributed linked list but still they are secure in a way that they are encrypted and can not be intercepted by unauthorized/ unintended users.

2.5.3. Messaging plugin

Messaging plugin is an prototypical implementation of email alike system. Messaging plugin provides all the basic functionality provided by ideal email application. Message plugin has two different views: Folder view where inbox, outbox folders are placed and Message view where user can read/write/open incoming and outgoing messages. it also provides label functionality using which user can organize messages according to label. It is also possible to send message to one or more person with a possibility of attached file in the message.

Unique aspect about this plugin is, that sending a file with using message plugin would be speedy with compare to traditional email application because of the fact that it does not send the real file but it just write the file into the individual storage of users. It is also possible to send any size of file using the messaging plugin.

Messaging plugin provides number of OSGi command which can be used to test the functionality of messaging plugin under given circumstances. These commands are SendMsg, GetInbox, GetOutbox, and DeleteInboxMsg.

2.5.4. Wall plugin

Wall plugin in LibreSocial provides basic functionality of social networking like sharing what is in your mind with one or more friends on the LibreSocial platform. Users can write on the other user’s wall as well as they can see what other users have shared if it is accessible by target users. Wall plugin is secure in a way that it has access control mechanism which enables use to share their information with either selected friends, friend list or publicly

with any LibreSocial user. Other user can also reply on the wall posts.

For the testing purpose, Number of OSGi commands are implemented for wall plugin. These commands are AddComments, AddPosts, RetrivePosts.

2.5.5. Chat plugin

In LifeSocial, chat plugin offers real time transmission of messages among the LibreSocial users. It supports one-to-one and many-to-many communications. The user can send the attachments, use the emoji, and the url parser in the chat interface. The user interface provides the chat history of the past conversations.

The available OSGi commands of chat plugin are SendChatMsg, SendInv, UnsubscribeTopic. UnsubscribeTopic allows user to leave the multichat at any time and he wont be able to recieve any update/notifications from that chat.

2.5.6. Group Plugin

Group plugin is developed to provide some basic functionality of forum/group like platform where users of LibreSocial can persue or discuss their common interest. Users can also share files and topics related to voting. Each user of LibreSocial is allowed to create group and he/she can add other participants to that group. Users can leave group at any time. Group features is really important feature for LibreSocial as it provides a basic platform in terms of collaboration and E-learning

Available OSGi commands for group plugin are AddFile, AddFolder, AddFriendstoGroup, AddGroup, DeleteContent, DeleteGroup, DeleteContent, GetGroupList, LeaveGroup, RenameContent.

2.5.7. Files Plugin

File plugin provides the interface to store files which are larger in size. Numerous file formats like pdf, Mp3, Mp4, exe etc... are supported by file plugin. This plugin can also be used in collaboration with other plugin. For instance, user can send uploaded file to other users through messaging plugin or share the file on the wall so other users can download/access that

fie.

As it is possible to store larger size of files, file plugin distributes the small chunks of file to the n number of different nodes with number of replicas in the network. This mechanism makes sure that, the data is evenly distributed across the network without burdening any particular node. Encryption mechanism is used to provide the enough level of security to file plugin.

2.5.8. Friends Plugin

Friends plugin is used to manage the friend list on the LibreSocial. All the friends in the friendlist can be viewd on the single page. It provide different functions related to user management like sending friend request to other users, unfriend, block/unblock other users etc...

OSGi commands implemented by friends plugin are SendFriendRequest and BlockFriendFromUserFriendList.

2.5.9. Calendar Plugin

LibreSocial users can use calender plugin in a private to schedule the events. it provides all the basic functionality which are provided by ideal clander application. OSGi command available for calender plugin are CreateEvent, EditEvent, DelEvent, GetEvent.

2.5.10. Search Plugin

Main aim of search plugin is to find other users on the LibreSocial network. It also allows user to search other users categorically.

The only available OSGi command for search plugin is GetSearchResults.

2.5.11. Profile Plugin

Profile plugin is used to store the personal user information in the LibreSocial. User data of profile plugin is also used to generate the unique identifier for user which can be used later to store the user data in the overlay network. Like other plugin, profile plugin is also wrapped with access control

mechanism which allow user to decide which kind of information he/she wants to share with whom.

Available OSGi command for the profile plugin are GetMyProfile, ShowCurrentProfile, StoreMyProfile.

2.5.12. Vote Plugin

Vote plugin provides explicit functionality to support the decision process. LibreSocial users can post/ask questions with some options and other invited users can select one of the option in order to express their opinions and views. Vote plugin also provides different way to present the answers to such questions/post. This includes pie charts, graph charts etc... and it can be exported as CSV file.

Available OSGi commands for vote plugin are CreateVote, DeleteVote, GetMyVottings, GetVottingResults, InviteUsers, Vote.

2.5.13. Audio/Video Chat Plugin

Video chat plugin interface provides direct audio video communication between LibreSocial users. Currently this functionality is using WebRTC [10] as base library which is browser specific. This feature does not support by all the browsers so it might create problems when used with incompatible browsers. Due to some challenges related to bandwidth problem with nodes in LibreSocial, Audio/Video chat plugin is limited to one to one communication only.

2.5.14. Apps Store

Apps store plugin is not a part of this thesis as it is not yet included in the application.

2.5.15. Login/Logout Plugin

Login plugin is one of the most important component of LibreSocial application in a way that, it provides basic functionality like creation of new users and login for existing users. Login plugin is an initial entry point for any

users in order to gain access to features provided by LibreSocial. Login plugin works in a combination with profile plugin. Whenever, new user is registered via login plugin, a new user profile will be created using the data input of login plugin. it also create unique user identifier based on the user name to store all the user related information in the network.

There are some other features like automatic login, cause of failure of login process, timeout in case of weak connections, and validation controls.

Available OSGi commands for login plugin are Register, Login, Logout.

2.5.16. WebProvider Plugin

LibreSocial is java based web application which is operated through front end with all the operations taking place into the back end. WebProvider plugin serves as a mediator between GUI and plugins. WebProvider uses embedded server Jetty [11] to serve multiple purposes. WebProvider basically allows user to use java api of plugin resided into LibreSocial architecture through REST interface. It also uses static web server which enables dynamic loading of pages into the browser with really small response time(faster loading of data into single web page).

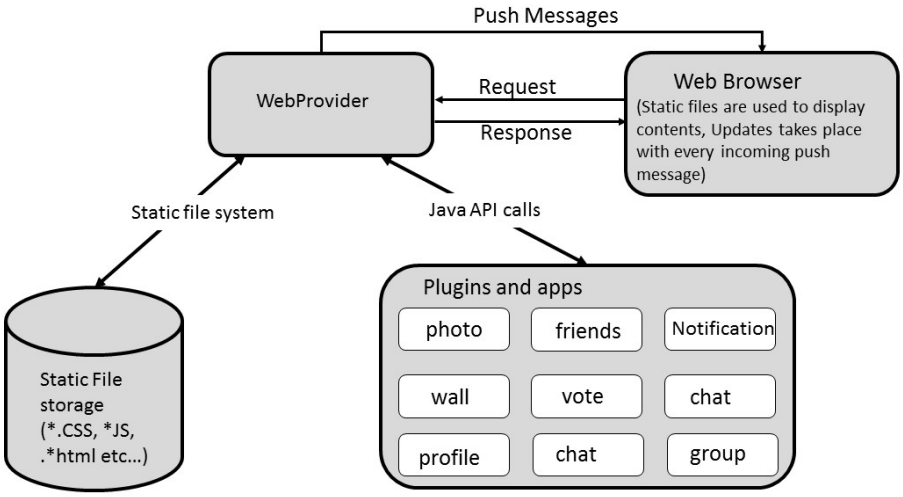


Figure 2.9.: WebProvide plugin interface [6]

Figure 2.9 provides detailed overview of WebProvider plugin. It also uses WebSocket connections ino roder to provide updates about the arrival of new data to responsible pugin. For example, WebSocket connection sends notification to front end whenever the new email is received.

2.6. Graphical User Interface

2.7. Improvements and bug fixing

This section provides an overview of the status of reported bugs and changes which were corrected during the making of this thesis. Detailed explanation is not provided for this section.

Error / Exception / Warning	**Status**
Send search results to browser before all results are found	Fixed
Automatically preselected network interface is incorrect	Not Fixed
Login fails right after logout	Fixed
German umlauts are not displayed properly in city selector	Fixed
Title- and mouse hover texts for buttons	Fixed
Quotationmarks in user names make the gui crash	Not Fixed
CSRF-Waekness Wall	Not Fixed
Deanonymization of user name	Fixed
Several null pointer exceptions	Fixed
XSS-Weakness if code is injected into user name and friend request is sent	Fixed
Time of Wall posts	Fixed
Memory is almost exhausted-Warning	Fixed
Warning: No Bootstraps reachable	Fixed
Network name is duplicated for each click on Discover	Fixed
Responsive Design / telescoping of elemente	Not Fixed
rice.p2p.past.PastException occurs	Not Fixed
Search does not find all results	Fixed
Popup windows cannot be moved	Fixed

Video chat: black window only	Fixed
Different default name for Chat Groups	Fixed
Group Messages not sent	Fixed
Tooltips for Group Chat Buttons	Not Fixed
OS X 10.10.3 Libresocial still not working	Not Fixed
Group Messages not sent	Fixed
Voting Results can not be displayed if opened a few times	Fixed
Abort image preload for albums when modal window was closed	Not Fixed
Upload the whole album even when the preload did not finish	Not Fixed
Serverinstallation / LibreSocial: Cannot open display	Not Fixed
DoS possible if actions are repeated	Not Fixed
Use selector element for country in search formular (like at the login page)	Not Fixed
Editor for Background/Proile images	Not Fixed
Background image on Friendlist	Not Fixed
Friendslist (Mainview) in multiple colums	Not Fixed
Telephony feature does not work correctly	Not Fixed
Firefox Videochat: alsa-devices only	Not Fixed
NodeIsFaultyException when Logout fails	Not Fixed

Table 2.1.: List of reported bugs and status

2.8. Chapter Summary

Main aim of this chapter was to provide a brief overview of underlying architecture of LibreSocial with detailed explanation of each layer of dedicated architecture. Current chapter is also important in a way that it provides brief detail about all the modifications performed by subsequent project groups and master thesis in order to make FreePasty compatible to LibreSocial. Following chapter will highlight all the important aspects related to test plugin used in the testing of LibreSocial.

3. Testing

"Software testing is a process of executing a program or application with the intent to finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product: Meets the business and technical requirements that guided it's design and development" [12].

As we have discussed in the above sections that, P2P systems are really complex to deal with. Building and testing P2P applications is difficult and important as they always deployed on huge number of anonymous nodes in the overlay network [13]. Behavior of nodes in the overlay network depends mostly on local circumstances like system parameters on which the node is being operated, bandwidth etc... Such volatile behavior of nodes in P2P systems is very common and must be interpreted as a fault during the test (failed nodes). Interpretation of failed nodes during the test is not just enough but it is also necessary to interpret behavior of nodes in P2P systems in terms of Statistical measurements. Statistical measure provides information about all the working nodes in P2P systems as numerical values (life time of node, capacity, joining time, leaving time) with respect to system time, memory, and storage space etc...

This chapter provides brief explanation about testing of LibreSocial application and all the requirements in order to conduct a successful testing of LibreSocial using dedicated "Test Plugin".

3.1. Test Plugin

Test plugin was developed with a purpose to test the performance and reliability of LibreSocial application. Test plugin can be found under the package "com.p2pframework.tester". Management of test plugin can be done using "config.txt" 3.1 file which resides under the "resource" folder of same package. Test plugin can be operated using two different modes: Master and

Slave. Master node is used to control other nodes in the overlay network and is a unique node in the network. Master node controls other nodes in a way that it sends test plan as command to other nodes. Master node also provides the information about users being registered on the different slaves. Slave nodes in a test mode are the nodes which takes instructions from the master node and execute all the actions performed on the master node.

This section explains about the details of the requirements, prerequisites, test plan, test case structure, and test case implementation of the developed test plugin.

3.1.1. Prerequisites

In order to enable Test plugin, we must execute certain necessary operations.

Updating Confix.txt File

We must enable test plugin in the config.txt file. The value of "*enableTest-Plugin*" must be true. One more change that must be done to config.txt file is to give proper IP address of server/Masternode. It also require to update the correct path of Test plan as the same confix.txt file can be used for different type of testing and test plan varies accroding to the type of test. Other changes are also possible accroding to the need of test strategy and planning. Figure 3.1 describes the structure of config.txt file in detail.

Resource Folder

Resource folder contains two types of files: configuration files like Config.txt, OnlineDuration.tx, testCaseRegistry.txt and data file like cityNames.txt, loginNames.txt, pictures etc... Resource folder plays important role for the testing as it severs login plugin with all the necessary instructions and input. For example, configuration files like config.txt are use for the configuration of test including identification of master and slave nodes. While, database like loginName.txt serves as input for test cses. For example, when a random slaves login, it will be assigned names from the loginName.txt file. Below, provided detailed information about each element of resource folder.

```
# IP address of master node (also used as bootstrap server)
server=131.234.31.40

# Enables the TestPlugin Values: true | false
enableTestPlugin=true

# Enables the random slave login mode: true | false
enableRandomLogin=true

# If the random slave login mode is not enabled, you can login
# the specific user you want: <username,password>
user=tester,123

# Slaves can logout according to an online duration distribution
# and log in again with another username
# In random mode only. Values: true | false
enableAutomaticLogout=true

# Path to testplan. Either starting with http:// or file:/
testplan= Dropbox  public link of testplan.txt file

# Path to a testplan with time slots for execution.
# Either starting with http:// or file:/
timedTestplan= Dropbox public link of timedTestplan.txt

# Path to a start script, that is executed when a new user registers
# in random mode.
startScript= Dropbox public link of startScript.txt

# Maximum time to wait after TestCase has been executed (in ms)
maxSleepTime=5000

# Minimum time to wait after TestCase has been executed (in ms)
minSleepTime=2000

# Time for Wait command (in ms)
waitTime=6000

# Time for startup delay (in ms)
startupDelay=5000
```

Figure 3.1.: Parameters in Config.txt

- **config.txt**
  As discussed in the above section this file is used to enable/disable test plugin as well as it provides a path of input test plans.

- **testCaseRegistry.txt**
  Used to register new test cases and provides class names of test cases.

- **onlineDuration.txt**
  Provides online duration distribution of each participants.

- **message.txt**
  Provides contents for test cases related to chat plugin.

- **loginNames.txt**
  Provides contents for test case related to login plugin.

- **groupNames.txt**
  Provides contents for test case related to group plugin.

- **pictures**
  Provides pictures for test cases related to profile and photo plugin.

- **countryNames.txt, cityNames.txt and interests.txt**
  Provides content related to profile plugin.

It is also important to put the resource folder in a correct location. Location of resource folder is different for different operation systems.

- **Windows OS**
  For Windows operating systems resource folder must me inside the eclipse folder: /eclipse/resources.

- **Linux OS**
  It is similar to Windows Operating system.

- **Mac OsX**
  For mac OsX it must be located at : /Applications/Eclipse/Eclipse.app/Contents/MacOS/resources

Test Script

Test scripts are basically used in a test procedure to test to verify whether the system under test performs the task as expected by end users. Test cases can be combined with other test cases in order to generate a test scripts. In our case we are using test scripts to test the behavior of nodes in P2P overlay network by generating data traffic. Test scripts also play an important role in monitoring the performance of these nodes. Test scripts varies with the different type of testing. In our case we will be using test scripts which comprises of all the system functionality aiming to test and monitor the performance of system during a long time of run.

Small example of test script is give in figure 3.2. Test scripts in LibreSocial contains OSGi commands which are implemented in order to initiate the test case action for particular functionality. For example, for the test case : change profile picture, the implemented OSGi command is TestCaseChangeProfilePicture. Please note that the path of test script (test plan) must be updated into config.txt file.

```
1x TestCaseChangeProfilePicture 1m
1x Wait 1m
1x TestCaseChangeProfileInformation 1m
1x Wait 1m
1x TestCaseViewOwnProfile 1m
1x Wait 1m
1x TestCaseViewFriendsProfile 1m
1x Wait 1m
5x TestCaseSendWallPost 2m
1x Wait 1m
5x TestCaseCommentWallPost 2m
1x Wait 1m
5x TestCaseViewOwnWall 2m
1x Wait 1m
```

Figure 3.2.: Test script in LibreSocial example

3.2. Test Cases

It is important to have an idea about how test cases works and what are the available test cases. This section provides short explanation about the structure of LibreSocial test cases. It also provides and overview of List of available test cases with their description, preconditions and valid parameters for respective test case.

3.2.1. Test Case Structure

Abstract class of test case in LibreSocial contains three explicit implementations: Preconditions, Parameters and OSGi commands.

In order to execute certain test cases, we must fulfill some conditions related to that test case. While, executing test case, it will check whether it fulfills required preconditions or not. if preconditions are not being fulfilled, it will execute fallback. The fallback is a TestCase, that needs to be executed to fulfill the preconditions of the certain test case [14].

Parameters are required for the execution of test cases as parameters contains all the necessary information like (IDs, Contents, Names etc..) which provides input content to the test case. if the method for checking valid parameters return true, the test case will be executed or proceed towards next step. In case of missing or invalid paramater, it will generate all the needed paramaters.

Successful execution of test case is only possible when all the preconditions are satisfied and all the available parameters are available. More information about implementaion fo test cases and test case structure can be found in the previous master thesis conducted. [15, 14].

3.2.2. Available Test Cases

List of available/valid test cases are given in table 3.1.

Test Case :	**Delete Album**
Description :	Delete album of user
Preconditions :	User logged in, user's albums loaded, at least one album
Parameters :	albumID

Test Case :	**Block User**
Description :	To block a friend from user's friends list
Preconditions :	User logged in, user's friends list loaded, at least one friend
Parameters :	Friend name

Test Case :	**View Friend's Photo**
Description :	To browse a photo of friend
Preconditions :	User logged in, at least one friend, albums of chosen friend loaded
Parameters :	Friend name, ID of album, ID of photo

Test Case :	**Send Message**
Description :	To send a message to another user
Preconditions :	User logged in
Parameters :	User name, subject of message, content of message

Test Case :	**Edit Album**
Description :	To edit album's information
Preconditions :	User logged in, user's albums loaded, at least one album
Parameters :	ID of album, album name, description

Test Case :	**Delete Photo**
Description :	To remove a user's photo
Preconditions :	User logged in, user's albums loaded, at least one photo
Parameters :	ID of album, ID of photo

Test Case :	**View Own Albums**
Description :	To browse own albums list
Preconditions :	User logged in
Parameters :	-

Test Case :	**View Friend's Albums**
Description :	To browse albums list of a friend
Preconditions :	User logged in, at least one friend
Parameters :	Friend name

Test Case :	**View Inbox Messages**
Description :	To browse user's inbox messages
Preconditions :	User logged in
Parameters :	-

Test Case :	**View Outbox Messages**
Description :	To browse user's outbox messages
Preconditions :	User logged in
Parameters :	-

Test Case :	**Delete Inbox Message**
Description :	To remove a messages from user's inbox
Preconditions :	User logged in
Parameters :	ID of message

Test Case :	**Send Multi Chat Invitation**
Description :	To invite all friends into a chat channel
Preconditions :	User logged in, at least one friend
Parameters :	Friend names

Test Case :	**Send Multi Chat Message**
Description :	To send a chat message
Preconditions :	User logged in, at least one chat channel
Parameters :	ID of chat channel, content of message

Test Case :	**Leave Multi Chat**
Description :	To quit from a chat channel
Preconditions :	User logged in, at least one chat channel
Parameters :	ID of chat channel

Test Case :	**Create Vote**
Description :	To create a vote
Preconditions :	User logged in
Parameters :	Topic, description, options, private or not, minimum choices, maximum choices

Test Case :	**Delete Vote**
Description :	To remove a vote
Preconditions :	User logged in, at least one vote
Parameters :	ID of vote

Test Case :	**Get Voting Result**
Description :	To obtain result of vote
Preconditions :	User logged in, at least one vote
Parameters :	ID of vote

Test Case :	**Get My Votings**
Description :	To retrieve user’s voting list
Preconditions :	User logged in
Parameters :	-

Test Case :	**Vote**
Description :	To do a voting
Preconditions :	User logged in, at least one vote
Parameters :	ID of vote, index of choice

Test Case :	**Create Group**
Description :	To create a group
Preconditions :	User logged in
Parameters :	Group name

Test Case :	**Invite Friend to Group**
Description :	To invite a friend into a group
Preconditions :	User logged in, at least one group, at least one friend
Parameters :	ID of group, friend name

Test Case :	**View Group**
Description :	To browse a group
Preconditions :	User logged in, at least one group
Parameters :	ID of group, ID of root folder

Test Case :	**Leave Group**
Description :	To quit from a group
Preconditions :	User logged in, at least one group
Parameters :	ID of group

Test Case :	**Create Folder**
Description :	To add a folder in a group
Preconditions :	User logged in, at least one group
Parameters :	Folder name, ID of group, ID of root folder

Test Case :	**Rename Folder**
Description :	To rename a folder
Preconditions :	User logged in, at least one group, at least one folder
Parameters :	Folder name, ID of group, ID of root folder, ID of folder

Test Case :	**View Folder**
Description :	To browse a folder
Preconditions :	User logged in, at least one group, at least one folder
Parameters :	ID of group, ID of folder

Test Case :	**Delete Folder**
Description :	To remove a folder
Preconditions :	User logged in, at least one group, at least one folder
Parameters :	ID of group, ID of root folder, ID of folder

Test Case :	**Upload File in Folder**
Description :	To add a file into a folder
Preconditions :	User logged in, at least one group, at least one folder
Parameters :	ID of group, ID of folder, path to file

Test Case :	**Delete File from Folder**
Description :	To remove a file from a folder
Preconditions :	User logged in, at least one group, at least one folder, at least one file
Parameters :	ID of group, ID of folder, ID of file

Test Case :	**View Forum**
Description :	To browse forum in a group
Preconditions :	User logged in, at least one group
Parameters :	ID of group

Test Case :	**Create Forum Thread**
Description :	To create a thread in forum
Preconditions :	User logged in, at least one group
Parameters :	ID of group, thread name

Test Case :	**Comment Forum Thread**
Description :	To add a comment on a thread
Preconditions :	User logged in, at least one group, at least on thread
Parameters :	ID of group, ID of thread, content of comment

Test Case :	**View Forum Thread**
Description :	To browse a thread
Preconditions :	User logged in, at least one group, at least on thread
Parameters :	ID of group, ID of thread

Test Case :	**View Calendar**
Description :	To browse user’s calendar
Preconditions :	User logged in
Parameters :	Current year, current month

Test Case :	**Create Calendar Event**
Description :	To add an event on calendar
Preconditions :	User logged in
Parameters :	Event name, start date, end data, description

Test Case :	**Edit Calendar Event**
Description :	To edit a calendar event
Preconditions :	User logged in, at least on calendar event
Parameters :	ID of event, event name, start date, end data, description

Test Case :	**Logout Login**
Description :	To logout current user account, then register a new user and login
Preconditions :	User logged in
Parameters :	-

Test Case :	**Send Wall Post**
Description :	To send a post on own wall
Preconditions :	User logged in
Parameters :	Content of post

Test Case :	**Comment Wall Post**
Description :	To add a comment on friend's post
Preconditions :	User logged in, at least one friend, chosen friend has wall entry
Parameters :	Friend name, ID of post, content of comment

Test Case :	**View Own Wall**
Description :	To browse user's own wall
Preconditions :	User logged in
Parameters :	ID of current user

Test Case :	**View Friend's Wall**
Description :	To browse a friend's wall
Preconditions :	User logged in, at least one friend
Parameters :	ID of friend

Test Case :	**Create Photo Album**
Description :	To create a photo album for current user
Preconditions :	User logged in, user's albums loaded, album name available
Parameters :	Album name, description

Test Case :	**Upload Photo**
Description :	To add a photo into an album of user
Preconditions :	User logged in, user's albums loaded
Parameters :	Album name, photo name, path to photo, description
Test Case :	**Delete Calendar Event**
Description :	To remove an event from user's calendar
Preconditions :	User logged in, at least on calendar event
Parameters :	ID of event
Test Case :	**Search**
Description :	To look for users
Preconditions :	User logged in
Parameters :	Criterion for searching

Table 3.1.: List of Available and Valid Test Cases in LibreSocial [14]

3.3. LibreSocial Configuration Parameters

LibreSocial Configuration Parameters are really important in any kind of testing conducted on LibreSocial as they have direct effect on the test results. Since the last long term test on LibreSocial conducted by Dheeraj Balabhadruni Weifeng Peng [14], new configuration parameters were added by Florian Schulte in his master thesis [9].

Same parameters are being used in the long term testing conducted as part of this thesis but the value of these parameters are different due to the nature of long term test as described in figure 3.3 and figure 3.4. LibreSocial configuration parameters can be found under "/org.lifesocial.core.utilities/config/LifeSocialConfig.txt". For more details please refer [14, 9]

```
# 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 = 50

# timeout for a canyouprovidemessage in ms
CanYouProvideTimeOut = 7000

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

# number of replications of a data object in the system
# NOTE: It is important that #strongNodes >= replicationFactor+1
in case weak nodes are present
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

# disable the parallel and iterative routing (1 = enabled, 0 = disabled)
disablePIRouting = 1

# only enable parallel routing (1 = enabled, 0 = disabled)
justParallel = 0

# configure the parallelity, since this number is not constant,
do not change it if necessary (0 = disabled, else = the parallelity u want )
parallelity = 1
```

Figure 3.3.: LibreSocial parameters (1)

```
# in monitoring, how many rounds you want to combine
combineParameter = 3

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

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

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

# enables message(email) automatically deleted after 1000 seconds
 (1 = enabled, 0 = disabled)
messageAutoDeleted = 0

# 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.
#mobileNodes = 0

# visualization of monitored data in runtime, this function
 will keep the memory usage increasing! (1 = enabled, 0 = disabled)
monitorDataVisual = 0

# url to the bulletin board
#Not used in the test
bulletinURL = Dropbox public url
```

Figure 3.4.: LibreSocial parameters (2)

3.4. Chapter summary

Goal of this chapter was to make reader aware of the functionality of test plugin as it is a special plugin developed with a purpose to test underlying functionality and performance of LibreSocial. Up to now all the important details were provided to put the reader into right context. Next two chapters provide details about conducted test and evaluation test results.

4. GUI based test

Main goal of GUI based testing of LibreSocial is to observe the relative behavior front-end (GUI) and back-end of LibreSocial during a long period of running under real environment. Single GUI based test was carried out as a part of this master thesis.

This chapter explains all the necessary steps which need to be performed in order to carry out GUI based test. It also gives detail about test environment and other details related to this test. Last covers all the observation made during GUI based test.

4.1. Test Environment

In order to match with realistic environment, we set up a network of 10 virtual machines. For this we used ten visualized computer systems that were running in the university of Paderborn pool area.

Hardware and Software configurations of all machines were same as below.

- Processor : Intel i7-2600, Intel Core i5
- RAM : 8 GB/4 GB
- Operating System : openSUSE 13:2 (x86 64), Windows 8 (x86 64), MacOS
- Storage Space: Different for different machine
- Network Interface : Gigabit Ethernet
- Java VM : OpenJDK JRE 1:8:0

4.2. Observations

Main aim of this test was to observe the behavior of LibreSocial front-end in real environment. For this purpose we made a LibreSocial network of 6 nodes under real environment with some of the nodes using different operation system. Nodes used to form the network were belong to different network as well. For example, 3 nodes were running in the same notwork while 2 nodes were running under different network. A node who was using MacOs was not able to join the ring in the same test due to NAT issues.

All the instances of LibreSocial were running absolutely fine for at least 15 to 18 hours of time. After that GUI of most of the instances become real slow and response time kept on increasing with the time passing by without any error in the eclipse console (As we were running one instance in development environment to observe exceptions).

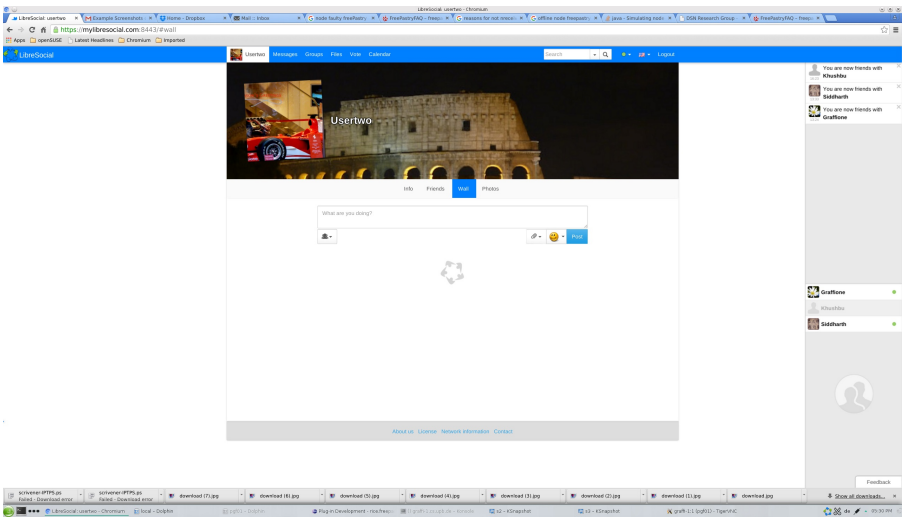


Figure 4.1.: LibreSocial:Wall

Figure 4.1 shows a screen shot of LibreSocial wall after approx 15 hours. It took almost 10 to 12 minutes to retrieve the original wall entries. Not just wall, but the same thing was observed with login and logout plugin. In 2 instances it became almost impossible to perform logout after 15 hours.



Figure 4.2.: LibreSocial:Chat

It was not possible to see the online/offline status of nodes as they became permanently dead after 18 hours of time as shown in figure 4.2 . while, Some nodes were able to send messages but at the same time recipient node was not able to receive messages. Most of the plugins including vote plugin, Group plugin, Message plugin, Photo plugin 4.3 etc... stopped responding after 17 to 18 hours.

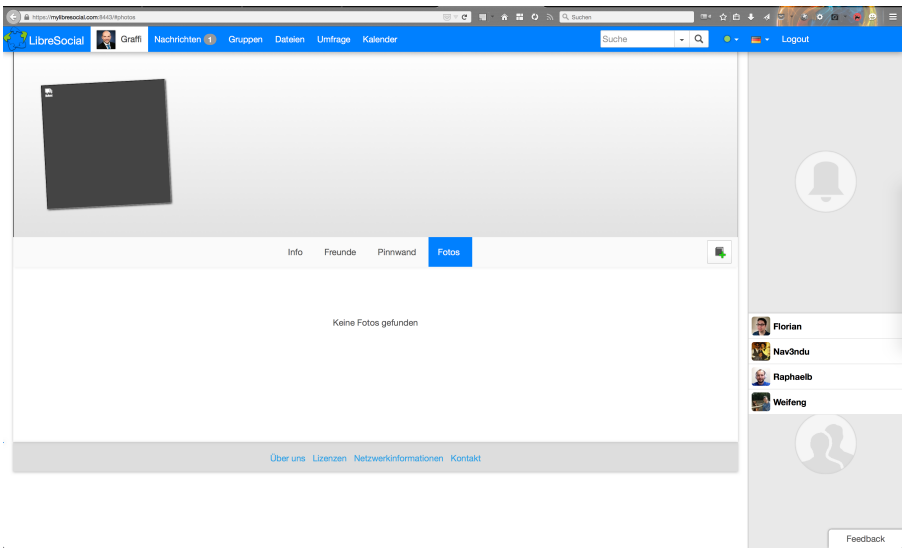


Figure 4.3.: LibreSocial:Photo

4.3. Outcome

The test has provided really important feedback which was helpful in order to conduct a long term test. We also observed that the storage layer of the application also causes problems when LibreSocial is used under real conditions especially with the data which are being stored in the specially designed distributed linked list. Message lost also occurred due to the fact that some of the nodes became dead after 19 hours and they were not able receive messages anymore. This problem is somehow related to Priority transport layer of FreePastry which is responsible for the transfer of messages from the source to destination. More detailed evaluation about this problem is done in the last section which concludes the work of master thesis.

5. Long Term Test

Main goal of long term testing of LibreSocial is to measure the performance and reliability of overlay network and related behavior of application during a very long period of running under the specified conditions and configuration. Long term testing was perform with an intention to uncover all the defects which leads to the instability of LibreSocial network by producing traffic and load into the network. Two long term test were carried out as a part of this master thesis.

This chapter explains all the necessary steps which need to be performed in order to carry out a long term test. It also gives detail about test environment and other detailes related to long term test. Last section covers the evaluation of long term tests.

5.1. Test Environment

In order to match with realistic environment, we set up a network of 10 virtual machines. For this we used ten visualized computer systems that were running in the university of paderborn pool area.

Hardware and Software configurations of all machines were same as below.

- Processor : Intel i7-2600
- RAM : 8 GB
- Operating System : openSUSE 13:2 (x86 64)
- Storage Space: 12 GB
- Network Interface : Gigabit Ethernet
- Java VM : OpenJDK JRE 1:8:0

5.2. Test Deployment

Test deployment is very important stage of long term testing. Since we are using virtual machines in the long term testing, we have to have an idea about remote login procedures and tools like SSH tunneling, VNC server, VNC viewer etc..

Though all these steps and procedures were describe by Dheeraj Balabhadruni and Weifeng Peng in their combined master thesis [14], this section provides a very short overview as some of the procedure and steps are not same as used for pervious long term tests.

5.2.1. Setting Up Development Environment

For 10 virtual machines, we are using Eclipse Keplar(x86 64) to run the application in a test mode in corporation with SmartGit repository to synchronize the application code on different virtual machines. Apart from that, it is also mendatory to enable the test plugin by updating confix.txt file and resource folder must be present as discussed in previous chapter.

5.2.2. Test configuration parameters

LibreSocial parameters must be changed depending on the test type and test goal. In the current test we have used configuration parameters as shown in figure 3.3 and figure 3.4.

5.2.3. Secure connections

We have used SSH tunneling in order to establish secure connection between local and virtual machines [16]. secure connection can be eestablished by using following command.

```
1      $ ssh -g -L <source port number>:<destination server>:
2        <destination port number> <host>
```

Basically, this command does the operation of local port forwarding. Here, the option -g is used to allow the remote hosts to connect the local forwarded ports, and option -L represents local port forwarding [16]. In order to carry

out local port forwarding operations successfully X11Forwarding must be enabled on both the server and client [16]. Root permission of the client is mandatory.

5.2.4. GUI access to Virtual Machines

As we have used virtual machines in long term test, it is necessary to have access to the GUI of these virtual machines. There are many tools availble for the purpose. For example, one can use XVFB display server, VNC, Teamviewer, Microsoft Remote Desktop(Windows OS), Apple remote desktop(MacOsX) etc...In our case we are using OpenSuse so we will use VNC server in order to gain secure connection to all virtual machines.

```
1
2   #!/bin/sh
3
4  unset SESSION_MANAGER
5  unset DBUS_SESSION_BUS_ADDRESS
6  OS=‘uname -s‘
7  if [ $OS = ’Linux’ ]; then
8    case "$WINDOWMANAGER" in
9      *gnome*)
10       if [ -e /etc/SuSE-release ]; then
11         PATH=$PATH:/opt/gnome/bin
12         export PATH
13       fi
14       ;;
15   esac
16 fi
17 if [ -x /etc/X11/xinit/xinitrc ]; then
18   exec /etc/X11/xinit/xinitrc
19 fi
20 if [ -f /etc/X11/xinit/xinitrc ]; then
21   exec sh /etc/X11/xinit/xinitrc
22 fi
23 [ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
24 xsetroot -solid grey
25 xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
26 twm &
```

Figure 5.1.: Xstartup.sh

Detailed information about basic operations related to VNC [17] server and VNC viewer can be found in the master thesis of Dheeraj Balabhadruni and

Weifeng Peng [14]. Figure 5.1 provides overview of the configuration file for VNC server used during this thesis. Figure 5.2 shows GUI of virtual machines which can be accessed using VNC viewer.

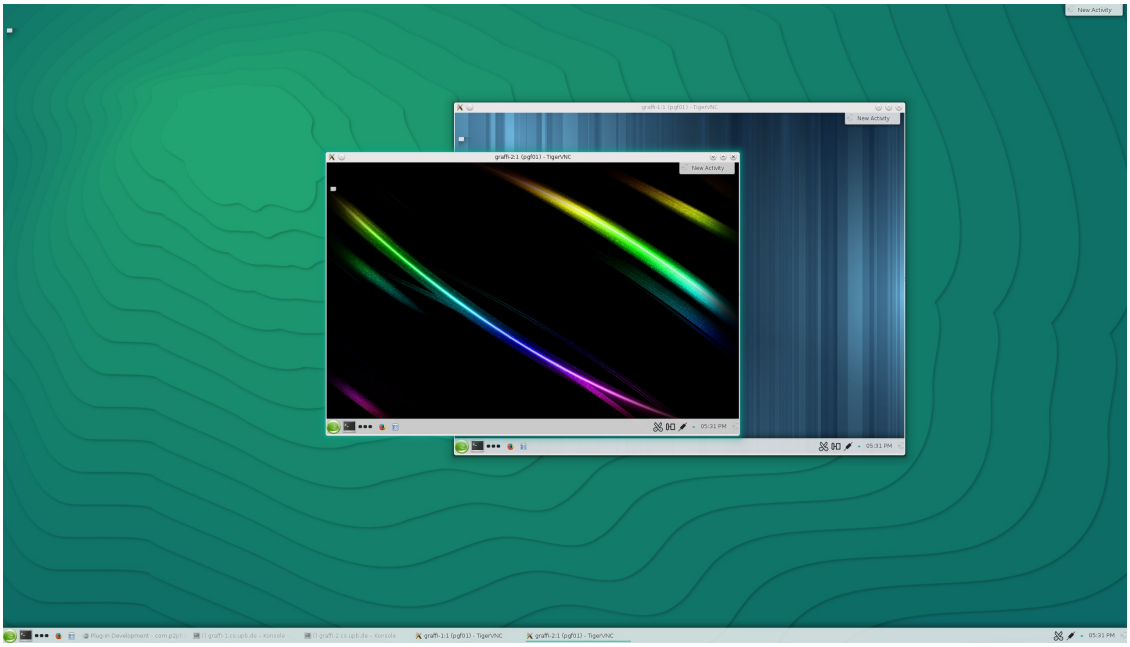


Figure 5.2.: GUI-Virtual machines-Graffi-1 and Graffi-2 in VNC viewer

5.2.5. Test Execution

Test execution contains two basic steps.

1. Delete old monitoring data before running a test

Every time when running, LibreSocial generates statistical storage and monitoring data. It is necessary to delete old data before running the LibreSocial application for test. This procedure can be done manually or by using shell script which aims to delete storage data for all the virtual machines used in this test.

Location of this generated data can be varied based on Operating system. In our test we are using custom location for storage and monitoring data. Following are the steps to customize the location of such data.

In the eclipse explorer, One can navigate to org.lifesocial.ui -> ProductConfiguration -> libresocial.product -> Run As -> Run Configuration ->Arguments -> VM arguments. In VM arguments one can change the location where they

want to store statistical storage and monitoring data.

For example in current test we are storing monitoring data in the "home" directory inside the folder "pgf01" of each virtual machine. So the VM argument would be

-Duser.home = /home/pgf01

Customization of location is beneficial in a case when you work with virtual machines with restricted permissions.

2. Input commands to execute test

There are two ways to start the test in LibreSocial [14]. Here, we are using master node's console as a input interface to execute the test script. It can be done by executing OSGi command "executeTimedTestPlan" in master node's eclipse console followed by "sendFriendRequestInOrder" in order to generate friendship between all slave nodes .

5.3. Test Script

Main purpose of test script is to generate traffic in the LibreSocial network and keep all the instances away from being idle. It should be also designed in a way it includes every single functionality of LibreSocial in order to measure the behavior of application in specified conditions. As discussed in above sections, Config.txt file inside test plugin is used specify the input test script. Timed test plan is used as input test script.

Overview of test script used during this test is shown under table 5.1. This test script is capable of running for more 2500 minutes.

5x TestCaseChangeProfilePicture 10m
1x Wait 15m
3x TestCaseChangeProfileInformation 10m
1x Wait 15m
4x TestCaseViewOwnProfile 10m
1x Wait 15m
4x TestCaseViewFriendsProfile 10m
1x Wait 15m
5x TestCaseSendWallPost 10m
1x Wait 15m
5x TestCaseCommentWallPost 10m
1x Wait 15m
5x TestCaseViewOwnWall 10m
1x Wait 15m
5x TestCaseViewFriendsWall 10m
1x Wait 15m
10x TestCaseSendMessage 10m
1x Wait 15m
5x TestCaseViewOutboxMessage 10m
1x Wait 15m
5x TestCaseViewInboxMessage 10m
1x Wait 15m
2x TestCaseDeleteInboxMessage 10m
1x Wait 15m
4x TestCaseSendMultiChatInvitation 10m
1x Wait 15m
10x TestCaseSendMultiChatMessage 10m
1x Wait 15m
4x TestCaseLeaveMultiChatMessage 10m
1x Wait 15m
5x TestCaseSearch 15m
1x Wait 15m
4x TestCaseViewCalendar 10m
1x Wait 15m

5x TestCaseCreateCalendarEvent 20m
1x Wait 15m
2x TestCaseEditCalendarEvent 10m
1x Wait 15m
2x TestCaseDeleteCalendarEvent 10m
1x Wait 15m
4x TestCaseGetMyVotings 10m
1x Wait 15m
10x TestCaseCreateVote 15m
1x Wait 15m
10x TestCaseVote 10m
1x Wait 15m
5x TestCaseGetVotingResults 10m
1x Wait 15m
2x TestCaseDeleteVote 10m
1x Wait 15m
5x TestCaseCreatePhotoAlbum 20m
1x Wait 15m
5x TestCaseUploadPhoto 10m
1x Wait 15m
5x TestCaseViewOwnAlbums 10m
1x Wait 15m
5x TestCaseViewFriendsAlbums 10m
1x Wait 15m
10x TestCaseViewFriendsPhoto 10m
1x Wait 15m
5x TestCaseDeletePhoto 10m
1x Wait 15m
4x TestCaseDeleteAlbum 10m
1x Wait 15m
2x TestCaseCreateGroup 10m
1x Wait 15m
4x TestCaseViewForum 10m
1x Wait 15m

5x TestCaseCreateForumThread 15m
1x Wait 15m
5x TestCaseCommentForumThread 20m
1x Wait 15m
5x TestCaseViewForumThread 20m
1x Wait 10m
10x TestCaseCreateFolder 3m
1x Wait 15m
2x TestCaseViewGroup 10m
1x Wait 15m
2x TestCaseRenameFolder 10m
1x Wait 15m
15x TestCaseUploadFileinFolder 10m
1x Wait 15m
5x TestCaseViewFolder 20m
1x Wait 15m
4x TestCaseLeaveGroup 10m
1x Wait 10m

Table 5.1.: Test Script Used in Long Term Test

5.4. Monitoring

As we discussed in section 5.2.5, LibreSocial generates monitoring data as SQlite file. In order to do statistical evaluation we must convert this SQlite file into human readbale format. for that purpose we can convert SQlite file int Gnuplot plot files which serves as input to different file formats. Steps to convert monitoring data into Gnuplot file are as follow,

Go to eclipse explorer, com.p2pframework.monitoring -> src -> com.p2pframework.monitoring.sqlite2gnuplot -> SQLiteToGnuplot.java -> run As ->run Configurations -> arguments -> Program arguments -> path to monitoring file (.SQlite file).

Above step will generate Gnuplot file which can easily be converted into PDF file format for detailed evaluation of monitoring results.

5.5. Test Evaluation

In this section, we will evaluate the statistical data gathered at the end of each long term test round. We will discuss only important results. All the results derived at the end of the test are attached in appendix. Due to the huge number of data, they are only part of digital thesis report (pdf).

Please note that two rounds of long term tests were conducted to check the consistency of statistical results. All the modifications were done in the first stage of thesis before conducting both tests. LibreSocial configuration parameters are also same for each test.

5.5.1. First Round Results Evaluation

This section evaluates the results of first round of long term test which was conducted with 10 instances.

Node Counts

Figure 5.3 gives an information about total number of network during the test. As it is obvious from the figure that availability of nodes in the network were pretty much constant as it shows average number of nodes were 9 during the whole period of time. There was some point in time around 3950 minutes when the node count dropped all of sudden but raised as quickly as it was dropped.Figure 5.4 shows the average amount of time the users took to join the network during the test which is around 7.8 seconds.

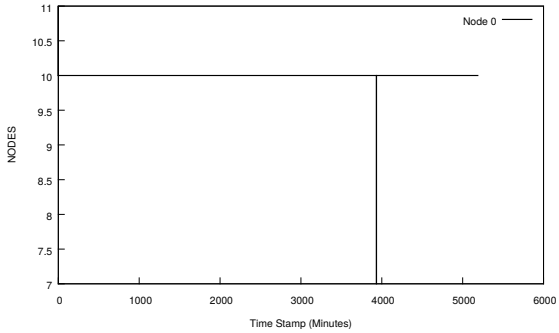


Figure 5.3.: Global Node count

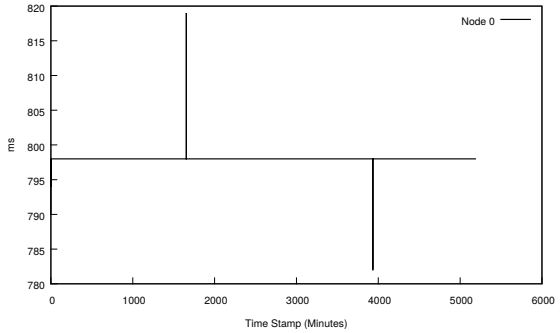


Figure 5.4.: Node joining time

Exceptions and Error Handling

Figure 5.5, 5.6 and 5.7 are very important results as they are directly related to the reliability and behavior of application.

In the past, it had been observed by various work on libresocial (Most recently during Florian Schulte [9] and Dheeraj Balabhadruni and Weifeng Peng's [14] master thesis) that message losts were critical issue which they faced during the evaluation of LibreSocial. In order to avoid message lost, some modification (increased size of message queue with prioritization)cwas done in the Message Queue of LibreSocial application and results are positive.

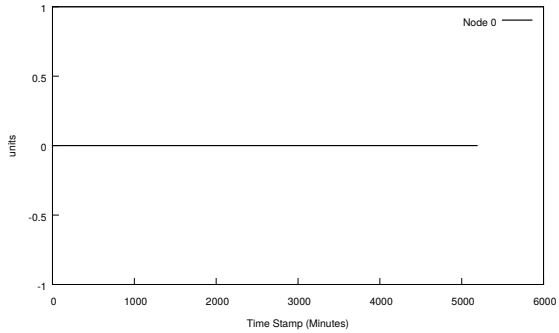


Figure 5.5.: Message Lost Count

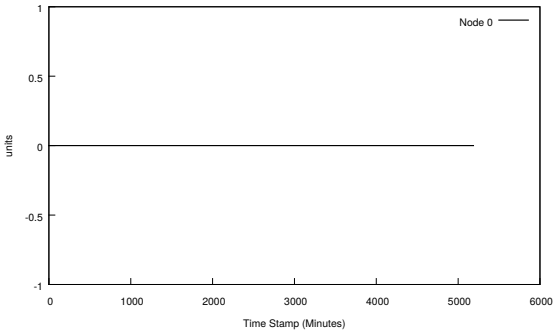


Figure 5.6.: Other Exception Count

Figure 5.5 shows that message count is decreased to zero for the whole period of test. Truthfulness of this result can also be verified from the Firgure 5.12 and Firgure 5.13 that the number of received messages are equal to the number of sent messages.

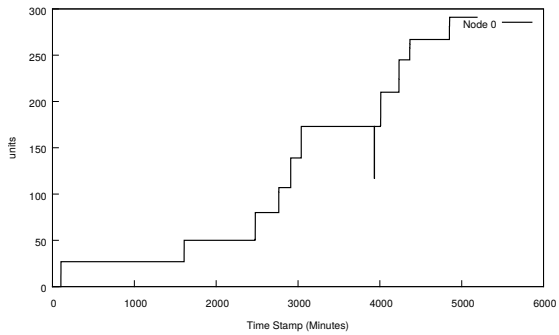


Figure 5.7.: Null Pointer Exception Count

While Figure 5.6 shows that no exceptions occurred during the whole period of test except from the null pointer exceptions which seem to be increasing with the time. It is really important to get rid of null pointer exceptions in order to improve the reliability and error handling of LibreSocial.

Data and Message Transfer Statistics

Figure 5.8 and 5.9 provides an overview of the number of bytes that were sent and received by all the nodes in the LibreSocial network. From the figures we can see that the shape of both the diagrams are almost similar as they represents the data of all nodes in the network combined which is all sent data received by another nodes in the network. Sent and Received data directly affected by the nodes presents in the network.

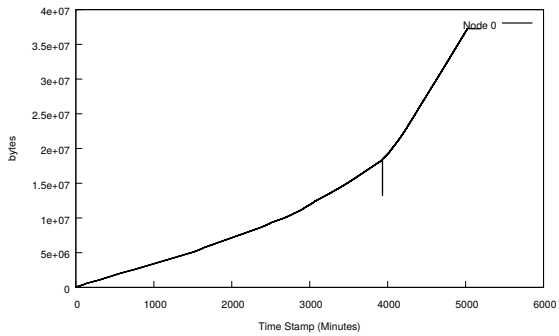


Figure 5.8.: Global Network Data Received

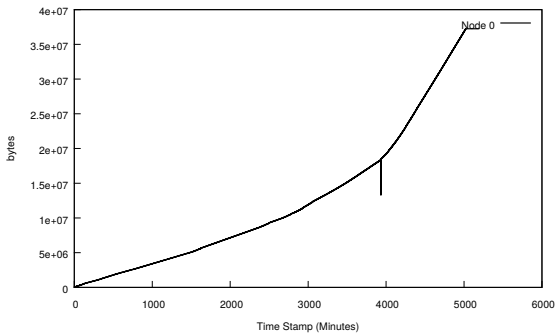


Figure 5.9.: Global Network Data Sent

In figure 5.3 we can see that node count suddenly dropped at 3950 minutes which was reflected in diagrams 5.8 and 5.9 by sudden drop of sent and received data at nearly 3950 minutes. Both the diagrams 5.8 and 5.9 seems to be moronically increasing as it shows no node left the network during the whole period of time except form the time near by 3950.

Since, no message lost occurred during the test figure 5.10 and 5.11 shows equal behavior with exact same sent and receive data rate.

Figure 5.12 and 5.13 shows amount of messages received and send by the nodes in the network. It can be seen from the figure that number of messages sent and received during the tests are directly proportionate to the life time of nodes in the network and shows equal behavior as the diagrams 5.8 and

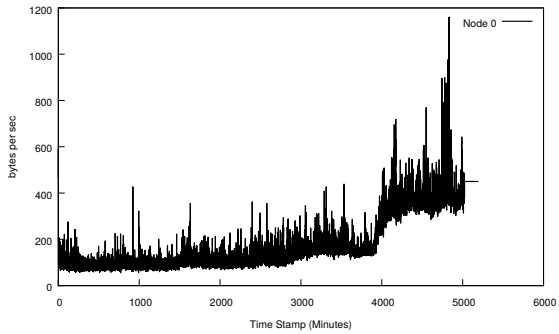


Figure 5.10.: Global Network Data Received Rate

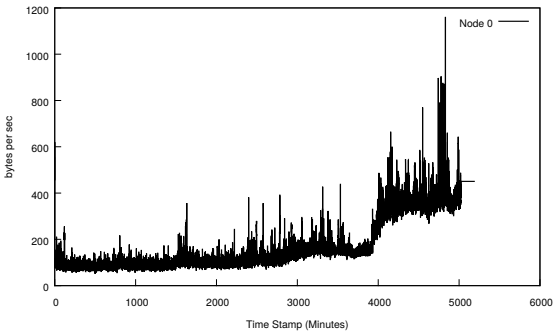


Figure 5.11.: Global Network Data Sent Rate

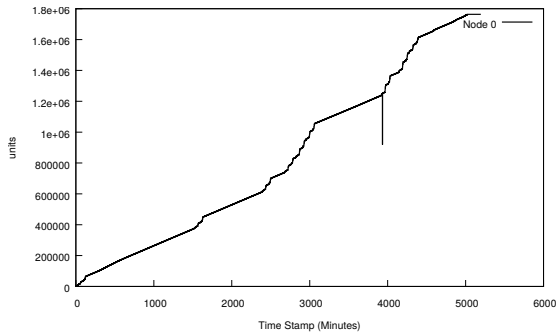


Figure 5.12.: Global Network Message Received

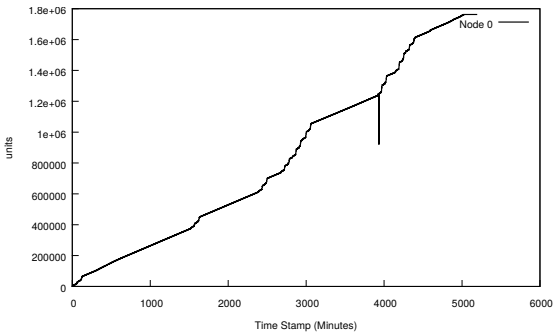


Figure 5.13.: Global Network Message Sent

5.9.

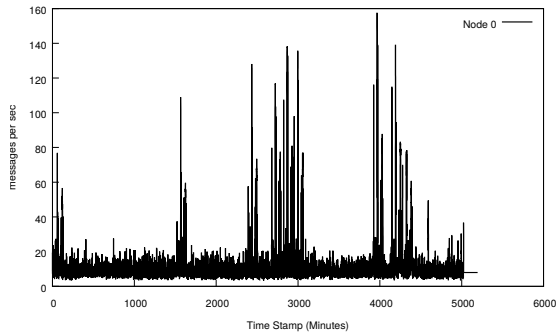


Figure 5.14.: Global Network Message Received Rate

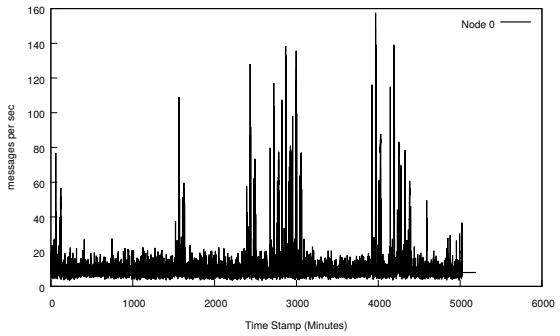


Figure 5.15.: Global Network Message Sent Rate

Figure 5.14 and 5.15 shows message sent rate and message received rate per second by all the nodes in network during the test. Here we can see that the large number of message generation per second during the some of the time intervals. Reflection of such behavior can be observed in the figure 5.12 and 5.13. For example, between 4000 to 46000 minutes number of messages suddenly increased from 1.2e+06 to 1.6e+06 which results into the increase rate of sent and receive message from 20 messages per second to 120 messages per second. The potential reason behind this might be the message queues flooded with huge amount of messages.

Memory Usage Statistics

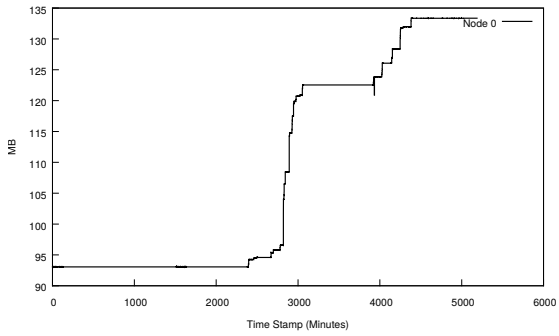


Figure 5.16.: Network-Total Memory

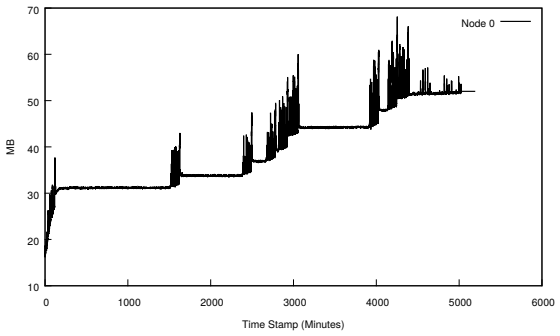


Figure 5.17.: Network-Used Memory

Figure 5.16 and 5.17 gives an overview of actual memory available in the network and memory used by the virtual machines for the nodes in the LibreSocial network. The value sums up the average calculated for all the

participant nodes.
When you observe both the figures carefully, the memory used by nodes in the network is relatively half of the total memory available which is really good in the context of any P2P systems. The main reason behind the consistency of usage of memory is improved monitoring process which now generates SQLite data files instead of traditional CSV data.

Pastry Statistics

Figures 5.18 and 5.19 have a similar behavior as the network message sent and receive statistics. But after careful evaluation of pastry data read amount and pastry data sent amount, we can see that the data sent amount is relatively higher than the data read amount with the similar behavior as graph approaches the pick. Potential reason behind this could be congested message queues which also which lead to drop in the amount of data read.

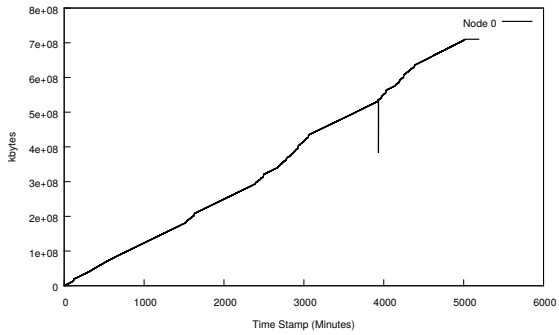


Figure 5.18.: Data Data Read Count

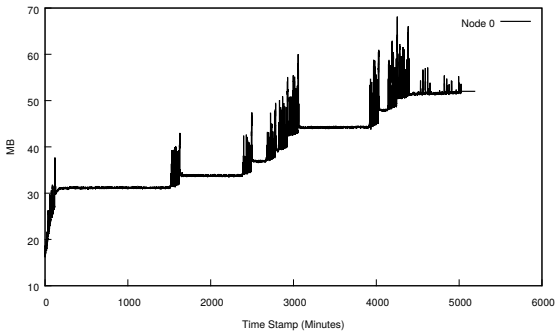


Figure 5.19.: Pastry Data Sent Count

Figures 5.20 and 5.21 are showing pastry data read rate and sent rate respectively where at some point sent rate exceeds the read rate due to same reason as before in the case of Figures 5.18 and 5.19 .

Replication Statistics

Figure 5.22 shows amount of replication which takes place with respect to time. Replication hugely depends on the intensity input test script. Here we have used input test script with an aim to generate high load in the network so it is obvious to see huge increment in replication count but at some point in time it needs be controlled in order to save the memory and space.

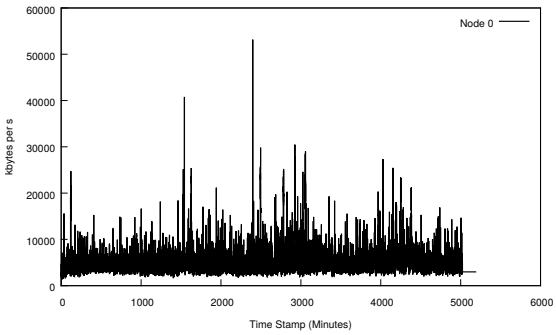


Figure 5.20.: Data Data Read Rate

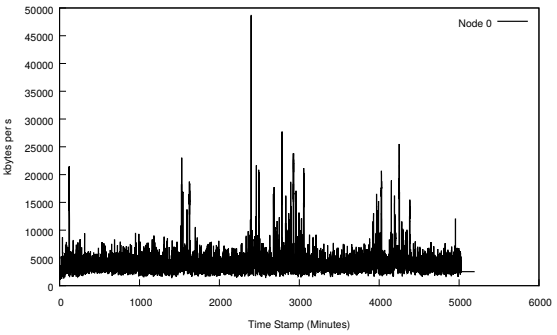


Figure 5.21.: Pastry Data Sent Rate

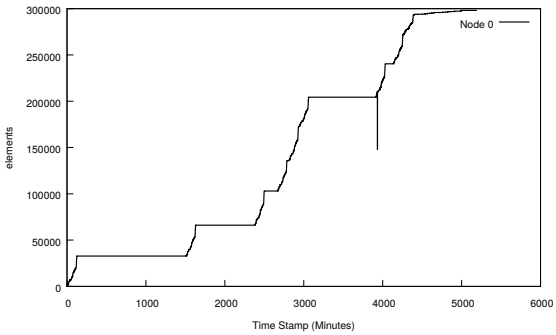


Figure 5.22.: Replication amount

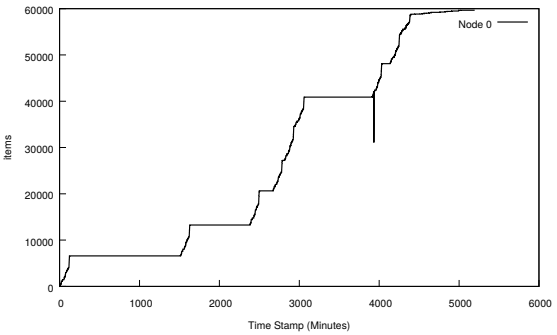


Figure 5.23.: Storagedispatcher Stored Items Count

Storagedispatcher Statistics

Figure 5.23 shows the number of data items that are stored by the storage dispatcher. The storage dispatcher directly associated with framework which stores all items that are persisted by the framework or the associated plugins. In contrast to Florian's master thesis [9] the storage dispatcher does not show decreasing behavior except from the point where number of nodes in the network dropped all of a sudden.

5.5.2. Outcome Of First Round

The test was able to run for at least 5 days which is approximately 5520 minutes. During the test all the functionality were repeatedly tested by using a test script which was also able to generate noticeable load and traffic in the LibreSocial network. Since, main aim of the test was to observe the behavior and performance of LibreSocial network and nodes belong to those network we were able to produce some good results. Main critical issues were related to handle overflowed message queues, message dropped and message lost due to the inability of nodes to receive and send messages. This issues were handled quite good as we found zero messages were lost during 5520 minutes and network was able to send and receive messages, though there was some time when queues were overflowed but it was handled as the size of queue was increased compare to tests conducted in the past.

Despite of produced good results, we noticed many other issues which need to be solved on a high priority basis. Such issues include Fault tolerance of nodes, replication of stored items and issues related to message queues.

Test had to be stopped because some nodes became faulty and becomes dead right after. This phenomenon lead to the crash of whole network. This behavior of LibreSocial was pretty strange as it supposed to handle dead nodes and new arrival of nodes. This needs to be investigated. Detailed analysis of problems, reason behind the problems and solutions to overcome such problems are given in the last section.

5.5.3. Second Round Results Evaluation

Motivation behind conducting the second round of test was to cross check the the results produces during the first round of long term test. Pleas note that test script used during second test was rather less intensed comapre to

first test.

Node Count and Jointing time

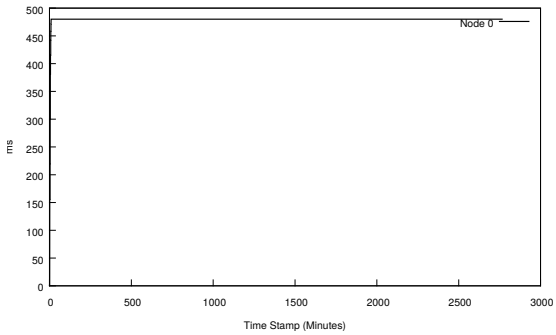


Figure 5.24.: Node count

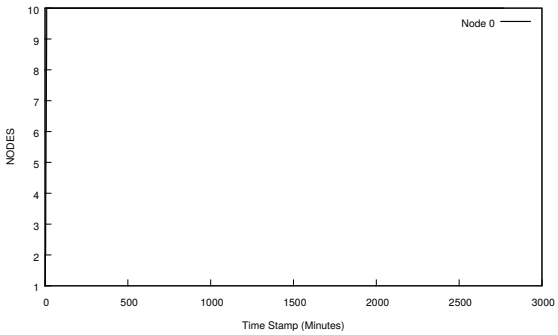


Figure 5.25.: Node joining time

Figure 5.24 and 5.25 shows the number of active nodes and joining time of the nodes in the notwork respectively. It is obvious from the figure 5.24 that all the nodes joined network within first 10 to 15 minutes of time and after that all the nodes were active throughout the whole duration of long term test. while, Figure 5.25 provides detailed overview of average jointing time of each node in network. From the figure it can be seen that the average joining time of all ten nodes was approximately 480 ms and it was constant throughout the whole period. When you compare the results produced during first long term test with the results of second long term test, you see that the results produced later were rather more positive as there was no deviation noticed in the graphs related to node counts and node joining time.

Exceptions And Error Handling

Figure 5.26 and 5.27 are showing the results related other exceptions and null pointer exceptions which occurred during the testing.

Compare to first long term test, we have observed 2 more other exceptions and increase in the count of null pointer exception. It is not possible to observe which particular exception other than the null pointer occurred during the test as the duration of test was really long and we do not have monitoring metrics for that. It is really hard to produced add new metrics just to observe particular exception type as there are hundreds of exceptions which may occur depending on the functionality and time. Moreover, it also

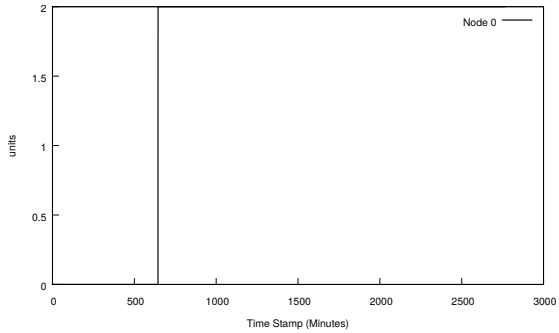


Figure 5.26.: Other Exception Count

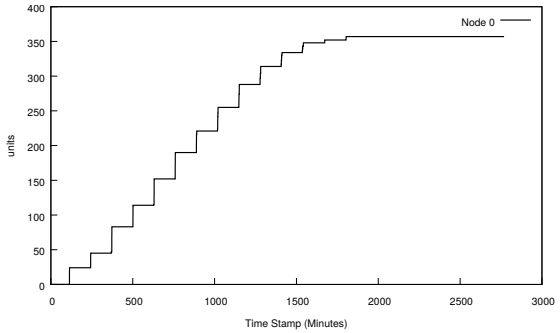


Figure 5.27.: Null Pointer Exception Count

depends on the selected test scripts.

Data and Message Transfer Statistics

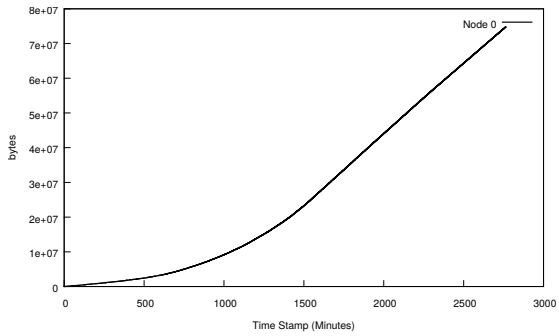


Figure 5.28.: Received Data

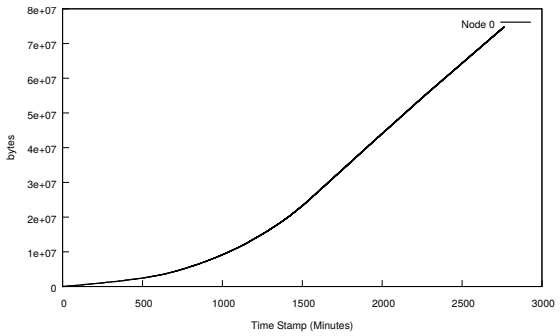


Figure 5.29.: Sent Data

Figure 5.28 and 5.29 shows the size of data in bytes received and sent by all the nodes in the network. By careful observation we can assume that the amount of data sent and received by each node in the network is same but in reality it might be different. In the graphs we can not notice the difference between sent and receive data as the size of data is in the range of ten to the power of 7 so it is hard to notice if very small amount of data is getting lost. This difference can be seen when we analyze data sent rate and data received rate.

Figure 5.30 and 5.31 shows receive and sent data rate of each node in the network. From the figures it can be seen that at some point in time the receive data rate gets lower than the sent data rate. This behavior shows the loss of input data. On the other hand not all data that was sent must be

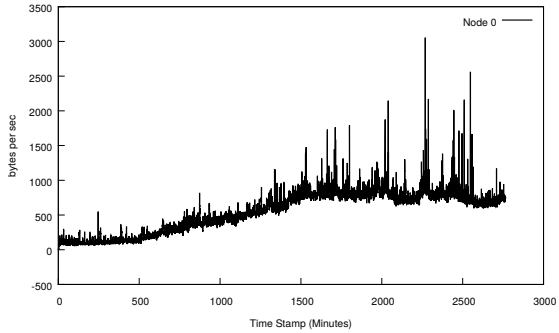


Figure 5.30.: Received Data Rate

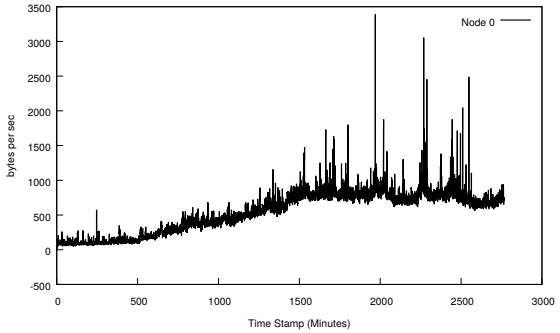


Figure 5.31.: Sent Data Rate

necessarily received by another node. As of now the only potential reason we could think of is the message lost. Since, we could not find any message lost in the network there are possibility of other reasons due to which the data is getting lost. These reasons must be found out and investigated.

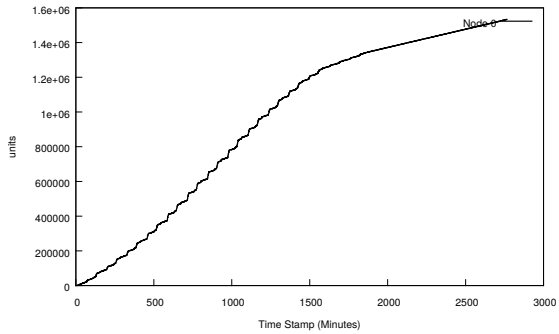


Figure 5.32.: Received Messages

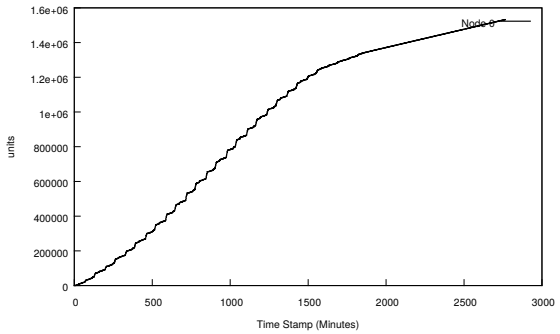


Figure 5.33.: Sent Messages

While, figure 5.32 and 5.33 explains the number of messages sent and received by each node in the network. Number of received and sent massages are equal which proves the correctness of figure 5.34 which represents the number of messages lost in the network during the process of sent and received messages by each node in the network.

Figure 5.35 and 5.36 represents the rate at which messages were sent and received per second. Observation shows that both diagrams are identical and shows the normal behavior.

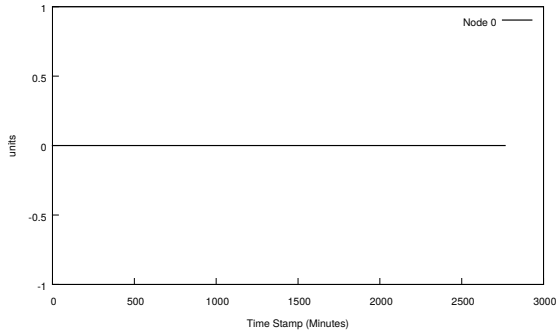


Figure 5.34.: Lost Message

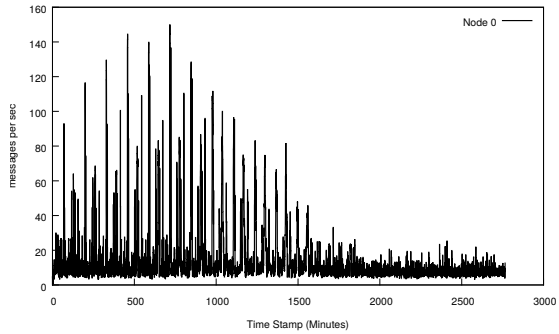


Figure 5.35.: Received Message Rate

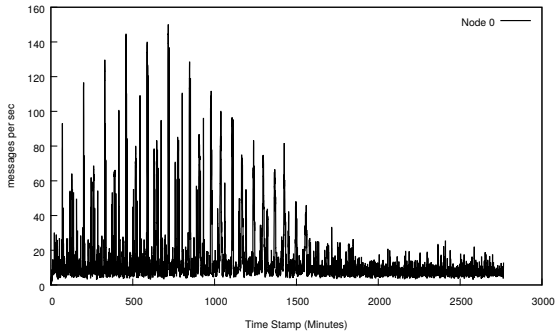


Figure 5.36.: Sent Message Rate

Other Evaluation Results

Statistical results of second test like memory usage, replication count, pastry data read and sent rate etc... have also shown the identical behavior as first test throughout the life time of test duration considering the only difference was between the duration of each round. Detailed information about statistical results can be found under appendix A which includes all the results of each test round.

5.5.4. Outcome Of Second Round

Second round of test conducted in order to verify the results of first round due to the reason that first round result had shown significant improvement compare to the tests conducted in the past. Due to the time constraint we had to stop second round to evaluate the results on time. Results produced during second round of test significantly conforms the correctness of results produced during the first round. Again, please note that all the improvements (except from the issues related to NAT) to the application were carried out before the beginning of testing phased.

6. Conclusion

Current chapter sums up all the work done as part of this master thesis, summarizes all achievements and observations. It addresses also, some bottlenecks which are still present in LibreSocial and needs to be address in the coming time in order to improve the reliability and stability of LibreSocial application network.

6.1. Achievement

Main goal of this master thesis was to fix all the existing error observed during the making of Florian's master thesis [9] and Dheeraj Balabhadruni and Weifeng Peng's combined master thesis [14] and investigate existing bugs present in the overlay layer of LibreSocial which cases instability of LibreSocial. In order to achieve both the aims GUI based test and Long term test was conducted and all the results were carefully observed.

In the first phase of thesis all the improvements in the code were carried out. List of all the improvements and errors which were fixed can be found under section 2.7. Next phase was started with GUI based test in order to observe front end and back end behavior of LibreSocial in real environment where LibreSocial was operated by real users (users familiar with LibreSocial). It was little surprising to see the results of GUI based test as it was lasted for really small period which was unexpected. Test had to be stopped due to the unresponsiveness of application as it became really slow without causing any critical exceptions after 18 hours of the test. After 19 hours of time, one or two nodes became dead (They were not able to) and message lost exceptions were generated as dead nodes were not able to receive messages anymore. After the careful evaluation of behavior and observation of exceptions we came to conclusion that the unresponsiveness was caused by storage layer especially with the distributed link list. Message lost were occurred due to the node failure. This is where the priority transport layer comes into the picture which uses single TCP sockets to send messages. In order to avoid

node failure or message routing to failed node we must make sure the pasty node handle feature works in correct manner. There are many forms of node failure. The most common one is simply the node going offline because the application which is running the DHT is shut down which seems be an unrealistic in the case of LibreSocial. Second is the change of dynamic IP of a domestic internet connection essentially has a subtly different effect as it invalidated all existing routing table entries about it but the over all node count does not go down. So basically you lose one node and you get new node but you try to send the message to the old node which we already lost and new node stays idle. Last reason is most common which is readability problem due to NATs. For this purpose we have a strong doubt on Transport layer used by FreePatry which contains node handle technique. Node failure also cause by low bandwidth due to which a node becomes really slow and it loses ability to answer the ping for liveliness check. Improved NATs techniques can efficiently solve this problem.

On the other hand we were able to produce good results using Long term test and results were different from the results of GUI based test considering the fact that we had front end disable during the long term test and we only observed the performance of beck end. Overall, the long term test was lasted for more than 4 days with no message lost and we were able keep all the nodes live in the network throughout the duration of whole test period. We were also able to observe that message processing can be handled efficiently by improving message queue size in the priority transport layer and effect of improved monitoring as memory consumption was really lower compare to the tests conducted in the past. Long term test had to be stopped due to the crash of ring resulted from one or more node failure. Despite of no message lost and lower memory consumption, Long term test was able to uncover some problems related to handling of dead nodes. same problems were also faced during GUI based test.

All uncovered bugs and reason for instability of LibreSocial are explained in next section and they should be given more attention on high priority bases.

6.2. Future Woke and Suggestions

This section describe bugs uncover during testing LibreSocial which needs to be solved on priority basis.

6.2.1. Length And Size Of Message queue

Since message queue in *PriorityTransportLayer* has to deal with big amount of messages, it some times gets overflowed and can not process lower priority messages as it has to discard all the lower priority messages in order to process high priority messages. In the process lower priority messages gets lost resulting into message lost. It is important to increase the size and number of such message queue in order to make message transmit procedure more effective.

6.2.2. Node handling

Message lost also caused by dead nodes which can note receive messages directed to them and causes message lost exceptions. Potential reasons behind this problems are NATs and DHT. NAT causes unreliability of node in various conditions which needs to be investigated. In order to provide access to firewall users, we must implement advanced NAT traversal.

We also argue that the DHT used in FreePastry suffers from the problem that it can not handle the churn efficiently because of continuous and rapid process of node arrival and departure. Performance of DHT can also be affected from different factors [18]. Such factors include: reactive versus periodic recovery from node failure, calculation of message time out during look ups, choice of nearby over distant neighbors. Because DHT should perform look ups rapidly and consistently under the churn rates at least as high as those observed during the evaluation results. Priority transport layer routes message to the destination nodes based on their entries in DHT. Message can not be deliver if DHT contains false node entry. We can also back this argument by comparing the results of long term tests during Dheeraj and Weifeng's thesis [14] and the long term test during current master thesis.

Figure 6.1 and 6.2 are showing results of node count and message lost produced during the long term test of Dheeraj and Weifeng's thesis. From the figure it can be seen that the node count and message lost showing same behavior in the period between 3800 minutes to 5200 minutes. For the first 1200 minutes DHT was able to perform looks ups and it was able to cope with speed of continuous node arrival and departure and message lost resulted was zero as DHT contained all the entries of newly arrived and departed node. But in the later phase (between 3800 to 5200 minutes) ratio of node arrival and departure became really high and DHT could not cope with the rapid arrival and departure of nodes which resulted into higher message lost as most of the messages were routed destination nodes using false node

id. On the other hand, we were manged to control node count which was pretty much constant (figure 5.3 and 5.5) throughout the whole duration and as result we had zero message lost as DHT was able to perform the look ups efficiently.

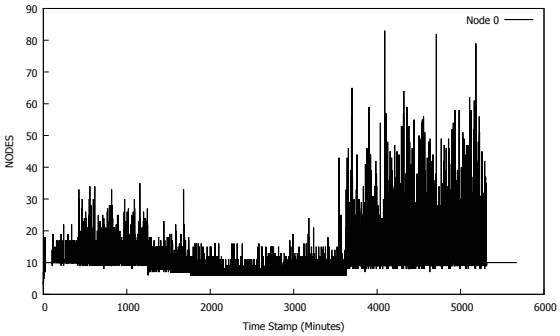


Figure 6.1.: Node count in long term test [Dheeraj and Weifeng] [14]

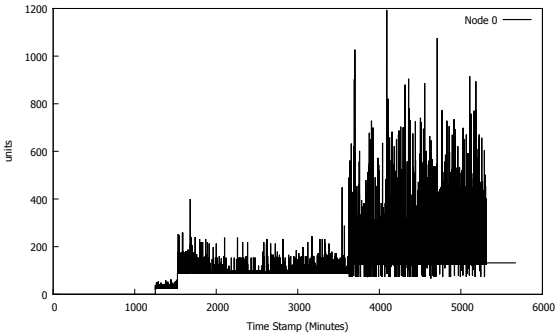


Figure 6.2.: Message lost in long term test [Dheeraj and Weifeng][14]

6.2.3. Error handling

LibreSocial is still vulnerable to unexpected user actions and produces lots of exceptions when used in realistic environment. Error handling should also be improved in order to make LibreSocial robust when used by real users.

A. Appendix

A.1. Long Term Test Results: Test 1

Part of digital version of thesis paper (PDF) as numbers of statistical results produced during test are really huge.

A.2. Long Term Test Results: Test 2

Part of digital version of thesis paper (PDF) as numbers of statistical results produced during test are really huge.

List of Figures

2.1. Peer-to-Peer Network 6
2.2. LibreSocial:Architecture [6] 8
2.3. LibreSocial:Overlay Network [6] 10
2.4. Distributed Linked List [6] 14
2.5. Replication work flow in LibreSocial [6] 15
2.6. Message routing in SkyEye.KOM [6] 16
2.7. SkyEye.KOM Tree topology [6] 17
2.8. Plugins in LibreSocial [6] 19
2.9. WebProvide plugin interface [6] 24

3.1. Parameters in Config.txt 29
3.2. Test script in LibreSocial example 31
3.3. LibreSocial parameters (1) 40
3.4. LibreSocial parameters (2) 41

4.1. LibreSocial:Wall 43
4.2. LibreSocial:Chat 44
4.3. LibreSocial:Photo 45

5.1. Xstartup.sh 48
5.2. GUI-Virtual machines-Graffi-1 and Graffi-2 in VNC viewer 49
5.3. Global Node count 54
5.4. Node joining time 54
5.5. Message Lost Count 55
5.6. Other Exception Count 55
5.7. Null Pointer Exception Count 55
5.8. Global Network Data Received 56
5.9. Global Network Data Sent 56
5.10. Global Network Data Received Rate 57
5.11. Global Network Data Sent Rate 57
5.12. Global Network Message Received 57
5.13. Global Network Message Sent 57
5.14. Global Network Message Received Rate 58

5.15 Global Network Message Sent Rate . . . 58
5.16 Network-Total Memory . . . 58
5.17 Network-Used Memory . . . 58
5.18 Data Data Read Count . . . 59
5.19 Pastry Data Sent Count . . . 59
5.20 Data Data Read Rate . . . 60
5.21 Pastry Data Sent Rate . . . 60
5.22 Replication amount . . . 60
5.23 Storagedispatcher Stored Items Count . . . 60
5.24 Node count . . . 62
5.25 Node joining time . . . 62
5.26 Other Exception Count . . . 63
5.27 Null Pointer Exception Count . . . 63
5.28 Received Data . . . 63
5.29 Sent Data . . . 63
5.30 Received Data Rate . . . 64
5.31 Sent Data Rate . . . 64
5.32 Received Messages . . . 64
5.33 Sent Messages . . . 64
5.34 Lost Message . . . 65
5.35 Received Message Rate . . . 65
5.36 Sent Message Rate . . . 65

6.1. Node count in long term test [Dheeraj and Weifeng] [14] . . . 69
6.2. Message lost in long term test [Dheeraj and Weifeng][14] . . . 69

List of Tables

2.1. List of reported bugs and status 26

3.1. List of Available and Valid Test Cases in LibreSocial [14] 39

5.1. Test Script Used in Long Term Test 53

Bibliography

[1] Dejan S Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing, 2002.

[2] Richard S. Hall Karl Pauls Stuart McCulloch David Savage. Osgi in action, 2011.

[3] Eclipse. https://eclipse.org. Accessed: 2015-11-15.

[4] Freepastry. http://www.freepastry.org, 2009. Accessed: 2015-11-15.

[5] Libresocial. https://libresoical.com. Accessed: 2015-11-15.

[6] Christy Jovita Bibiana Arockiaraj Kailaash Balachandran Navendu Barua Bing Gao Eshant Tulsidas Gour Christopher Holomek Thomas Jacobs Siddharth Kumar Bhagvanjibhai Kaneriya Maitri Modasia Weifeng Peng Chander Prakash Manasa Veena Rapolu Florian Schulte Rohan Sukumaran. Final report - a peer-to-peer framework for social networks. Project group, Uni-Paderborn, 2013.

[7] Antony Rowstron and Peter Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 188–201. ACM, 2001.

[8] K. Graffi, A. Kovacevic, Song Xiao, and R. Steinmetz. Skyeye.kom: An information management over-overlay for getting the oracle view on structured p2p systems. In *Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on*, pages 279–286, Dec 2008.

[9] Florian Schulte. Public testing of a peer-to-peer framework based social network application. Master thesis, Uni-Paderborn, 2015.

[10] Webrtc. www.webrtc.org/reference. Accessed: 2015-11-15.

[11] Jetty web server. eclipse.org/jetty/. Accessed: 2015-11-15.

[12] Istqb: What is software testing? istqbexamcertification.com/what-is-a-software-testing/. Accessed: 2015-11-15.

[13] Eduardo Cunha De Almeida, Gerson Sunyé, Yves Le Traon, and Patrick Valduriez. Testing peer-to-peer systems. *Empirical Software Engineering*, 15(4):346–379, 2010.

[14] Dheeraj Balabhadruni Weifeng Peng. Systematic evaluation of a decentralized social network. Master thesis, Uni-Paderborn, 2014.

[15] Pascal Kallien. Systematic evaluation of a peer-to-peer framework in the use case of a social network. Master thesis, Uni-Paderborn, 2013.

[16] Ssh portforwarding. https://help.ubuntu.com/community/SSH/OpenSSH/PortForwarding, 2014.

[17] Virtual network computing. http://en.wikipedia.org/wiki/Virtual_Network_Computing, 2014.

[18] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a dht. In *Proceedings of the USENIX Annual Technical Conference*, pages 127–140. Boston, MA, USA, 2004.

[19] opensuse operating system. https://www.opensuse.org/en/, 2014.