Benchmarking Decentralized Monitoring Mechanisms in Peer-to-Peer Systems

Dominik Stingl, Christian Gross,
Sebastian Kaune, Ralf Steinmetz
Multimedia Communications Lab
TU Darmstadt
Darmstadt, Germany
{stingl,chrgross,kaune,
steinmetz}@kom.tu-darmstadt.de

Karsten Saller
Real Time Systems Lab
TU Darmstadt
Darmstadt, Germany
karsten.saller@es.tu-darmstadt.de

ABSTRACT

Decentralized monitoring mechanisms enable obtaining a global view on different attributes and the state of Peer-to-Peer systems. Therefore, such mechanisms are essential for managing and optimizing Peer-to-Peer systems. Nonetheless, when deciding on an appropriate mechanism, system designers are faced with a major challenge. Comparing different existing monitoring mechanisms is complex because evaluation methodologies differ widely. To overcome this challenge and to achieve a fair evaluation and comparison, we present a set of dedicated benchmarks for monitoring mechanisms. These benchmarks evaluate relevant functional and non-functional requirements of monitoring mechanisms using appropriate workloads and metrics. We demonstrate the feasibility and expressiveness of our benchmarks by evaluating and comparing three different monitoring mechanisms and highlighting their performance and overhead.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.4.1 [**Performance of Systems**]: Performance attributes

General Terms

Performance, Standardization

Keywords

Benchmarking, Decentralized Monitoring Mechanisms, Peer-to-Peer Systems, Performance Comparison

1. INTRODUCTION

In the last decade, monitoring of Peer-to-Peer (P2P) systems has gained much research interest resulting in a plethora of different monitoring approaches, each providing different performance characteristics. All approaches have in common, that they reveal general insights about the network and application [24], or summarize the characteristics of the participants [7].

Given the multitude of existing solutions, a fair comparison between several solutions is hard to achieve, if not impossible. This lack of comparability results from the widely differing evaluation methodology for decentralized monitoring mechanisms: (i) although designed for the same purpose with similar functionality, the addressed non-functional requirements vary, (ii) the applied workloads to evaluate the quality of a mechanism differ widely in their composition, and (iii) different metrics are used to quantify the quality of the system.

To overcome this lack of comparability, we present the following contributions:

(i) We identify the relevant non-functional requirements for decentralized monitoring mechanisms for P2P systems, such as scalability and robustness. Given these requirements, we propose a set of benchmarks that investigate how decentralized monitoring mechanisms meet these non-functional requirements. Therefore, each developed benchmark consists of one or several workloads, which evaluate a specific non-functional requirement by a predefined set of appropriate metrics. Based on the provided benchmarks, the quality of decentralized monitoring mechanisms can be evaluated and compared in a reproducible and unbiased way. Furthermore, our benchmarks can be applied to tune the parameter setting of a system to identify an optimal configuration for a particular workload scenario.

(ii) To exemplify our methodology, we present a case study and discuss the benchmarking results of three monitoring approaches (a gossip-based and tree-based approach as well as a simple centralized approach as reference). Thus, we are not interested in declaring one approach "better" or "worse" than another as denoted by Rhea et al. [19], but in showing the applicability and expressiveness of our presented benchmarks.

The rest of this paper is structured as follows: Section 2 provides the background on decentralized P2P monitoring mechanisms followed by Section 3 presenting our benchmarking methodology. The benchmarking results are presented in Section 4. Subsequently, we discuss related work in Section 5, summarize this paper in Section 6 and give an outlook on future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICPE'12, April 22–25, 2012, Boston, Massachusetts, USA
Copyright 2012 ACM 978-1-4503-1202-8/12/04 ...$10.00.

2. DECENTRALIZED MONITORING MECHANISMS

In this section, we give a brief overview on decentralized monitoring mechanisms highlighting their offered functionality and composition. For the design of benchmarks, it is indispensable to understand, which functionality is provided by the considered class of mechanisms, because it influences the identification of the relevant non-functional requirements for this class. Based on these requirements, the different benchmarks can be defined, as outlined in Subsection 3.1. Moreover, the offered functionality serves to design an interface for the class of mechanisms to access and execute the relevant operations during a benchmark.

2.1 Functional Description

A decentralized monitoring mechanism [12,16,23,24] gathers different types of data from the whole system to assess and calculate the *global state* of the system and its participants. The information to collect is represented by a set of *attributes*, measured by every participating peer. Depending on the focus of a monitoring mechanism, the gathered attributes range from the transmitted traffic [18], over application-related information [23], to the user and its utilized communication device [7].

Due to the large number of users, the transmission of the measured attributes and its subsequent collection at one or several data sinks results in a high amount of data, consuming a considerable amount of bandwidth resources in the system. Existing approaches, therefore, apply aggregation of the monitored attributes to compress the size of the data and to save the bandwidth of the participating peers. Using this aggregation, a monitoring mechanism calculates the so-called *global view* of a monitored attribute, which can be subsequently used to deduce the aforementioned global state of the whole system. Typical aggregates that are used for the calculation of a global view cover functions, such as minimum, maximum, sum, average, or standard deviation [3,16]. After the computation of a global view for a set of aggregates, each participating peer in a P2P system can retrieve the newly created information.

2.2 Architectural Description

In the following, we present the architecture of a decentralized monitoring mechanism, which provides the previously mentioned functionality. We sketch how a decentralized monitoring mechanism is composed and integrated into a P2P system. In contrast to the functional description, the information about a mechanism's composition is not mandatory for the design of benchmarks, because we evaluate the whole mechanism and do not study the impact of internal components on the overall behavior. The overview, however, justifies the choice of a tree- and gossip-based approach for the case study in Section 4, because it becomes apparent that the topology mainly influences the behavior of the monitoring mechanism. In the following, we present the three basic components, every monitoring mechanism can be reduced to.

Topology Construction and Maintenance.

In literature, *trees* [16, 24] and *meshes* [11, 12] constitute the two prominent topologies for a decentralized monitoring mechanism. Within a tree topology, information is only exchanged between children and parents. Within a mesh topology, one or several neighbors are randomly chosen to exchange monitored information [5]. This results in *gossip-based* communication, which is often used as a synonym when describing the communication within mesh-based monitoring mechanisms. Furthermore, there are several hybrid approaches, combining gossip-based aggregation in trees [23] or creating trees of mesh-based networks [2, 18].

The topology maintenance depends on the network environment and its network topology. Monitoring approaches that are deployed in static and structured environments, such as in the area of grid computing [18], heavily differ from approaches for autonomous systems with highly dynamic users. The herein considered mechanisms for P2P systems must actively maintain the monitoring topology and additionally manage the arrival and departure of peers [1,2]. Therefore, they rely on additional mechanisms, such as Distributed Hash Tables (DHT) or membership protocols [10].

Data Collection.

This component sketches how monitored data is exchanged. Typically, gossip-based monitoring approaches actively send data to neighboring peers [12], also denoted as *push*. If the message sent triggers the transmission of an answer at the receiver, the gossip-based approach applies *push-pull*-based data collection [11]. Tree-based approaches can decide to either *push* data [7] or to alternatively *pull* data from neighbors [16, 18]. To trigger the collection of measured data, monitoring mechanisms rely on a periodic or event-based collection. For the latter case, the activating event may be, for example, (i) a newly measured value of an attribute at a peer, (ii) a query for the global view of an attribute, (iii) or the attempt of the system to generate a snapshot of an attribute at a certain point in time.

Result Dissemination.

This component highlights the possibilities to disseminate the global view of the monitored attributes. The existing strategies comprise *proactive* and *reactive* result dissemination. While proactive dissemination transmits the created global view to all or only a subset of peers, reactive dissemination sends the global view of attributes only to requesting peers. Tree-based monitoring approaches allow choosing between the different dissemination strategies, whereas proactive dissemination is implicitly integrated in gossip-based monitoring, due to the push-based collection of data.

3. BENCHMARKING DECENTRALIZED MONITORING MECHANISMS

In this section, we describe the design of our benchmarks, which will be used for the comparison of the different monitoring mechanisms in our case study. The designing process for a particular benchmark consists of the following three aspects: (i) The system specification provides the basis for the definition of benchmarks (Section 3.1). It illustrates the functional and non-functional requirements, each system has to fulfill. (ii) Given the requirements, appropriate workload schemes to benchmark a system are identified (Section 3.2). (iii) To quantify the obtained results of an applied workload, a set of metrics is created (Section 3.3). Finally, Section 3.4 outlines the combination of the three mandatory aspects in one or several benchmarks.

3.1 System Specification

Our system under test (SUT) consists of a decentralized monitoring mechanism, which is set up on top of a P2P system. To benchmark the SUT, it provides an interface to apply different workloads on the system and to measure the produced results. In case that the class of mechanisms being benchmarked does not provide a predefined interface, it must be derived based on the functional requirements of that class. To cover a wide range of existing approaches, the common functionality must be carefully analyzed and merged in a set of methods within the interface.

Due to the fact that neither an interface nor the provided functionality of a decentralized monitoring mechanism is specified, we examined existing approaches to highlight the key aspects. As outlined in Section 2, a decentralized monitoring mechanism calculates and provides the global view for a set of attributes. For that reason, each participant locally measures and stores the specified attributes for the overall collection. When the collection process is finished, the global view of attributes can be retrieved by the participating peers. Based on this description, the common functionality of a decentralized monitoring mechanism can be defined within the following interface.

- `setLocalValue(String name, double value)` stores a locally measured value of an attribute for the latter collection.
- `getGlobalViewOfAttributes()` returns the global view of all monitored attributes.

Every monitoring approach, applying our benchmarks, must provide this functionality and implement the specified interface in order to be evaluated or compared to another solution. Thus to apply the different workloads and to measure the produced results, the resulting interface of the SUT is located at each peer, which participates and monitors the system.

Besides the architecture and the design of the interface, the system specification also outlines the non-functional requirements of a system. Therefore, we identified the following *quality aspects*, representing the relevant non-functional requirements of decentralized monitoring mechanisms. These requirements build the basis for the subsequent identification of workloads and metrics.

- **Performance** characterizes the quality of the provided functionality of a mechanism. In the context of monitoring, we divide performance into *validity* and *timeliness*. With validity, we address the accuracy of the delivered results, which can be characterized through the difference between the measured and the actual global view of an attribute. Since the provisioning of correct information is the primary function of a decentralized monitoring mechanism, validity represents a central aspect. Besides the delivery of correct results, timeliness covers the aspect how fast the monitoring mechanism captures the global view and how fast it can deliver or distribute this view in the system.
- **Costs** comprise the communication or computation overhead produced by the monitoring mechanism to perform its task with a certain performance.
- **Fairness** can be evaluated with respect to performance and costs. On the one hand, a fair system should offer the same access to the provided services and avoid starving peers. On the other hand, a fair system should distribute the operational costs that peers are not overloaded.
- **Scalability** refers to the ability of a monitoring mechanism to preserve its performance at reasonable costs, while increasing the number of participating nodes or monitored attributes. A threshold for acceptable performance or costs must be defined by the application scenario.
- **Robustness** deals with the behavior of the whole P2P system in the presence of external and unpredictable events. These events mainly comprise massive fluctuations of participants due to, e.g., a network collapse or flash crowd behavior.
- **Stability** characterizes the ability of a decentralized monitoring mechanism to deal with the random behavior of autonomous peers in a P2P system. We consider the random behavior in terms of churn, which describes the varying frequency of arriving and leaving peers.

The identified non-functional requirements can be divided into two classes of quality aspects. On the one hand, there are quality aspects, such as performance, costs, and fairness, which can be quantified by metrics. Based on these metrics, it is possible to estimate if a mechanism meets these requirements. In contrast, the second class of quality aspects cannot directly be assessed by individual metrics, but is quantifiable by metrics, which are related to the first class of quality aspects. Instead, the second class of quality aspects characterizes the properties of a workload.

3.2 Workloads

For benchmarking decentralized monitoring mechanisms, we elaborated several workloads to address the identified quality aspects. These workloads are applied on the SUT, while the participating peers perform their tasks and measure a set of predefined attributes. Using the captured attributes, the monitoring mechanism calculates the global view for each attribute, as described in Section 2.1. Afterwards, this global view is disseminated to the peers. To examine validity of a monitoring mechanism under the specified workloads, the measured global view is compared to the so-called *correct global view*. In contrast to the global view obtained by the monitoring mechanism, the correct global view of an attribute is calculated based on a snapshot of the system at a certain point in time. Except for the peer count of a monitoring mechanism, we do not measure common system attributes (e.g., network traffic or number of messages) nor domain-specific attributes (e.g., lookup-rates or file-downloads for file-sharing systems) to evaluate validity. Instead, the peers in our benchmark obtain their monitored values from a *value generator*, as presented by Graffi [6]. This generator calculates a new value for each monitoring peer based on the current time and on the defined function. Afterwards, the calculated global view is compared with the actual value retrieved from the value generator to assess the validity of the monitoring mechanism.

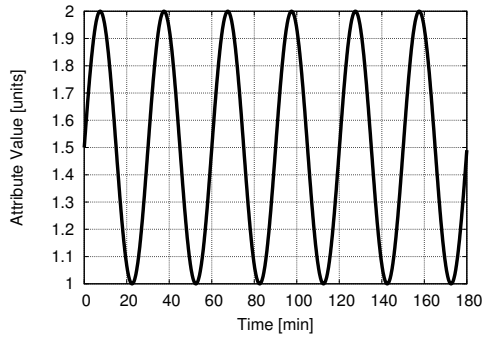


Figure 1: Sine reference signal with a period of 30min.

The value generator facilitates a more detailed analysis, because we can define functions with differing complexity, which refer to constant or highly varying attributes. It is possible to design individual functions that exhibit desired characteristics, such as steep slopes or periodicity, in order to estimate to which extent a monitoring mechanism is able to capture a varying signal. For example, it is easier to capture the values of a slightly increasing linear function than of a sine or a rectangular function. Moreover, the value generator improves the comparability of results in terms of validity. The monitored values of a function are independent from the enclosing P2P system (e.g., P2P file-sharing application) and current workload scenario (e.g., churn), thus, the values are not biased. In order to evaluate validity within our benchmarks, we implement a sine function, as displayed in Figure 1.

Besides this function for the value generator, we propose a set of workloads. These workloads are not application-related but synthetic workloads. They are domain-specific and model typical scenarios that are common for P2P system, such as churn or an increasing number of peers. We decided to apply synthetic workloads (i) to provide application-independent results and (ii) to stress a system regarding a particular, isolated quality aspect.

Baseline.

The baseline workload represents idealized conditions and provides insights on the behavior of the monitoring mechanism under these conditions. In contrast to the remaining workloads, this workload does neither include message loss, which is enabled for the rest of workloads, nor consider churn, which is addressed in a separate workload. Moreover, other workload parameters, such as the number of peers or the amount of monitored attributes, are fixed during this workload. This results in a static workload scenario with a fixed number of peers and perfect network conditions.

Churn.

In this workload scenario, we evaluate stability of a monitoring mechanism in the presence of churn. As underlying model, we employ an exponential churn model, which assumes an exponentially distributed mean session time per peer. The workload consists of several runs, which model the peers with different mean session times per run.

Massive join/crash.

The massive join workload consists of one run. During this run, we assume that a predefined fraction of new peers simultaneously joins the system. Within the context of a massive crash, the workload consists of a single run as well and covers the ungraceful departure of a predefined fraction of peers. These workloads evaluate the robustness of a decentralized monitoring mechanism, since it has to deal with a sudden change in the system state as well as in the amount of peers. For both workloads, one has to differentiate between a collapse of the monitoring mechanism due to the breakdown of the whole P2P system or due to the inability of the monitoring mechanism to reorganize itself.

Symbol	Description
T	The set of time samples
$P(t)$	The set of online peers at time $t \in T$
$A(t)$	The set of attributes being monitored at time t
$X_m(a,t,p)$	The *measured* global aggregate X of an attribute $a \in A(t)$ at time $t \in T$ available at a peer $p \in P(t)$
$X_c(a,t)$	The *correct* global aggregate X of an attribute $a \in A(t)$, which is calculated based on a snapshot of the system at time $t \in T$
$\tau_{\min}(X(a,t,p))$	The time of the oldest sample being included into an aggregate
$\tau_{\max}(X(a,t,p))$	The time of the most recent sample being included into an aggregate
$\Delta t_{\text{agg}}(X(a,t,p))$	The aggregation time considers all included values for a global view and is calculated as $\Delta t_{\text{agg}}(X(a,t,p)) = \tau_{\max} - \tau_{\min}$
$\Delta t_{\text{prop}}(X(a,t,p))$	The propagation time for a global aggregate from a data sink to a peer
$\Delta t_{\text{req}}(X(a,t,p))$	The time to answer a request for a global aggregate

Table 1: List of mathematical symbols

Increasing number of attributes.

In this workload scenario, we investigate scalability of a monitoring mechanism by scaling the amount of transmitted and processed data. We denote this type of scalability as *vertical scalability*. The workload consists of several runs. For each run, we increase the number of monitored attributes, which results in a higher amount of transmitted and processed data.

Increasing number of peers.

With the linear increase of peers in the system, this workload investigates another type of scalability of a monitoring mechanism, which we denote as *horizontal scalability*. In contrast to the previously described workload, which addresses vertical scalability, this workload increases the number of peers to an upper bound during one run and not during several runs.

3.3 Metrics

In this subsection, we introduce the metrics being measured to evaluate a decentralized monitoring mechanism. In order to describe the metrics below, we use the set of symbols shown in Table 1.

3.3.1 Per peer metrics

The following metrics are measured on a per peer basis for each participant of the P2P system. They can be mapped onto the quality aspects performance and costs.

Performance Metrics:

- $t_{\text{stale}}(X(a,t,p))$ denotes the staleness or age of an aggregate in seconds, observed at peer $p \in P(t)$ and is calculated as

$$t_{\text{stale}}(X(a,t,p)) = \Delta t_{\text{agg}} + \Delta t_{\text{prop}} + \Delta t_{\text{req}}$$

- $\epsilon_X(a,t,p)$ represents the relative monitoring error in percent for an aggregate X of on attribute $a \in A(t)$ at peer $p \in P(t)$ at time $t \in T$ and is defined as:

$$\epsilon_X(a,t,p) = \frac{|X_m(a,t,p) - X_c(a,t)|}{X_c(a,t)} * 100\%$$

Cost Metric:

- $l(t,p)$ represents the total traffic in $\frac{\text{kB}}{\text{s}}$ at peer $p \in P(t)$ at time $t \in T$. It comprises the up- and download traffic and is calculated as follows:

$$l(t,p) = l_{\text{up}}(t,p) + l_{\text{down}}(t,p)$$

3.3.2 Global Metrics

Based on the per peer metrics the following global metrics can be calculated:

- The mean of a metric x over the set of peers at time $t \in T$:

$$\overline{x}(t) = \frac{1}{|P|} \sum_{p \in P} x(p,t).$$

- The mean of a metric x over the set of time samples per peer $p \in P$:

$$\widetilde{x}(p) = \frac{1}{|T|} \sum_{t \in T} x(p,t).$$

- The total mean of a metric x:

$$\widehat{x} = \frac{1}{|T||P|} \sum_{t \in T} \sum_{p \in P} x(p,t).$$

3.4 Benchmark Implementation

Having introduced the system specification, workloads, and metrics, we present the benchmark implementation. This implementation combines the three components and creates the different benchmarks to evaluate the SUT. We have derived four different benchmarks that investigate and evaluate the system in a baseline, robustness, stability, and scalability benchmark. Before presenting all benchmarks, we describe the basis for each benchmark, which consists of three different phases as shown in Figure 2: (i) the setup phase of 60min in which 1,000 peers join the system, (ii) the stabilization phase of additional 20min, which ensures that the whole P2P system is set up correctly and stable, and (iii) a workload and measurement phase of 180min, where the different workload schemes are applied and where the benchmarking metrics are captured.

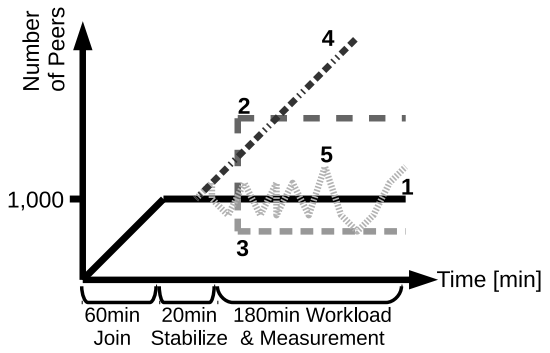


Figure 2: Schematic drawing of the schemes for varying the number of peers: (1) constant number of peers, (2) massive join, (3) massive leave, (4) linear increase, and (5) regular churn.

Baseline Benchmark.

The baseline benchmark provides insights on performance and costs in an idealized environment without message loss or peer churn. Using the *baseline* workload, this benchmark represents a reference for the remaining benchmarks regarding (i) the examined quality aspects of a particular monitoring mechanism as well as (ii) the comparison between the different monitoring mechanisms.

Scalability Benchmark.

To examine the scalability of a decentralized monitoring mechanism, we divide scalability into horizontal and vertical scalability. Horizontal scalability is benchmarked by the workload with an *increasing number of peers*. Within this workload the number of peers is linearly increased from 1,000 to 10,000 peers during the workload and measurement phase. In contrast, the workload with an *increasing number of attributes* benchmarks vertical scalability. The workload consists of three runs and covers scenarios with 10, 100, and 1,000 attributes, which are monitored by the system.

Stability Benchmark.

For investigating the stability of the system, we apply the workload for *churn*. The workload consists of three runs, which model peers with a mean session time of 60, 30, and 15min. With the increasing frequency of arriving and leaving peers per run, this workload examines the stability of a decentralized monitoring mechanism.

Robustness Benchmark.

In the robustness benchmark, we investigate the system behavior in two different scenarios defined by the *massive join* and *massive crash* workloads. We look at the system behavior when (i) 50% of the peers simultaneously leave and (ii) 100% new peers simultaneously join the system. We consider a system to be robust if these metrics reach predefined levels after a crash or a massive join. While the levels must be defined by the application scenario in which the particular monitoring approach should be used, we restrict the evaluation of robustness to a comparison of the three different systems.

4. BENCHMARKING RESULTS

In order to evaluate our benchmarks, we chose three different monitoring mechanisms and implemented them in the P2P simulation framework PeerfactSim.KOM [21]. We benchmarked all three systems using the previously defined benchmarks. Before presenting the results for each benchmark and outlining the most important conclusions, Subsection 4.1 summarizes the simulation setup and details the three chosen monitoring mechanisms.

4.1 Simulation Setup

We simulate each of the three monitoring mechanisms on top of a Chord overlay [22], since the tree-based approach requires a DHT to build its monitoring topology. Out of the four presented benchmarks, each benchmark is simulated with its corresponding workloads and metrics. During the workload phase, which lasts 180min (cf. Subsection 3.4), we periodically measure the produced data of the simulation with an interval of a minute. The data comprises the produced results of the monitoring mechanism and the traffic of the whole system, including the overlay as well. After sketching the basis for our simulations, we detail our selected monitoring mechanisms and briefly justify our choice.

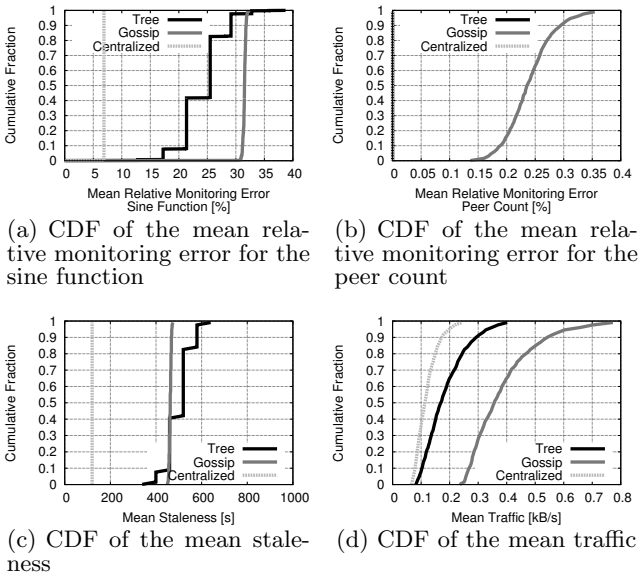
Based on the description of decentralized monitoring mechanisms in Section 2, it becomes apparent that the selection of the topology heavily influences the decisions for the remaining two components of a monitoring mechanism. Thus, the topology constitutes the main decision criterion for a monitoring mechanism, as outlined by Makhloufi et al. [17]. Therefore, we select two decentralized monitoring approaches, which rely on different topologies, while their mechanisms for data collection and result dissemination are similar. Data collection and result dissemination are part of the discussion for future work (cf. Section 6). For the benchmark, we selected (i) a tree-based approach, (ii) a gossip-based approach, and (iii) a centralized monitoring solution as reference, which are detailed in the following.

4.1.1 A Tree-Based Monitoring Mechanism

The monitoring mechanism, introduced by Graffi et al. [7], relies on the lookup-functionality of the underlying DHT to build its tree topology based on the given peer IDs. Using the created topology, every participating peer, either leaf or inner node of the tree, periodically sends its set of attributes towards the root, which calculates the global view of all monitored attributes. This results in a push-based data collection mechanism. Simultaneously, the root regularly sends the information down the tree to every inner node and leaf, leading to a proactive result dissemination. We set both update intervals to 60s as proposed by Graffi et al. [7].

4.1.2 A Mesh-Based Monitoring Mechanism

To evaluate our benchmark on mesh-based systems, this subsection details the approach by Jelasity et al. [11], which relies on gossip-based communication to monitor the P2P system. For this type of communication, the underlying overlay network must only allow for the retrieval of neighbors to periodically communicate with a randomly chosen subset of them. The mesh-based monitoring mechanism divides the time into *epochs*, which in turn consist of a predefined amount of *cycles* to calculate the global view of the monitored attributes. We set the amount of cycles per epoch to 30 with a cycle length of 10s, which correspond to the values indicated by Jelasity et al. [11]. In the beginning of a new epoch, every participating peer measures its attributes and periodically sends the current values to a randomly chosen neighbor. Through the aggregation of the measured attributes at each peer, the values converge to the average at the end of an epoch. Besides periodically pushing the own data to a neighbor, every peer that receives such a message, replies to this message with its own data. Thus, the system implements a *push-pull-based* data propagation.



(a) CDF of the mean relative monitoring error for the sine function

(b) CDF of the mean relative monitoring error for the peer count

(c) CDF of the mean staleness

(d) CDF of the mean traffic

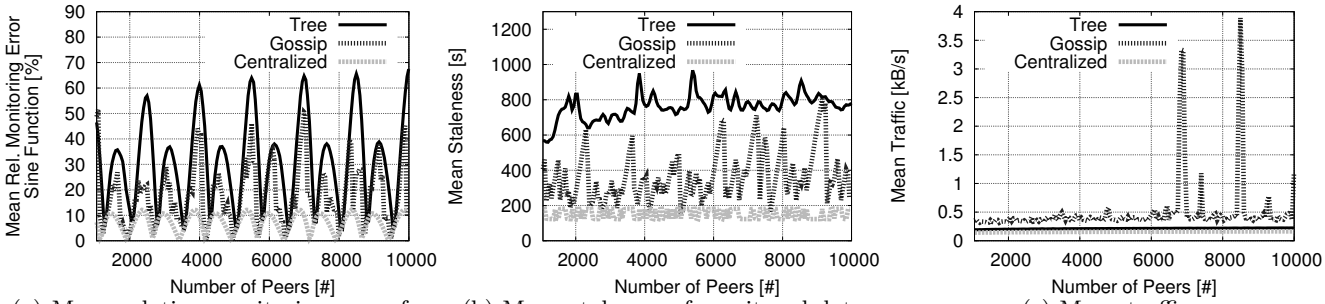
Figure 3: Per peer results for performance and costs, measured during the baseline workload.

4.1.3 A Centralized Monitoring Approach

In order to have a reference for decentralized monitoring mechanisms, we implemented a centralized monitoring approach, which is set up on top of the overlay. All participating peers of the centralized monitoring mechanism periodically push their measured attributes to a central server, which calculates the global view of the monitored attributes. Afterwards, the server proactively disseminates the computed global view to all peers in the system, resulting in a proactive result dissemination. Similar to the tree-based approach, we set both update intervals to 60s. In the following evaluation, the statistics of the centralized approach represent an optimal monitoring solution and serve as reference. Therefore, we mainly detail the comparison of the decentralized solutions and only refer to the centralized approach where appropriate.

4.2 Baseline Benchmark

We first study the performance and costs of the different approaches under idealized conditions within the baseline benchmark. Starting with performance in terms of validity, Figure 3(a) and 3(b) show the cumulative fraction of the mean relative monitoring error per peer averaged over a simulation denoted as $\widetilde{\epsilon_{avg}}(a,t,p)$. Both plots outline that the tree-based approach outperforms the gossip-based approach and in terms of the relative error for the peer count even catches up with the centralized approach. Although, each mechanism is able to capture the total amount of peers, the



(a) Mean relative monitoring error for the sine function (b) Mean staleness of monitored data (c) Mean traffic

Figure 4: Per peer results for performance and costs, measured during the horizontal scalability workload. The x-axes show the actual number of peers in the system, which increases from 1,000 to 10,000 peers over an interval of 180min.

gossip-based approach exhibits a small deviation. Dealing with fairness, every participating peer of the gossip-based approach is provided with a similar monitoring error. Contrary to this, the error of the tree-based approach is spread over a larger interval, due to its hierarchical topology and the stepwise data propagation down the tree.

Considering the mean staleness per peer $t_{\text{stale}}(\widetilde{X(a,t,p)})$, as displayed in Figure 3(c), we observe that the tree-based approach only partially outperforms the gossip-based approach, because a larger fraction of peers obtains older results in contrast to the gossip-based approach. This results in a mean staleness of 462s for the gossip-based and 501s for the tree-based approach. The obtained results for staleness lead to the interesting finding that the provided validity of the tree-based approach is slightly higher than of the gossip-based approach, whereas the staleness does not support this trend. Dealing with the distribution of staleness among the peers, the same characteristics as for the previously presented error distribution in Figure 3(a) can be observed.

Considering the costs of a peer in terms of the mean total traffic $\widetilde{l(t,p)}$ (Figure 3(d)), the produced communication overhead of the tree-based approach nearly reaches the minimum overhead of the centralized approach. In contrast, the tree-based approach does not evenly balance the load among the peers. The gossip-based approach produces the highest traffic due to a shorter update frequency of the mechanism. Compared to the other monitoring approaches, the distribution of costs is even worse, because the traffic depends on the amount of neighbors in the network, which differs among the peers and is not limited as for the tree-based and centralized approach.

4.3 Scalability Benchmark

At first, we study performance and costs of the considered monitoring mechanisms during the *horizontal scalability workload*. Looking at validity, displayed in Figure 4(a), we notice that the tree-based monitoring mechanism produces a higher mean error $\overline{\epsilon_{avg}(a,t,p)}$ than the gossip-based approach. This results from the fact that the dynamic and growing system has a higher impact on the tree than on the mesh topology. Due to the arrival of new peers, the continuous reorganization of the monitoring tree delays the result calculation and dissemination, because the data remains longer in tree. In contrast, we observe a smaller impact on the mesh topology, because the arrival of new peers does not require a reorganization of the underlying topology. Instead, new peers can be inserted anywhere into the mesh.

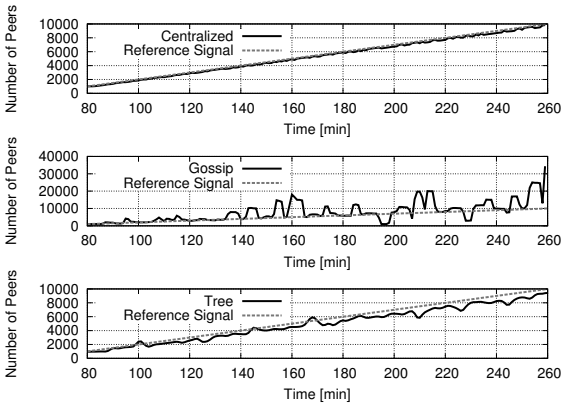


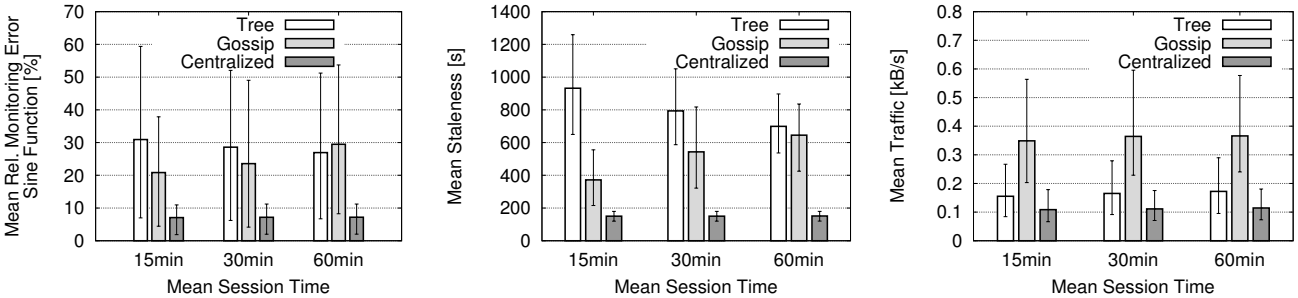
Figure 5: Actual vs. monitored number of peers over time during the horizontal scalability workload.

Figure 4(b) outlines the results for the mean staleness $\overline{t_{\text{stale}}(X(a,t,p))}$ and confirms the previous statements regarding the mean monitoring error for the reference signal. Due to the reorganization and the resulting delay, the tree-based approach exhibits a higher mean staleness of results with values up to 967s, whereas the gossip-based approach performs better, but exhibits highly fluctuating values, which vary between 146s and 805s.

In contrast to the results for the mean error of the monitored function, the drawn conclusion does not hold for the monitored number of peers. Figure 5 shows that the tree-based approach outperforms the gossip-based approach, which exhibits considerable outliers. The opposed outcome in terms of the peer count originates from the underlying peer counting procedure of the considered gossip-based approach. Contrary to the measurement of other attributes, e.g., the reference signal, peer counting is more susceptible to the dynamic of the system.

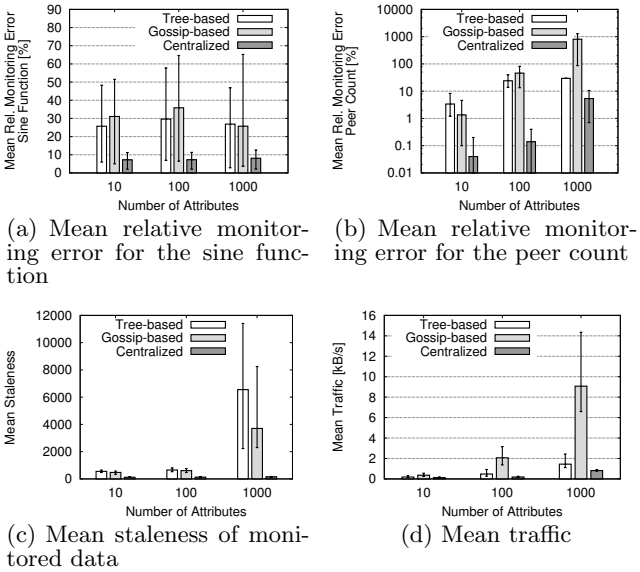
Dealing with the mean total traffic $\overline{l(t,p)}$ (Figure 4(c)), we observe a similar trend as for the baseline benchmark. While the tree-based approach nearly produces as less traffic as the centralized approach, the gossip-based approach causes the highest amount of traffic. However, it becomes apparent that both decentralized monitoring approaches scale well with the increasing number of peers.

Next, we study the results for the *vertical scalability workload*. The plots show the truncated mean after discard-



(a) Mean relative monitoring error for the sine function (b) Mean staleness (c) Mean traffic

Figure 6: Per peer results for performance and costs, measured during the churn workload.



(a) Mean relative monitoring error for the sine function (b) Mean relative monitoring error for the peer count

(c) Mean staleness of monitored data (d) Mean traffic

Figure 7: Per peer results for performance and costs, measured during the vertical scalability workload.

ing the values, which are below the 10- and above the 90-percentile. For each mechanism, Figure 7(d) shows that the total traffic per peer increases with the growing number of monitored attributes. While the mean relative monitoring error for the sine function in Figure 7(a) still indicates that the decentralized alternatives are able to handle the increased traffic, the mean relative error for the peer count (cf. Figure 7(b)) as well as the mean staleness of the provided results (cf. Figure 7(c)) outline contrary results. In terms of the peer count error, the underlying procedure for the peer count of the gossip-based approach reveals again its weakness in the presence of dynamic and unreliable environments. Although there are several paths between two peers inside a mesh, whereby bottlenecks, such as overloaded or slow peers, can be bypassed, the peer count procedure does not benefit from the mesh topology. Dealing with staleness, the age of the provided results of the tree-based approach significantly increases with a growing number of attributes. The reason for the degrading performance in terms of timeliness originates from the underlying tree-topology: If a path from the root to a sub-tree, or the other way round, is congested, the information cannot be forwarded. It resides at an inner node, leading to a bottleneck in the tree topology.

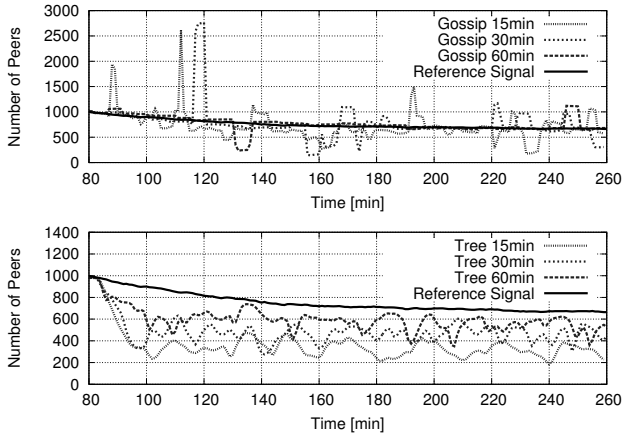
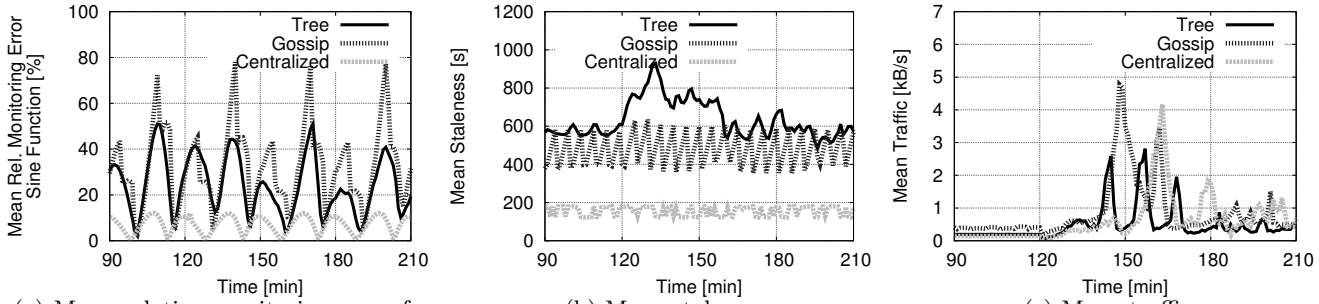


Figure 8: Actual vs. monitored number of peers over time with a mean peer session time of 15, 30, and 60 minutes.

4.4 Stability Benchmark

Examining the different monitoring approaches with respect to stability, we start with the evaluation of performance. Figure 8 shows the monitored number of peers averaged over all currently participating peers in the system. We omit the outcome of the centralized approach, since the results are accurate and do not significantly differ for the different mean session lengths. Instead, we plot the results for the two decentralized approaches dependent on the mean session time. Based on the displayed results, it becomes apparent that the tree-based approach suffers from the increasing peer fluctuations, because it cannot handle the resulting dynamic of the P2P system and degrades in terms of the provided performance. In contrast, the gossip-based approach manages the increasing dynamic of the system in a better way. Although exhibiting some outliers, whose occurrences increase with a decreasing mean session time, the gossip-based approach is capable of monitoring the current number of peers in the system.

A similar trend can also be observed, when looking at performance of the tree-based approach in terms of validity and timeliness. Figure 6(a) and 6(b) show an increase of the mean relative monitoring error $\widehat{\epsilon_{avg}(a,t,p)}$ as well as of the mean staleness of results $t_{\text{stale}}(\widehat{X(a,t,p)})$, whereas the gossip-based approach outperforms the tree-based approach in terms of validity and reduces the age of monitored results.



(a) Mean relative monitoring error for the sine function (b) Mean staleness (c) Mean traffic

Figure 9: Per peer results for performance and costs over time, measured during the massive crash workload.

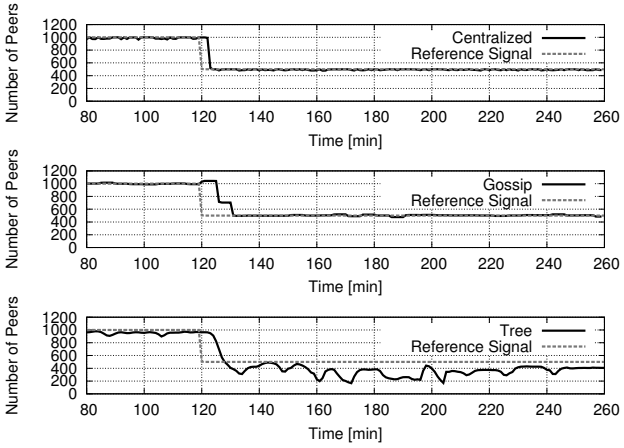


Figure 10: Actual vs. monitored number of peers over time during the massive crash workload.

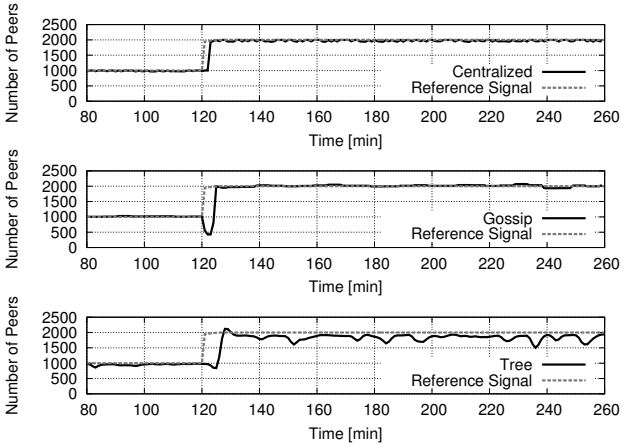


Figure 11: Actual vs. monitored number of peers over time during the massive join workload.

Dealing with costs, as displayed in Figure 6(c), the increasing churn rate shows little effect on the mean total traffic $\widehat{\overline{l(t,p)}}$ of the decentralized monitoring mechanisms.

4.5 Robustness Benchmark

Starting with the *massive crash workload*, Figure 10 displays the peer count to examine validity for the considered monitoring mechanisms during this workload. The gossip-based approach handles the sudden change in the system, settles down after 10min at the correct number of peers, and delivers stable results over time. Figure 9(b) and 9(a) reveal as well the robust behavior of the gossip-based approach. Irregardless of the sudden change in the system, the mean staleness $\overline{t_{\text{stale}}(X(a,t,p))}$ oscillates around 481s, while the mean relative monitoring error for the sine function $\overline{\epsilon_{avg}(a,t,p)}$ retains its characteristic oscillation. In contrast, the tree-based approach is not able to recover from the crash and delivers incorrect and fluctuating results, especially in terms of the mean error for the peer count. The reason for this failure originates from the collapse of the underlying Chord overlay. The monitoring topology cannot be created and maintained without the lookup-functionality of the overlay.

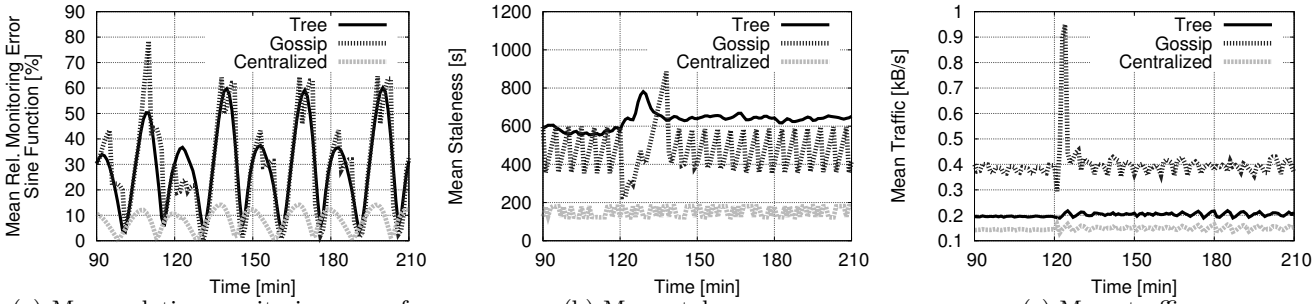
Examining the costs during and after the crash, Figure 9(c) displays a highly varying mean traffic $\overline{l(t,p)}$ for each approach. The highly fluctuating traffic originates from Chord's maintenance mechanisms, which react on the departure of peers and calm down after a certain amount of time.

For the *massive join workload*, Figure 11 displays the provided results of the three alternatives in terms of the monitored number of peers. Contrary to the previously discussed results of the massive crash workload, each mechanism manages the sudden increase of peers in the system and returns to its normal state after a period of time. This trend can also be observed, when looking at other metrics that quantify the performance of our alternatives: (i) In terms of validity, Figure 12(a) outlines that all mechanisms recover and provide similar results. (ii) Figure 12(b) shows that the mean staleness $\overline{t_{\text{stale}}(X(a,t,p))}$ of the provided results does not degrade due to the massive join and the sudden increase in the system size. Dealing with costs, Figure 12(c) shows that the mean total traffic does not change and levels off after short fluctuations.

4.6 Discussion of Results

Having presented the benchmark of the different monitoring approaches, this section summarizes the obtained results. Subsequently, we discuss and compare the initial performance evaluation of the respective papers that introduced the herein utilized monitoring mechanisms.

Under idealized conditions, the tree-based monitoring approach outperforms the gossip-based approach in terms of validity, while producing less traffic, which is balanced more regularly among the peers. Regarding the mean monitoring error for the peer count, the tree-based approach even catches up with the centralized approach. In contrast, the hierarchical structure results in a biased distribution of results, because leaves or distant nodes from the root obtain



(a) Mean relative monitoring error for the sine function (b) Mean staleness (c) Mean traffic

Figure 12: Per peer results for performance and costs over time, measured during the massive join workload.

inaccurate and old results. In the presence of churn, the performance of the tree-based approach significantly decreases dependent on the mean session time of the peers in the system. On the contrary, the underlying mesh topology of the gossip-based approach exhibits a better stability and is capable of handling the increasing dynamic in the system, still providing valid and fresh results. Nevertheless, the peer count procedure shows its susceptibility to the altering mesh topology. In terms of extreme peer fluctuations, which characterize the robustness of a mechanism, the corresponding benchmark outlines that the gossip-based approach is robust enough to handle sudden changes in the system and even provide results on top of a crashed overlay. Due to the intensive application of the lookup functionality of the overlay by the tree-based approach, the monitoring mechanism collapses during the crash. Although, it is capable of handling a sudden increase of peers in the P2P system. Dealing with scalability of a decentralized monitoring mechanism, the horizontal workload outlines that the decentralized mechanisms scale with a growing number of peers. The increased amount of peers has only a negligible influence on performance and costs, except for the underlying peer count procedure of the gossip-based approach, which exhibits considerable outliers. Dealing with the vertical scalability workload, the resulting traffic of both decentralized monitoring mechanisms increases and leads to a decreasing validity and increasing staleness of the monitored attributes. We observe that the tree-based approach already suffers from a smaller increase in traffic regarding its performance, while the participants in the gossip-based approach must handle a considerable amount of traffic.

In the following, we discuss and compare the initial performance evaluation for the gossip- and subsequently the tree-based approach. Jelasity et al. [11] evaluate their approach with respect to scalability, robustness, and stability. In terms of scalability, the paper only addresses horizontal scalability, which is evaluated based on mathematical analysis. The paper omits an evaluation in terms of vertical scalability. While the authors rely on validity, or accuracy as denoted by the authors, to characterize the performance of the presented approach, costs are not considered during the simulative and experimental evaluation. Consequently, the drawn conclusions in terms of scalability, stability, and robustness cover only performance. Using our presented methodology, we have shown in Section 4.3 and 4.5 that the resulting costs in terms of traffic are not influenced by peer-related workloads, e.g., horizontal scalability, massive crash or join. Instead, we could show in Section 4.5, that a growing amount of monitored attributes results in a considerable increase of traffic and that the produced traffic influences the provided results (cf. Section 4.2). In terms of accuracy, Jelasity et al. only consider the peer count as measurable attribute in their evaluation, because it represents a "worst-case" due to its sensitivity to failures. The assessment of accuracy based on "normal" attributes, such as modeled by our value generator, is omitted. Based on our methodology, we showed the increased susceptibility of the peer counting procedure in contrast to the robust calculation of "normal" attributes (cf. Section 4.3 and 4.4). Dealing with the experiments on stability and robustness, Jelasity et al. present an exhaustive evaluation, which examines the effect of peer crashes, different message loss rates and churn on the performance of the mechanism. Within these experiments, the authors only concentrate on one epoch of the protocol, while long-term effects are ignored. Thus, out of the presented results, it is not obvious if the presented approach can recover and how long this might take.

In contrast to the previous and our methodology, the tree-based approach [6, 7] is just evaluated in terms of scalability and stability, but set up on two different overlays. With respect to stability, the corresponding workload consists of different churn levels, which are applied on the system during one run. In terms of scalability, the decentralized monitoring mechanism was evaluated under a varying amount of peers in separate runs. Similar to our methodology, Graffi et al. evaluate the performance of the presented approach in terms of validity and timeliness, which they denote as precision and freshness. On the contrary, they evaluate validity and timeliness of the obtained results only at the root, while dissemination of results back to the remaining peers is not taken into account. Dealing with validity, they look at the peer count and other attributes, which are either measured by the peers or modeled by their implemented value generator. While examining the resulting costs and their distribution among the peers, they do not evaluate how validity of the obtained results differs among the peers. In this regard, Section 4.2 outlines that the topology of the tree heavily influences validity and timeliness. Moreover, we showed that the tree-based approach provides a similar performance as the centralized approach under idealized conditions. On the contrary, Section 4.3 outlines that the approach suffers from an increasing amount of attributes, while it cannot handle massive crashes, in contrast to massive joins (cf. Section 4.5), and that performance degrades if peer fluctuation increases (cf. Section 4.4).

Based on the two examples of performance evaluation, it

becomes apparent that there is no standardized way for the evaluation of decentralized monitoring mechanisms. Moreover, the examples outline that a comparison of several mechanisms based on the differing initial evaluations is hard to achieve. The presented evaluations only agree on a fraction of quality aspects, such as validity, costs, or scalability, which are examined. On the contrary, other important aspects, e.g., robustness, or fairness are neglected. The resulting workloads, evaluation scenarios, and setups differ widely and cannot be compared. Besides a standardized set of quality aspects or workloads, a unified approach must be established to capture the measurements for the evaluation. As shown by the examples, measurements can be taken at all peers, while other evaluations rely on measurements at single peers, such as the root.

5. RELATED WORK

The related work in the area of benchmarks for decentralized systems details the methodology and aspects as well as existing implementations for the performance evaluation. The considered implementations range from distributed hash tables (DHT) [15], over networked virtual environments [8, 13], to decentralized monitoring mechanisms [3, 4].

Haeberlen et al. [9] discuss the general benefits of a benchmark for decentralized systems, leading to an improved comparability between different approaches and a better classification of the obtained results. In addition to the positive features of a standardized methodology, their paper also highlights common dangers of a benchmark, which might originate from inappropriate or false standardization, incomplete tests or ossification of a standard. Besides this general description of benchmarks in decentralized systems, we already focus on a benchmarking methodology in the area of P2P systems in our previous work [14]. We outline the specifics for the design of a P2P benchmark and give a concrete definition for benchmarking search overlays and overlays for networked virtual environments.

Apart from the description of the benchmarking methodology, several approaches exist that present the implementation of a benchmark for a P2P system. Li et al. [15] develop a methodology to evaluate the efficiency of different DHTs by examining the trade-off between performance and cost. Therefore, they define different types of workloads to test the overlays under varying conditions and to investigate overlays with different parameter settings. Kovacevic et al. evaluate in [13] the suitability of DHTs in networked virtual environments. They develop a dedicated benchmark that addresses the investigation of relevant quality aspects for these environments by defining appropriate metrics. An extended version of the benchmark has been proposed by Gross et al. [8], which allows for the comparison of arbitrary overlays for networked virtual environments implementing a certain interface definition.

Regarding the benchmark for decentralized monitoring mechanisms, Bawa et al. [3] present a benchmark for three different aggregation approaches ranging from a tree-based over a gossip-based to a hybrid topology to monitor a P2P network. Given the made assumptions for the benchmark (e.g., network topology, distributed state, and communication failures), the paper compares the three approaches regarding different quality aspects, covering flexibility, generality, termination, and correctness. Our presented benchmarks extend the work by Bawa et al. concerning the examination of the identified non-functional requirements. For the evaluation of accuracy, we define a detailed analysis for a monitoring mechanism and its produced monitoring error based on reference signals of the value generator, besides peer count. Moreover, we identified different workloads to stress the monitoring mechanisms under different conditions for the examination of quality aspects, such as robustness. In [4], Cappos and Hartman compare their developed tree-based monitoring mechanism with another tree-based approach [24] and a centralized solution, using analytical models, simulations, and experiments. We extend the extensive evaluation in their work by including the examination of accuracy for decentralized monitoring mechanisms. In addition, we add the investigation of robustness for decentralized monitoring mechanisms by massive join/crash workloads.

The problem of missing comparability becomes even more clear in a survey of decentralized aggregation mechanisms by Makhloufi et al. [17]. While giving a good overview about different schemes for aggregation protocols, highlighting the different design decisions, the concluding table, which lists the performance of the considered approaches, does not enable a fair comparison between them. This results from the fact that the summary only summarizes the results of the respective papers.

6. FUTURE WORK

In this paper, we have presented our approach for a set of benchmarks, which establishes a standardized evaluation of decentralized monitoring mechanisms to facilitate comparability of results. For the standardized evaluation, we (i) defined a common interface for a unified access of the provided functionality, (ii) identified relevant non-functional requirements of the considered class of mechanisms, and (iii) designed a set of workloads and metrics to evaluate and quantify the non-functional requirements. We presented the implementation of four different benchmarks (baseline, stability, robustness, scalability) for evaluating performance and costs of decentralized monitoring mechanisms. Thereby, we identified characteristic performance and cost profiles as well as monitoring capabilities for two decentralized monitoring mechanisms (a gossip-based and a tree-based approach) as well as for a centralized approach, which served as a reference.

We plan to apply our benchmarks on different decentralized monitoring mechanisms, since the presented application of benchmarks only considered monitoring approaches with push-based data collection and proactive result dissemination. Therefore, we intend to benchmark pull-based and reactive monitoring mechanisms as well, in order to determine the trade-off between push- and pull-based data aggregation, or proactive and reactive result dissemination, as already analyzed in our previous work [20].

In the future we plan to exchange the underlying overlay with other well known overlays in order to investigate the interdependencies in terms of performance and costs between monitoring mechanisms and underlying overlays. Besides, we will not only focus on the communicational overhead caused by a decentralized monitoring mechanism, but also consider the computational overhead, such as the resulting I/O- or CPU-usage. Moreover, we plan to execute our benchmarks in larger simulations, which exceed the capabilities of typical testbeds, such as PlanetLab.

7. ACKNOWLEDGMENTS

This work has been supported by the German Research Foundation (DFG), Research Group 733, "QuaP2P: Quality Improvement of Peer-to-Peer Systems".

8. REFERENCES

[1] K. Albrecht, R. Arnold, M. Gahwiler, and R. Wattenhofer. Aggregating Information in Peer-to-Peer Systems for Improved Join and Leave. In *Proc. of the 4th Internat. Conf. on Peer-to-Peer Computing*, pages 227–234, 2004.

[2] M. S. Artigas, P. García, and A. F. G. Skarmeta. DECA: A Hierarchical Framework for DECentralized Aggregation in DHTs. In *Large Scale Management of Distributed Systems*, volume 4269, pages 246–257. Springer, 2006.

[3] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating Aggregates on a Peer-to-Peer Network. Tech. Rep. 2003-24, Stanford InfoLab, 2003.

[4] J. Cappos and J. H. Hartman. San Fermín: Aggregating Large Data Sets Using a Binomial Swap Forest. In *Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 147–160, 2008.

[5] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. Epidemic Information Dissemination in Distributed Systems. *IEEE Computer*, 37(5):60–67, 2004.

[6] K. Graffi. *Monitoring and Management of Peer-to-Peer Systems*. PhD thesis, Technische Universtiät Darmstadt, 2010.

[7] K. Graffi, D. Stingl, J. Rueckert, A. Kovacevic, and R. Steinmetz. Monitoring and Management of Structured Peer-to-Peer Systems. In *Proc. of the 9th Internat. Conf. on Peer-to-Peer Computing*, pages 311–320, 2009.

[8] C. Gross, M. Lehn, C. Münker, A. Buchmann, and R. Steinmetz. Towards a Comparative Performance Evaluation of Overlays for Networked Virtual Environments. In *Proc. of the 11th Internat. Conf. on Peer-to-Peer Computing*, pages 34–43, 2011.

[9] A. Haeberlen, A. Mislove, A. Post, and P. Druschel. Fallacies in Evaluating Decentralized Systems. In *Proc. of the 5th Internat. Workshop on Peer-to-Peer Systems*, 2006.

[10] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations. In *Proc. of the 5th ACM/IFIP/USENIX Internat. Conf. on Middleware*, pages 79–98, 2004.

[11] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-Based Aggregation in Large Dynamic Networks. *ACM Transactions on Computer Systems*, 23(3):219–252, 2005.

[12] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2003.

[13] A. Kovacevic, K. Graffi, S. Kaune, C. Leng, and R. Steinmetz. Towards Benchmarking of Structured Peer-to-Peer Overlays for Network Virtual Environments. In *Proc. of the 14th Internat. Conf. on Parallel and Distributed Systems*, pages 799–804, 2008.

[14] M. Lehn, T. Triebel, C. Gross, D. Stingl, K. Saller, W. Effelsberg, A. Kovacevic, and R. Steinmetz. Designing Benchmarks for P2P Systems. In *From Active Data Management to Event-Based Systems and More*, volume 6462, pages 209–229. Springer, 2010.

[15] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. A Performance vs. Cost Framework for Evaluating DHT Design Tradeoffs Under Churn. In *Proc. of the 24th Annual Joint Conf. of the IEEE Computer and Communications Societies*, pages 225–236, 2005.

[16] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A Tiny AGgregation Service for Ad-hoc Sensor Networks. In *ACM SIGOPS Operating Systems Review*, volume 36, pages 131–146, 2002.

[17] R. Makhloufi, G. Bonnet, G. Doyen, and D. Gaiti. Decentralized Aggregation Protocols in Peer-to-Peer Networks : A Survey. In *Modelling Autonomic Communications Environments*, volume 5844, pages 111–116. Springer, 2009.

[18] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, 30(7):817–840, 2004.

[19] S. Rhea, T. Roscoe, and J. Kubiatowicz. Structured Peer-to-Peer Overlays Need Application-Driven Benchmarks. In *Peer-to-Peer Systems II*, pages 56–67. Springer, 2003.

[20] K. Saller, D. Stingl, and A. Schürr. D^4M, a Self-Adapting Decentralized Derived Data Collection and Monitoring Framework. In *Workshops der wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen*, pages 245–256, 2011.

[21] D. Stingl, C. Gross, J. Rückert, L. Nobach, A. Kovacevic, and R. Steinmetz. PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems. In *Proc. of the Internat. Conf. on High Performance Computing and Simulation*, pages 577–584, 2011.

[22] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, 2001.

[23] R. Van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *ACM Transactions on Computer Systems*, 21(2):164–206, 2003.

[24] P. Yalagandula and M. Dahlin. A Scalable Distributed Information Management System. *ACM SIGCOMM Computer Communication Review*, 34(4):379–390, 2004.