[KGK+08] *Aleksandra Kovacevic, Kalman Graffi, Sebastian Kaune, Christof Leng, Ralf Steinmetz;* **Towards Benchmarking of Structured Peer-to-Peer Overlays for Network Virtual Environments.** In: Proceedings of the 14th International Conference on Parallel and Distributed Systems (ICPADS 2008), IEEE Computer Society Press, December 2008.Seite 799-804

# Towards Benchmarking of Structured Peer-to-Peer Overlays for Network Virtual Environments

Aleksandra Kovacevic, Kalman Graffi, Sebastian Kaune, Christof Leng, and Ralf Steinmetz
Technische Universität Darmstadt
Email: {sandra$^1$,graffi$^1$,kaune,steinmetz}@kom.tu-darmstadt.de, cleng$^2$@dvs.tu-darmstadt.de

## Abstract

*Network virtual environments (NVE) are an evolving trend combining millions of users in an interactive community. A distributed NVE platform promises to lower the administration costs and to benefit from research done in the peer-to-peer (p2p) domain. In order to reuse existing mature p2p overlays for NVEs, a comparative evaluation has to be done in the same environment (e.g. resources of peers, peer behavior, churn, etc.), using appropriate test cases (scenarios) and observing relevant performance metrics. In this paper we present a benchmarking approach for p2p overlays in the context of NVEs. We define related quality attributes, scenarios, and metrics and use them to evaluate Chord and Kademlia as most popular p2p overlays and assess their suitability to NVE.*

## 1 Introduction

Network virtual environments (NVE) [15] gained more and more attention in research community in recent years, as they address a broad community and offer various technical challenges. NVEs combine 3D graphics, consistent world representation and community interaction. These large-scale entertaining networks with millions of users are typically provided by companies in a client/server based infrastructure. Due to the nature of these communities, a peer-to-peer based solution promises to lower the costs dramatically [12]. Before building another overlay specifically for the needs of NVEs, it is reasonable first to evaluate already existing deployed p2p overlays in order to decide on their applicability for NVEs. Peer-to-peer research nowadays offers a variety of different overlay networks aiming for the same goal - building an efficient, scalable, stable, and robust peer-to-peer system. However, the evaluation of overlays is commonly done using individually defined criteria and metrics. This makes the evaluation results incompatible and comparing the existing solutions nearly impossible. A comparative evaluation of overlays is valid only if it is done in the same environment (e.g. resources of peers, peer behavior, churn, etc.) and using an appropriate usage scenario with relevant performance metrics.

In this paper we show the systematic steps to enable valid comparison of overlays to be applied for NVEs, show the impact of different design decisions on their performance (trade-offs), identify the criteria for an appropriate overlay for NVEs, and discuss the applicability of Chord and Kademlia for NVEs. In order to achieve it, we introduce *benchmark sets* consisting of quality attributes, appropriate metrics, and scenarios. We assess the performance of two popular overlays using these benchmarking sets that are tightly related to the needs of NVEs. P2P overlays provide basic operations for resolving queries and routing using a common object addressing scheme. They can be classified into structured ([13, 16, 14]) and unstructured ([1]), hybrid ([4, 10]) and hierarchical ([2]). In this paper we will focus on structured overlays due to their strict determinism in storing and retrieving objects in the network identified by their key. We evaluate Chord [16] as the most cited structured p2p overlay and Kademlia [13] as the most deployed p2p overlay and discuss their applicability for NVEs.

In the next section, revelant quality attributes are defined. A benchmark set is presented in Section 3 with the given metrics and scenarios. Two structured overlay networks, Chord and Kademlia are evaluated against the proposed benchmarks and the results are presented and discussed in context of NVEs in Section 4. The discussion on future work and a conclusion is given in Section 5.

## 2 Relevant Quality Attributes for NVEs

The nature of peer-to-peer overlay networks introduces some key quality aspects each application has to consider. For example, stability, scalability, and load balancing are issues brought by the fact that peers, as autonomous entities can randomly leave, join, or perform queries. It results in a big variation of network size, number of exchanged messages, number of stale contacts in routing tables, etc. Furthermore, peers have variable connectivity and failures can appear at a random point of time, which brings robustness as

[1,2] Authors supported by the German Research Foundation, Research Group 733, "QuaP2P: Improvement of the Quality of Peer-to-Peer Systems" [3].

another important quality aspect. In this paper we focus on purely performance attributes and omit not performance related attributes like reliability, availability, consistency, and security. In the following, we present the quality attributes that have been considered, for further details see [7].

*Efficiency* is defined as the ratio of performance (performance of overlay operations and service provisioning) and costs (from the view of individual node, the whole p2p overlay and the IP infrastructure).

*Stability* is tightly connected with resilience and is not clearly distinguished from *robustness*. Therefore we will here try to make the difference between them, considering the focus of this paper. Stability, resilience, and robustness describe the behavior of the system under changed conditions. From the point of view of the system architecture there are two classes of changes: expected and unexpected. All changes that are predicted and therefore described by a protocol are referred to as *expected*. All appropriate actions that have to be performed under those changes are included in the protocol. Still, predicted changes can jeopardize the robustness and resilience of the system when they occur too often, therefore creating bottlenecks of the system. An example of expected changes would be a large number of joins, leaves, or searches during a short time period or a sudden big number of lookup or routing messages directed to one node. On the other side, *unexpected* changes of the systems are not considered by protocols. Examples are connection failures, attacks, etc. Accordingly, additional mechanisms must be developed to make systems resistant to intensive expected or unexpected changes.

A *resilient (stable) system* is able to maintain all functions and services the system offers under expected or unexpected conditions and environment changes. A basis for metrics describing resilience is performance variation that occurs when a system is exposed to the mentioned changes. Exact performance metrics as well as reference values depend on the observed system and usage scenarios. Here variation refers to the relative difference of a parameter from its value in stable state of a system. Therefore, evaluating resilience of a system requires previous evaluation under stable conditions.

A *robust system* is recovering from both intensive expected and unexpected changes of the system in a reasonable time interval. A basis for metrics describing robustness is therefore the time a system needs to recover to the performance it had in stable state. More precisely, performance in stable state is described with an upper and a lower performance bound. Time is measured from the point of time where the system left the stable state performance.

*Scalability* is the quantitative adaptability of the overlay to a changing number of participants or services in the overlay, while preserving the performance.

An overlay is *load balanced* when the costs for publishing, storing, and resolving queries are uniformly distributed over the peers. The load of a peer should be proportional to its individual capacity. Load balancing presents the distribution of traffic load (received/sent messages) on the individual peers. Overload can occur due to a more significant role a peer or popular data/services it offers.

## 3 Benchmark Sets

A benchmark set can be presented as a tuple of quality attributes $Q$, metrics $M$, and scenarios $S$. Regarding a quality attribute the scenarios and observed metrics differ in a benchmarking test. In order to make statements on e.g. the scalability of an overlay, a suitable scenario and the appropriate metrics are needed. We define basic $M$ and $S$ sets used to evaluate P2P overlays in the context of NVEs.

### 3.1 Metrics

Following set of metrics is relevant to the mentioned quality attributes $Q$ of P2P overlays:

**Number of hops** $N_{hops}$ is the common used metric for evaluating the performance of peer-to-peer overlays. It presents the number of contacted peers on the way from the source to the destination for an observed query (e.g. lookup in structured overlays) message. Routing in distributed hash tables (DHTs) can be either recursive or iterative. In comparison with recursive routing (e.g. in Kademlia), our measurements of the iterative routing (e.g. in Chord) takes into account also those contacted nodes that are not involved in routing the message to the destination.

**Response time** $t_r$ is defined as the duration of a query operation. It is different in iterative and recursive routing even if the number of hops is the same. The parallelization of lookup queries like in Kademlia brings significant performance benefits which is evidently reflecting on this metric.

**Overall success rate** $R_{success}$ is important as both the number of hops and the response time cannot show the share of successfully answered query operations. Therefore, the metrics catalog for the evaluation has to include the average success rate of requests defined as ratio $R_{success}$ of number of successfully resolved and overall number of query operations:

$$R_{success} = \frac{N_{success}}{N_{total}}$$

**Relative Delay Penalty (RDP)** describes how well the overlay structure matches the underlying network topology. It is defined as the ratio $RDP$ of the measured latency introduced by sending a message from point A to B through the overlay structure and the corresponding latency when sending it directly through the underlay [8]:

$$RDP = \frac{d_{overlay}(A, B)}{d_{underlay}(A, B)}$$

**Stale contact ratio** $R_{stale}$ measures the usage of stale contacts in the routing table. Peers are joining and leaving the network and thereby their contact information in routing tables of other peers can be stale. This influences the overall performance of the protocol. Stale contact ratio

$$R_{stale} = \frac{\sum N_{messages}(stale\_contact)}{\sum N_{messages}}$$

gives the share of messages sent to peers which already left the network in the overall number of sent messages.

**Message distribution** shows the exact portion of the total number of received messages for each peer and directly shows the load distribution.

**Message type distribution** sorts all received messages into different types, in order to depict protocol overhead or load balancing. Currently, the following main five different types of messages are identified:

- *join* which includes all message types which are necessary in a bootstrapping phase,
- *leave* consisting of all message types which are supporting a leaving process,
- *maintenance* including all messages for stabilization of network structure (e.g. updating routing table),
- *user messages* which presents all messages of overlay operations involved in user interaction, and
- *result* including all necessary messages for transfer of the data.

**Stale message ratio** determines the percentage of lost messages caused by *churn*:

$$R_{lost} = \frac{\sum N_{lost\_messages}}{\sum N_{messages}}$$

### 3.2 Scenarios

An observed overlay $O$ performs different depending on the used scenario $S$. The scenario defines which overlay operations certain peers or a group of peers perform at which point of time. The scenarios considered in this benchmark set are described in the following and a detailed scenario setup for experiments (Section 4) is given.

*Ideal* is the scenario where peers first join the network and once the bootstrapping process is over, peers start to perform specific overlay operations. A new overlay operation will not take place before an appropriate stabilization phase is over, churn is not expected. For example, in experiments with 10.000 nodes, during the time interval $t_j = [0 : 10.000]ms$ all participating peers join the network ($1ms$ per peer). Besides joining, peers will publish their data using *put(key,data)*. In the next time interval, $t_{st} = [10.000 : 90.000]ms$ the system stabilizes. The peers perform a number of lookup/search operations starting at $T_{get} = 100.000\,ms$. *Volatile Joins* is a scenario with changing network size. The participants are divided in groups $g_0$, $g_1$, and $g_2$, after joining and completing the bootstrapping phase of the first group $g_0$, all peers from $g_0$ publish their data. Once the publishing process is done, peers start random *get(key)* operations and group $g_1$ joins the network. Peers from the group $g_0$ will continue performing *get(key)* operations and group $g_2$ joins the network and perform the appropriate actions analogue to above. *Churn* is a scenario where a significant number of peers leaves in a short time interval. In our experiments, two groups of peers are considered - $g_0$ with $1/3$ and $g_1$ $2/3$ of the overall number of peers. The actions in this scenario are described in Figure 1. *Failures* is the same like scenario 'Churn' with the difference that peers are randomly failing during the simulation so that messages are getting lost, contacts in routing table outdated, etc.

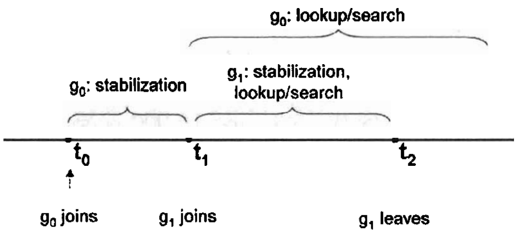


**Figure 1. Scenario 'Churn'**

## 4 Experiments Using the Benchmark Sets

NVEs state various requirements on the described quality attributes. However, with the proposed benchmark sets we are able to evaluate existing (and new) P2P overlays systematically for e.g. the purpose of NVEs. For each quality attribute we define the metrics and the scenario in which the overlays are tested, by this the results become comparable. As NVEs require identifiable objects in a virtual world we consider DHTs as most fitting to the need of NVEs. Among them we identified Chord [16] as the most cited one in literature and Kademlia [13] as the most used in real world peer-to-peer applications, as promising candidates for benchmarking in context of NVEs. All experiments are done in the peer-to-peer simulator PeerfactSim.KOM [11]. The common setting for all experiments are the following. Maximal latency is set to 350ms, timeout penalty is 750ms. Kademlia has $b = 1$, replacement cache is 2, timeout multiplier is 2.5 while $k$ and $\alpha$ are variable. In Chord number of successors are 10 and stabilization interval is 650ms.

### 4.1 Efficiency

In this experiments, the scenario 'Ideal' is used, metrics $N_{hops}$, $t_r$, and $RPD$, the experiment size is 10.000 peers, Kademlia bucket size $k$ is 10 or 20, and the degree of parallelism $\alpha$ is 1 or 3. The results of the simulations are presented in Figure 2(a). Efficient lookup operations are crucial for NVEs to quickly retrieve objects identifiable by their keys.

Chord needs around 35% more hops than Kademlia with $k = 10$ and $\alpha = 3$. Turning off parallel lookup resolution in Kademlia ($\alpha = 1$) and increasing the size of buckets to 20 significantly decrease the number of necessary hops, as less peers are involved in the lookup query. However, parallelism decreases the response time as shown in Figure 2(b). Here, Chord needs to much time for query resolving, which shows that it cannot be used without modifications for NVEs. As a result, we conclude, that overlays like Kademlia, that parallelize the lookup operations are to prefer as P2P infrastructure for P2P based NVEs. With regard to the RDP, we see that overlay distances in Kademlia are very close to underlay distances. However, as we have shown in [9] by using information about the underlay, lookup times and overlay distances can be decreased even more.

## 4.2 Scalability

The experiments on scalability use the scenario 'Volatile Joins' with the metrics $N_{hops}$, $t_r$, and $RDP$. The Experiment size is 100, 1000, 2500, 5000, and 10.000 peers, and in Kademlia the bucket size $k$ is fixed on 10 and $\alpha$ on 3. As the complexity of routing performance in both protocols is $O(log_2(n))$, Kademlia and Chord are not scaling linear but logarithmic. Figure 3 shows that Chord again scales significant worse then Kademlia as the number of hops increase from 4 to 7 with the increasing the experiment size, whereas Kademlia peers need in average 3 instead of 1,5 hops. However, both protocols have a lookup complexity of $O(log_2(n))$, which is state of the art for DHTs. Both overlays fit the with this complexity to the needs of modern infrastructures for large-scale NVEs.

## 4.3 Stability

In these experiments, we used the scenario 'Churn', and metrics $N_{hops}$ and $t_r$, whereas the experiment size is 10.000 peers. In Kademlia the bucket size $k$ is fixed on 10 and $\alpha$ on 3. The churn rate is set on mixed log-normal rate in order to investigate the behavior of the overlays under realistic peer participation.

Due to the extreme increase of network size (double size) in short time interval, the performance of both protocols is significantly decreasing between $t_1$ and $t_2$ as shown in Figures 5 and 4. The size of peers in the network before $t_1$ is 3000 nodes, therefore it is corresponding to the performance values of $N_{hops}(Chord) = 6$, $N_{hops}(Kademlia) = 2$, $t_r(Chord) = 600\ ms$, and $t_r(Kademlia) = 120\ ms$, see Figures 3(a) and 3(b). Between $t_1$ and $t_2$, the performance corresponds to the performance values for 10.000 nodes: $N_{hops}(Chord) = 7$, $N_{hops}(Kademlia) = 3$, $t_r(Chord) = 700\ ms$, and $t_r(Kademlia) = 200\ ms$. The results in Figures 4 and 5 show severe unstable performance values for Chord, with much bigger confidence intervals. Additionally, when 7000 peers leave the network at the point $t_2$, Chord has one additional peak caused by dead contacts in the routing tables waiting for timeouts. In contrast to that, Kademlia sustains no significant performance decrease. This results from contact updating and data replication mechanisms used in Kademlia, which are more fitting to the NVE scenario than the data management strategies in Chord. Caching of results may help in fastening the lookup time of objects, but also introduces data consistency issues. However, for immutable data like landscape information in NVEs caching is a good strategy to improve the response time as well as prioritized processing of specific overlay operations [5].

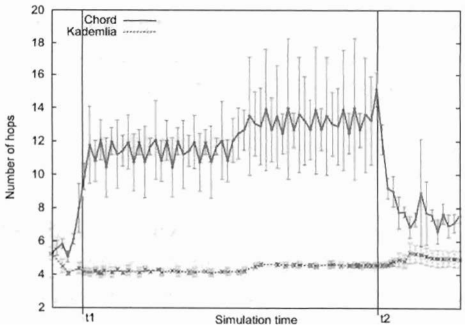


**Figure 4. Stability (Number of Hops) of Kademlia and Chord**

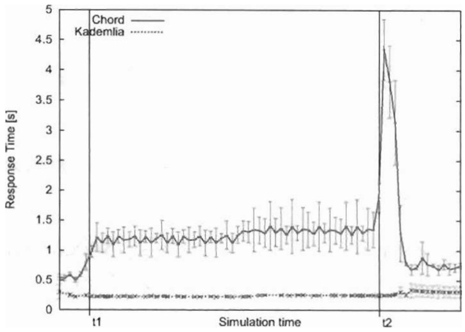
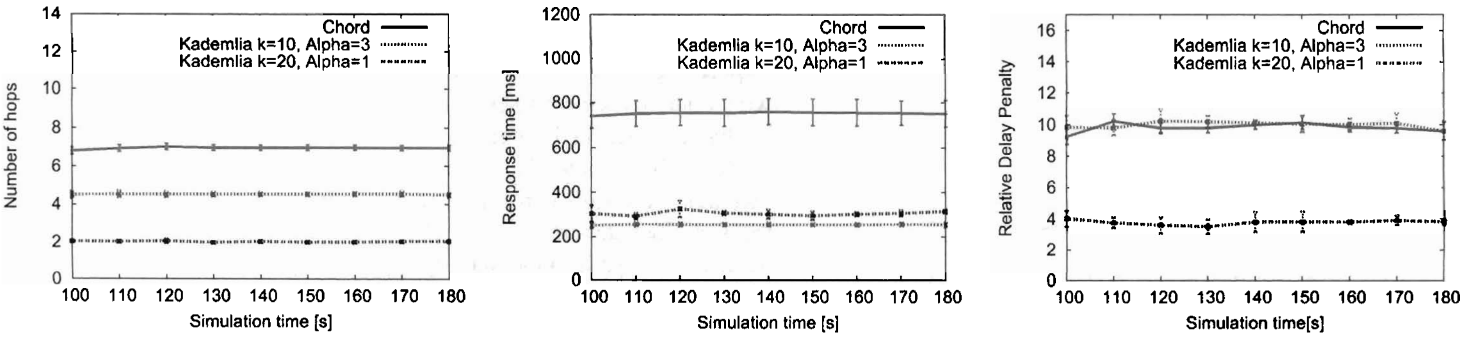


**Figure 5. Stability (Response time) of Kademlia and Chord**

## 4.4 Robustness

Experiments on robustness require the scenario 'Failures', as we are interested in the behavior of the overlays in unexpected situations. We present here results concerning metrics $N_{hops}$ and $t_r$ with an experiment size is 10.000 peers. Kademlia's bucket size $k$ is fixed to 10, $\alpha$ on 3 and the churn rate is set on mixed log-normal.

Comparing with the results regarding stability, performance variations are here considerably bigger. The response time is drastically increasing, especially in the case

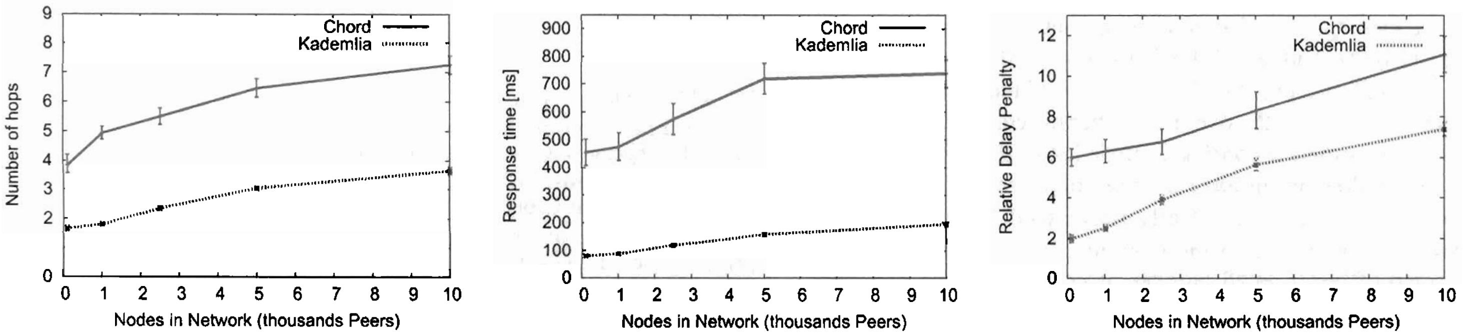


(a) Number of Hops

(b) Response time

(c) Relative Delay Penalty

**Figure 2. Efficiency of Kademlia and Chord**



(a) Number of Hops

(b) Response time

(c) Relative Delay Penalty

**Figure 3. Scalability of Kademlia and Chord**

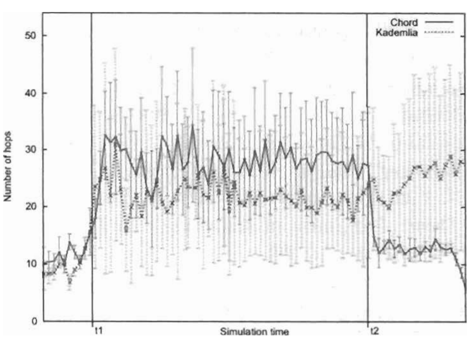


**Figure 6. Robustness (Number of hops) of Kademlia and Chord**

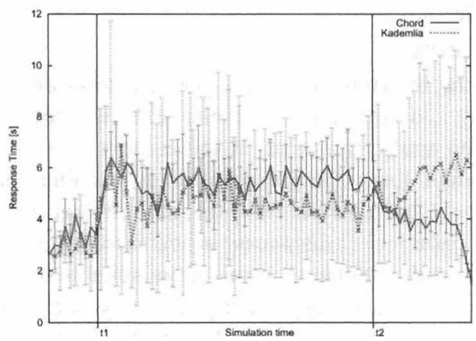


**Figure 7. Robustness (Response time) of Kademlia and Chord**

of Kademlia where even parallelism with $\alpha = 3$ could not help to maintain the performance under high failure rate. Both Kademlia and Chord need a much bigger number of hops to resolve the lookup queries after the group $g_1$ joined the network at the time $t_1$. While the average hop count $N_{hops}$ in Kademlia decreases slowly after getting its highest value, the number of hops in Chord stays around its maximum until the group $g_1$ leaves the network at the time point $t_2$. The Chord ring was broken between $t_1$ and $t_2$. The response time $t_r$ of Kademlia constantly grows between $t_1$ and $t_2$. Timeouts of lookup operations are increasing because sudden leaving of 2/3 of participants caused a big number of dead contacts in the routing tables. The reason for the large confidence intervals is that the number of online peers in the systems vary greatly responding to the mixed log-normal churn rate which reflects extreme variations of performance values. P2P infrastructures for NVEs should adopt contact management strategies like in Kademlia, where every message passing a reveals information about the network and may provide better contacts. The strict finger algorithm in Chord cannot cope with to much churn at the same time, as the Chord ring may break by that.

## 4.5 Load Balancing

As the load balancing experiments can be done with any scenario, we present experiments with the scenario 'Ideal',

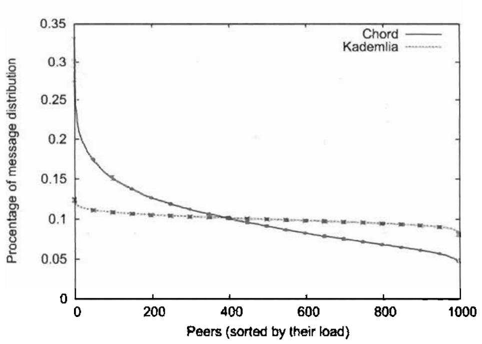


**Figure 8. Load Balancing of Chord and Kademlia**

using *message distribution* as main metric. The experiment size is 1000, Kademlia's bucket size $k$ is 20 and $\alpha$ is 3.

Figure 8 presents the share of received messages on each peer in relation to the overall number of received messages. Peers are sorted according to their load. Peers in Kademlia are following uniform distribution of the load whereas some Chord peers are clearly overloaded. The reason is that the Chord ring in the beginning of the simulation is formed of just a small number of nodes and therefore the most of the participants have fingers to those nodes. A suitable NVE supporting P2P overlay should either implement suitable strategies to share the load among the peers, like in Kademlia, or it should use a load dispatching strategy like presented in [6].

### 4.6 Summary

By having the benchmarking sets, we can fairly evaluate any overlay and compare the results with the requirements of a given application area. In the context of NVEs, Kademlia provides low response times using caching and parallel lookups. Chord suffers from its strict contact strategy (fingers in exponential distances) in scenarios with lots of churn. As a result, Kademlia seems more suitable to the needs of NVEs. However, any other overlay can be evaluated against Kademlia by adopting the benchmarking set, which enables researches to compare evaluation results.

## 5 Conclusion and Future Work

In this paper we defined a benchmark set in order to enable valid comparison of overlays, determining the impact of different design decision on performance (trade-offs), and identifying the a suitable overlay for the purpose of NVEs. We presented the benchmark set as a tuple of quality attributes $Q$, metrics $M$, and scenarios $S$ and identified the relevant quality attributes each peer-to-peer system has to cope with - efficiency, scalability, stability, robustness, and load balancing. The suitable definitions of those quality attributes for peer-to-peer overlay networks were given. We identified and defined the important metrics, and scenarios which are forming the benchmark for the certain quality attribute. The described benchmark set we used to evaluate and compare two overlays - Chord, as the most cited one in literature and Kademlia, as the most used in real world peer-to-peer applications with regard of NVEs. All experiments were done in the peer-to-peer simulator PeerfactSim.KOM. By having a defined scenario, a clear definition of metrics we were able to objectively compare these two overlays in a way that the results can validated with any simulator using the same benchmarking set. The results show the weaknesses of Chord in poor stability and robustness as well as worse efficiency and scalability than Kademlia, which benefits from its parallel lookup strategy and the caching mechanisms used. Future work will be focused on identifying trade-offs between the described quality attributes and evaluating various overlays against our benchmarks.

## References

[1] RFC-Gnutella 0.4. `http://rfc-gnutella.sourceforge.net/developer/testing/`.

[2] V. Darlagiannis. *Overlay Network Mechanisms for P2P Systems.* PhD thesis, TU Darmstadt, Germany, July 2005.

[3] DFG Research Group 733. QuaP2P: Improvement of the Quality of Peer-to-Peer Systems. http://www.quap2p.de.

[4] FastTrack. http://www.fasttrack.nu.

[5] K. Graffi et al. Overlay Bandwidth Management: Scheduling and Active Queue Management of Overlay Flows. In *IEEE LCN '07: Local Computer Networks*, 2007.

[6] K. Graffi et al. Load Balancing for Multimedia Streaming in Heterogeneous P2P Systems. In *ACM NOSSDAV*, May 2008.

[7] O. Heckmann et al. Qualitätsmerkmale von Peer-to-Peer-Systemen. Technical Report KOM-TR-2006-03, Multimedia Communications Lab, TU Darmstadt, May 2006.

[8] S. Jain, R. Mahajan, and D. Wetherall. A Study of the Performance Potential of DHT-based Overlays. In *USENIX Symposium on Internet Technologies and Systems*, 2003.

[9] S. Kaune, T. Lauinger, and A. Kovacevic. Embracing the Peer Next Door: Proximity in Kademlia. In *Proc. of Int. Conf. on P2PComputing*, 2008.

[10] A. Kovacevic, N. Liebau, and R. Steinmetz. Globase.KOM - a P2P Overlay for Fully Retrievable Location-based Search. In *Proc. of the 7th IEEE Int. Conf. on P2P Comp.*, Sep 2007.

[11] A. Kovacevic et al. Benchmarking platform for p2p systems. *it - Information Technology*, 49(5):312–319, Sep 2007.

[12] N. Liebau et al. The Impact of the P2P Paradigm. In *AMCIS*, Aug 2007.

[13] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. `http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.ps`.

[14] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM'01*, pages 329–350, Nov. 2001.

[15] S. Singhal and M. Zyda. *Networked Virtual Environments: Design and Implementation.* ACM Press/Addison-Wesley Publishing, 1999.

[16] I. Stoica et al. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.