Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

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

Newton Masinde, Liat Khitman, Iakov Dlikman, Kalman György Graffi

2020

Preprint:

This is an accepted article published in MDPI Future Internet - Special Issue "Security and Privacy in Social Networks and Solutions". The final authenticated version is available online at: https://doi.org/10.3390/fi12090140

future internet

MDPI

Article

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

Newton Masinde [1,*], Liat Khitman [1], Iakov Dlikman [1] and Kalman Graffi [1,2]

[1] Technology of Social Networks, Heinrich Heine University, Universitatsstrasse 1, 40225 Dusseldorf, Germany; liat.khitman@hhu.de (L.K.); iakov.dlikman@hhu.de (I.D.); kalman.graffi@honda-ri.de (K.G.)

[2] Honda Research Institute Europe GmbH, Carl-Legien-Strasse 30, 63073 Offenbach am Main, Germany

* Correspondence: newton.masinde@hhu.de

Received: 14 July 2020; Accepted: 18 August 2020; Published: 20 August 2020



Abstract: Peer-to-peer (P2P) networks have been under investigation for several years now, with many novel mechanisms proposed as is shown by available articles. Much of the research focused on showing how the proposed mechanism improves system performance. In addition, several applications were proposed to harness the benefits of the P2P networks. Of these applications, online social networks (OSNs) 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 are 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 up to 2000 nodes and show the performance against costs of the system in general. From the results it is evident that LibreSocial's performance is capable of meeting the needs of users.

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 demonstrated by number of online social networks (OSNs) available, with studies showing a general overlap between the online and offline networks of many of these users [1]. As a consequence of the growth of these platforms some concerns have arisen on two fronts, technical and social [2]. The technical concerns arose due to a high dependence on centralization in administering 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 have emerged. As it currently stands, the OSN providers have succeeded in developing mitigating solutions for the scalability concerns, such as using distributed data management solutions such as Cassandra [3] and Haystack [4] in Facebook or using cloud services such as Amazon's AWS storage services (https://aws.amazon.com/products/storage/), with the more popular OSNs with very large user networks, such as Facebook (2.6 billion), YouTube (2 billion) and WhatsApp (2 billion) as of July 2020 (https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/). However, this level of scaling does not come cheap, and more often that not requires the providers to develop monetization models for revenue generation which result in certain violations of the user's rights leading into the second concern. The social concerns have to do with the users' privacy and data

ownership. As the centralized OSNs have grown, the providers, who own all the uploaded data, have developed personal data markets [5], in which the user's private data are the product on sale. 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 [6]. The collected user data may be sold to third parties who use it for personalized advertising [7], increasing brand awareness [8], emotional manipulation [9] or even online activity surveillance [10]. Even though most of the OSNs include privacy settings to give users some control over how much information they reveal about themselves, in many cases more is revealed when aggressively using these privacy settings than when having fewer setting [11].

To tackle these concerns, a move from centrally managed solutions to decentralized solutions was proposed in two main directions: web-based and P2P-based solutions [12]. *Web-based solutions*, also commonly referred to as federated solutions, are formed by having nodes that run software which supports one of the federated social web protocols which allow the nodes to communicate with one another. Some federated networks have a fairly large user base such as Mastodon (2.7 million), Diaspora (0.77 million), Prosody (0.23 million), although they are not as popular and widespread as the centralized OSNs. However, they have a heavy reliance on a distributed web server infrastructure which requires some in-depth knowledge of web-server administration to setup the OSN hence not many users can effectively use them. In addition, running a private web server for thousands of users also reintroduces the pressure of monetization. On the other hand, *P2P-based solutions* are easy to set up, requiring only downloading, possible installation and running of the application to join the P2P OSN community. They are therefore suitable to regular Internet users. Most of the P2P-based OSNs incorporate mechanisms give users the ability to control their own privacy settings and also maintain data ownership, which drastically reduces or completely eliminates the pressure to monetize. Also, since the current OSNs are Web 2.0 applications and are at a logical level inherently P2P in nature [13], 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 (peers) have 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 [14,15]. However, the design and implementation of P2P OSNs is not trivial, 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 [13]. The research on P2P OSNs has given rise to many proposals such as PeerSoN [16], Safebook [17], SuperNova [18], DECENT [19], LotusNet [20] and DiDuSoNet [21], each using different techniques to solve these design challenges. For most of the proposed solutions, the observed focus reported is on showing how the system achieves a particular function vis-à-vis an identified design concern, such as security, privacy, storage, replication and so on. Therefore there is little, if any, literature showing a realistic assessment of the quality these OSNs in terms of the overall performance against the associated costs.

In this paper, we aim at presenting a systematic evaluation of the performance of one of the more advanced P2P OSN solutions, LibreSocial [22] (previously called LifeSocial.KOM [23–25]) which is currently undergoing iterative development. We note that the existing publications on the previous iterations of LibreSocial were not focused on showing how well the system performs in general, but rather on its design [23,24] and the security features [25]. Specifically, we aim at showing the general system performance by evaluating the application comprehensively to show the 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 can be used for P2P systems and also describes the system quality properties that we aim to investigate. In Section 3, we briefly give a description of LibreSocial, our P2P-based OSN framework solution. 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 with a discussion that interprets them. Finally, in Section 6, we give the concluding remarks.

2. Benchmarking P2P Systems

As an experimental methodology for evaluating a system's performance, *benchmarking* uses a synthetic/generic application on an existing, real environment [26], 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 with this approach is its inability to fully represent a realistic situation. However, it helps in directing system designers on where to make adjustments based on results obtained. 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

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. 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 [27]. A benchmark must satisfy the following requirements: (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 [28,29]. In order for the benchmark to meet these requirements, three important benchmark parameters, system, workload and environment, which interact with the system under test must be defined [30].

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* 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, and only a selected subset of parameters may be varied. This selected subset of parameters are termed *factors* and the most preferred factors are those that have the largest impact on the systems performance [31]. 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 and Relevant Metrics

The efficacy of any benchmark design relies on a well defined set of quality properties and quality metrics. While *quality metrics* focus on only describing a single attribute of a mechanism within a scenario, workload or configuration, the *quality properties* describe the system's/mechanism's characteristics putting into consideration various individual measurements of quality metrics. Quality properties are distinguished into *workload-independent* and *workload-dependent* [30], which are discussed next and the relevant quality properties with their respective metrics are given in Table 1.

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

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. They are obtainable through direct measurements or indirectly calculated using other quality aspects. In our test application, we are concerned with performance and cost.

- Performance: For a feel of the system's performance, aspects 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 used 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. They are useful in helping us understand and define the workload for the system to be tested and are evaluated using the workload-independent quality properties. 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 stability and scalability.

- Stability: This describes the system's ability to continue performing despite inherent system behavior dynamics, such as high churn rate, and eventually converge to a stable state if the workload remains the same. Stability correlates to resilience under adverse conditions, i.e., sudden changes that may occur in the system architecture due to the workload. These changes may be *expected* since the defined protocols account for them, or *unexpected* because the protocols do not consider them. To ascertain the level of stability, 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 system's ability to handle changing workloads. Two scaling dimensions are considered: *horizontal* which affects number of peers in the system, i.e., 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 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 insecure 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 protocols (publish-subscribe, unicast and multicast) and monitoring mechanisms, to be incorporated. In a previous article [32], we present a detailed description of these requirements and the options available for implementing each required P2P mechanisms for an OSN.

LibreSocial [22] (previously LifeSocial.KOM [23–25]) is a P2P-based framework that provides all these. The application sits on a P2P framework that was developed based on the Open Services Gateway Initiative (OSGi) service platform (https://osgi.org/download/r7/osgi.core-7.0.0.pdf). 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 1. In [22], 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 of the TCP/IP model. The overlay provides a degree of distribution transparency by abstracting the complexities of the physical connections. The overlay supports logarithmic message routing and object location using a heavily modified FreePastry (http://www.freepastry.org/FreePastry/), an implementation of Pastry [33], 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.

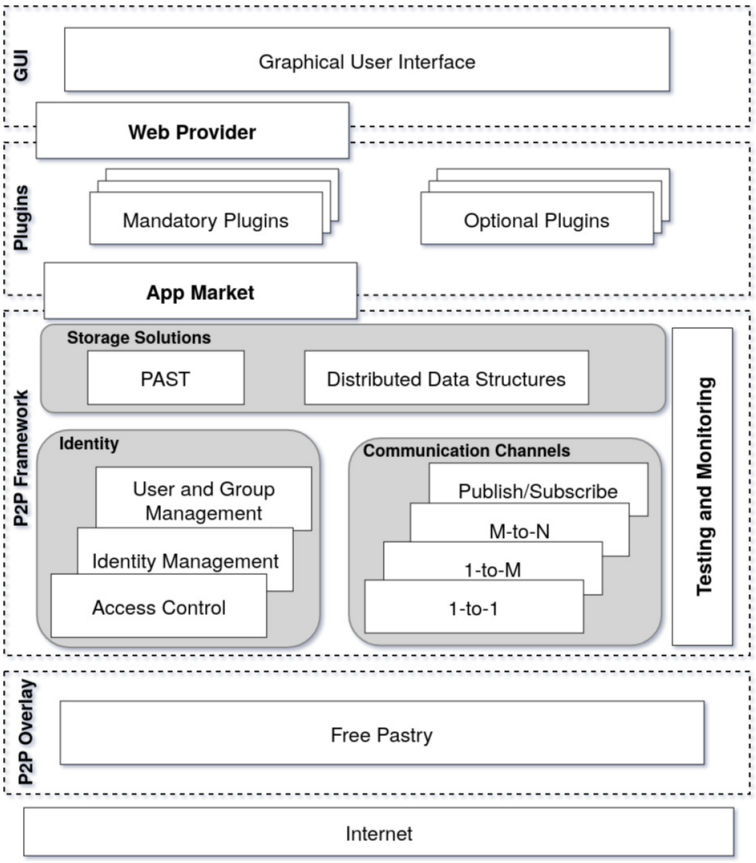


Figure 1. A P2P Social Network architecture.

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, i.e., storage, identity, communication, and testing and monitoring which are further discussed.

3.2.1. Storage

Simple file storage is provided via a heavily modified PAST [34,35], a persistent P2P storage utility bundled along with FreePastry which also includes replication services. In addition, three types of distributed data structures (DDSs), i.e., distributed sets, distributed linked lists and prefix hash trees [36] are also used 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.

3.2.2. Communication

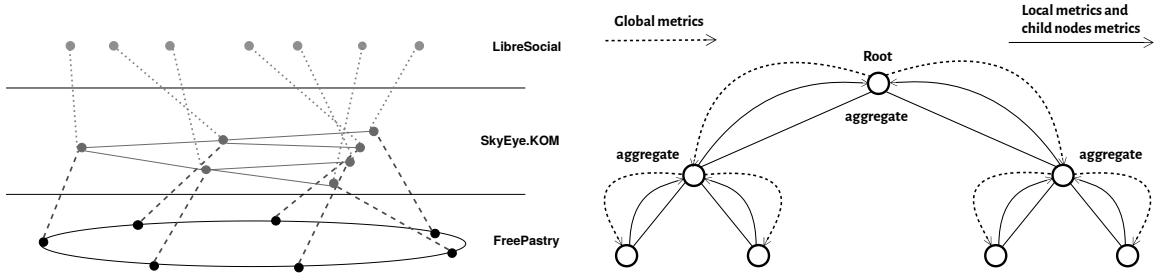
The framework supports both synchronous and asynchronous messaging depending on the channel used. It also supports unicast (1-to-1) messaging such as in direct messaging, multicast (1-to-N) messaging such as streaming to a group as well as using aggregation (N-to-M) mechanisms for distribution and aggregation 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 also uses Scribe [37], a push-based publish/subscribe utility bundled with FreePastry, offers streaming via WebRTC (https://webrtc.org/) for audio/video conferencing, and has a secure message channel that can be encrypted and signed.

3.2.3. Identity

Three key identity features offered are identity management, user and 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* uses the AES symmetric encryption algorithm and it ensures relevant users are authorized to read/write data. 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 is possible if it is signed with the corresponding private key of a specific public key after verification by the storing node.

3.2.4. Testing and Monitoring

The testing and monitoring plugins work in tandem during system testing. The test plugin sends instructions that mimic actual user activities to the different application plugins based on a defined test plan. 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 [38,39] monitors and gathers relevant measurements during the testing process. The implementation of SkyEye.KOM is independent of the underlying P2P overlay, as seen in Figure 2a, and it provides the ability to aggregate and disseminate information within the tree structure which is constructed on top of the overlay thus providing a global view on the performance and costs of the P2P network. The aggregation process for the monitoring data is shown in Figure 2b. The aggregation functions include *count*, *min*, *max*, and *mean*.



(**a**) Overview of the monitoring setup (**b**) SkyEye.KOM aggregation

Figure 2. Monitoring setup.

The services provided via the framework are then used 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 supports developing of 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 are then presented 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 between the user and the system. Figure 3 shows screenshots of LibreSocial's GUI. The GUI is composed of three sections, i.e., 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.

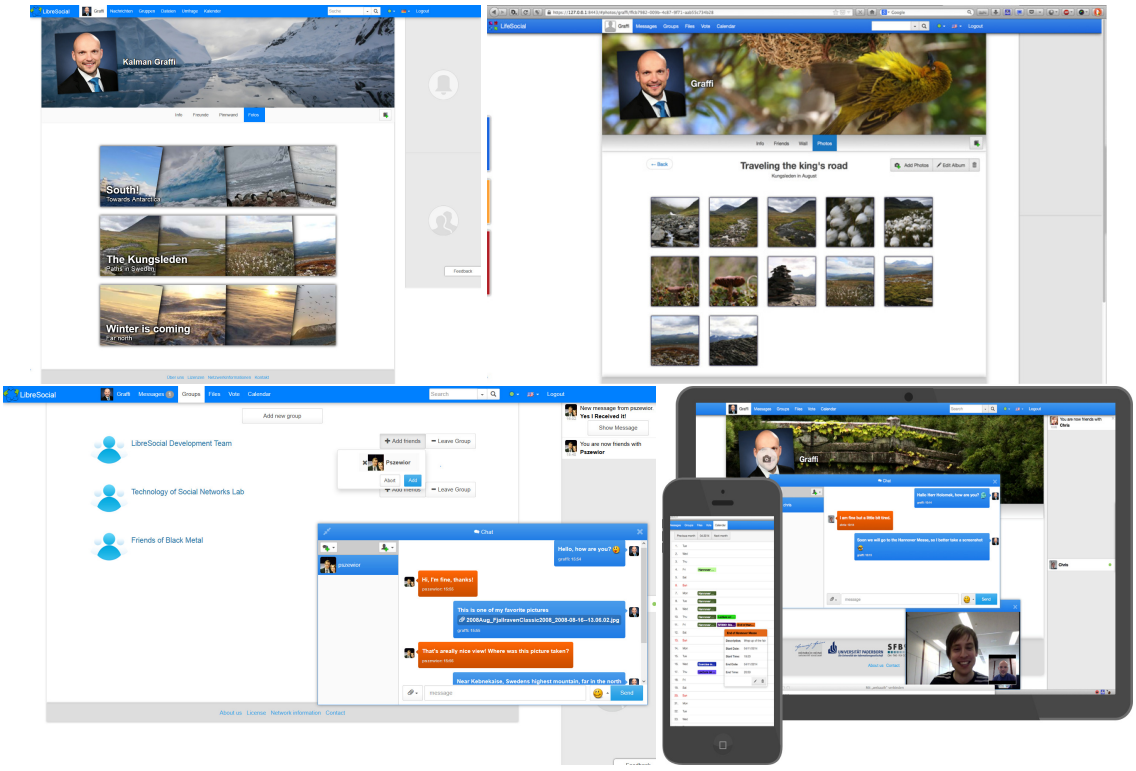


Figure 3. Screenshots of LibreSocial.

Summary: In this section we have introduced LibreSocial, our P2P Framework for OSNs, briefly giving a description of how the core P2P components are integrated into 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 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 [40]. The quality attributes and relevant metrics to be used are introduced in Section 2.2. We now discuss the relevant scenarios designed for evaluating the system. Our test environment for all scenarios is the same so as to guarantee result consistency. The tests are run on the

high performance computing (HPC) cluster at Heinrich Heine University (https://www.zim.hhu.de/high-performance-computing).

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 KBs, with an average of about 300 KBs, and the average message length is 150 characters. Also, for the testing, the photo images are used in place of files/documents. The system parameters for each of the test scenarios are shown in Table 2. A description of the different scenarios of the benchmarking test follows.

Table 2. System Parameters

Test	**Network Size**	**Number of Nodes**	**Leafset Size**	**RF**	**Duration** *(mins)*	
					Setup	*Test*
Plugin analysis	-	100	127	4	40	240
	-	500	127	4	184	240
Pseudo-random	-	500	24	6	160	200
Scalability/stability	-	1000	24	4	480	200
Replication	Medium	100	127	4, 16, 32	40	141
		200	127	4, 16, 32	80	141
	Large	1000	127	4, 16	360	141
		2000	127	4, 16	800	141

4.1.1. Baseline Tests: Plugin Analysis

These set of tests are used to baseline the system's performance. LibreSocial is primarily composed of plugins, bundles that implement various functionalities of the OSN, and includes a test plugin which interacts with each plugin via a command interface to trigger the functionalities as shown in Figure 5b. This test allows us to see the plugins in action and the overall effect due to the different plugins on the overall system. The system configurations for this test are shown in Table 2, with Table 3 as the workload generated via the test plugin. Two network sizes of 100 and 500 nodes are chosen, and for each of the networks we initiate two different sets of workload specifically, light load (with only two repetitions per test case or action) and medium load (five repetitions). The repetitions for each test case are allowed to complete over a duration of five minutes.

4.1.2. Pseudo-Random Behavior

The testing seeks to model realistic randomized user behavior. System parameters defined are given in Table 2. The random behavior algorithm selects tests in a random pattern as shown in Figure 4, mimicking normal user behavior. The algorithm was designed based on the work in [41] wherein aggregated data about the behavior of more than 35,000 users in four different social networks was studied. In the study, key parameters used are: 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 the sequence of activities based on probabilities. At the end of that time period the user logs out, and may later log back in.

Table 3. Plugin test Workload.

Plugin	Action	Repetitions *(count)*		Duration* *(mins)*
		Light	Medium	
Messaging	Send message	2	5	5
	View inbox message	2	5	5
	View outbox message	2	5	5
Live chat	Send multichat invitation	2	5	5
	Send multichat message	2	5	5
Group	Create group	2	5	5
	Invite friend to group	2	5	5
	View group	2	5	5
	View my group list	2	5	5
File storage	Create folder	2	5	5
	Upload file in folder	2	5	5
	View folder	2	5	5
Forum	Create forum thread	2	5	5
	Comment forum thread	2	5	5
	View forum	2	5	5
Photos	Create photo album	2	5	5
	Upload photo	2	5	5
	View own album	2	5	5
	View friend's album	2	5	5
Calendar	Create calendar event	2	5	5
	Edit calendar event	2	5	5
	View calendar	2	5	5
Voting	Create vote	2	5	5
	Add public vote	2	5	5
	Voting invite user	2	5	5
	Vote	2	5	5
	Get my votings	2	5	5
	Get voting results	2	5	5
Wall	Send wall post	2	5	5
	Comment wall post	2	5	5
	View own wall	2	5	5
	View friend's wall	2	5	5

* Two minute pause after completion of the repetitions for each test case.

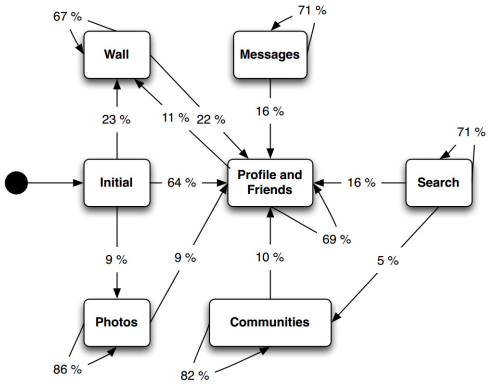


Figure 4. Transition probabilities.

4.1.3. Scalability/Stability

This test scenario focuses on realizing a network with 1000 nodes to evaluate the system scalability and stability. Tables 2 and 4 shows the system parameters and the workload for this scenario. The test aims to show the ability of the network to smoothly scale up and the effect of churn on the network and proceeds as a single test in two phases as follows.

(i) Phase 1-Network growth: The test begins with 250 nodes, which is incremented in steps of 250 nodes representing 100%, 50% and 33.3% increments, respectively. At each step, the workload shown in Table 4 is executed.

(ii) Phase 2-Network churn: Once we reach 1000 nodes, i.e., 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. Similar to the growth phase, after each churn, the workload is again executed.

Table 4. 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 completion of the repetitions for each test case.

4.1.4. Replication Factor

The replication factor (RF) is responsible for the number of duplicate items in the underlying network so that in case a peer leaves the network, any previously uploaded files can be accessed by other peers. In this test scenario the replication factor and its effects are considered. The system parameters are presented in Table 2 and the workload parameters which are defined in the test plan are shown in Table 5.

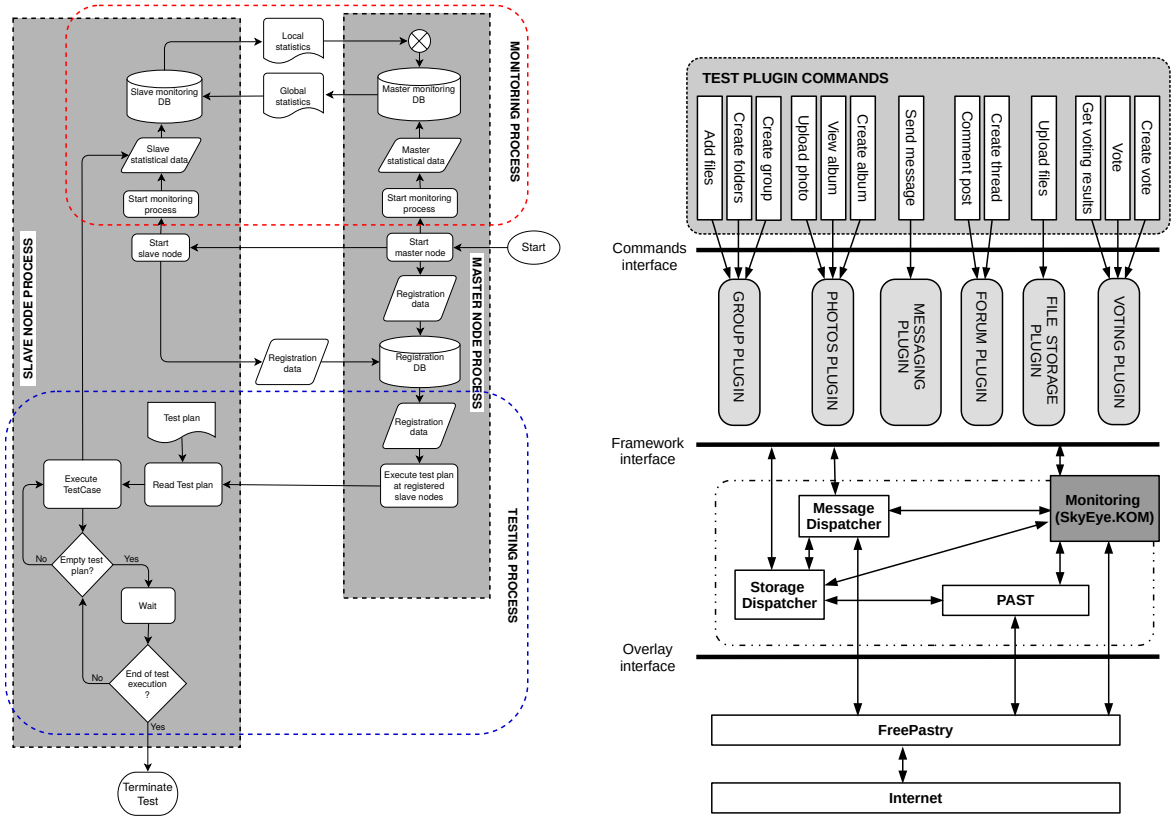
Table 5. Replication Workload.

Plugin	Action	Repetitions *(Count)*	Duration * *(mins)*
Messaging	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 completion of the repetitions for each test case.

4.2. Test Execution and Data Collection

In Figure 5 the setup of the test is portrayed. During testing, we differentiate between two roles for the nodes in the network, the *master* and the *slave* nodes. There can only exist a single master node in the P2P network during the testing, and it must also be the first node that is started. Once the network is established, the master node sends the signal to begin testing to the slave nodes, and does not execute any test cases itself. Slave nodes then bootstrap onto the network formed by the master node. To perform any test, there must be at least $k + 1$ 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.



(**a**) Flowchart of testplan execution (**b**) Configuration at peer level

Figure 5. Test setup.

The test plan is a simple text file with test cases listed in order a desired order, with the number of repetitions per test case and the total time allotted for a particular test case (including the repetitions). A *test case* is defined as the smallest unit that can be executed during a test and corresponds to a single activity initiated 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 checks that the necessary preconditions defined are met or else meets them first before the actual test case is then executed. The entire process of the test execution with a test plan is shown in Figure 5a. Figure 5b shows the interaction of the test plugin with other plugins at peer level. For each plugin, there are several test cases that can be executed and similarly, respective system metrics that can be collected.

During the test execution process, monitoring data is collected at five seconds intervals and stored in SQL format as it is easy to handle and presents minimal load to the test instances. The collected

data is aggregated by the master node and distributed to the slave nodes as global monitoring data. Hence, each node has a local view as well as a global view of the monitoring statistics.

5. Evaluation of Results

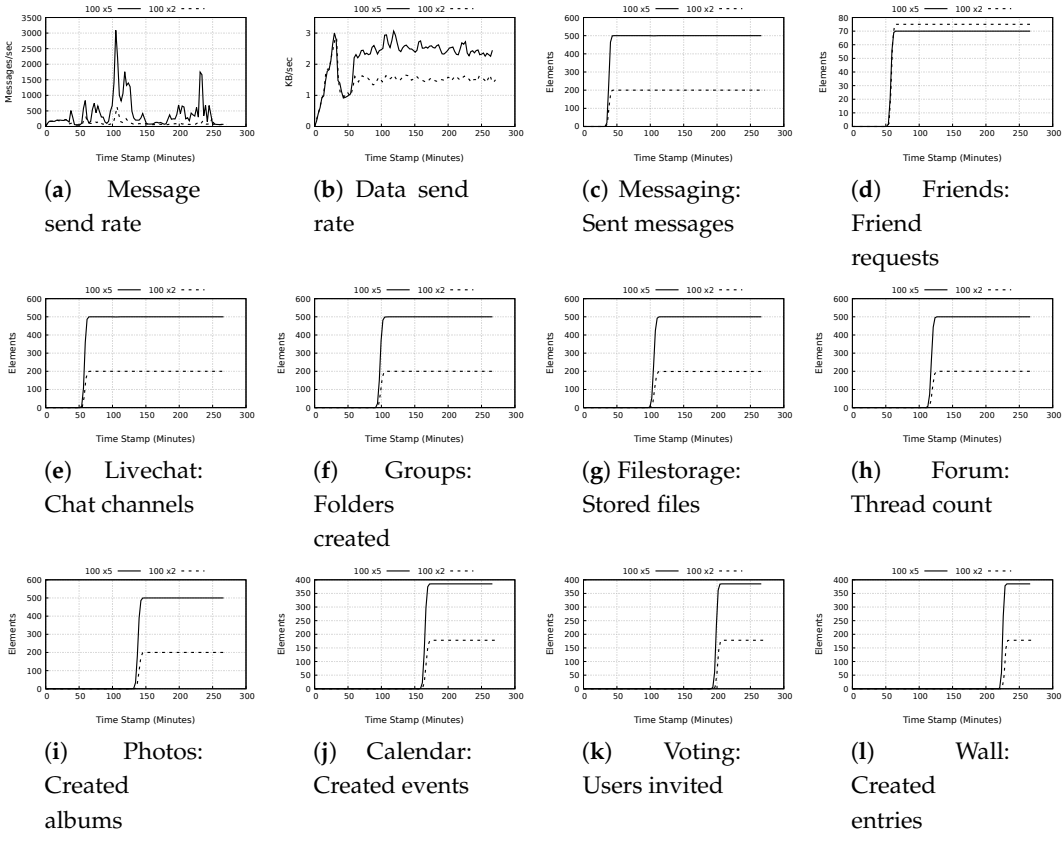
A detailed discussion of the test results obtained from each of the test scenarios follows.

5.1. Baseline Tests: Plugin Analysis

The results are shown in Figures 6 and 7, for the 100 and 500 node tests respectively. Table 6 is a comparative analysis of the completion times, the network messages and network data generated as a consequence of the tests conducted. Table 7 is an analysis of the impact of each plugin action on the message rate and data rate. The results are discussed in the following.

5.1.1. Plugin Actions

Figure 6c–l for the 100 node tests and Figure 7c–l for the 500 node tests, are representative of the results due to the workload in Table 3. We include one additional plugin, the Friends plugin (Figures 6 and 7), from which we can deduce that the nodes actually form friendship links with each other. Because of these links, the nodes are able to send direct messages to friends, establish livechat sessions or post comments on the walls of their established friends. The graphs are testament to the usefulness and accuracy of the monitoring component in particular as it is easy to see the difference due to the different workloads as well as the network sizes.



(**a**) Message send rate (**b**) Data send rate (**c**) Messaging: Sent messages (**d**) Friends: Friend requests

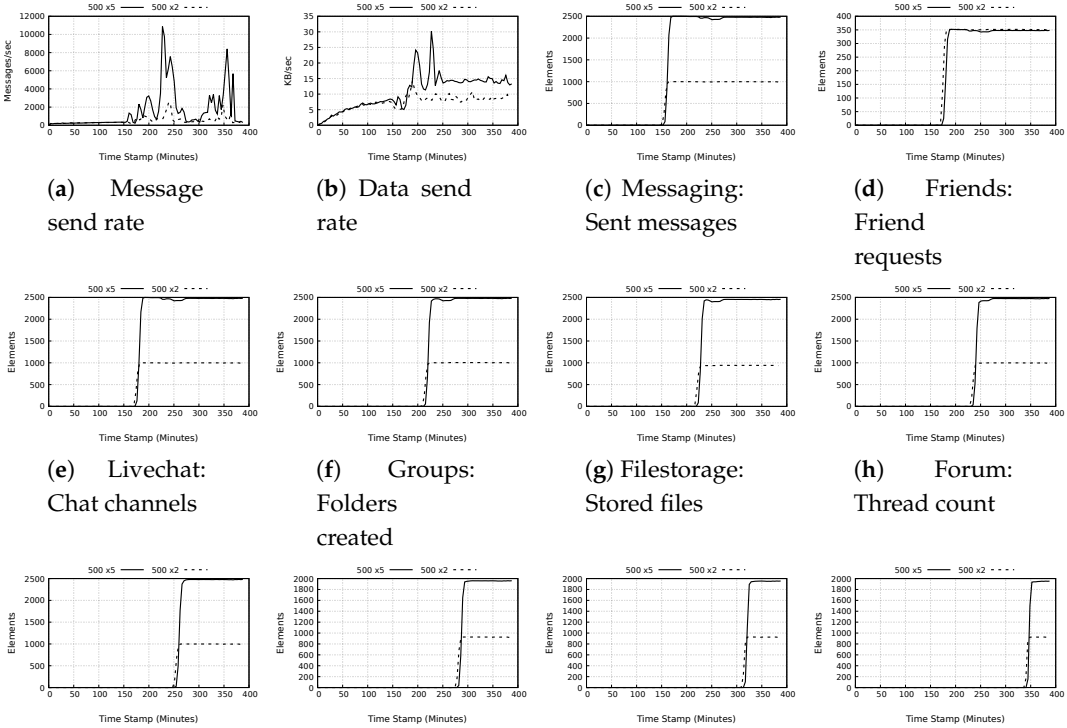
(**e**) Livechat: Chat channels (**f**) Groups: Folders created (**g**) Filestorage: Stored files (**h**) Forum: Thread count

(**i**) Photos: Created albums (**j**) Calendar: Created events (**k**) Voting: Users invited (**l**) Wall: Created entries

Figure 6. Analysis of plugin for 100 Nodes tests.

Table 6. Plugin test analysis.

Plugins	Actions	Max. Completion Time (secs)				Network Messages Generated				Network Data Generated (MBs)			
		100 Nodes		500 Nodes		100 Nodes		500 Nodes		100 Nodes		500 Nodes	
		Light	*Medium*	*Light*	*Medium*	*Light*	*Medium*	*Light*	*Medium*	*Light*	*Medium*	*Light*	*Medium*
Messaging	Send message	15	35	5	15	101,820	169,408	694,239	1,147,379	0.152	0.191	4.206	4.907
	View inbox	15	35	15	15	3463	4466	14840	40446	0.048	0.061	0.256	0.670
	View outbox	15	35	15	35	3310	4107	12,820	18,342	0.045	0.054	0.211	0.303
Livechat	Invitations	15	35	15	35	187,904	268,320	857,484	1,248,916	0.169	0.229	3.464	3.644
	Send message	15	55	15	35	13,996	21,793	79,543	83,693	0.081	0.196	0.474	0.987
Group	Add public group	15	55	15	55	112,349	275,182	586,848	1,434,662	0.125	0.290	4.007	8.124
	Invite user	25	55	15	75	95,112	225,731	264,031	1,218,640	0.116	0.224	0.514	4.886
	View group	15	30	15	30	29,279	85,149	172,107	518,014	0.080	0.137	0.805	1.432
Filestorage	Create folder	25	45	25	55	64,227	160,869	324,113	885,438	0.102	0.182	1.023	1.819
	Store files	15	90	5	100	352,122	1,074,538	1,742,799	5,427,956	0.172	0.484	7.614	11.665
	View folder	15	80	15	75	64,324	258,735	380,631	1,485,086	0.076	0.232	0.491	1.153
Forum	Create thread	25	75	25	75	210,476	547,374	1,074,085	2,917,062	0.138	0.344	2.758	3.473
	Comment post	25	95	25	95	164,276	538,011	898,757	2,173,601	0.117	0.276	1.466	2.339
	View forum	10	35	10	30	17,943	66,398	186,365	437,562	0.075	0.168	1.113	1.328
Photo	Create album	15	45	15	45	31,916	76,467	162,415	240,881	0.095	0.186	0.789	0.651
	Upload photo	15	55	15	55	57,109	141,724	309,335	728,483	0.090	0.243	1.097	2.026
	View own album	15	20	10	25	5373	9407	19,843	84,185	0.082	0.136	0.500	1.363
	View friend's album	5	55	15	45	4972	10,899	9583	55,011	0.074	0.176	0.280	1.011
Calendar	Create event	15	45	5	45	16,168	31,192	89,276	132,274	0.085	0.202	0.559	1.129
	Edit event	15	45	5	55	14,734	29,985	81,296	155,881	0.079	0.190	0.514	1.437
	View calendar	10	20	10	25	3893	4668	17,044	59,337	0.074	0.132	0.355	1.105
Voting	Create vote	15	35	15	45	62,783	120,471	365,981	632,771	0.092	0.197	1.085	1.503
	Invite user	15	45	15	65	56,349	191,021	307,391	1,152,736	0.098	0.198	0.699	1.845
	Vote	15	55	15	75	60,632	201,765	353,137	1,200,553	0.087	0.224	0.508	1.607
	View voting list	10	30	10	35	15,092	71,721	100,671	446,388	0.081	0.154	0.616	1.113
	Get voting results	15	45	15	35	29,753	144,261	191,097	811,686	0.081	0.159	0.386	0.883
Wall	Create entry	15	45	15	55	94,473	179,048	472,950	940,023	0.106	0.237	0.969	1.937
	Comment post	15	75	25	80	121,486	524,681	855,278	2,876,806	0.090	0.245	1.263	1.764
	View own wall	10	35	15	30	30,130	162,590	190,977	962,130	0.089	0.152	0.532	1.457
	View friend's wall	10	25	10	30	32,085	185,969	263,153	1,108,552	0.072	0.152	0.849	1.341



(**a**) Message send rate

(**b**) Data send rate

(**c**) Messaging: Sent messages

(**d**) Friends: Friend requests

(**e**) Livechat: Chat channels

(**f**) Groups: Folders created

(**g**) Filestorage: Stored files

(**h**) Forum: Thread count

(**i**) Photos: Created albums

(**j**) Calendar: Created events

(**k**) Voting: Users invited

(**l**) Wall: Created entries

Figure 7. Analysis of plugin for 500 Nodes tests.

Table 7. Network data and message rates for each plugin action.

Impact	Plugin::Action	Message Rate (msg/sec)				Data Rate (Kb/sec)			
		100 Nodes		500 Nodes		100 Nodes		500 Nodes	
		Light	*Medium*	*Light*	*Medium*	*Light*	*Medium*	*Light*	*Medium*
Low messaging, low data	Calendar::Create Event	101.05	164.17	595.17	696.18	0.54	1.09	3.82	6.08
	Calendar::Edit Event	92.09	157.82	541.97	779.41	0.51	1.02	3.51	7.36
	Calendar::View Calendar	25.12	28.29	109.96	349.04	0.49	0.82	2.35	6.66
	Livechat::Send Message	87.48	108.97	497.14	464.96	0.52	1.00	3.03	5.61
	Messaging::View Inbox	21.64	24.81	92.75	252.79	0.31	0.35	1.64	4.29
	Messaging::View Outbox	20.69	22.82	80.13	101.90	0.29	0.31	1.35	1.72
	Photo::View Friend's Album	33.15	54.50	59.89	289.53	0.51	0.90	1.79	5.45
	Photo::View Own Album	33.58	57.01	128.02	495.21	0.52	0.84	3.30	8.21
High messaging, low data	Filestorage::Create Folder	377.81	846.68	1906.55	4427.19	0.61	0.98	6.16	9.31
	Filestorage::View Folder	402.03	1149.93	2378.94	6750.39	0.49	1.06	3.14	5.37
	Forum::Comment Post	966.33	2241.71	5286.81	9056.67	0.70	1.18	8.83	9.98
	Forum::View Forum	115.76	368.88	1202.35	2500.35	0.50	0.96	7.35	7.77
	Group::View Group	182.99	486.57	1075.67	2960.08	0.51	0.80	5.15	8.38
	Photo::Create Album	199.48	402.46	1015.09	1267.79	0.61	1.00	5.05	3.51
	Photo::Upload Photo	356.93	708.62	1933.34	3642.42	0.58	1.24	7.02	10.37
	Voting::Create Vote	392.39	669.28	2287.38	3330.37	0.59	1.12	6.94	8.10
	Voting::Get Voting Results	185.96	759.27	1194.36	4509.37	0.52	0.86	2.47	5.02
	Voting::Invite User	352.18	1005.37	1921.19	5489.22	0.63	1.07	4.47	9.00
	Voting::View Voting List	97.37	409.83	649.49	2479.93	0.54	0.90	4.07	6.33
	Voting::Vote	378.95	1008.83	2207.11	5457.06	0.56	1.15	3.25	7.48
	Wall::Comment Post	759.29	2384.91	5031.05	12,785.80	0.58	1.14	7.61	8.03
	Wall::Create Entry	590.46	942.36	2955.94	4700.12	0.68	1.28	6.20	9.92
	Wall::View Friend's Wall	207.00	1093.94	1697.76	6334.58	0.48	0.92	5.61	7.85
	Wall::View Own Wall	194.39	903.28	1193.61	5497.89	0.59	0.86	3.40	8.53
High messaging, high data	Filestorage::Store Files	2200.76	4572.50	11,618.66	22,154.92	1.10	2.11	51.98	48.75
	Forum::Create Thread	1238.09	2488.06	6318.15	13,259.37	0.83	1.60	16.61	16.17
	Group::Add Public Group	702.18	1375.91	3667.80	7173.31	0.80	1.48	25.64	41.59
	Group::Invite User	559.48	1128.66	1681.73	5539.27	0.70	1.15	3.35	22.74
	Livechat::Invitations	1174.40	1490.67	5359.28	6938.42	1.08	1.30	22.17	20.73
	Messaging::Send message	636.38	941.16	4628.26	7171.12	0.97	1.09	28.71	31.40

5.1.2. Test Completion Times

Table 6 presents the time taken for each individual test case to complete, with the monitoring capturing data in intervals of 5 s. All tests completed within the allotted test time (5 min), with the maximum completion times for the light load and medium workload generally oscillating around 15 s (7.5 sec/action) and 55 s (11 sec/action) respectively. The longest maximum completion times are observed in the 500 node test under medium workload for Filestorage—store files (100 s); Forum—comment post (95 s); Wall—comment post (80 s); and Voting—vote (75 s). It is also noteworthy that there is generally little or no disparity in completion times for the same workload despite increasing number of nodes, and an increase in the workload results in longer completion times as would be expected. The completion times point to a well balanced network that ensures that there are few overall delays experienced.

5.1.3. Message and Data Rates Characteristics

Figure 6a,b for the 100 node tests, and Figure 7a,b for the 500 node tests portray the network message and data rates throughout the test duration. From the monitoring data collected, the number of messages and the amount of data due to the plugin actions was deduced and this is presented in Table 6. The message and data rates for the duration of each test is then presented in Table 7, from which three distinct cluster groups are visible when comparing the message rates and data rates consequent to the different test cases. We discuss the three groups and base the classification against the 500 node test taking into account the medium workload.

a) Low message and low data rates: These test cases result in lower than 1000 messages/sec with data rates not exceeding 10 Kb/s. The majority of the test in this groups focus on data retrieval as opposed to acting on the data. With the use of the available caching mechanisms, it is expected that there will be only minimal messaging as well as network calls. The exception to this observation is the Create Calendar action, which is a local action rather than a global action

with data replication for redundancy, and the livechat message which sends a short message that does not include large data. The actions in this class are associated with four plugins, Calendar, Livechat, Messaging and Photos.

b) High message and low data rates: The test cases classified in this group are characterized as having message rates above 100 and less than 15,000 messages/sec with data rate not exceeding 15 Kb/s. The plugin actions in this class are associated with Filestorage, Forum, Photo, Voting and Wall plugins. Of particular interest is the Forum, Photo voting and wall plugins because they use DDSs to store data. Most of the plugin actions in this category make repeated calls for the DDSs relating to the data requested with each call resulting in significant messaging as the entire DDS is retrieved.

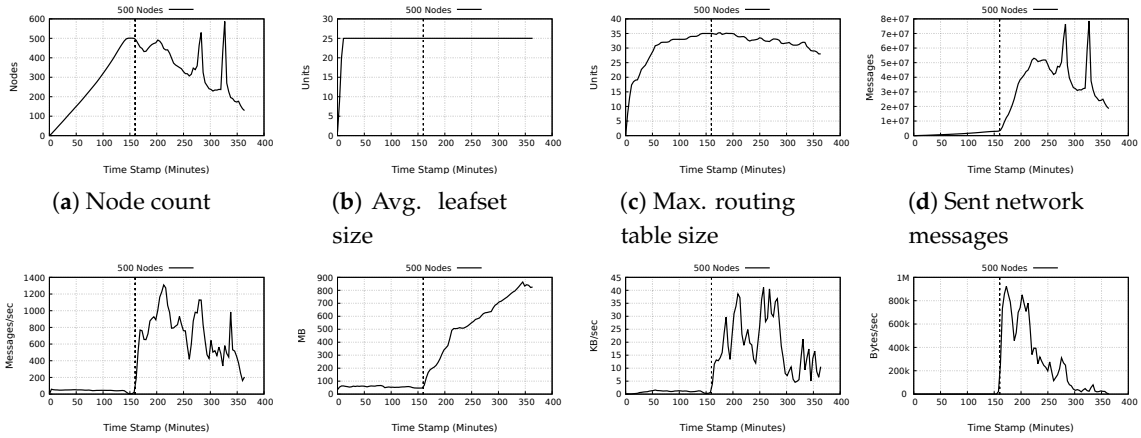
c) High message and high data rates: These tests have the largest overall impact on the message and data rates. The message rates range from 500 to slightly less than 23,000 messages/sec. The highest data rates recorded is 51.98 Kb/s observed for the plugin Filestorage store files action in the 500 node test with light workload. The actions in this classification are associated with Filestorage, Forum, Group, Messaging and Livechat. The actions, such as store files by the Filestorage plugin, or send message by the Messaging plugin generate high messages as well as data because they use DDSs.

LibreSocial's modular design, which allows separation of the various OSN functions into plugins (or bundles) from the P2P core service components, supports the implementation of SkyEye [38,39], a tree-based monitoring solution. From the collected monitoring statistics, it is demonstrated that there is a smooth synergy between the application layer and the underlying components, seen by the effect of each plugin on the data and messaging rates. The worst case test completion times for each plugin action are tolerable being on average about 7.5 and 11 secs/action for the light and medium loads irrespective of the network size. These times may be caused by the replication factor value and lower times can be anticipated with lower replication factor values, although such a move may significantly affect the retrieval times as will be demonstrated in the replication test (Section 5.4). The message and data rates are dependent on the plugin action performed. Actions that require storage of significant data tend to generate more messages as well as data overall, such as uploading files or photos. Also because of the manner in which DDSs are stored (distributed across the network), there is a significant messaging as the DDS items are being retrieved, which is expected. This interaction renders the conclusion that the storage mechanism composed of both the PAST for simple files as well and the DDS for complex data types, coupled with the caching and replication, integrate well within the application. With a maximum recorded network data rate 51.98 and 48.75 Kb/sec for the light and medium load tests, we believe that the network is capable of handling more and can easily scale up.

5.2. Pseudo-Random Behavior

The evaluation looks at the node count, routing maintenance (leafset and routing table sizes), messaging (count and rate), memory usage size, network data rate and DDS data retrieval rate. Figure 8 shows the results from this test. Figure 8a portrays the active nodes in the network. There is a notable gradual decline in the network composition over the experimental period. This may probably be directly attributable to two aspects: the randomization algorithm and the absence of new participants joining the network. The leafset size, seen in Figure 8b, is stable at 25 throughout the experimental period, as is expected. On the other hand, the routing table size shown in Figure 8c depicts an increase as the network grows, with the maximum size of 35 at the end of the network joining phase, followed by a gradual decline corresponding to the network size reduction. FreePastry's routing management algorithm performs cleanup by link reorganization, i.e., as one link becomes unusable, the algorithm greedily selects a new link, since several paths exist between any two nodes. Hence, from our observation, we deduce the presence of dynamic route readjustments, and proactive refreshing of the routing table. The network messages and network messaging rate as shown in Figure 8d,e respectively are indicative of a increased network activity as the random action of the plugins begin

(roughly after the 160th minute), 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 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 8f 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 8g, is recorded at about 40 KB/s. In general, the data rate oscillates between 10 and 30 KB/s, which we consider acceptable performance for a network of this size, and also because of the data used during the testing process, which was generally less than 1 MB. 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 8h.



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

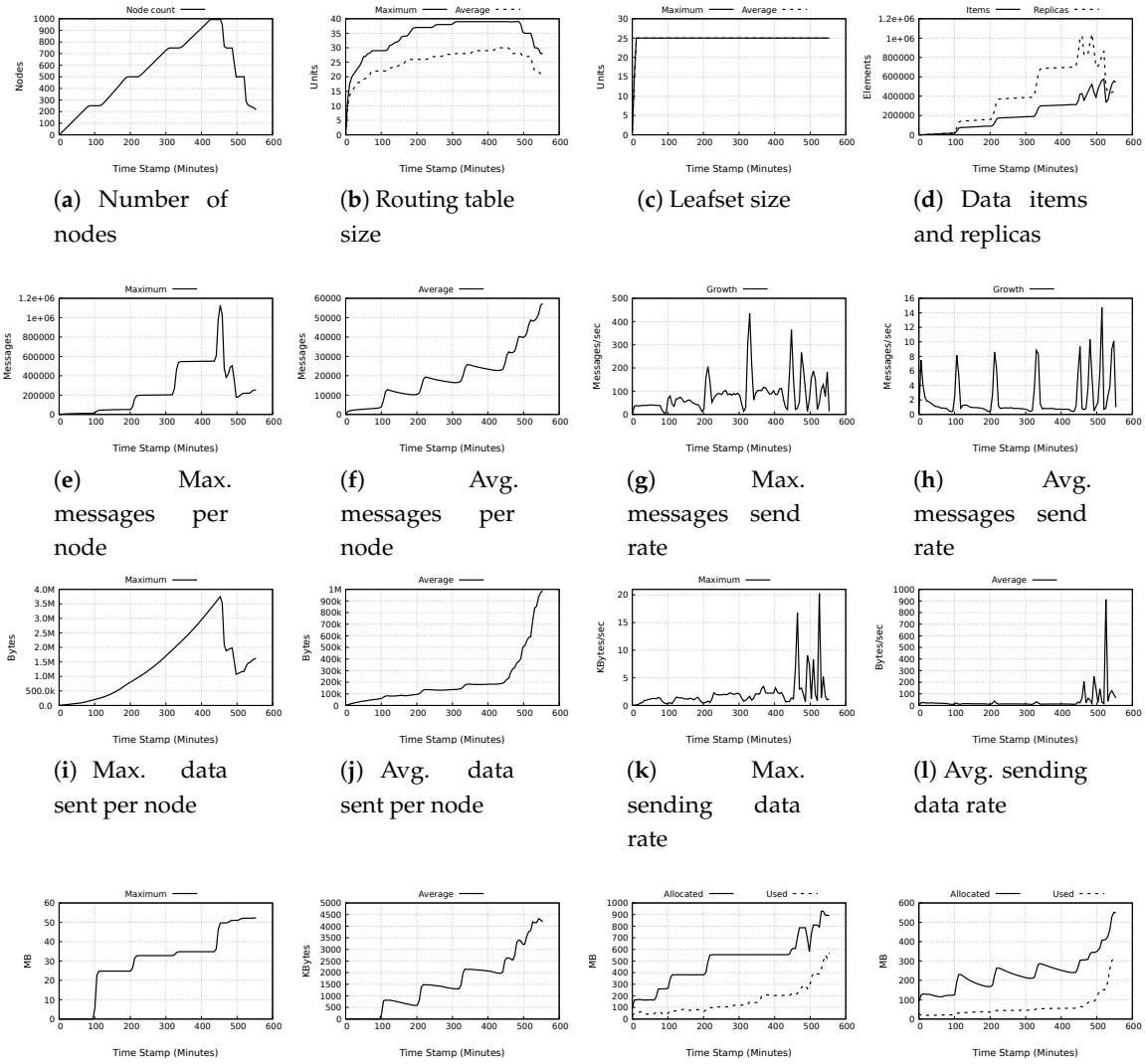
(**e**) Max. network message send rate (**f**) Max. network memory used (**g**) Max. send data rate (**h**) Max. DDS retrieval rate

Figure 8. Realistic user behavior.

The results reveal the usability of LibreSocial in the real world. The social network is stable and the monitoring component captures network statistics throughout the test period, which helps in gathering insights 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. From the results of this test, we may conclude that the randomization algorithm may have a part to play in the eventual reduction in the network size over the test duration. It is also probable that there may be other factors that may have caused the network to diminish other than the randomization algorithm. We believe that the randomization model is a good portrayal of realistic human interaction and hence, such a significant drop in the network size is not to be expected. This calls for further investigation into how the model interacts with the application in triggering the test cases.

5.3. Stability and 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 are carried out in the same experiment, it is easy to make observations and deductions on system behavior in both situations and contrast them. We discuss the results shown in Figure 9 in the following.



(**a**) Number of nodes

(**b**) Routing table size

(**c**) Leafset size

(**d**) Data items and 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 and used mem (max)

(**p**) Total and used mem (avg)

Figure 9. Stability and scalability analysis.

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

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 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 and 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 nonetheless, some are still present.

e) Messages sent: The maximum and average messages sent in a single node are shown in Figure 9e,f 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 25,000 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 200,000 messages, the average messages show a sharp increase to a value of 58,000 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 Figure 9g,h 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 Figure 9i,j. 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 as 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: Figure 9k,l 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 as 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.

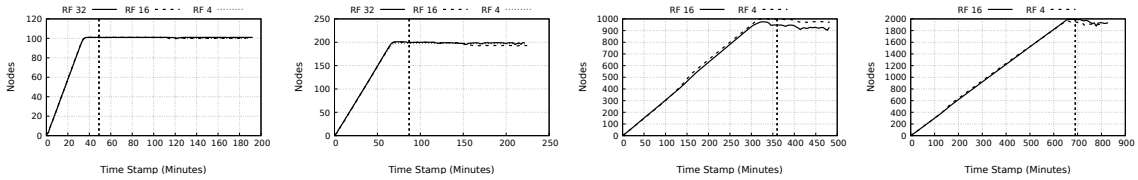
i) DDS data stored: The maximum and average DDS data stored are shown in Figure 9m,n, respectively. During the growth phase, there is an increase in the DDS data after the first growth step. No DDS data is recorded during the first growth phase as the workload is yet to be executed. At the end of the 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 and used memory: A comparison of the maximum and average values for total allocated and used memory is shown in Figure 9o,p, 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 has a large impact on the memory allocated. The maximum memory allocated is 900 MB, and the average maximum is about 550 MB. However, the memory that is used is less than half of the allocated memory. The maximum used is about 580 MB and the average maximum used memory is about 300 MB.

From the results, we can conclusively state that the network can scale up to at least 1000 nodes. There is a smooth growth of the network which remains stable even during the churn phase. There is an evident dynamic readjustment of the core network management protocols, such as the routing table entries and possibly the leafset entries, to accommodate the changing network in both the growth and churn phases. During the growth phase, the shared network memory is increased proportionally to match the demand and nodes are added to the routing table. During the churn phase, the nodes make adjustments by taking on more load, shown by an increase in memory rather than a decrease. Despite the significant reduction in number of nodes during churn (up to 50% network size reduction), the routing's link reorganization mechanism ensures that the routing table adjusts accordingly and the leafset remains stable. During the tests, the ability to increase the number of nodes was only limited by the resources available in the HPC cluster in terms of memory for the entire setup. Beyond this limitation, it is possible to scale up the network without any hindrance. Additionally, as long as a significant number of nodes is still online (possibly several hundred nodes), the network can adjust itself when there is a high churn rate.

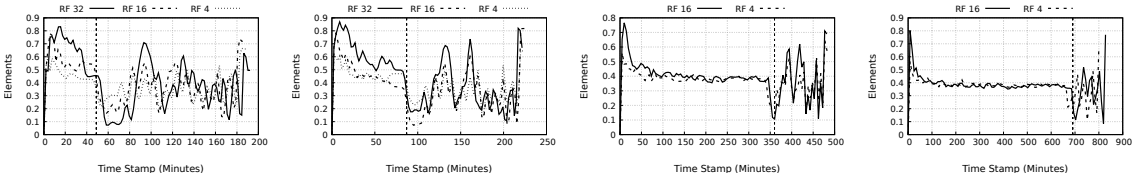
5.4. Effect of the Replication Factor

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 8. We considered the performance of the system against the costs and discuss the relevant metrics in the following.



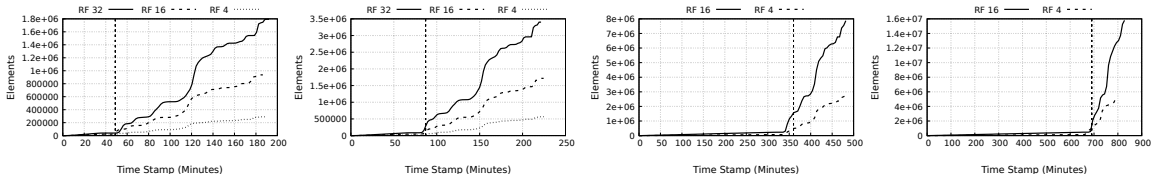
(**a**) 100 nodes (**b**) 200 nodes (**c**) 1000 nodes (**d**) 2000 nodes

Figure 10. Node count analysis.



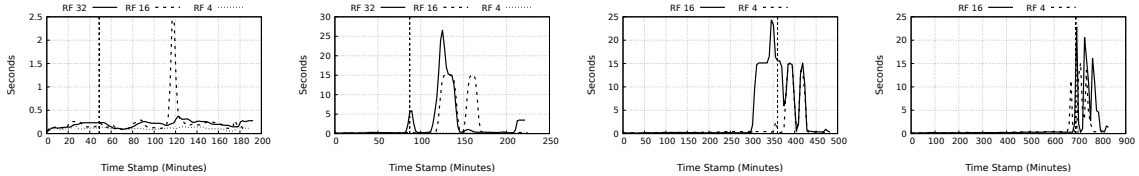
(**a**) 100 nodes (**b**) 200 nodes (**c**) 1000 nodes (**d**) 2000 nodes

Figure 11. Mean hop count analysis.



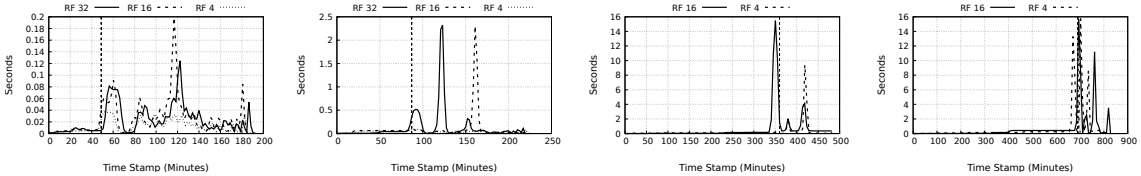
(**a**) 100 nodes (**b**) 200 nodes (**c**) 1000 nodes (**d**) 2000 nodes

Figure 12. Number of replicas analysis.



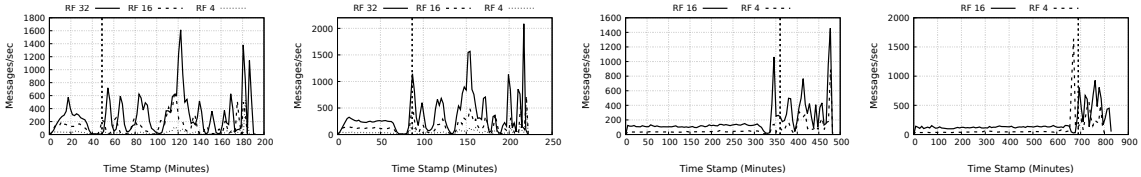
(**a**) 100 nodes (**b**) 200 nodes (**c**) 1000 nodes (**d**) 2000 nodes

Figure 13. Maximum storage time analysis.



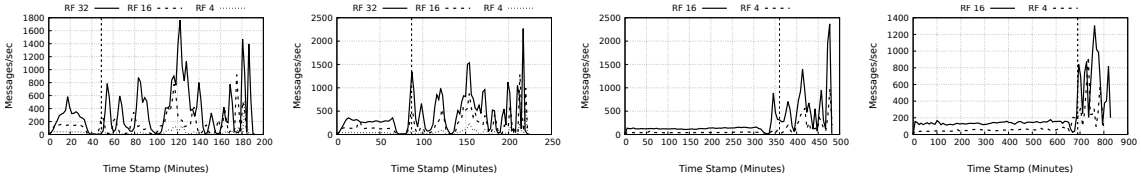
(**a**) 100 nodes (**b**) 200 nodes (**c**) 1000 nodes (**d**) 2000 nodes

Figure 14. Maximum retrieval time analysis.



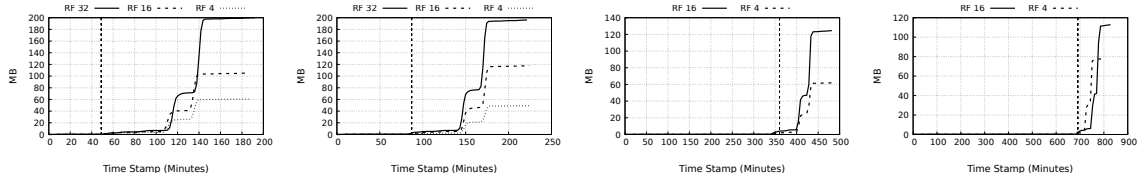
(**a**) 100 nodes (**b**) 200 nodes (**c**) 1000 nodes (**d**) 2000 nodes

Figure 15. Maximum message send rate analysis.



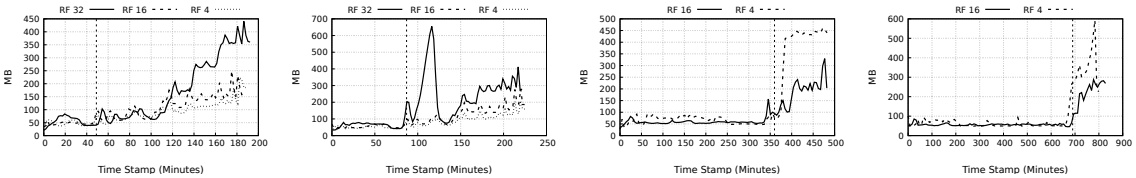
(**a**) 100 nodes (**b**) 200 nodes (**c**) 1000 nodes (**d**) 2000 nodes

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.

5.4.1. 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: Figure 10a–d portray the behavior nodes in the systems. The figures indicate successful monitoring of the network nodes, and in general the network, which is stable throughout the experimental period. Of note, is the slight reduction by about 100 nodes for both the 1000 and 2000 nodes experiments. A possible reason for this, evident from the logs, is that the nodes experienced insufficient memory errors resulting in overall node failure. However, this did not affect the aggregation of monitored statistics in the network.

b) Hop count: The hop count for the experiments are given in Figure 11a–d. 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 Figure 12a–d, 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 expected.

d) Storage time: From Figure 13a–d, 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 8, are less than one second, but we note the extreme cases of maximum storage time in the order of 10 s of seconds, particularly the 2000 node experiment with replication factor of 4, which recorded a maximum storage time of 91 s. 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 Figure 14a–d. 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 system retrieval times are still less than one second. A significant observation is that for the 100 node tests, for replication factor of 16, a higher maximum retrieval time of 15.12 s is seen, in comparison to 0.189 and 0.6 s for replication factors 4 and 32 respectively. It is probable this is the result of a DDS data retrieval and is an outlier.

f) Message send/receive rate: From Figure 15a–d, for message sending rates and Figure 16a–d 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.

5.4.2. Cost Analysis

The three aspects to the costs incurred to be evaluated are storage space usage, memory usage and network data rate. The analysis follows.

a) Used storage space: From Figure 17a–d, 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 Figure 18a–d, 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 use for replication factor 4 and 16, but a replication factor of 32 shows a notably large maximum memory usage. For the 1000 and 2000 node tests, with lower replication factor of 4, there is 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, i.e., increase in replication factor causes an increase in memory consumption.

c) Network bandwidth: The last four columns of Table 8 show the average and maximum sending and receiving network data rates 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 shown by the increasing maximum data rates. The average data rates for both sending and receiving are 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.

Table 8. Replication Factor Analysis.

Nodes	RF	PERFORMANCE									COST							
		Average Hops	t_{retr} (secs)		t_{store} (secs)		m_{send} (msgs/s)		m_{rcv} (msgs/sec)		Used Storage (MB)		Used Memory (MB)		Data Rate (KB/s)			
			Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Send*		*Receive*	
															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	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	910	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

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 are much smaller than one second. Storage and retrieval times below one second are essential in meeting

the user's needs in terms of quality of experience. 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. Conclusions

As opposed to many other system tests carried out in the form of simulation tests, in this evaluation of LibreSocial, a P2P framework for OSNs, we endeavored to perform benchmarking for large networks of the actual application, reaching up to 2000 active network nodes. Our tests do not just measure the quality and costs of the system, but also demonstrate the possibility of having a wide set of OSN functions that work well together in our P2P solution. We demonstrate clearly that the plugins constituting the OSN application integrate well with framework layer which provides the required P2P elements for the plugins. Our contention is that other solutions do not have this proof and also not this range of functionality. In addition, to our knowledge, such large tests are heretofore yet to be performed with purely P2P-based OSNs, with a majority of the tests being simulations such as [19,21,42–44]. The evaluation in general presents very insightful information. The P2P framework, and the OSN application functions designed in the form of plugin extension works well. We clearly show how these OSN plugins impact the messaging and data rates in the scheme of the overall network. In addition, tests were modeled to mimic the real environment for random behavior characteristics, its ability to scale up and remain stable under high churn, as well as review the effect of the replication factor on the system performance. The system's performance in a pseudo-real environment based on our randomization model was satisfactory, showing that the system works quite well with minimal errors. However, the network was observed to diminish, a behavior that is attributed to the design of the randomization algorithm. During the scalability and stability testing, there was evidence of dynamic adjustments to suit the situation in terms of memory requirements, routing readjustments and replica diversion as well as placement, which renders to the ability of the P2P OSN to smoothly scale up and also continue to function well under high churn. 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. This non-linear behavior due to the replication factor on system performance may require a separate data-driven study to determine the optimal replication factor and ascertain the reason(s) for such any optimal behavior observed. In conclusion, we believe that this benchmark can be a useful reference source for future P2P OSNs in terms of performance and cost analysis.

Author Contributions: Conceptualization, N.M., L.K., I.D. and K.G.; methodology, N.M., L.K, I.D. and K.G.; formal analysis, L.K. and I.D.; investigation, N.M., L.K. and I.D.; writing—original draft preparation, N.M. and K.G.; writing—review and editing, N.M. and K.G.; supervision, N.M. and K.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: Computational support and infrastructure was provided by the "Centre for Information and Media Technology" (ZIM) at the University of Dusseldorf (Germany).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DDS	Distributed data structure
ECC	Elliptic curve cryptography
GUI	Graphical user interface
HPC	High performance computing
KB	Kilobyte
MB	Megabyte
MVVM	Model-View-View-Model
OSGi	Open Services Gateway Initiative
OSN	Online social network
P2P	Peer-to-peer
RF	Replication factor
SN	Social network

References

1. Subrahmanyam, K.; Reich, S.M.; Waechter, N.; Espinoza, G. Online and offline social networks: Use of social networking sites by emerging adults. *J. Appl. Dev. Psychol.* **2008**, *29*, 420–433. doi:10.1016/j.appdev.2008.07.003.
2. Guidi, B.; Conti, M.; Ricci, L. P2P architectures for distributed online social networks. In Proceedings of the IEEE International Conference on High Performance Computing & Simulation (HPCS) 2013, Helsinki, Finland, 1–5 July 2013; pp. 678–681. doi:10.1109/HPCSim.2013.6641493.
3. Lakshman, A.; Malik, P. Cassandra: A Decentralized Structured Storage System. *SIGOPS Oper. Syst. Rev.* **2010**, *44*, 35–40. doi:10.1145/1773912.1773922.
4. Beaver, D.; Kumar, S.; Li, H.C.; Sobel, J.; Vajgel, P. Finding a Needle in Haystack: Facebook's Photo Storage. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*; OSDI'10; USENIX Association: Berkeley, CA, USA, 2010; pp. 47–60.
5. Spiekermann, S.; Acquisti, A.; Bohme, R.; Hui, K.L. The challenges of personal data markets and privacy. *Electron. Mark.* **2015**, *25*, 161–167. doi:10.1007/s12525-015-0191-0.
6. Taddei, S.; Contena, B. Privacy, trust and control: Which relationships with online self-disclosure? *Comput. Hum. Behav.* **2013**, *29*, 821–826. doi:10.1016/j.chb.2012.11.022.
7. Tucker, C.E. Social networks, personalized advertising, and privacy controls. *J. Mark. Res.* **2014**, *51*, 546–562.
8. Barreda, A.A.; Bilgihan, A.; Nusair, K.; Okumus, F. Generating brand awareness in Online Social Networks. *Comput. Hum. Behav.* **2015**, *50*, 600–609. doi:10.1016/j.chb.2015.03.023.
9. Bakir, V.; McStay, A. Fake News and The Economy of Emotions. *Digit. Journal.* **2018**, *6*, 154–175. doi:10.1080/21670811.2017.1345645.
10. Roy, R.; Gupta, N., Digital Capitalism and Surveillance on Social Networking Sites: A Study of Digital Labour, Security and Privacy for Social Media Users. In *Digital India: Reflections and Practice*; Kar, A.K., Sinha, S., Gupta, M.P., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; pp. 67–81. doi:10.1007/978-3-319-78378-9_4.
11. Brandimarte, L.; Acquisti, A.; Loewenstein, G. Misplaced Confidences: Privacy and the Control Paradox. *Soc. Psychol. Personal. Sci.* **2013**, *4*, 340–347. doi:10.1177/1948550612455931.
12. Paul, T.; Buchegger, S.; Strufe, T. Decentralizing Social Networking Services. *Int. Tyrrhenian Workshop Digit. Commun.* **2010**, *1230*, 1–10.
13. Buchegger, S.; Datta, A. A case for P2P Infrastructure for Social Networks-Opportunities & Challenges. In Proceedings of the 2009 Sixth International Conference on Wireless On-Demand Network Systems and Services, Snowbird, Utah, 2–4 February 2009.
14. Buford, J.F.; Yu, H., Peer-to-Peer Networking and Applications: Synopsis and Research Directions. In *Handbook of Peer-to-Peer Networking*; Shen, X., Yu, H., Buford, J., Akon, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Chapter 1, pp. 3–45. doi:10.1007/978-0-387-09751-0_1.
15. Rodrigues, R.; Druschel, P. Peer-to-Peer Systems. *Commun. ACM* **2010**, *53*, 72–82. doi:10.1145/1831407.1831427.

16. Buchegger, S.; Schioberg, D.; Vu, L.H.; Datta, A. PeerSoN: P2P Social Networking: Early Experiences and Insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*; SNS '09; ACM: New York, NY, USA, 2009; pp. 46–52. doi:10.1145/1578002.1578010.
17. Cutillo, L.A.; Molva, R.; Strufe, T. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Commun. Mag.* **2009**, *47*, 94–101. doi:10.1109/MCOM.2009.5350374.
18. Sharma, R.; Datta, A. SuperNova: Super-peers based architecture for decentralized online social networks. In Proceedings of the IEEE 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012), Bangalore, India, 3–7 January 2012; pp. 1–10. doi:10.1109/COMSNETS.2012.6151349.
19. Jahid, S.; Nilizadeh, S.; Mittal, P.; Borisov, N.; Kapadia, A. DECENT: A decentralized architecture for enforcing privacy in online social networks. In Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Lugano, Switzerland, 19–23 March 2012; pp. 326–332. doi:10.1109/PerComW.2012.6197504.
20. Aiello, L.M.; Ruffo, G. LotusNet: Tunable Privacy for Distributed Online Social Network Services. *Comput. Commun.* **2012**, *35*, 75–88. doi:10.1016/j.comcom.2010.12.006.
21. Guidi, B.; Amft, T.; De Salve, A.; Graffi, K.; Ricci, L. DiDuSoNet: A P2P architecture for distributed Dunbar-based social networks. *Peer-to-Peer Netw. Appl.* **2016**, *9*, 1177–1194. doi:10.1007/s12083-015-0366-7.
22. Graffi, K.; Masinde, N. LibreSocial: A Peer-to-Peer Framework for Online Social Networks. *CoRR* **2020**, arXiv:cs.SI/2001.02962.
23. Graffi, K.; Podrajanski, S.; Mukherjee, P.; Kovacevic, A.; Steinmetz, R. A distributed platform for multimedia communities. In Proceedings of the 2008 Tenth IEEE International Symposium on Multimedia, Berkeley, CA, USA, 15–17 December 2008; pp. 208–213. doi:10.1109/ISM.2008.11.
24. Graffi, K.; Gross, C.; Mukherjee, P.; Kovacevic, A.; Steinmetz, R. LifeSocial. KOM: A P2P-based platform for secure online social networks. In Proceedings of the 2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P), Delft, The Netherlands, 25–27 August 2010; pp. 1–2.
25. Graffi, K.; Gross, C.; Stingl, D.; Hartung, D.; Kovacevic, A.; Steinmetz, R. LifeSocial. KOM: A Secure and P2P-based Solution for Online Social Networks. In Proceedings of the 2011 IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2011; pp. 554–558.
26. Buchert, T. Managing Large-Scale, Distributed Systems Research Experiments with Control-Flows. Ph.D. Thesis, Universite de Lorraine, Lorraine, France, 2016.
27. Lehn, M.; Triebel, T.; Gross, C.; Stingl, D.; Saller, K.; Effelsberg, W.; Kovacevic, A.; Steinmetz, R., Designing Benchmarks for P2P Systems. In *From Active Data Management to Event-Based Systems and More: Papers in Honor of Alejandro Buchmann on the Occasion of His 60th Birthday*; Sachs, K., Petrov, I., Guerrero, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 209–229. doi:10.1007/978-3-642-17226-7_13.
28. Kounev, S. Performance Engineering of Distributed Component—Based Systems. Ph.D. Thesis, Technischen Universitat Darmstadt, Darmstadt, Germany, 2005.
29. Sachs, K.; Kounev, S.; Carter, M.; Buchmann, A. Designing a Workload Scenario for Benchmarking Message-Oriented Middleware. In Proceedings of the 2007 SPEC Benchmark Workshop (SPEC), Dresden, Germany, 22 June 2007.
30. Saller, K.; Panitzek, K.; Lehn, M. Benchmarking Methodology. In *Benchmarking Peer-to-Peer Systems: Understanding Quality of Service in Large-Scale Distributed Systems*; Effelsberg, W., Steinmetz, R., Strufe, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 19–45. doi:10.1007/978-3-642-38673-2_3.
31. Jain, R. *The Art of Computer Systems Performance Analysis—Techniques for Experimental Design, Measurement, Simulation, and Modeling*; Wiley Professional Computing, Wiley: Hoboken, NJ, USA, 1991.
32. Masinde, N.; Graffi, K. Peer-to-Peer based Social Networks: A Comprehensive Survey. *CoRR* **2020**, arXix:2001.02611.
33. Rowstron, A.; Druschel, P. *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. Middleware 2001*; Guerraoui, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 329–350. doi:10.1007/3-540-45518-3_18.
34. Rowstron, A.; Druschel, P. Storage Management and Caching in PAST, a Large-scale, Persistent Peer-to-peer Storage Utility. *SIGOPS Opering Syst. Rev.* **2001**, *35*, 188–201. doi:10.1145/502059.502053.

35. Druschel, P.; Rowstron, A. PAST: A large-scale, persistent peer-to-peer storage utility. In Proceedings of the IEEE Eighth Workshop on Hot Topics in Operating Systems, Elmau, Germany, 20–22 May 2001; pp. 75–80. doi:10.1109/HOTOS.2001.990064.
36. Ramabhadran, S.; Ratnasamy, S.; Hellerstein, J.M.; Shenker, S. Brief Announcement: Prefix Hash Tree. In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*; PODC '04; ACM: New York, NY, USA, 2004; pp. 368–368. doi:10.1145/1011767.1011823.
37. Rowstron, A.; Kermarrec, A.M.; Castro, M.; Druschel, P. Scribe: The Design of a Large-Scale Event Notification Infrastructure. In *Networked Group Communication*; Crowcroft, J., Hofmann, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 30–43. doi:10.1007/3-540-45546-9_3.
38. Graffi, K.; Kovacevic, A.; Xiao, S.; Steinmetz, R. SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems. In Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems, Melbourne, Australia, 8–10 December 2008; pp. 279–286. doi:10.1109/ICPADS.2008.8.
39. Graffi, K.; Disterhoft, A. SkyEye: A tree-based peer-to-peer monitoring approach. *Pervasive Mob. Comput.* **2017**, *40*, 593–610. doi:10.1016/j.pmcj.2017.07.003.
40. Kovacevic, A.; Graffi, K.; Kaune, S.; Leng, C.; Steinmetz, R. Towards Benchmarking of Structured Peer-to-Peer Overlays for Network Virtual Environments. In Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems, Melbourne, Australia, 8–10 December 2008; pp. 799–804. doi:10.1109/ICPADS.2008.68.
41. Benevenuto, F.; Rodrigues, T.; Cha, M.; Almeida, V. Characterizing User Behavior in Online Social Networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*; IMC '09; ACM: New York, NY, USA, 2009; pp. 49–62. doi:10.1145/1644893.1644900.
42. Olteanu, A.; Pierre, G. Towards Robust and Scalable Peer-to-Peer Social Networks. In Proceedings of the Fifth Workshop on Social Network Systems, SNS '12, Bern, Switzerland, 10 April 2012.
43. Nilizadeh, S.; Jahid, S.; Mittal, P.; Borisov, N.; Kapadia, A. Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*; CoNEXT '12; ACM: New York, NY, USA, 2012; pp. 337–348. doi:10.1145/2413176.2413215.
44. Franchi, E.; Poggi, A.; Tomaiuolo, M. Blogracy: A peer-to-peer social network. *Int. J. Distrib. Syst. Technol. (IJDST)* **2016**, *7*, 37–56. doi:10.4018/IJDST.2016040103.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).