Structured Overlay Networks as an Enabler for Future Internet Services

Strukturierte Overlay Netze als Wegbereiter für NGI Dienste

Gerald Kunzmann, Technische Universität München (TUM),
Andreas Binzenhöfer, University of Würzburg,
Fabian Stäber, Siemens Corporate Technology, Munich

Summary The next generation Internet will not only be defined by its technological progress but also by innovative Internet applications which offer new features, more interactivity, and a better user experience. Structured overlay networks, which create a well-defined virtual topology above the basic transport network, are a powerful means to easily create such Internet applications. There are many different approaches to realize structured overlay networks which in their core functions share the same basic principles. In this work we summarize the fundamentals of structured overlay networks, describe their inherent problems, and present an overview of our solutions. We then show how all these ideas have been put into practice in terms of a distributed carrier grade communication platform. ►►► **Zusammenfassung** Das Internet der nächsten Generation wird sich nicht nur durch einen weiteren technischen Fortschritt auszeichnen, sondern vor allem auch durch innovative Anwendungen, die neue Funktionen, mehr Interaktivität und eine verbesserte Nutzerfreundlichkeit mit sich bringen. Solche Anwendungen lassen sich besonders einfach mit strukturierten Overlay Netzen realisieren, die eine virtuelle Topologie oberhalb des eigentlichen Transportnetzes aufbauen. In der Literatur werden verschiedene Ansätze beschrieben, wie strukturierte Overlay Netze konstruiert werden können, im Kern basieren diese aber alle auf den gleichen Prinzipien. In dieser Arbeit fassen wir diese zusammen, beschreiben die grundlegenden Probleme von Overlay Netzen und geben einen Überblick über von uns entwickelte Lösungen. Abschliessend zeigen wir am Beispiel einer verteilten Kommunikationsplattform wie sich diese Ideen in die Praxis umsetzen lassen.

KEYWORDS C.2.1 [Computer Systems Organization: Computer-Communication Networks: Network Architecture and Design] distributed networks; C.2.4 [Computer Systems Organization: Computer-Communication Networks: Distributed Systems] distributed applications; overlay networks, Internet applications, next generation Internet / Overlay Netze, Internet Anwendungen, Internet der nächsten Generation

1 Introduction

In the last decades, the Internet has drastically changed the way we publish, exchange, and process information. While the technological progress provides an ever more powerful basis, it is the applications that account for the success of the Internet at the end of the day. Software vendors recognizing this situation no longer tend to provide off-the-shelf products but are already working on new ways to deliver the next generation of Internet applications. Those applications will be less static and include two-way web functionality like blogging, wikis, or interactive sharing and collaboration features.

It begins to show that this diversity of applications can no longer alone be shouldered by the still predominant client-server architecture. In this context overlay networks promise to be a means to already support next generation Internet applications today. The end-to-end characteristics of this architecture provide the possibility to place intelligence at the edge of the network, subsequently enabling any user to offer new custom developed services. Inspired by the success of Peer-to-Peer (P2P) applications such as file-sharing, the research community has been tightly focused on the fundamentals of overlay networks and P2P algorithms in the last few years [14]. Now it is time to put this new paradigm for creating Internet applications into practice.

In this work we give an overview of the basic principles behind structured overlay networks. In particular, we show what the different algorithms have in common, what makes them work, and most importantly we uncover the inherent problems and research challenges in the field of structured overlay networks. For the current set of the most important problems we give an overview of our own solutions and discuss how to utilize overlay architectures as the basis for next generation Internet applications. As an example we describe the main achievement of our work, a successfully implemented communication tool which is based on overlay technology containing our modifications. It revolutionizes the possibilities offered by a Private Branch eXchange (PBX) by moving its functionality from a central location to the user at the edge of the network.

2 Structured Overlay Networks

In P2P overlay networks peers establish logical connections between each other that are independent of the underlying physical network. While peers in unstructured P2P overlays set up random connections, connections in structured P2P overlays follow deterministic rules, thus creating a specific overlay structure, e. g., a ring or a torus. All peers and data items are assigned unique identifiers (IDs) that exactly determine their place in the overlay structure. This structure can then be exploited to efficiently resolve lookups for content. Thereby, each node is responsible for data items with IDs that are "close" to the node's ID. Churn, i. e., nodes joining and leaving the overlay, requires stabilization mechanisms that maintain the correct overlay structure.

2.1 Topologies

In Distributed Hash Tables (DHT) a hash function is applied to map nodes as well as objects to a common m-bit ID space. A node's ID can, e. g., be computed by hashing the node's IP address, whereas the filename could be used to obtain a hash value for shared data. The network structure is then set up by positioning nodes in the structure according to their ID.

In *Chord* [15] IDs are ordered in an m-bit identifier circle modulo 2^m (all operations are performed using modulo 2^m arithmetics). A key k is assigned to the first node (called successor node of key k) whose ID is equal to or follows k in the ID space. Nodes maintain connections to their successors as well as some shortcuts through the ring.

CAN [9] uses a d-dimensional ID space in order to set up a d-torus that is partitioned into zones. Each node is responsible for a certain zone, including all documents in that zone. Nodes maintain connections to all immediate neighbors along all dimensions.

Kademlia [8] sets up a binary tree, with the prefix of each node's ID corresponding to its position in the tree. Thereby, the exclusive or (XOR) function is used to determine the distance between two points in the ID space.

2.2 Routing

Most DHTs provide two kinds of pointers among peers: connections to all adjacent nodes to provide a correct resolution of lookups, and pointers to more distant peers, i. e., shortcuts through the overlay, to accelerate lookups. In Chord these shortcuts are called *fingers*, with the i^{th} finger of node n being the first node that succeeds n by at least 2^{i-1} $(1 \leq i \leq m)$. Then, the successor of a key k can be looked up by recursively searching the closest known predecessor of k, i. e., each peer transmits the query to its closest finger that is still preceding k. Thereby, the distance to k is at least halved with every iteration. If k's predecessor is reached, the query is forwarded to the successor of k. Like Chord, most DHTs provide a $O(\log_2 N)$ hop routing by storing $O(\log_2 N)$ deterministic pointers to other nodes in a network with N live nodes.

2.3 Scalability

We distinguish two types of scalability: Functional scalability indicates that the overlay scales well with the size of the network, and stochastic scalability refers to the influence of stochastic parameters, like the churn rate [1]. Thereby, the number of erroneous pointers and the corresponding stabilization overhead are the metrics for evaluating the correctness of the structure (stability), and metrics like path length [hops], latency [seconds], and success rate [%] evaluate the protocols' efficiency.

DHTs are designed for functional scalability: The path length scales with $O(\log_2 N)$, and the stability is almost independent of N, as stabilization is only performed in a limited local area of the overlay. In contrast to that, high churn rates are critical for maintaining the overlay as changes in the overlay must be detected, propagated, and repaired. Thereby, the detection of failed nodes consumes time as the absence of stabilization messages must be distinguished from a link congestion, and the propagation of changes consumes a high amount of network bandwidth.

3 Research Challenges and Solutions

DHTs come along with different research challenges, like short lookup delays, efficient overlay maintenance, self-organization, and complex queries. To be able to study these performance aspects in detail, we developed a highly scalable ANSI-C simulator for DHT overlays [2]. In the following, we will concentrate on Chord as its simple ring structure can be used to intuitively explain the basic functionalities. The results however are valid for DHTs in general. Results are evaluated by calculating the mean $E[X]$ of all samples X within one

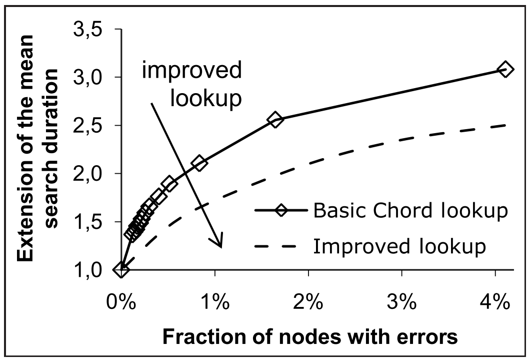

Figure 1 Mean lookup latency.

simulation. All experiments were run until the confidence intervals became negligibly small and could thus be omitted in the following figures.

In this section we concentrate on the stability of the overlay and its maintenance at a moderate bandwidth consumption. We show, that the number of erroneous pointers significantly increases for high churn rates. Moreover, the more erroneous pointers exist, the higher the lookup latency and the lower the lookup success rate. Figure 1 shows this correlation for two different lookup algorithms, the basic Chord implementation and an improved variant. In this article we concentrate on reducing the number of erroneous pointers by improving the stabilization scheme. Improved lookup strategies may be found in related work, e. g., [3; 12].

3.1 Overlay Stability

The correctness of the structure is crucial for structured overlay networks. Only valid neighbor information assures that content is stored and queried at the correct node. If nodes use a pointer to a node that is no longer participating in the network the sent packet is lost. Therefore, a timer monitors if an acknowledgment for the packet is received, otherwise the packet must be retransmitted. The timer value must be significantly larger than the average round trip time (RTT) in order to avoid duplicate packets. Thus, each timeout noticeably increases the average lookup delay.

Chord uses a very simple stabilization scheme. Each node n stores contact information (ID, IP:port) of its direct successor s and predecessor p on the ring. Node n periodically sends a *stabilize* message to its direct successor s. Receiving this message node s returns the contact information of its predecessor p.

Joining and *leaving* nodes inform their neighbors, thus avoiding any inconsistencies in the structure. However, the *failure* of nodes, e. g., due to a link break or power cut, must be detected by their neighbors. In Chord, if no answers are received on successive stabilization messages, nodes must assume that their successor has failed. In order to replace a failed successor, each node maintains a list of about $2 \cdot \log_2 N$ successors. Node s transmits its successor list l in its stabilize responses to its predecessor p. Node p adds s at the front of l, thereby deleting the last element, and replacing its own successor list with l.

We propose an improved stabilization algorithm, where nodes store neighbor lists including successors and predecessors and transmit these lists in the stabilize messages. In contrast to Chord, the messages are transmitted to both neighbors on the ring each T seconds, and there is no reply to them. However, the failure of nodes can still be detected if no stabilize messages are received from a neighbor for a certain period. This timeout is set to $2.5 \cdot T$ in our implementation.

In order to update the neighbor lists even faster, nodes that detect a failed neighbor send notification messages to all nodes in their neighbor list, informing them about the failure. Thus, almost all neighbor lists that contain the failed peer can be updated immediately. Accordingly, new nodes, after copying the neighbor list from their successor, announce their presence to all nodes in that list. This variant is a mixture of periodic and reactive stabilization [10]. The extra traffic introduced by reactive stabilization is justified by a significantly more robust overlay structure. Attacks that exploit such failure messages might be prevented by signed messages, however security issues are not addressed in this article. The extra traffic introduced by the notifications can be compensated by longer stabilization periods, as the notifications already update the neighbor lists almost completely.

However, we noticed that sometimes the order of the stabilization messages from different neighbors becomes mixed up as nodes send stabilization messages independent of each other. Therefore, we apply some of the well understood techniques from token rings to the Chord protocol [6]. We use token-like stabilization packets that circulate in both directions of the ring. Each token contains a list l of the last nodes that it has passed. When receiving a token, nodes can copy the information included in the token to their neighbor list. Thereby, tokens that circulate in order of the ID space are used to update the predecessor entries and vice versa. If a node n receives a token from another node that is not a direct neighbor, n notifies the sender of the token about its correct neighbor or at least a node that is located closer to the sender. This ensures that the token is always passed along the complete ring without skipping any nodes. If n receives the token from a neighbor, the list is shifted by one entry, discarding the farthest entry, and n inserts itself at the top of the list. Finally, the token is forwarded to the next node in the direction

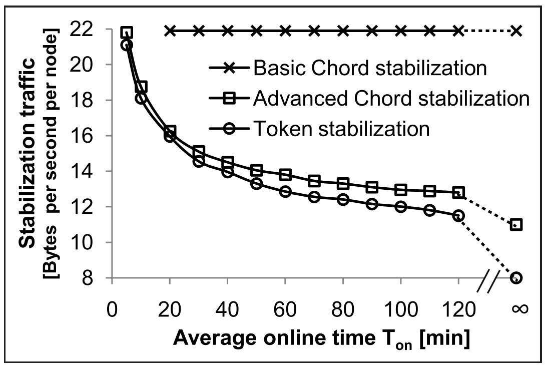

Figure 2 Mean stabilization overhead.

of the token. Hereby, Chord's ring-shaped structure is especially suited for this algorithm. Other topologies are not able to use token-like stabilization messages, and thus must cope with asynchronous stabilization messages.

Figure 2 shows the resulting stabilize traffic for different average online times for the basic Chord stabilization, the advanced stabilization using notifications and the token stabilization. The rightmost x-value corresponds to a stable system without any churn. In order to be able to compare the different protocols to each other, we defined the stabilization periods T for the different algorithms in such a way, that the signaling traffic is identical for simulations with a mean online time of 5 min. These settings for T are kept constant for all simulations. Thus, all curves theoretically intersect at this x-value. (Note, Chord is not able to maintain a correct overlay with this setting for high churn rates, thus, no values for mean online times smaller than 20 min exist.) The average traffic is constant for the basic Chord protocol, as stabilize is executed periodically and independent of the churn rate. In contrast to that, the signaling traffic is significantly less in the other two implementations, especially for relatively stable scenarios. While the improved algorithms are able to maintain the overlay structure even for short average online times, the stabilize traffic considerably increases for high churn rates.

However, for all algorithms the fraction of nodes with errors in their neighbor list increases with high churn rates, causing serious damage to the overlay structure (see Fig. 3). The figures also show that token stabilization results in the most stable network, although it generates less overhead than the other implementations. Of course, a more stable network might be achieved for all variants if the stabilization period is increased.

3.2 Self-Organization

In order for a structured overlay network to efficiently operate in practice, one has to adjust parameters like the number of overlay connections to other peers or the frequency at which information about the current overlay status is exchanged with those peers. Mahajan et al. [7] investigated the trade-off between high maintenance cost and poor stability in dynamic networks. The results show that it is crucial to adapt parameters dynamically. However, the optimal amount of such maintenance overhead directly depends on the current size of the overlay as well as the current online/offline behavior of the participating peers. The main problem in this context is that to a single peer the remaining system essentially appears as a black box. In practice, the maintenance overhead in structured overlay networks is therefore set to a fixed value which is dimensioned for the expected worst case.

In this section, we sketch a simple three-step concept as illustrated in Fig. 4 which leads toward a more self-organizing overlay structure that automatically adapts itself to the current state of the system. Applying this concept the peer measures the current conditions in the overlay, evaluates the corresponding performance and then adapts its parameters accordingly. The evaluation step can be implemented using basic methods from stochastic calculus to derive important performance measures like the probability to lose all known overlay neighbors. In the measurement process, however, the peer needs to estimate variables like churn or the system size, information which is inherently not available to the peer.

Through its routing tables a peer usually knows the IDs of its direct neighbors in the overlay. Given that there are 2^m possible identifiers, a very simple idea to estimate the number of peers is to measure the average distance of the neighbors in the ID space and dividing 2^m by this number. To obtain a more sophisticated estimate, it can be shown that, assuming random identifiers, the distance between two direct overlay neighbors is approximately geomet-

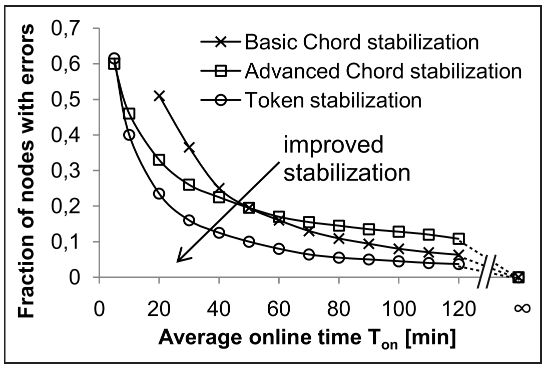

Figure 3 Mean overlay stability.

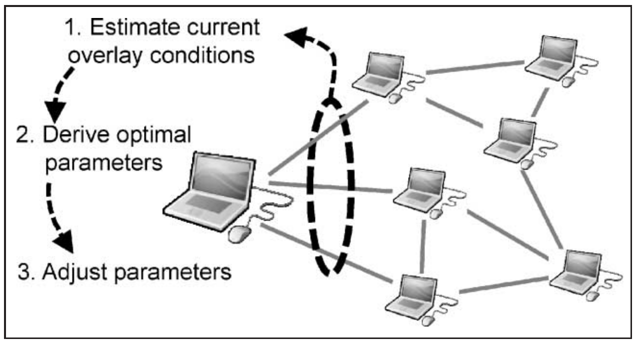
1. Estimate current overlay conditions
2. Derive optimal parameters
3. Adjust parameters

Figure 4 Self-organization concept for overlay networks.

rically distributed with $p = \frac{n}{2^m}$ [1]. Due to the memoryless property of this distribution, we can then find additional intervals like the distance between the theoretical and the actual position of a finger in Chord which are also geometrically distributed with the same parameter p. Such intervals can then be used as additional input to a maximum likelihood estimator which then results in a more accurate estimate.

The behavior of the peers in the overlay can be estimated by observing a peer's overlay neighbors and from this deriving the distribution of their online time. In general, the more observations a peer maintains the more accurate its estimate is going to be. To express accuracy, we regard the deviation of the estimate from the actual value in percent. In Fig. 5 we consider how much the 97.5% and 2.5% quantiles of the estimated values based on k observations differ from the actual value in percent. The accuracy of the estimate increases exponentially with k.

An increased accuracy, however, comes at the cost of a reduced responsiveness of the estimator. Thereby responsiveness is defined as the time it takes until the estimator reacts to changes of the general peer behavior and thus the time it takes to obtain k fresh results. This responsiveness can either be improved at the cost of accuracy or by increasing the overhead involved in the estimation process, i. e., by increasing the number c of overlay neighbors a peer shares its observations with.

In Fig. 6 the responsiveness of the estimator is expressed in multiples of the mean online time $E[T_{on}]$ of a peer. The responsiveness degrades linearly with k while it can be improved by increasing the number of overlay neighbors, e. g., from 20 to 40 as shown in the figure. In practice, the desired accuracy, the responsiveness as well as the overhead one is willing to spend can thus directly be adjusted by k and c.

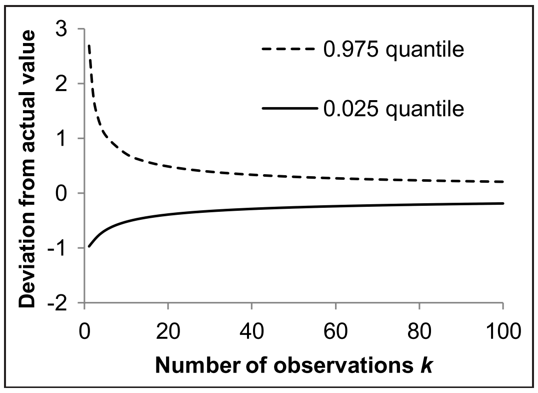

Figure 5 Accuracy of the estimates.

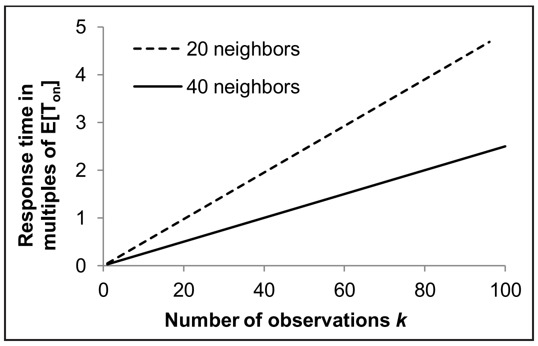

Figure 6 Responsiveness of the estimator.

4 The PeerThings Communication Platform: An Industrial Application Scenario

In this section we introduce the Siemens PeerThings project[1] as a real-live industrial application for large-scale structured overlay networks. As part of the PeerThings project, Siemens developed a decentralized communication platform supporting video communication, voice communication, instant messaging, etc. using an underlying Chord-like DHT. The scalability as well as the reliability of this DHT were evaluated and improved in a joint research-project with both the University of Munich and the University of Würzburg.

In order to compete with a highly reliable fixed public switched telephone network (PSTN) the correct, fast, and efficient lookup of data stored in the DHT is a crucial requirement. However, a basic Chord implementation was not able to cope with these conditions. Amongst other things, the challenges presented in this paper had to

[1] PeerThings was presented at the CeBIT trade show in 2006.

be solved. Especially, the high churn rates as observed in already deployed overlay networks presented a serious challenge. In addition, a dynamic adjustment of several design parameters was required to build a highly stable network while keeping maintenance costs low.

Furthermore, additional challenges, like a distributed lookup of users in a phone-book-like user directory, had to be solved by the authors. Like a printed phone book, the user directory must support range queries, like queries for all users with a certain last name in a certain city. The main challenge of such a user directory rises from the non-uniform distribution of last names. That is, there are a few very common names, and many names that appear very infrequently. As part of the PeerThings project, we evaluated related work on search indexes for DHTs [11], and we found that these approaches result in heavily overloaded peers or high latency when common names are queried. As a solution, we introduced the Extended Prefix Hash Tree (EPHT) [13], which is a tree-shaped data structure that is mapped onto the DHT and stores the user data in the leaf nodes. An evaluation with the phone book of Munich (> 500 000 entries) showed that the requirements for data load and latency can be fulfilled.

5 Conclusion

The size and complexity of current computer networks call for new functionality which was not a part of their initial design. Structured overlay networks are a means to provide the necessary corrections on top of the actual network. However, while theoretically understood in detail, structured overlays have not yet been widely deployed in the Internet. In this paper we therefore uncovered the practical challenges which are inherent to structured overlay mechanisms and described how to approach them. In particular, we summarized how to improve the stability of the overlay using only negligible maintenance overhead. To be able to apply these modifications in practice we described how to estimate important system parameters, which are needed as input to our algorithms, at runtime. Finally, citing the Siemens prototype PeerThings as an example we showed that in combination with our modifications structured overlay networks offer an easy way to realize new functionality on top of a network which itself would be hard to modify. Based on our results, we strongly believe that future Internet applications will heavily rely on such highly distributed overlay networks and thereby shift the intelligence from the core to the edges of the network.

References

[1] A. Binzenhöfer. Performance Analysis of Structured Overlay Networks *Dissertation*, University of Würzburg, 2008.

[2] A. Binzenhöfer, T. Hossfeld, G. Kunzmann, and K. Eger. Efficient Simulation of Large-Scale P2P Networks. In: *IJCSE – Special Issue on 'Parallel, Distributed and Network-Based Processing'*, Inderscience Publishers, 2008.

[3] M. Castro, P. Druschel, Y. Hu, and A. Rowstron. Exploiting network proximity in distributed hash tables. In: *Proc. FuDiCo 02*, 2002.

[4] R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In: *Proc. ACM SIGCOMM 03*, 2003.

[5] G. Kunzmann, A. Binzenhöfer, and R. Henjes. Analyzing and Modifying Chords Stabilization Algorithm to Handle High Churn Rates. In: *Proc. MICC i.c.w. ICON 05*, 2005.

[6] G. Kunzmann, R. Nagel, and J. Eberspächer. Increasing the reliability of structured P2P networks. In: *Proc. DRCN 05*, 2005.

[7] R. Mahajan, M. Castro, and A. Rowstron. Controlling the Cost of Reliability in Peer-to-Peer Overlays. In: *Proc. IPTPS 03*, 2003.

[8] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In: *Proc. IPTPS 02*, 2002.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content Addressable Network. In: *Proc. ACM SIGCOMM 01*, 2001.

[10] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. *UCB/CSD-03-1299*, EECS Department, University of California, Berkeley, 2003.

[11] J. Risson and T. Moors. Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. In: *Computer Networks*, vol. 50, Elsevier Science, 2006.

[12] D. Rossi and I. Stoica. Gambling Heuristics on a Chord Ring. In: *Proc. GLOBECOM 05*, 2005.

[13] F. Stäber, G. Kunzmann, and J. P. Müller. Extended Prefix Hash Trees for a Distributed Phone Book Application. In: *Proc. ICPADS 07*, 2007.

[14] R. Steinmetz and K. Wehrle. Peer-to-Peer Systems and Applications. In: *LNCS*, Vol. 3485, Springer, 2005.

[15] 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. ACM SIGCOMM 01*, 2001.

1

2

3

1 **Dipl.-Ing. Gerald Kunzmann** works as member of the research staff at the institute of communication networks at the TUM. His main research interests are the stability and resilience of structured P2P networks, as well as the deployment of P2P methods in Voice-Over-IP applications.

Address: Technische Universität München (TUM), Institute of Communication Networks, Arcisstr. 21, 80290 Munich, Germany, Tel.: +49-89-28923506, Fax: +49-89-28923523, E-Mail: gerald.kunzmann@tum.de

2 **Dr. Andreas Binzenhöfer** studied in Würzburg, Germany, and at the UCLA in California, USA. Since 2002 he is a research assistant at the chair of distributed systems at the University of Würzburg. His current research areas are performance analysis of structured P2P systems and their application to distributed network management.
Address: University of Würzburg, Institute of Computer Science, Am Hubland, 97074 Würzburg, Germany, Tel.: +49-931-8886654, Fax: +49-931-8886632, E-Mail: binzenhoefer@informatik.uni-wuerzburg.de

3 **Dipl.-Inf. Fabian Stäber** is a doctoral candidate, working at the peer-to-peer center of competence at Siemens Corporate Technology. His focus is on industrial applications of peer-to-peer overlays.
Address: Siemens Corporate Technology, Information and Communications, Otto-Hahn-Ring 6, 81730 Munich, Germany, Tel.: +49-89-63641619,
E-Mail: fabian.staeber.ext@siemens.com