Systematic Evaluation of LibreSocial - a Peer-to-Peer Framework for Online Social Networks

Newton Masinde[1,*], Liat Khitman[1], Iakov Dlikman[1], Kalman Graffi[1]

Technology of Social Networks
Heinrich Heine University
Universitätsstrasse 1, 40225
Düsseldorf, Germany

Abstract

Peer-to-peer (P2P) networks have been under investigation for several years now, with many novel mechanisms proposed as is evidenced by articles available. Much of the research has focused on showing how the proposed mechanism affects the system performance. In addition, several applications have been proposed to harness the benefits of the P2P networks. Of these applications, online social networks (OSNs) have raised much interest particularly because of the scalability and privacy concerns with centralized OSNs, hence several proposals are in existence. However, accompanying studies on the overall performance of the P2P network under the weight of the OSN applications outside simulations are very few, if any. In this paper, the aim is to undertake a systematic evaluation of the performance of a P2P framework for online social networks called LibreSocial. Benchmark tests were designed, taking into account the random behavior of users, effects of churn on system stability and effect of replication factor. We manage to run benchmark tests for upto to 2000 nodes and show the performance against costs of the system in general. From the results it is evident that LibreSocial provides a performance that is up to par with current user needs.

Keywords: Distributed systems, Peer-to-peer Networks, Social Networks, Framework, Performance Evaluation

1. Introduction

Social networking has experienced tremendous growth since the turn of the 21st century, a fact evidenced by number of online social networks (OSNs) available. In addition, studies have shown that there is a general overlap between the online and offline networks of many of these users [1]. As a consequence of the growth of these platforms, there have been some concerns raised. These concerns can be considered on two fronts, technical and social [2]. The technical concerns arise because of the high dependence on the use of centralized management for the OSNs, and with a rapidly growing user base various scalability performance issues and hence increasing cost of management and maintenance of the overall system infrastructure are the obvious consequence. The social concerns have to do with the users' privacy and data ownership, and in many respects, can be looked at as a direct result of the first concern. As the networks have grown, the providers, who own all the data uploaded, have developed a new market, called personal data markets [3], in which the product on sale is the user's private data. Although self disclosure is a hallmark of activities over the Internet, the nature of OSNs has increased the level of online disclosure as users exchange data and reveal information about themselves [4]. The private data is sold to third parties who utilize it for personalized advertising [5], increasing brand awareness [6], emotional manipulation [7] or even online activity surveillance [8]. Even though most of the OSNs have included privacy settings to allow so that users have control of how much data they reveal about themselves, in many cases they end up revealing much more by using these privacy settings than with fewer setting [9].

To tackle the aforementioned concerns, it has been suggested to move from centrally managed solutions to decentralized solutions with two main directions observed here: web-based and P2P-based solutions [10]. Web-based solutions, such as Diaspora, however have a heavy reliance on a distributed web server infrastructure requiring in-depth knowledge of web-server systems and administration to setup the OSN hence only expert users can effectively use them. In addition, running a private web server for thousands of users also raises the monetization pressure. On the other hand, P2P-based solutions are easy to setup, requiring only an application to be downloaded and ran so as to join the P2P OSN community. Therefore, they are suitable to regular internet users. In any case, no monetization pressure arises with P2P OSNs. Also since the

*Corresponding author
URL: Newton.Masinde@hhu.de (Newton Masinde), Liat.Khitman@hhu.de (Liat Khitman), Iakov.Dlikman@hhu.de (Iakov Dlikman), Graffi@hhu.de (Kalman Graffi)

current OSNs are Web 2.0 applications and are, at a logical level, inherently P2P in nature [11], it is more sensible to couple the logical level with an actual physical level that is P2P in nature.

Peer-to-Peer (P2P) technology has been in existence for quite some time, with a notable upsurge in research in this field at the turn of the millennium. P2P systems distinguish themselves from other computing paradigms by the fact that participating units, referred to as peers, exhibit equal rights within the system, regardless of their computing capacities. Simply put, a P2P network or system is a virtual, self-organized network formed over the existing physical communication network through introduction of specialized protocols that enable the heterogeneous nodes to autonomously interact with each other and share resources. P2P networks are characterized as having high degree of decentralization, self-organization, multiple administrative domains, low barrier to deployment, organic growth, resilience to faults and attacks and abundance and diversity of resources [12, 13]. However, the design and implementation of P2P OSNs is not a trivial matter, requiring considerations into key P2P aspects such as the topology design, distributed data storage, update propagation and versioning, search and addressing, robustness under churn and security [11]. The research on P2P OSNs has given rise to many proposals such as PeerSoN [14], Safebook [15], SuperNova [16], DECENT [17], LotusNet [18] and DiDuSoNet [19], each using different techniques to solve these design challenges. For most of the proposed solutions, the observed focus reported has been on showing how the system achieves a particular function vis-à-vis a particular concern, such as security, privacy, storage, replication and so on. Therefore, there is little, if any literature on the quality of overall performance of these OSNs.

In this paper, we aim at presenting a systematic evaluation of the performance of one of the more advanced P2P OSN solutions, LibreSocial (previously called LifeSocial.KOM [20, 21, 22]). In specific, we show the system's expected degree of availability, scalability and robustness. The rest of the paper is arranged as follows. Section 2 introduces the benchmarking as an experimental methodology that is used for P2P systems and also describes the system quality properties that we aim to investigate. In Section 3, we describe the make up of LibreSocial, our P2P OSN framework solution, briefly. Thereafter, Section 4 introduces the metrics that will be employed in evaluating the system, and further describes the test setup, including the test scenarios and the different workload specifications. Section 5 shows the results obtained and clearly giving an interpretation for them. Finally, in Section 6, we give the concluding remarks.

2. Benchmarking P2P Systems

As an experimental methodology to evaluate a system's performance, *benchmarking* uses a synthetic/generic application on an existing, real environment [23], and relevant performance metrics are chosen based on the application's domain. Benchmarking is normally standardized within a given domain of operation, and the results in such experiments are easily obtainable and comparable. The disadvantage of this approach is that it does not fully represent a realistic situation. We now briefly focus on the key aspects of the P2P benchmarking process and thereafter highlight the P2P quality properties with respective metrics that are of concern to this work.

2.1. P2P benchmarking model

In contrast to other standard benchmarks used for other computing systems, P2P-benchmarks need to define important aspects of the underlying network so that they are reproducible [24]. Benchmarking can either be vertical or horizontal. In *vertical benchmarking* only one system is evaluated to identify its boundaries of operation. For *horizontal benchmarking*, several similar systems are evaluated against each other. Our focus is on the former, meaning that no other alternative implementations are tested. Any benchmark needs to satisfy a set of basic requirements, that is, it must: *i*) be based on workloads representative of real-world applications, *ii*) exercise all critical services provided by platforms, *iii*) not be tuned/optimized for any specific product, *iv*) generate reproducible results, and *v*) not exhibit any inherent scalability limitations [25, 26]. So as to meet these requirements, Saller et al. [27] define three important benchmark parameters, system, workload and environment, which interact with the system under test as illustrated in Figure 1.

The *system parameters* are system specific and bound the system under test, such as the size of the routing table or replication factor. The *workload parameters* are the parameters that affect the workload generation such as the number of peers and also include other application-specific settings such as number of queries for a given activity. *Environment parameters* are bound up in the host and the underlying communication links. In the test process, not all parameters are of concern. Therefore, only a selected subset of parameters may be varied so as to meet a particular need. This selected subset of parameters are termed *factors* and the most preferred factors are those that have the largest impact on the systems performance [28]. By altering the parameters appropriate *results* are obtained. These results are usually a collection of metrics which can then be used to measure the quality of the system. Therefore, it is important to define suitable quality properties that can be used along with the metrics collected. Next, we give a description of useful quality properties.

2.2. P2P quality properties & relevant metrics

The test of the benchmark design is determined by a clearly defined set of quality properties. *Quality metrics* only focus in describing a single attribute of a mechanism within a scenario, workload or configuration, while *quality*

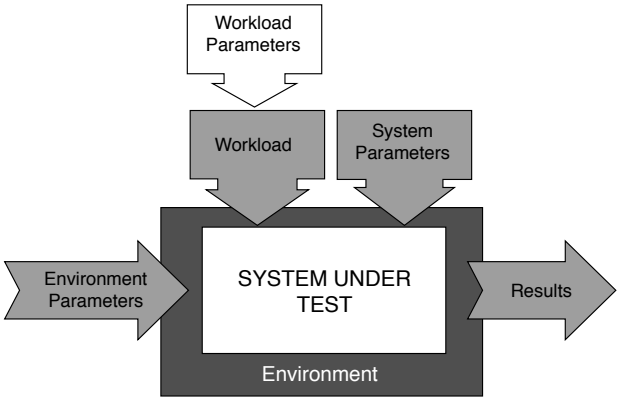


Figure 1: Benchmarking model

Table 1: Quality aspects with relevant metrics

Quality Aspect	Relevant Metrics
Performance	Hop-count Data storage time Data retrieval time Message sending rate Message receiving rate
Cost	Used space Used memory Network bandwidth
Stability/Scalability	Number of nodes Leafset size Routing table size memory size Messages sent Message sending rate Data transferred Data transfer rate Data items stored Number of replicas DDS data stored

properties describe the system's/mechanism's characteristics in consideration of various individual measurements of quality metrics. Quality properties are distinguished into *workload-independent* and *workload-dependent* [27], which are discussed next and the relevant quality properties with their respective metrics are given in Table 1.

2.2.1. Workload-independent quality properties

These are easily adopted from the computer system performance analysis using metrics that represent the system behavior under workload. These quality properties are obtainable through direct measurements or indirectly calculated using other quality aspects. In our test application, we are concerned with performance, cost, stability and scalability.

Performance. To obtain a feel of the performance of the system, points to consider are responsiveness (how fast the system reacts), throughput (how much load that system can handle in a given time frame) and the extent to which the results match the expectation. The metrics chosen for performance measurements are *hop-count*, *storage time* (t_{store}), *retrieval time* (t_{retr}), *message sending rate* (m_{send}) and *message receiving rate* (m_{rec}).

Cost. This property describes the amount of resources that are utilized to ensure a given task is fulfilled or a service is provided. Relevant cost metrics are *used storage space*, *used memory* and *network bandwidth*.

2.2.2. Workload-dependent quality properties

These have a relation to the manner in which the workload is introduced into the system. These properties therefore help us understand and define the workload for the system to be tested and are evaluated using the workload-independent quality properties. In order to show the efficacy of using these properties, there is need to conduct a baseline evaluation on the system to bring a clear comparison. The properties which we focus on are dependability and scalability.

Stability. This is the ability of the system to continue to perform despite inherent system behavior dynamics, such as churn within the system, and should eventually converge to a stable state if workload remains the same. Therefore, stability correlates to resilience under adverse conditions, that is, sudden changes that may occur in the system architecture due to the workload. These changes may be *expected* as the defined protocols account for them, or *unexpected* because the protocols do not consider them. To ascertain stability level, the system needs to be evaluated against a baseline to show the relative differences of a particular parameter from the stable, baselined system. The relevant metrics are *number of nodes* in the network at any given time, the *leafset size*, *routing table size*, *memory size*, *number of messages sent* and the *messaging rate*; the *amount of data transferred* and the *data transfer rate*; the *data items stored* and the *number of replicas*; and the *DDS data stored*.

Scalability. Revolves around the systems ability to handle changing workload. Two scaling dimensions are considered: *horizontal* which affects number of peers in the system, that is, increase and decrease in the number of peers in the network, and *vertical* which concerns the ability of the P2P system to handle increasing workload from the participating peers. For this study, the focus was on horizontal scaling. The relevant metrics that will be looked at are the same as in the case of stability.

Summary. In this section, we introduced key terminologies that will repeatedly be used throughout this paper. We have also briefly described the factors to consider in the creation of a P2P benchmark and finally described important quality properties with the relevant metrics, which are summarized in Table 1, where necessary, that are used

in the evaluation. In the next section, Section 3, we introduce the P2P framework that we have developed in previous years, aimed at meeting the necessary service requirements for a functional OSN, briefly explaining all the key components of the framework.

3. A P2P Framework for Online Social Networks

The process of designing and eventually building a P2P-based OSN has to consider the combination of various, reliable and secure functions that are implemented on top of unreliable and unsecure devices, such as a robust overlay connecting the participants, user management, reliable distributed data storage, access control and secure communication. The users of the system will usually expect a variety of elements, such as messaging walls, support for photo albums, messaging and chatting as well as audio-visual conversational support. To ensure controlled quality, these services require further supporting elements such as distributed data structures, communication principles (publish-subscribe, unicast and multicast) as well as monitoring mechanisms need to be implemented.

LibreSocial (previously LifeSocial.KOM [20, 21, 22]) is a P2P-based framework that provides all these. The application sits on a P2P framework that has been developed based on the Open Services Gateway Initiative (OSGi) service platform[1]. The OSGi platform allows the various components of the application to be defined as bundles that can easily be loaded at runtime and plugged into the running application at will. The architecture of LibreSocial is made up of four distinct layers: *the P2P overlay*, *the P2P framework*, *plugins and applications* and *the graphical user interface*. These four layers are located on top of the Internet (network) layer as shown in Figure 2. In [29], the architecture is discussed in greater detail, but we nevertheless endeavor to briefly highlight the core functionalities. Thereafter, we give a description of the test environment that is bundled with the application.

3.1. P2P Overlay

The P2P overlay is the lowest layer of the framework, and connects to the network layer on the TCP/IP model. The overlay provides a degree of distribution transparency as provides a level of abstraction that hides the complexities of the physical connections. The overlay supports logarithmic message routing and object location using a heavily modified FreePastry[2], an implementation of Pastry [30], which provides an ID space of size 2^{160}. All higher layers of the framework, depend on the overlay for secure and reliable routing. The next layer above the P2P overlay is the P2P framework.

[1] https://osgi.org/download/r7/osgi.core-7.0.0.pdf
[2] http://www.freepastry.org/FreePastry/

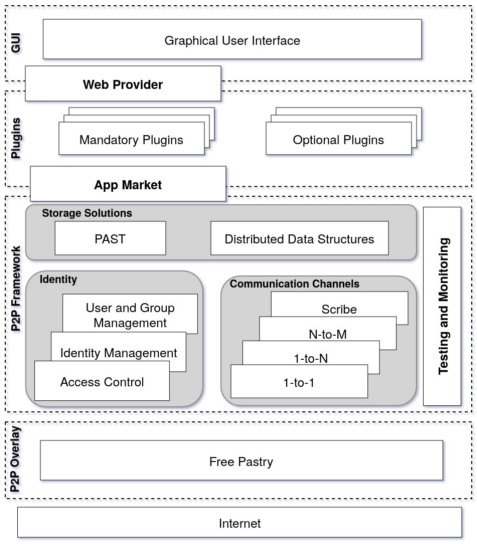


Figure 2: A P2P Social Network architecture

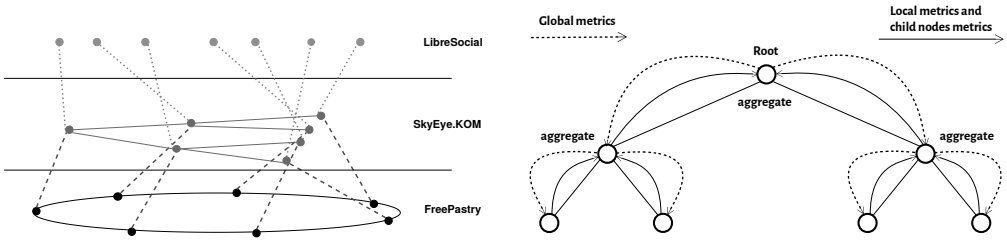
3.2. P2P Framework

The P2P framework provides all the essential services that make up the bulk of the P2P system. There are four core services that the framework offers, that is storage, identity, communication and testing and monitoring which are further discussed.

Storage. Simple file storage is provided via a heavily modified PAST [31, 32], a persistent P2P storage utility bundled along with FreePastry which also includes replication services. In addition, three types of distributed data structures (DDSs), that is, distributed sets, distributed linked lists and prefix hash trees [33] are also utilized for storage. These are useful for complex linked data such as albums with photos having comments or wall messages with comments. The storage service offers an intelligent caching mechanism that takes into account updates.

Communication. The framework supports both synchronous and asynchronous messaging depending on the channel utilized. It also supports unicast (1-to-1) messaging such as in direct messaging, multicast (1-to-N) messaging such as streaming to a group as well as using aggregation (N-to-M) mechanisms for distribution and aggregation of network information. In all cases authentication, confidentiality and integrity of communication is supported. It also allows using of IP-based communication where the communicating node retrieves the IP information of the other node and communicates directly with it. LibreSocial, in addition, includes publish/subscribe mechanism that is push based, streaming via WebRTC[3] for audio/video conferencing and a secure message channel that can be encrypted and signed.

[3] https://webrtc.org/



(a) Overview of the monitoring setup

(b) SkyEye.KOM aggregation

Figure 3: Monitoring setup

Identity. Three key identity features offered are identity management, user & group management and access control. These features required significant modification of FreePastry to ensure secure node identification based on a public key infrastructure mechanism, in this case elliptic curve cryptography (ECC), in which the `nodeID` is the public key. *User management* is made possible by performing a mapping of the `nodeID` to an immutable `userID`. *Access control* is based on AES as the symmetric encryption algorithm and it ensures relevant users are authorized to read/write data and is achieved using symmetric encryption based on the AES standard. The owner of new data generated a symmetric cryptographic key and encrypts the stored data item, encrypts this key with the public key of each user/group who requires read writes. The encrypted data is signed and combined with the public key of the owner as well as the list of encrypted keys to generate the secure storage item that is then stored in the network. Overwriting of the data with a new storage data item if it is signed with the corresponding private key of a specific public key after verification by the storing node.

Testing and monitoring. These two components work together and are used to test the systems performance for quality of service (QoS). The monitoring plugin gathers data via an aggregation channel during the testing and allows the tester, and later the interested user, to get a global view of the network status at any point in time during the testing process. The tree-based aggregation protocol, SkyEye.KOM [34, 35] monitors and gathers relevant measurements during the testing process. The implementation of SkyEye.KOM is independent of the underlying P2P overlay, as seen in Fig. 3a, and it provides the ability to aggregate and disseminate information within the tree structure that is constructed on top of the overlay thus providing a global view global view on the performance and costs of the P2P network. The aggregation process for the monitoring data is shown in Fig. 3b. The aggregation functions include *count*, *min*, *max*, and *mean*.

The services provided via the framework are then utilized by the application, which is made up of the plugin components. Next we describe the plugins which constitute the application.

3.3. Plugins and application

The application components are broken down into plugins which are loaded dynamically at runtime. The plugins include, login, profile, notifications, files, search, friends, group/forum, calendar, messaging, multichat, audio/video chat, wall, photos, voting, testing and monitoring. The plugins or bundles are software components that add a specific feature to the system and enhance the systems capabilities. This is possible because the OSGi design framework makes it possible to develop the different plugins or bundles as modules that are easily extensible and simple to integrate. Each plugin provides an OSGi command interface that can be accessed during testing by the test plugin, which works in connection with the monitoring plugin to support quality monitoring and testing. The plugin contents being availed to the user via the web application.

3.4. Graphical user interface (GUI)

The GUI is the top most layer and it is the point of interaction of the user with the system. Figure 4 shows screenshots of LibreSocial's GUI. The GUI is composed of three sections, that is, the plugin template, the plugin logic and the WebProvider. The plugin template is typically html files and a standard JavaScript implementation based on the Model-View-View-Model (MVVM) paradigm. The plugin logic works at transmitting user events from the front-end to the REST handler via the WebProvider and then transmits the results via the same channel back to the front-end. It also renders the data that it has obtained into the desired template. Lastly, the WebProvider is the interface between the plugins and the user's web browser. Via a desired web browser, the user can easily access all required aspects of OSN at runtime.

Summary. In this section we have introduced our P2P Framework for OSNs, LibreSocial and we have briefly given a description of how different mechanisms are realized so as to offer a secure, scalable and fully decentralized OSN. The Framework constitutes of four layers, the overlay, the framework, the plugins and applications and the graphical user interface, with each offering or receiving necessary services from the adjacent layers to guarantee a working OSN. In the section that follows, we define the metrics and

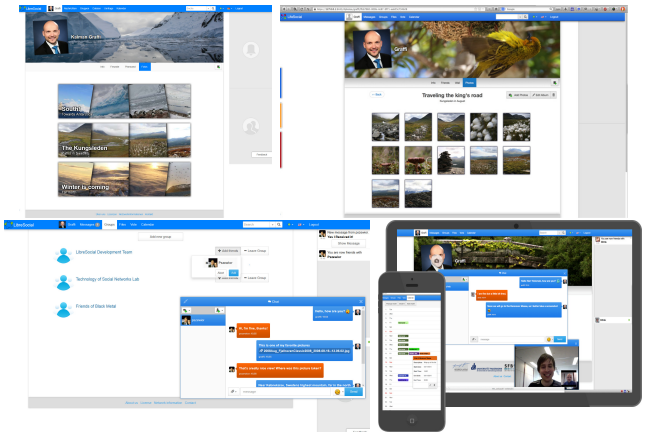


Figure 4: Screenshots of LibreSocial

the describe the experimental scenarios and their respective workloads for the benchmarking process.

4. The Test Environment

A benchmark set can be considered as a tuple made up of the quality attributes Q, metrics M and the test scenarios S [36]. The quality attributes and relevant metrics to be used have been introduced in Section 2.2. In this section, we discuss relevant scenarios designed for evaluating the system. The test environment for all test scenario was the same for result consistency. The tests were deployed to the high performance cluster (HPC) at Heinrich Heine University[4]. The HPC offers the capability to support up to 4000 LibreSocial nodes.

4.1. Workload

The workloads for the benchmark tests are dependent on the particular test that is being carried out. The actual workload is generated using photo images and text messages. Photo images sizes ranged from 100 - 800 kilobytes, with an average of about 300 kilobytes. Messages sent are on average 150 characters in length. Also, for the testing, the photo images are used in place of files/documents. The system parameters for each of the test scenarios is given in Table 2. A description of the different scenarios of the benchmarking test follows.

Pseudo-random behavior. The testing seeks to model realistic randomized user behavior. System parameters defined are given in Table 2a. The random behavior algorithm selects tests in a random pattern as shown in Figure 5 mimicking normal user behavior. It was designed based on the work in [37] wherein aggregated data about the behavior of more than 35,000 users in four different social networks was studied. In the study, key parameters utilized were: the average time spent on a social network website, login frequency throughout the day, activities the user participates in and the order. The statistics gathered made it feasible to implement a mechanism that selects a period of time that a user spends online and at the end of that time period the user logs out. Two types of pseudo random tests were conducted, *realistic user behavior* and *controlled random behavior*. The difference is that for the realistic user behavior, nodes were allowed to login and logout at will as dictated by the behavior algorithm, while in the controlled random behavior, nodes were not allowed to logout during the testing period.

Scalability/Stability. This test scenario, focuses on realizing a network with 1000 nodes to evaluate the system scalability and stability. Table 2b are the system parameters and Table 3 gives the workload for this scenario. The test was conducted to show the effect of churn on the network and proceeds as follows: 1000 nodes join the network and the workload shown in Table 3 is executed after a stabilizing period of about 10 minutes. Thereafter a churn of 250 nodes is introduced in a step-wise manner representing, a 1/4, 1/3 and 1/2 of the nodes respectively) till the network has 250 nodes. At each point, the workload is executed to test the ability of the network reaching a stable state after churn occurs. The tests were conducted in two parts.

(*i*) *Network growth*: The test begins with 250 nodes, which is incremented in steps of 250 nodes representing 100%, 50% and 33.3% increment respectively. At each step, the workload shown in Table 3 is executed.

(*ii*) *Network churn*: Once we reach 1000 nodes, that is, at the end of the network growth phase, joined the network, the churn phase begins. 250 nodes are removed a step-wise manner similar to network growth, representing, a 25%, 33.3% and 50% churn, until the network has only 250 nodes. After each churn, the workload is again executed.

Replication factor. The replication factor (RF) is responsible for the number of duplicate items in the underlying network so in a case that a peer leaves the network, its previously uploaded files will stay accessible to other peers. In this test scenario the replication factor and its effects are considered. The system parameters are presented in Table 2c and the workload parameters which are defined in the test plan are shown in Table 4.

4.2. Test execution

In Figure 6 the setup of the test is portrayed. During testing, we differentiate between the *master* and the *slave* nodes. There can only exist a single master node in the P2P network that is established for testing and must also be the first node that is started. Its purpose is to send test case execution commands to the slave nodes, while the master itself does not execute any test cases. The slave nodes then bootstrap onto the network formed by the master node. To perform any test, there must be at least

[4] https://www.zim.hhu.de/high-performance-computing

Table 2: System Parameters

(a) Pseudo-random test

Parameters	Realistic	Controlled
Leafset size	24	24
Replication factor	6	6
Number of nodes	500	100, 200
Setup time (mins)	160	80
Test duration (mins)	200	120
Enable logout	true	false

(b) Scaling test

Parameters	Value
Leafset size	24
Replication factor	4
Number of nodes	1000
Setup time (hrs)	8
Test duration (mins)	200

(c) Replication test

Parameters	Medium	Large
Leafset size	127	127
Replication factor	4, 16, 32	4, 16
Number of nodes	100, 200	1000, 2000
Test duration (min)	141	141

Table 3: Scalability Workload

Plugin	Action	Repetitions	Duration* *(mins)*
Messaging	Send message	5	2
Photos	Upload photo	5	2
Forum	Comment forum thread	5	2
Wall	Send wall post	5	2

* One minute pause after each test case

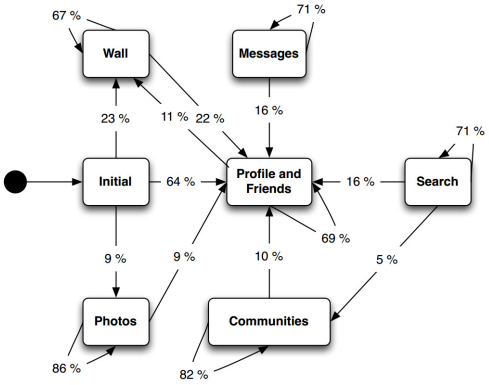
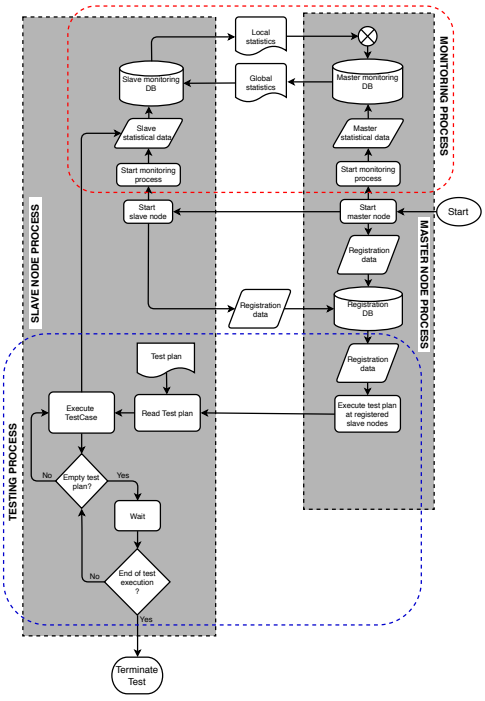


Figure 5: Transition probabilities

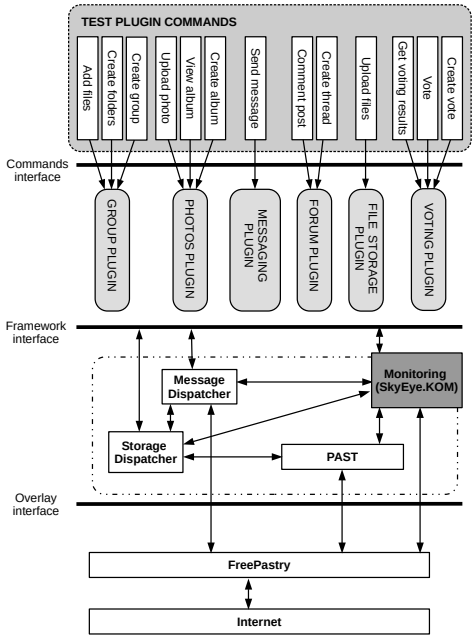
Table 4: Replication Workload

Plugin	Action	Repetitions *(Count)*	Duration *(mins)*
Messsaging	Send message	10	1
	Delete inbox message	10	1
Live chat	Send multichat invitation	5	1
	Send multichat message	10	1
	Leave multichat message	5	1
Group	Create group	5	1
	Invite friend to group	10	1
	View group	5	2
	View my group list	5	1
File storage	Create folder	5	1
	Upload file in folder	10	1
Forum	Create forum thread	10	2
	Comment forum thread	10	1
Photos	Create photo album	5	1
	Upload photo	10	1
	View friend's album	8	1
Voting	Create vote	4	1
	Add public vote	4	1
	Voting invite user	4	1
	Vote	10	1
	Get voting results	10	1
Wall	Send wall post	10	1
	Comment wall post	10	1

* One minute pause after each test case



(a) Flowchart of testplan execution



(b) Configuration at peer level

Figure 6: Test setup

$k+1$ slave nodes on the network, where k is the replication factor chosen for the data items in the network. The test roles then enable the execution of a test plan.

The test plan is a simple text file the has test cases enqueued. A *test case* is defined as the smallest unit that can be executed during a test and corresponds to a single activity execution done by a user such as, sending a friend request, writing on a wall, sending a chat message, uploading a file and so on. For a test case to be executed, in many cases there are certain preconditions that must be met. For example, to send a chat message to someone, the other person must also be a friend. Therefore in executing a test case to send a chat message, the precondition to have a friend must also be fulfilled. Thus each test case has all necessary preconditions defined which must be met or else executed before the actual test case is then executed. The entire process of the test execution with a test plan is shown in Figure 6a. Figure 6b shows how the test plugin interacts with other plugins at peer level by initiating a particular test case for that peer. For each plugin, there are several test cases that can be executed and similarly, respective system metrics that can be collected.

5. Evaluation of Results

The evaluation of the test results that follows is based on the test scenarios that have been previously described.

5.1. Pseudo-random behavior

We consider the realistic user and controlled random behavior tests separately. The evaluation looks at the node count, routing maintenance (leafset and routing table sizes), messaging (count and rate), memory usage size, nework data rate and DDS data retrieval rate.

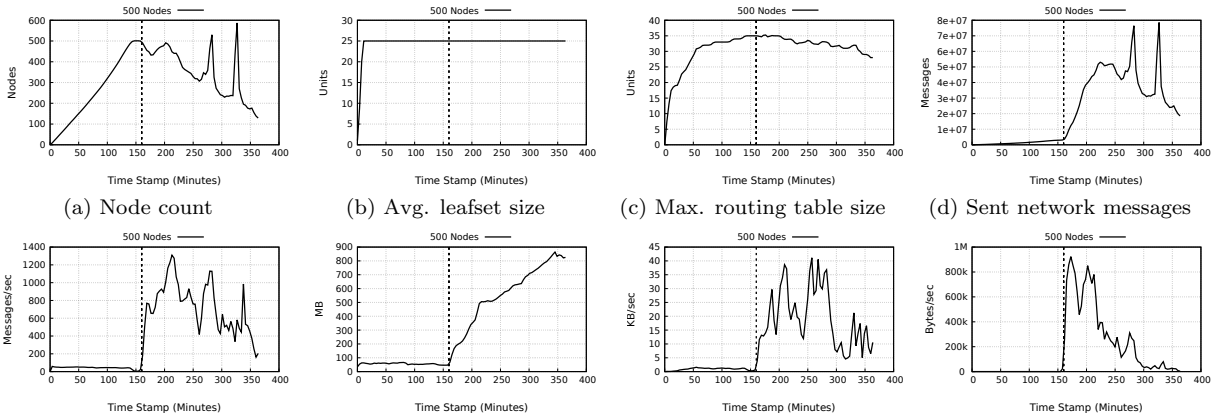
5.1.1. Realistic user behavior

Figure 7 shows the results from this test. Figure 7a portrays the active nodes in the network. There is a notable gradual decline in the network composition over the experimental period. This is directly attributable to two aspects: the randomization algorithm and the absence of new participants joining the network. The leafset size, seen in Figure 7b, is stable at 25 throughout the experimental period, as is expected. On the other hand, the routing table size shown in Figure 7c depicts an increase as the network grows, with the maximum size of 35 at the end of the network joining phase, followed by a decline in that follows the network size. This phenomenon leads to the postulation that there is dynamic route adjustments, and proactive refreshing of the routing table. The network messages and network messaging rate as shown in Figures 7d and 7e respectively are indicative of an active network post network startup (after 160 minutes), with a maximum message rate of about 1300 messages/sec. The number of messages sent and the message rate drop with reduction in network size as was anticipated. One of the requirements for the network is that each node provides a portion of its own memory for the network, which is dynamically adjusted as more memory is needed. Figure 7f shows the maximum memory provided by a single node through the test, which grows steadily to a maximum of about 850 MB at the close of the experiment. At its peak, the sending data rate, seen in Figure 7g, is recorded at about 40 KB/s. In general, the data rate oscillates between 10 and 20 KB/s, which we consider acceptable performance. The maximum retrieval rate for DDSs is recorded at about 900 kB/s at the start of execution of the algorithm, thereafter dropping steadily as the active node count reduce as seen in Figure 7h.

5.1.2. Controlled user behavior

The results for controlled behavior are shown in Figure 8. The node count shown in Figure 8a indicates that both the 100 and 200 nodes tests had a drop in the number of nodes, as opposed to a constant node count as expected, although not as rapid as the realistic randomization test. Nevertheless, other test results show that the system in general operated as expected. The leafset sizes for all tests remained stable as seen in Figure 8b for both tests. Also the routing table size is seen to change with the reduction in node count as in Figure 8c, similar to the realistic test, hence indicative of a robust network that recognizes node failures and makes appropriate adjustment dynamically. As expected, the larger networks has more messages, as seen in Figure 8d, and there is also a notable drop in messages recorded that parallels the drop in the node count in both tests. The maximum message sending rates are about 610 and 880 messages/sec for the 100 node and 200 node tests respectively as seen in Figure 8e. The network memory usage is expected to increase as the network data grows due to the interactions between the node just as was previously seen in the realistic test. This is the case as depicted in Figure 8f. The notable drop in the network memory correlates with the drop in network nodes. The 200 node test records a maximum data sending rate of about 11.5 KB/s while oscillating between 2 and 4 KB/s against a maximum of 4 KB/s for the 100 node test which oscillates between 0.5 and 1.5 KB/s as shown in Figure 8g. Similar to the realistic test, the retrieval rate for the DDSs is highest just after the end of the startup phase and declines steadily, with maximum recorded values being 600 KB/s and 240 KB/s for the 200 and 100 node tests respectively as given in Figure 8h. As previously noted, there was an unexpected drop in the node count. From the analysis of the error log files, it was realized that in some instances, the allocated node memory was exceeded resulting in the general failure of the node.

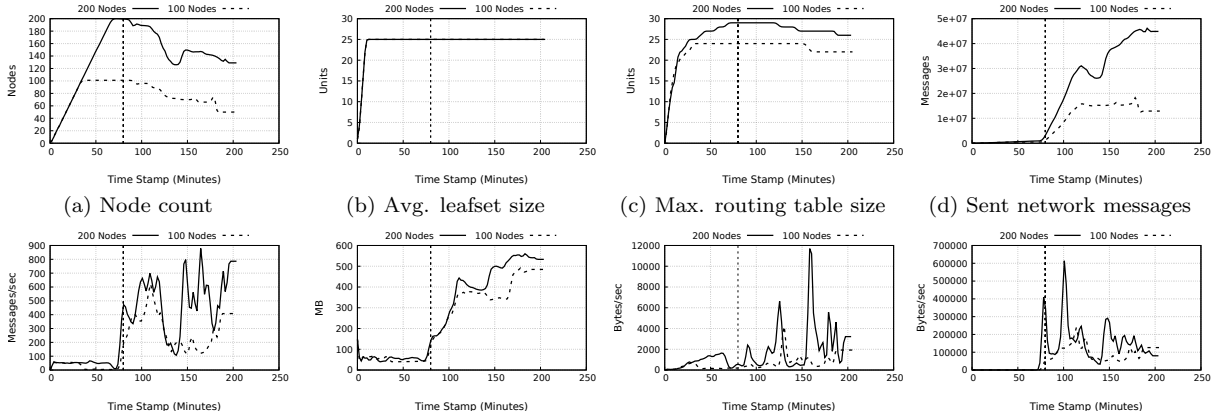
Summary. From the results of the realistic test, it is evident that for the network to remain stable and alive for a longer period of time, there is need for an ever growing network, taking into consideration the randomization



(a) Node count (b) Avg. leafset size (c) Max. routing table size (d) Sent network messages

(e) Max. network message send rate (f) Max. network memory used (g) Max. send data rate (h) Max. DDS retrieval rate

Figure 7: Realistic user behavior



(a) Node count (b) Avg. leafset size (c) Max. routing table size (d) Sent network messages

(e) Max. network message send rate (f) Max. network memory used (g) Max. send data rate (h) Max. DDS retrieval rate

Figure 8: Controlled random behavior

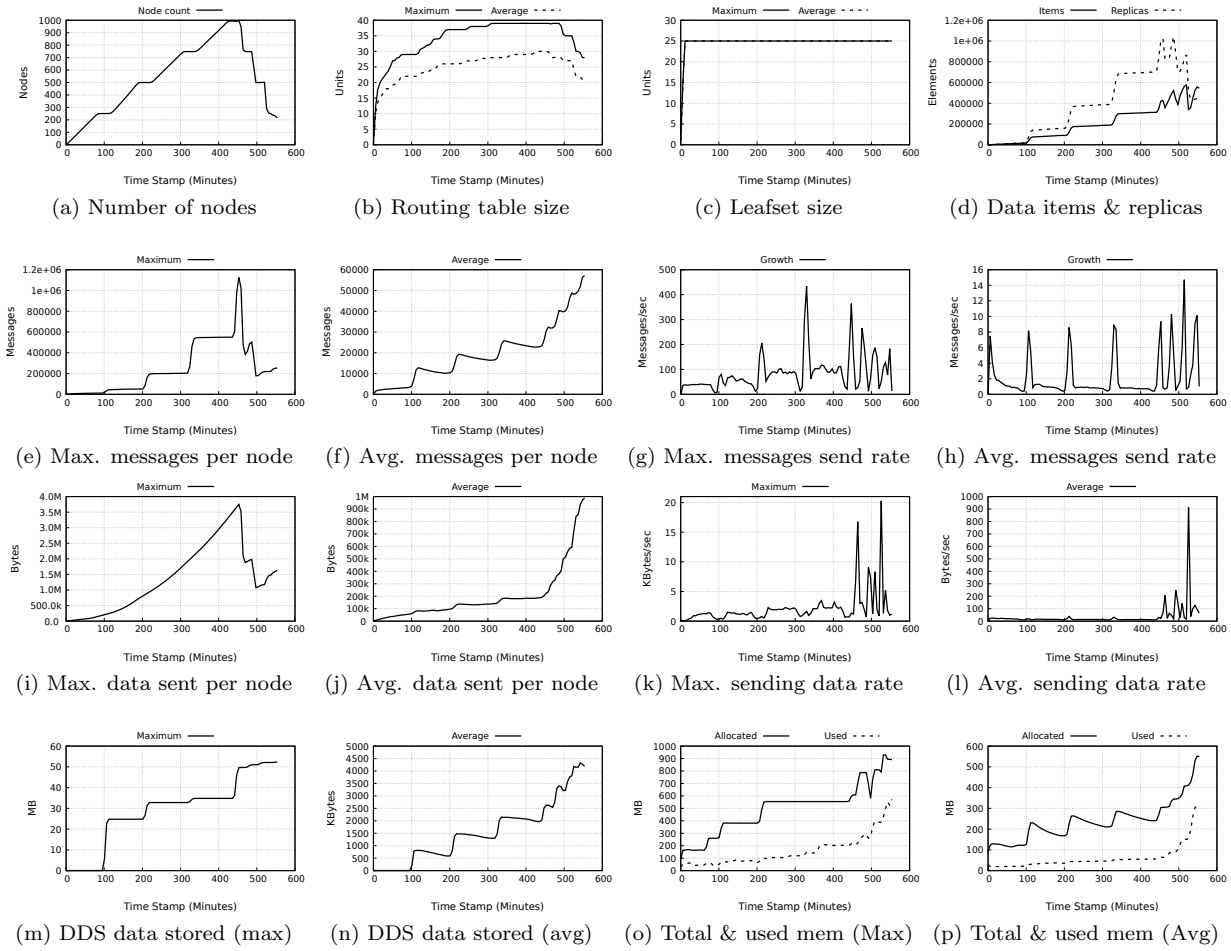
algorithm employed. The results of the controlled test, despite the memory-induced limitations that resulted in node failures, proved that overally, the system functions well. Thus, the social network is stable throughout and and the monitoring component is able to capture data throughout, allowing us to gather statistics relevant to explain the finer workings of the system. The results thus presented attest to a properly working social network application, and also provide sufficient evidence that the underlying FreePastry network structure is stable and reliable even in a randomly behaving network.

5.2. Stability & scalability

To show the ability of the network to reach a stable state, we look at the systems reaction to both network growth and churn. From the results we can deduce how stable the network is and also make remarks on the ability to scale up and down. As the test for both growth and churn were carried out in the same experiment, it is easy to see how the system reacts in both situations and contrast them. We discuss the results shown in Figure 9 in the following.

a) Node count: Throughout the experiment, the node count remained stable in both the growth and churn phases as depicted in Figure 9a. There was also seamless transition from the growth phase into the churn phase. As the network was able to support up to 1000 nodes, we can deduce that the system can easily scale up. Further, during the churn phases, we notice that despite massive churn, (of up to 50%), the network remains in operation.

b) Routing table size: Figure 9b shows the maximum and average table size for a single node through the experimental duration. Observable is a corresponding adjustment of the routing table during the growth phase as well as during churn. The maximum value recorded is 39 nodes, but on average, the maximum is 30 nodes. Despite the relatively high growth (as



(a) Number of nodes (b) Routing table size (c) Leafset size (d) Data items & replicas

(e) Max. messages per node (f) Avg. messages per node (g) Max. messages send rate (h) Avg. messages send rate

(i) Max. data sent per node (j) Avg. data sent per node (k) Max. sending data rate (l) Avg. sending data rate

(m) DDS data stored (max) (n) DDS data stored (avg) (o) Total & used mem (Max) (p) Total & used mem (Avg)

Figure 9: Stability and scalability analysis

well as churn) the routing table size does not significantly alter. If we take 250 nodes in the network as the base, hence a maximal value of 29 nodes and average of 22 nodes for the routing table size, the routing table maximally adds about 10 nodes and on average up to 9 nodes during the entire growth phase, and looses the same number of nodes during the entire churn phase. This means the the routing table adjusts accordingly to meet the demands of the network.

c) Leafset size: From Figure 9c, we observe, that the leafset is maintained at 25 nodes. The experimental setting for the leafset size is 24 nodes. The additional value is because the first and last value in the leafset is the node's own `ID`. As the network has more nodes than 25, it is expected that the average and maximum values for the leafset size will be the same.

d) Stored data items & replicas: The replication factor for the experiment is set to 4. It would be therefore expected that the number of replicas should be at least 3 times more than the local items. However, each node also stores some replica, which become local to it. That notwithstanding, we note that generally, there is about twice as many replicas as locally stored data items. This holds true throughout the experimental duration, except at the last churn, for which the network is reduced by 50%. This may have caused a drastic reduction in the number of replicas, but essentially, there some replicas are still present.

e) Messages sent: The maximum and average messages sent in a single node are shown in Figures 9e and 9f respectively. There is a steady rise in messages sent per node in the growth phase, evident in both figures, with the maximum value recorded being about 1.1 million messages by a single node corresponding to the end of the growth phase. On average, during the growth phase, the maximum value is about 25000 messages, which is acceptable. During the churn phase, we observe different characteristics from the two figures. While the maximum messages per node shows a sharp decline to a value of about 200000 messages, the average messages show a sharp increase to a value of 58000 messages. This rise in average messages may be due to route adjustment queries as

nodes are removed from the network and new routes have to be established.

f) Message send rate: The maximum and average rates are shown in Figures 9g and 9h respectively. The maximum recorded message rate is about 430 messages/sec corresponding to the last increment of growth phase. Essentially, the maximum message rate during growth oscillates around 100 messages/sec. The average value for the message rate on the other hand, reaches a maximum during the last step of churn, with a value of about 15 messages/sec. On average, the message rage oscillates between 1 and 10 messages/sec.

g) Data sent: The values recorded for maximum and average data sent per node are shown in Figures 9i and 9j. Both figures show a direct correlation with the messages sent. The maximum recorded value for data sent by a single node is about 3.7 MB which is seen at the end of the growth phase which then drops to a minimum of slightly more than 1.0 MB during the churn phase. On the average values, during the growth phase, we see a steady rise in data sent to reach 190 kB at the end of the growth phase. During the churn phase however, rather than a decline, there is a sharp rise in data sent, reaching just below 1.0 MB at the end of the experiment. This sharp rise just like in the case of messages stored may be due to high messaging, as well as readjustments in replica placements for nodes still in the network.

h) Data send rate: Figures 9k and 9l show the maximum and average data sending rates. During the growth phase, the maximum sending rate hardly exceeds 2.5 kB/s. However, during the churn phase, especially as nodes leave the network, there are significant surges in the data rate, with a maximum data rate of 20 KB/s during the last step of churn. On average, the values are quite different. During growth phase, the data rate hardly exceeds 20 bytes/sec, and just like in the case of maximum data send rates, the average shows surges during the churn phase, with a notable high of 900 bytes/sec at the last churn step. The reason for this phenomenon is explained in the same way as the messages stored and data sent.

i) DDS data stored: The maximum and average DDS data stored are shown in Figures 9m and 9n respectively. During the growth phase, there is an increase in the DDS data after the first growth step. No DDS data is recorded during the first growth phase as the workload is yet to be executed. At the end of the growth phase, the DDS data stored is maximum size of DDS data stored is about 34 MB, with the average maximum size being 2 MB. During the churn phase, rather than a decline, there is an increase in the amount of DDS data stored per node, reaching a maximum of about 52 MB and an average maximum of 4.3 MB just before the end of the experiment.

j) Total & used memory: A comparison of the maximum and average values for total allocated and used memory is shown in Figures 9o and 9p respectively. We note that both the maximum and average for allocated memory correlate with the graphs for messages sent and DDS data stored. This is indicative of the fact that messages sent and DDS data stored had a large impact on the memory allocated. The maximum memory allocated was 900 MB, and the average maximum was about 550 MB. However, the memory that was actually utilized was in comparison less than half of the allocated memory. The maximum utilized was recorded as about 580 MB and the average maximum used memory was about 300 MB.

Summary. From the results, we can conclusively state that the network renders itself scalable as is evident by the smooth growth of the network, as well as stable as evidenced by the results during the churn phase. The system adjusts automatically adjusts the relevant aspects of the system to accommodate changes in the network. As the network grows memory allocated is increased to match demand and nodes are added to the routing table. When churn occurs, we see that the nodes make adjustments by taking on more load, evidenced by an increase in memory rather than a decrease, and there is restructuring of the routing table.

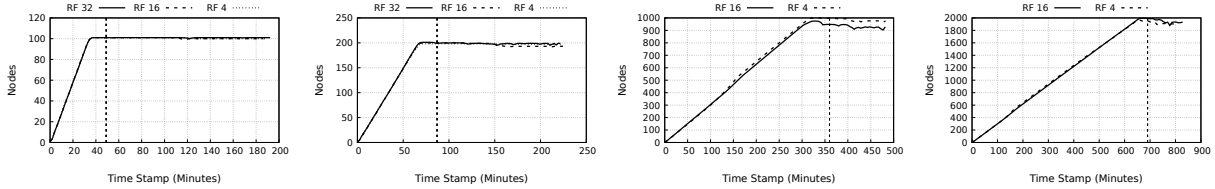
5.3. Effect of the replication factor

Figures 10-18 depict the system dynamics with the focus narrowed down to the previously selected metrics for analysis. The deductions for the results are given in the summary provided in Table 5. We considered the performance of the system against the costs and discuss the relevant metrics in the following.

5.3.1. System performance

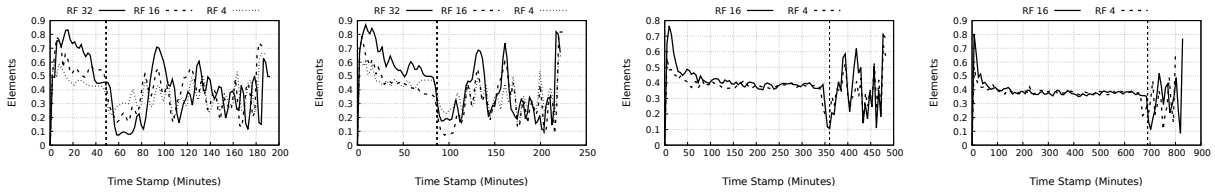
The analysis of the performance of the system considered the number of nodes in the system throughout the experiment, the hop counts, the number of replicas in the system, the maximum data storage and retrieval times, the message sending and receive rates, the maximum used storage space and the maximum used system memory.

a) Node count: Figures 10a-10d portray the behavior nodes in the systems. The figures indicate successful monitoring of the network nodes and in general the networks were seen to be stable throughout the experimental. It is noteworthy that the test duration. Of note though, is that the 1000 and 2000 nodes experiments had a slight reduction by about 100 nodes or less. One explanation for this, evident from the logs generated, is that the nodes experienced insufficient memory errors during testing resulting in overall node failure. However, this did not affect the aggregation of monitored statistics in the network.



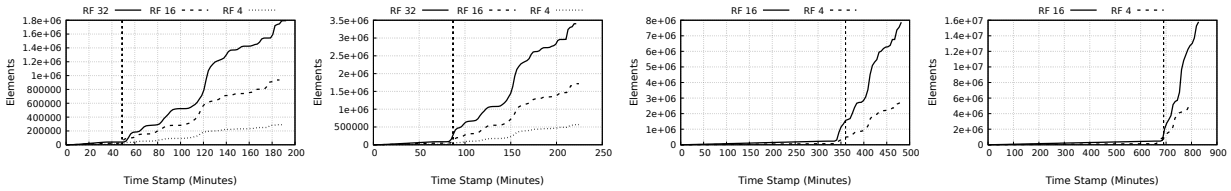
(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 10: Node count analysis



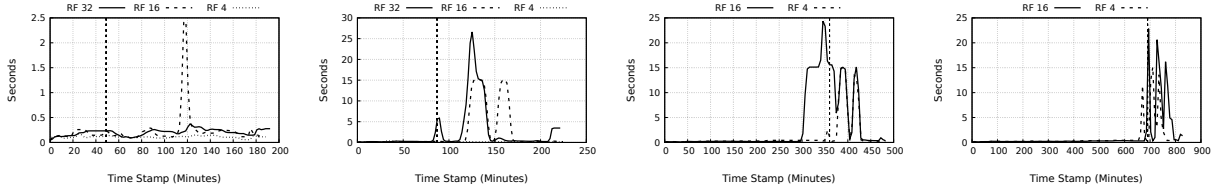
(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 11: Mean hop count analysis



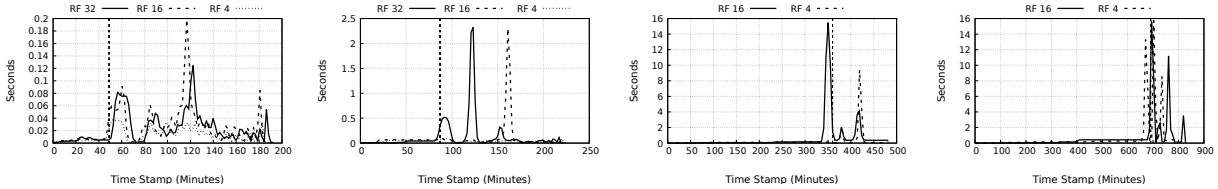
(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 12: Number of replicas analysis



(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 13: Maximum storage time analysis



(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 14: Maximum retrieval time analysis

b) Hop count: The hop count for the experiments are given in Figures 11a-11d. In general, the number of hops from sender to receiver is 1 because the number of nodes in the experiments do not exceed the maximum possible limit for the routing table of $160.20 = 3200$ entries. Hence the average hop counts tends to be less than one in the network. It is however observed that as the network gets larger, there is less disparity in the network as the graphs show an overlap, as is the case with 1000 and 2000 node graphs.

c) Number of replicas: As the number of nodes increases, as can be seen in Figures 12a-12d, the number of data replicas data also increases as well. In addition, with increasing replication factor, there is an equivalent increase in the replicas with the same magnitude, which is as expected.

d) Storage time:From Figures 13a-13d, two important observations can be made. First, is that increasing the number of nodes results in increased maximum storage time, although the increase is in the order of milliseconds, hence can be ignored. The second more significant deduction is that increasing the replication factor results in increased maximum storage time. In general, the average storage times, as shown in Table 5, are less than one second, but we note the extreme cases of maximum storage time in the order of 10s of seconds, particularly the 2000 node experiment with replication factor of 4, which recorded a maximum storage time of 91 seconds. Bearing in mind that storage time includes time to store replicas, it is possible that this discrepancy may be related to storage of data related to distributed data structures such as wall posts and forum threads, which must be distributed and replicated at the same time.

e) Retrieval time: This is shown in Figures 14a-14d. The same observations as with storage time are also seen here. Again, we make the deduction that large retrieval times are related to distributed data structures. Interestingly, even with more replicas in the network, the maximum retrieval time does not significantly drop. The average retrieval times are still less than one second for the system in general. A significant observation is that for the 100 node tests, for replication factor of 16 a higher maximum retrieval time of 15.12 seconds was recorded, in comparison to 0.189 and 0.6 seconds for replication factors 4 and 32 respectively. It is probable this was the result of a DDS data retrieval and is an outlier.

f) Message send/receive rate: From Figures 15a-15d, for message sending rates and Figures 16a-16d for message receiving rates, it is noted that a higher replication factor there is a drop in the average messaging rates. Also surprising is that when comparing the 100 and 200 node experiments, it seems that a replication factor of 16 seems to generate higher maximum rates than the lower replication factor tests. This result may be an indicator that when considering replication factor, there are values that are accounted as optimum for the network performance. This result may need further investigation. The average rates also dropped as the network became larger, and as the replication factor increased.

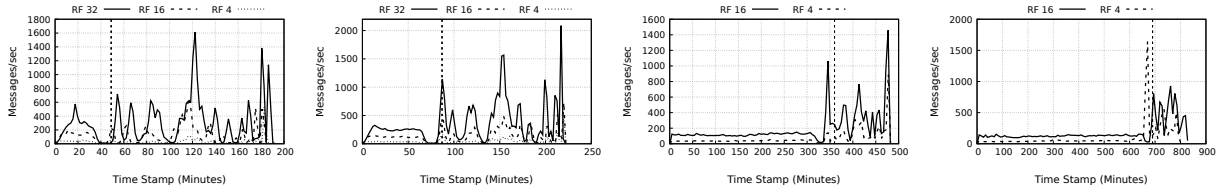
5.3.2. Cost analysis

The three aspects to the cost on the system that were evaluated are storage space usage, memory usage and network data rate. The analysis follows.

a) Used storage space: From Figures 17a-17d, it is seen that if the replication factor is held constant, the average used storage space per node is almost invariably similar with deviation of about ± 10 MB. The maximum used storage space on the other hand varies and no general trend can be deduced. In general, as the replication factor increases, used space increases, both on average and also the maximum used space.

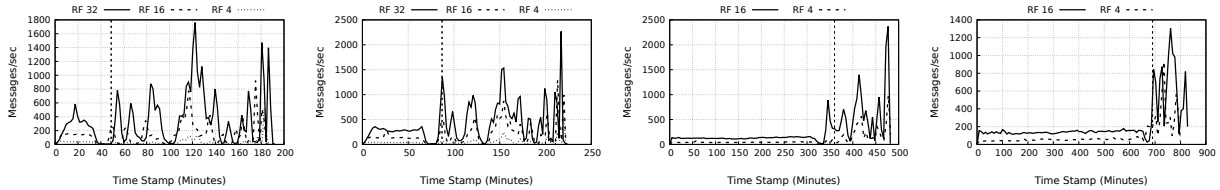
b) Used memory: In Figures 18a-18d, the maximum used memory for the four node groups is shown. With fewer nodes in the network, there is very little disparity in maximum memory utilization for replication factor 4 and 16, but replication factor 32 had a notably large maximum memory usage. For the 1000 and 2000 node tests, with lower replication factor of 4, there was a drastically higher maximum memory consumption than for replication factor 16. This may be because there are fewer replicas in the network thus requiring more requests generated leading to more memory consumption as a suitable replica is located. It is also possible that with fewer replicas, the storing nodes may also experience a bottleneck in replying to all the requests. The average memory consumption tended to follow the expected norm, that is, increase in replication factor had an increase in memory consumption.

c) Network bandwidth: In Table 5, the last four columns indicate the average and maximum network data rate for sending and receiving due to the workload. It is observed that increasing the nodes or replication factor leads to a corresponding increase in the needed network bandwidth as is evidenced by the increasing maximum data rates. The average data rates for both sending and receiving were nevertheless less than 1 KB/s except for the 2000 node test with replication factor of 16, which recorded values slightly greater than 1 KB/s.



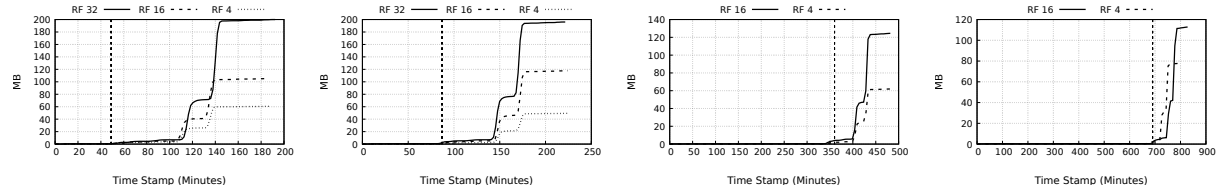
(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 15: Maximum message send rate analysis



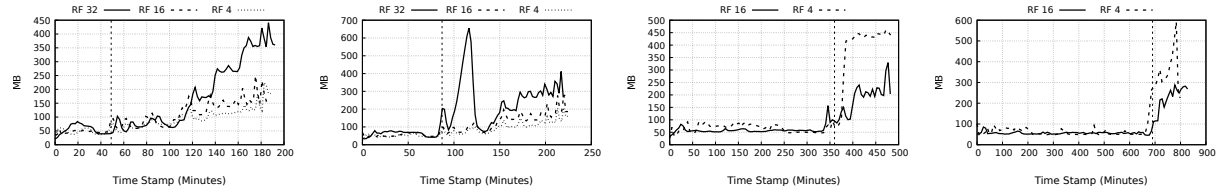
(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 16: Maximum message receive rate analysis



(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 17: Maximum used space analysis



(a) 100 nodes (b) 200 nodes (c) 1000 nodes (d) 2000 nodes

Figure 18: Maximum used memory analysis

Table 5: Summary of Replication Factor Analysis

Nodes	RF	PERFORMANCE									COST							
		Avg Hops	t_{retr} (secs)		t_{store} (secs)		m_{send} (msgs/s)		m_{rcv} (msgs/s)		Used storage (MB)		Used memory (MB)		Data rate (kB/s)			
															Send		Receive	
			Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
100	4	1	0.081	0.189	0.012	0.138	15.19	90.54	15.19	95.59	25.24	57.77	101.38	350.51	0.13	0.97	0.07	1.21
	16	1	0.314	15.120	0.037	0.553	13.68	105.08	13.68	105.08	82.23	100.22	148.35	334.96	0.18	1.30	0.12	1.58
	32	1	0.176	0.600	0.033	0.302	12.13	87.35	12.13	87.35	152.80	190.71	221.35	525.75	0.15	1.40	0.15	1.57
200	4	1	0.068	0.368	0.018	0.232	14.72	84.52	14.72	84.52	24.76	47.03	111.67	301.14	0.06	15.53	0.05	3.98
	16	1	0.767	15.212	0.023	15.147	11.69	120.10	11.69	201.64	83.27	112.29	158.27	383.30	0.35	2.81	0.17	2.74
	32	1	0.861	30.053	0.070	15.073	11.50	120.72	11.50	120.72	152.83	186.97	224.24	788.62	0.13	15.43	0.13	15.64
1000	4	1	0.194	15.283	0.111	15.158	10.00	230.56	9.99	230.56	25.14	59.07	118.73	567.17	0.39	63.10	0.38	32.27
	16	1	0.764	30.179	0.200	18.075	7.22	211.48	7.22	211.48	74.97	118.72	163.67	397.23	0.77	60.50	0.51	40.39
2000	4	1	0.156	30.084	0.640	91.000	7.58	211.53	7.58	211.53	24.68	74.33	104.09	645.46	0.62	100.49	0.70	71.20
	16	1	0.755	41.954	0.803	30.397	7.11	222.49	7.11	222.49	68.18	107.59	168.69	431.38	1.03	60.75	1.28	52.05

Summary. Evidently, the replication factor plays a significant role in the system performance. It is generally expected that, the higher the replication factor is, the better is the system performance, but at higher costs to the network. Invariably, it was observed that the system averages for storage and retrieval were much smaller than one second. High maximum storage/retrieval times are seen because of the DDS structures. In essence, the use of DDS, although allowing the system to handle complex data such as forums, albums and wall posts, has a considerable impact on the data access and storage times. When looking at average message sending/receiving rate, higher replication factors are desired because the rates drop, notwithstanding increasing maximum message rates, which can be also attributed to storage or retrieval of complex data types. The cost implications to replication factor increase are increased storage space usage and memory usage. In general, therefore, it can be said that the choice of replication factor in for storage must be carefully considered based on a performance-cost analysis. Higher replication values tend to have adverse effects on performance in smaller networks and lower replication values have a similar effect on large networks. Hence a balance must be made based on the network size.

6. Conclusion

As opposed to many other benchmarking tests which are carried out in simulation tests, in this evaluation of LibreSocial, a P2P framework for OSNs, we endeavoured to perform benchmarking for large networks, reaching upto 2000 active nework nodes. In addition, tests were modeled to mimic the real environment for random behavior characteristics. The evaluation in general presents very insightful information. The network easily scale up and also continues to function well under high churn, making adjustments to suit the situation in terms of memory requirements, route readjustments and replica diversion as well as placement. This behavior points to a network that can achieve stable state quickly. The replication factor chosen for the system plays a very important role in the performance of the system. The higher the replication factor, the higher the costs associated with it. However, as the network size grows, lower replication factors generally tend to result in poor performance. In view of the system's performance in a real environment based on our randomization model, we can safely say that the system works quite well with minimal errors. It is worth noting that the randomization algorithm designed had the tendency of causing the network to diminish.

References

[1] K. Subrahmanyam, S. M. Reich, N. Waechter, and G. Espinoza, "Online and offline social networks: Use of social networking sites by emerging adults," *Journal of Applied Developmental Psychology*, vol. 29, no. 6, pp. 420–433, 2008, social Networking on the Internet.

[2] B. Guidi, M. Conti, and L. Ricci, "P2P architectures for distributed online social networks," in *International Conference on High Performance Computing & Simulation, HPCS 2013, Helsinki, Finland, July 1-5, 2013*, 2013, pp. 678–681.

[3] S. Spiekermann, A. Acquisti, R. Böhme, and K.-L. Hui, "The challenges of personal data markets and privacy," *Electronic Markets*, vol. 25, no. 2, pp. 161–167, Jun 2015.

[4] S. Taddei and B. Contena, "Privacy, trust and control: Which relationships with online self-disclosure?" *Computers in Human Behavior*, vol. 29, no. 3, pp. 821–826, 2013.

[5] C. E. Tucker, "Social networks, personalized advertising, and privacy controls," *Journal of Marketing Research*, vol. 51, no. 5, pp. 546–562, 2014.

[6] A. A. Barreda, A. Bilgihan, K. Nusair, and F. Okumus, "Generating brand awareness in online social networks," *Computers in Human Behavior*, vol. 50, pp. 600–609, 2015.

[7] V. Bakir and A. McStay, "Fake news and the economy of emotions," *Digital Journalism*, vol. 6, no. 2, pp. 154–175, 2018.

[8] R. Roy and N. Gupta, *Digital Capitalism and Surveillance on Social Networking Sites: A Study of Digital Labour, Security and Privacy for Social Media Users.* Cham: Springer International Publishing, 2018, pp. 67–81.

[9] L. Brandimarte, A. Acquisti, and G. Loewenstein, "Misplaced Confidences: Privacy and the Control Paradox," *Social Psychological and Personality Science*, vol. 4, no. 3, pp. 340–347, 2013.

[10] T. Paul, S. Buchegger, and T. Strufe, "Decentralizing social networking services," in *International Tyrrhenian Workshop on Digital Communications*, 2010, pp. 1–10.

[11] S. Buchegger and A. Datta, "A case for P2P infrastructure for social networks-opportunities & challenges," in *Proceedings of the Sixth International Conference on Wireless On-Demand Network Systems and Services, 2009. WONS 2009.* IEEE, 2009, pp. 161–168.

[12] J. F. Buford and H. Yu, "Peer-to-peer networking and applications: Synopsis and research directions," in *Handbook of Peer-to-Peer Networking*, X. Shen, H. Yu, J. Buford, and M. Akon, Eds. Springer, 2010, ch. 1, pp. 3–45.

[13] R. Rodrigues and P. Druschel, "Peer-to-peer systems," *Communications of the ACM*, vol. 53, no. 10, pp. 72–82, 2010.

[14] S. Buchegger, D. Schiberg, L.-H. Vu, and A. Datta, "Peerson: P2P social networking: Early experiences and insights," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems.* ACM, 2009, pp. 46–52.

[15] L. A. Cutillo, R. Molva, and M. nen, "Safebook: privacy preserving online social network," in *Internet Privacy Workshop, IAB/W3C/ISOC/MIT*, 2010, pp. 1–2.

[16] R. Sharma and A. Datta, "Supernova: Super-peers based architecture for decentralized online social networks," in *Proceedings of the 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS).* IEEE, 2012, pp. 1–10.

[17] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia, "DECENT: A decentralized architecture for enforcing privacy in online social networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on.* IEEE, 2012, pp. 326–332.

[18] L. M. Aiello and G. Ruffo, "Lotusnet: Tunable privacy for distributed online social network services," *Computer Communications*, vol. 35, no. 1, pp. 75–88, January 2012.

[19] B. Guidi, T. Amft, A. De Salve, K. Graffi, and L. Ricci, "DiDuSoNet: A P2P architecture for distributed dunbar-based social networks," *Peer-to-Peer Networking and Applications*, pp. 1–18, 2015.

[20] K. Graffi, S. Podrajanski, P. Mukherjee, A. Kovacevic, and R. Steinmetz, "A distributed platform for multimedia communities," in *Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on.* IEEE, 2008, pp. 208–213.

[21] K. Graffi, C. Gross, P. Mukherjee, A. Kovacevic, and R. Steinmetz, "LifeSocial. KOM: A P2P-based platform for secure online social networks," in *Peer-to-Peer Computing (P2P), 2010*

IEEE Tenth International Conference on. IEEE, August 2010, pp. 1–2.

[22] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz, "LifeSocial. KOM: A secure and P2P-based solution for online social networks," in *Proceedings of the 2011 IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, January 2011, pp. 554–558.

[23] T. Buchert, "Managing large-scale, distributed systems research experiments with control-flows," Thesis, Université de Lorraine, Jan 2016.

[24] M. Lehn, T. Triebel, C. Gross, D. Stingl, K. Saller, W. Effelsberg, A. Kovacevic, and R. Steinmetz, *Designing Benchmarks for P2P Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 209–229.

[25] S. Kounev, "Performance engineering of distributed component-based systems," PhD, Technischen Universität Darmstadt, Darmstadt, Germany, 2005.

[26] K. Sachs, S. Kounev, M. Carter, and A. Buchmann, "Designing a workload scenario for benchmarking message-oriented middleware," in *SPEC Benchmark Workshop*, 2007.

[27] K. Saller, K. Panitzek, and M. Lehn, *Benchmarking Methodology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 19–45.

[28] R. Jain, *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling*, ser. Wiley professional computing. Wiley, 1991.

[29] K. Graffi and N. Masinde, "LibreSocial: A peer-to-peer framework for online social networks," *arXiv preprint arXiv:2001.02962*, 2020.

[30] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*. Springer, 2001, pp. 329–350.

[31] ——, "Storage Management and Caching in PAST, a Large-scale, Persistent Peer-to-peer Storage Utility," *SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 188–201, October 2001.

[32] P. Druschel and A. Rowstron, "PAST: A large-scale, persistent peer-to-peer storage utility," in *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*. IEEE, 2001, pp. 75–80.

[33] S. Ramabhadran, S. Ratnasamy, J. M. Hellerstein, and S. Shenker, "Brief Announcement: Prefix Hash Tree," in *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '04. New York, NY, USA: ACM, 2004, pp. 368–368.

[34] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz, "SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems," in *2008 14th IEEE International Conference on Parallel and Distributed Systems*, Dec 2008, pp. 279–286.

[35] K. Graffi and A. Disterhöft, "SkyEye: A tree-based peer-to-peer monitoring approach," *Pervasive and Mobile Computing*, vol. 40, pp. 593–610, 2017.

[36] A. Kovacevic, K. Graffi, S. Kaune, C. Leng, and R. Steinmetz, "Towards Benchmarking of Structured Peer-to-Peer Overlays for Network Virtual Environments," in *14th International Conference on Parallel and Distributed Systems, ICPADS 2008, Melbourne, Victoria, Australia, December 8-10, 2008*. IEEE, 2008, pp. 799–804.

[37] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behavior in online social networks," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC 09. New York, NY, USA: ACM, 2009, pp. 49–62.