Hybrid DHT Design for Mobile Environments

Stefan Zoels[1], Simon Schubert[1], Wolfgang Kellerer[2], and Zoran Despotovic[2]

[1] Institute of Communication Networks, Munich University of Technology, Germany
stefan.zoels@tum.de, corecode@fs.ei.tum.de
[2] Future Networking Lab, DoCoMo Communications Laboratories Europe, Germany
{kellerer, despotovic}@docomolab-euro.com

Abstract. In this paper we present a hybrid design concept for Distributed Hash Tables (DHTs), in order to increase the performance of DHTs in scenarios with mobile participants. By defining two classes of nodes (static and temporary) and assigning critical overlay networking tasks to reliable static nodes, our concept allows the disburdening of resource-constrained temporary nodes such as PDAs or mobile phones. Further we present an implementation of our system design, based on the Chord protocol, in the Network Simulator 2 (NS-2) and in the overlay simulator L7Sim and show simulation results that prove the significant advantages of our extension in comparison to conventional DHTs.

1 Introduction

Distributed Hash Tables (DHTs) are currently a major subject of research in the area of distributed computing and Peer-to-Peer (P2P) networks in particular. Their two key properties – hash table like lookup interface and extreme scalability – turn out to be sufficient for building large scale distributed applications. Additionally, in contrast to unstructured P2P networks, they avoid flooding of query messages, thus reducing the average number of search hops to $O(\log n)$ for a network with n nodes. As a result the signaling traffic in the overlay network decreases significantly.

However, the current mainstream research on P2P concentrates on fixed IP networks consisting of functionally equal nodes. As such, it usually neglects eventual heterogeneity among the participating computing devices. In this paper we focus on extending current DHTs to mobile environments. In order to do so, we have to be aware of the challenges resulting from this large heterogeneity of participating nodes, ranging from hard-wired work stations to GPRS-connected mobile phones:

- Limited resources of mobile devices (CPU power, RAM size, storage capacity) as well as low access data rates have to be addressed. Moreover, devices cannot be modeled as one class of nodes but their heterogeneity requires different consideration of different types of nodes.
- High costs for mobile data transfer lead to short online times of mobile participants. Resulting we face a highly dynamic environment, characterized by high churn rates.

S. Joseph et al. (Eds.): AP2PC 2006, LNAI 4461, pp. 19–30, 2008.

– The increased failure probability of mobile devices (due to wireless link breaks, discharged batteries...) can result in a high number of lost object references, which in turn may result in the (at least temporary) unavailability of shared objects.

In this paper we address these requirements by proposing a hybrid DHT design. We define two classes of nodes, which we call 'static' and 'temporary', and we assume that static nodes both perform routing tasks and maintain references to the available objects in the system, while temporary nodes only perform routing. In this way we disburden temporary nodes and avoid shifting object references when temporary nodes join or leave the system. The result is significantly decreased overall maintenance traffic.[1] Besides, we emphasize that this approach has another important advantage: It enables the low performance nodes (e.g. mobile terminals) to participate in a DHT based P2P network.

The work presented in this paper is an extension and a generalization of our previously proposed Hybrid Chord Protocol [1].

The paper is organized as follows. Section 2 gives an overview of distributed hash tables. Section 3 presents our hybrid DHT design in detail. In Section 4 we present an illustrative set of simulations we performed to test the performance of the proposed design. Section 4.1 illustrates the setup of simulation scenarios with our traffic generator, while Section 4.2 shows and discusses simulation results. Section 5 concludes this paper and gives an outlook to future work.

2 Distributed Hash Tables - Overview

The basic problem DHTs address is self-organized distribution of a set of objects among a set of peers, enabling their subsequent fast lookup. In a DHT peers collaboratively manage specific subsets of objects, identified by keys from a key space K, which depend on the set of all peers and the set of all objects available in the system. This is done by associating each peer with a key taken from K and also associating with this key a partition of the key space such that the peer becomes responsible to manage all objects identified by keys from the associated partition. Typically the key partition consists of all keys closest to the peer key in a suitable metric. Thus the key space K is equipped with a distance function d. To forward query requests peers form a routing network by taking into account the knowledge on the association of peers with key partitions.

In short, any DHT is equipped with a function $key : P \to K$ that associates peers with keys and, given $key(P)$, a function $partition : K \to 2^K$ associating peers with partitions of K, and a function $neighbors : K \to 2^P$ that associates each peer with a subset of other peers, making thus an overlay graph G [2].

Function *key* is a hash function mapping a peer's IP address or a randomly chosen string into a hash value. Please note that the set of all participating peers at any time can be considered as parameter of the function *partition*; the interpretation is that the objects must be assigned to the peers that are currently

[1] Throughout the remaining paper we use the term 'maintenance traffic' synonymously for the number of object references shifted to a joining node or from a leaving node.

present in the system. A side goal of using a hash function to map object keys to peers is balancing the load distribution: each peer should be responsible for approximately the same number of keys. The function *neighbors* is responsible for building the DHT routing graph. Using the metric of the key space, it normally enables peers to maintain short-range links to all peers with neighboring keys and in addition a small number of long-range links to some selected peers. Using thus established routing graph, peers forward query requests in a directed manner to other peers from their routing tables trying to greedily reduce the distance to the key that is being looked up. Most of DHTs achieve by virtue of this construction and routing algorithms lookup with a number of messages logarithmic in the size of network by using routing tables which are also logarithmic in the size of the network [3,4]. However, in recent few years there have been also some works that achieve constant outdegree graph topologies and consequently constant sized routing tables while retaining logarithmic routing [5,6]. To sum up, the specific designs of these structures depend on the choice of key space, distance function, key partitioning, and linking strategy. They have been subject of intensive research over the recent years and resulted in numerous designs of structured overlay networks.

However, the good properties related to the efficiency of routing do not come for free. For constructing and maintaining a structured P2P network peers have to deal in particular with the problem of node joins and failures. Since the freedom to choose neighbors in a structured P2P network is constrained by the conditions imposed by the function *neighbors*, maintenance algorithms are required to re-establish the consistency of routing tables in the presence of network dynamics. Depending on the type of guarantees given by the network different deterministic and probabilistic maintenance strategies have been developed. Maintenance actions can be triggered by various events, such as periodical node joins and leaves or routing failures due to inconsistent routing tables. The different maintenance strategies trade-off maintenance cost versus degree of consistency and thus failure resilience of the network.

3 Hybrid DHT Design

The main goal of our hybrid DHT design is to enable participation of mobile devices such as PDAs or mobile phones in a DHT based P2P lookup system. It sets up a hybrid overlay structure by extending a given conventional DHT protocol as to define two different types of nodes: static nodes and temporary nodes. Static nodes are reliable nodes in the overlay network that are characterized by long online times, low failure probabilities and good hardware resources (e.g. office computers with hard-wired connections to the Internet). All other, low-performance nodes in the overlay network (e.g. all mobile participants such as GSM mobile phones or WLAN PDAs) are temporary nodes.

We require a minor modification to the object mapping rules of the DHT: In contrast to the conventional DHT protocol, a reference to a shared object is stored on the closest *static* node of the object's key. (The term "closest" refers to

the conventional DHT's distance metric: in Pastry it is the height of the smallest tree containing the two considered nodes, in Chord it is the simple difference of the nodes' identifiers.) In contrast, temporary nodes maintain only a pointer to their succeeding static node. Thus whenever a temporary node in the network receives an INSERT or QUERY request (due to its responsibility for the key given in the request) it simply forwards this request to its closest static node which in turn stores the according reference(s). Such hybrid structure can be realized by calling different JOIN and LEAVE procedures when nodes connect to or quit the overlay network, depending on the node class that this node belongs to (see pseudocode in Figure 1).

```
n.joinStatic()
    setupRoutingTable();
    n = find_next_static(n.id);
    transfer_matching_references(n);
    start_timers();

n.joinTemporary()
    setupRoutingTable();
    n = find_next_static(n.id);
    set_next_static(n);
    start_timers();

n.leaveStatic()
    n = find_next_static(n.id);
    transfer_references(n);
    inform_neighboring_nodes();
    stop_timers();

n.leaveTemporary()
    inform_neighboring_nodes();
    stop_timers();
```

Fig. 1. Pseudocode for hybrid system structure setup

The differentiation between static and temporary nodes has three major advantages:

- The heterogeneity of the participating nodes that results from extending the overlay network to mobile environments is addressed.
- The maintenance traffic in the overlay network can be decreased significantly, as object references have to be shifted only when static nodes (which have long online times) join or leave the overlay network. Moreover, resource-constrained temporary nodes are prevented from storing and providing object references.
- Only reliable static nodes with low failure probability store references to shared objects. Resulting, the probability that an object is available in the system but the node(s) that is (are) responsible for storing a reference to it has failed is reduced. Consequently, the availability of provided content increases.

The extension we just presented assumes that there are two classes of nodes, static and temporary, defined independently of the current state of the system (i.e. the properties of the nodes available in the system). Thus any node can unambiguously determine to which class it belongs. In principle, it should be possible to make a step further and remove this constraint as to make a node class dependent on the system state at any time instant (a joining node might assess the current state of the system based on the properties of the nodes encountered in the joining process) and enable the total work division according to the nodes relative capabilities. We plan to investigate such extension in the future.

4 Simulations

In this section, we present the results from a series of simulations, in which we compare the conventional Chord protocol [4] and the Hybrid Chord Protocol (HCP), obtained by applying the above described modifications to Chord. The results show the significant advantages of HCP in scenarios with resource-constrained mobile participants.

4.1 Simulation Setup

To evaluate and compare the performance of conventional Chord and HCP we implemented both protocols in the Network Simulator 2 (NS-2) [7]. Since NS-2 simulates the complete packet flow through all layers of the ISO/OSI reference model, it requires a high amount of CPU power and Random Access Memory. Resulting, the size of the simulated overlay network is limited to only a few hundred nodes. In order to be able to simulate even larger overlay networks (typical for P2P networks) we modified the NS-2 implementation of both protocols so that only the overlay network is simulated. With this so-called L7Sim (Layer 7 Simulator), messages are exchanged directly between the P2P applications, without making use of the underlying layers of the ISO/OSI reference model. The delay of physical links is thereby represented by an equally distributed delay between 10 ms and 200 ms. The convergence of both approaches is shown in Section 4.2. To set up simulation scenarios we implemented a traffic generator that performs the following tasks:

- **Definition of different node classes.** For the simulated overlay networks, different classes of participating nodes can be defined. Appropriate parameters for node classes are the mean online time, the failure probability (i.e., the probability that a node leaves the overlay network without notifying other nodes), the number of shared objects, and the average query rate. For NS-2 simulations, also the data rate and the delay of the physical link to the core network can be defined.
- **Creation of an initial overlay network.** The traffic generator creates an initial Chord/HCP overlay structure with a given number of nodes, including the setting of predecessor pointer, successor list, finger table, next static pointer (for temporary HCP nodes) and provided content. For NS-2 simulations, it also connects the overlay nodes to the core network. The core network emulates the physical IP connections between the overlay nodes. It consists of 100 core nodes and is created with the BRITE [8] topology generator.
- **Generation of an eventfile.** The eventfile is created according to the specified parameters and is used as input for both network simulators.

Figure 2 shows the setup of an exemplary simulation scenario with the traffic generator using three node classes. In detail the simulation process runs as follows: Firstly, the traffic generator reads the scenario file and generates – according to the parameters given in the scenario file – an output file containing the initial

```
nodeclass WLAN_NOTEBOOK
        mean_online_time 3600s
        failure_probability 10%
        shared_objects 50
        query_rate 300s
        link datarate 1Mb delay 10ms
nodeclass UMTS_PHONE
        mean_online_time 1800s
        failure_probability 25%
        shared_objects 20
        query_rate 120s
        link datarate 384kb delay 120ms
nodeclass GPRS_PHONE
        mean_online_time 900s
        failure_probability 50%
        shared_objects 5
        query_rate 60s
        link datarate 100kb delay 400ms
initial
        100 WLAN_NOTEBOOK
        100 UMTS_PHONE
        100 GPRS_PHONE
simulation-duration 1h
```

Fig. 2. Scenario file for the setup of a simulation scenario

overlay network as well as simulation events. Simulation events are composed of NODE-JOIN, NODE-LEAVE, NODE-FAILURE and QUERY events. Secondly, the generated file is taken as input for the used network simulator, which in turn produces a tracing file that can be analyzed with appropriate evaluation tools.

4.2 Simulation Results

Based on multiple independent simulations, we evaluate HCP in comparison to the conventional Chord protocol. The focus of the following simulations is put onto the decreased maintenance traffic and the increased availability of provided content that can be achieved with HCP.

In a first simulation we set up a network with 100 overlay nodes which are connected randomly to the core network. The main goal of this simulation is to evaluate the differences between simulating the overlay network on top of a physical network using the complete protocol stack (NS-2) and simulating the overlay network independently, without considering the physical topology (L7Sim). Thus we want to determine the impact of lower-layer parameters such as queue length, packet loss or link latency on our simulations. Table 1 shows all relevant simulation parameters for this simulation scenario.

By varying the mean online time[2] of temporary nodes from 600 s to 1800 s, we create seven different independent eventfiles. We simulate each eventfile with

[2] The traffic generator determines the online time of each participating node following a negative-exponential distribution, with mean value given in the node class definition in the scenario file.

Table 1. Simulation parameters for scenario 1

Number of node classes:	2	
Node class:	*STATIC*	*TEMPORARY*
Mean online time:	1800 s (neg. exp. dist.)	600 s - 1800 s (neg. exp. dist.)
Number of shared objects:	10 per node	10 per node
Physical link:	1 Mb/s, 10 ms delay	100 kb/s, 100 ms delay
Mean number of nodes:	100	
Partitioning:	10 static, 90 temporary	
Simulation duration:	4 hours	

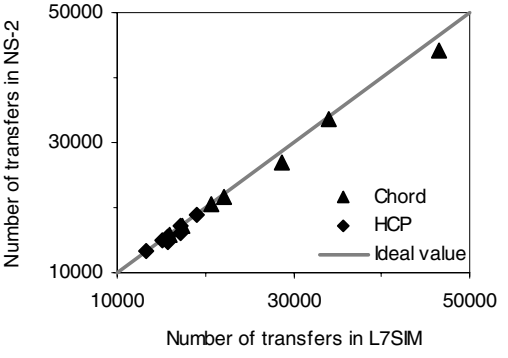

Fig. 3. Scenario 1: Maintenance traffic in both simulators

both protocols in NS-2 and in L7Sim, and compare the total number of transferred object references (i.e. the resulting maintenance traffic) in both simulators.

As we see from Figure 3, the measured numbers of transferred object references are nearly the same in both simulators. The marginal differences in NS-2 result from a negligible packet loss in the physical layer. Resulting we can state that simulating only the overlay network, without considering the underlying physical topology, is sufficient for our analysis.

The basic criterion for comparing maintenance traffic in HCP and in Chord is the ratio α of the mean online time of static nodes in HCP and the mean online time of all nodes in Chord:

$$\alpha = \frac{\text{Mean online time of static nodes}}{\text{Mean online time of all nodes}}$$

As stated in section 3, HCP stores object references only on static nodes. Therefore, the mean online time of static nodes is crucial for the maintenance traffic in HCP, as object references have to be shifted whenever a static node joins or leaves the overlay network. In contrast, Chord stores object references on all nodes in the overlay network, so the mean online time of all nodes is decisive for the maintenance traffic in Chord. By theoretical evaluation (see Appendix) we can show that the maintenance traffic in HCP is lowered by a factor of $1/\alpha$ in comparison to Chord.

In the above simulation scenario we obtain different values for the ratio α from the varying mean online time of temporary nodes. Figure 4 shows a comparison of the resulting maintenance traffic. It illustrates the percentage of transferred object references in HCP in comparison to Chord, depending on the ratio of mean online times α. The simulation results coincide with our theoretical evaluation that HCP reduces maintenance traffic by a factor of $1/\alpha$ compared to Chord.

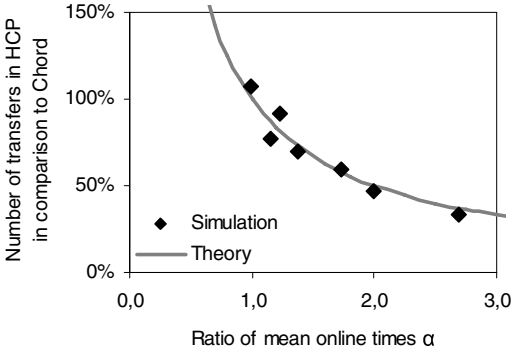

Fig. 4. Scenario 1: Maintenance traffic in HCP compared to Chord

Our next simulation aims at the verification of this theoretical evaluation in a large overlay network with a high percentage of mobile participants, and by a lot of different independent simulation runs. Therefore, we create multiple simulation scenarios according to the setup parameters given in Table 2. Please note that we vary the mean online time of static nodes from five to fifty minutes.

Table 2. Simulation parameters for scenario 2

Number of node classes:	2	
Node class:	*STATIC*	*MOBILE*
Mean online time:	300 s - 3000 s (neg. exp. dist.)	300 s (neg. exp. dist.)
Number of shared objects:	1 per node	1 per node
Mean number of nodes:	1000	
Partitioning:	100 static, 900 mobile	
Simulation duration:	2 hours	

Due to the high number of overlay nodes, and based on the findings of our first simulation we confine ourselves to simulate this scenario only in the overlay simulator L7Sim. We generate 56 different eventfiles with a ratio of mean online times α ranging from 0.99 to 8.93. The individual values for α result directly from the varying mean online time of all participating nodes in each scenario file. In Figure 5, the resulting maintenance traffic of all 56 simulation runs is depicted for both protocols. As expected, the total number of transferred object

references in Chord nearly stays at a constant level, because the mean online time of all nodes in the overlay network only changes slightly (please note that 90% of the overlay network is formed by mobile nodes that have a constant mean online time of about 300 s). On the other hand the mean online time of static nodes rises from 300 s to 3000 s in average. Along with this increasing ratio α comes significantly decreased maintenance traffic in HCP. With this simulation we can prove our theoretical evaluation: As we can see in Figure 5, the amount of transferred object references in HCP decreases with $1/\alpha$, while it remains constantly high in Chord.

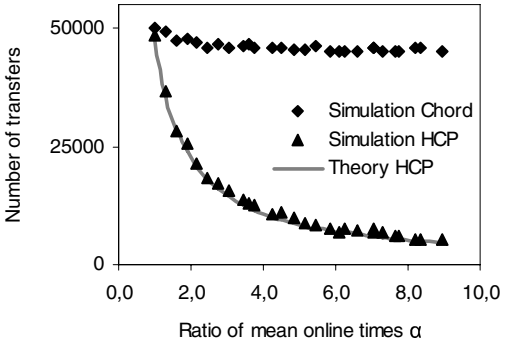

Fig. 5. Scenario 2: Maintenance traffic

So far, we have considered theoretical simulation scenarios with only two different node classes. To evaluate HCP in a more realistic scenario, we set up a heterogeneous overlay network with 2000 nodes, partitioned into five different node classes: 100 office computers, 700 DSL subscribers, 400 ISDN users, 400 PDAs, and 400 mobile phones. Table 3 illustrates the modeling of these nodes classes.

Table 3. Simulation parameters for scenario 3

Number of node classes:	5				
Node class:	*OFFICE*	*DSL*	*ISDN*	*PDA*	*PHONE*
Mean online time:	24 h	2 h	30 min	10 min	2 min
Failure probability:	0.1%	5%	10%	35%	50%
Number of shared objects:	0-30	0-30	0-15	0-8	0-5
Average query rate: 1 every...	10 min	8 min	5 min	1 min	20 s
Simulation duration:	1 hour				

Again, the mean online time and the average query rate of overlay nodes are negative exponentially distributed and the number of shared objects is distributed equally between the given minimum and maximum value. The simulated time is one hour. When simulating HCP, only nodes that belong to the node

classes *OFFICE* and *DSL* are allowed to become static nodes, and thus to store references to shared objects. All following simulation results represent the average values calculated from 10 independent simulation runs.

Figure 6 shows the maintenance traffic of both protocols over time, simulated with L7Sim. Since a large part of the network consists of nodes with low mean online times, we notice a continuously high amount of transferred object references in Chord. In contrast, HCP offers significantly decreased maintenance traffic, as object references are stored only on static nodes (*OFFICE* and *DSL* nodes) which are characterized by long online times.

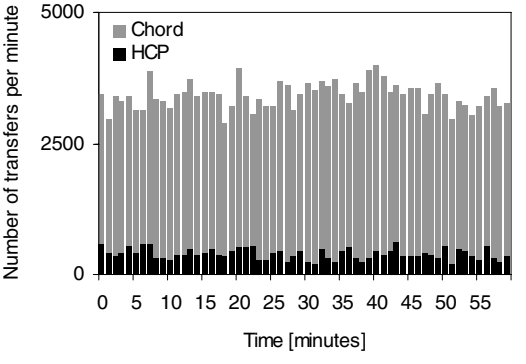

Fig. 6. Scenario 3: Maintenance traffic over time

In a second step we evaluate the content availability in both protocols, represented by the success rate of queries. We define the success rate λ of a query by dividing the number of providing hosts given in the query result by the real number of hosts currently providing the searched object. For example, when object X is shared by hosts A, B and C, and a query for X returns B and C as sharing hosts, the success rate of the query is $\lambda = 2/3 = 67\%$.

In Figure 7 the cumulative distribution of queries is plotted against the query success rate.[3] Chord can resolve 61.2% of all queries with 100% query success (i.e., the query result contains all providing hosts), but at the same time shows a sizeable fraction of non- or low-successful queries that return no or only a small number of currently providing hosts. These non- or low-successful queries result from failures of nodes that store the references to providing hosts, and from the fact that the providing hosts have not yet republished their shared objects. In contrast to Chord, HCP offers excellent query success rates. 95.5% of all queries in HCP return all currently providing hosts ($\lambda = 100\%$) and only 1.1% of all queries have a success rate less than 80%.

Thus, the above simulations prove the increased content availability in HCP that results from storing object references only on reliable static nodes. From

[3] An important parameter for this simulation is the refresh period for shared objects. It was set to 900 s, i.e. every shared object is republished by its owner every 15 minutes, in order to keep the object references in the system up-to-date.

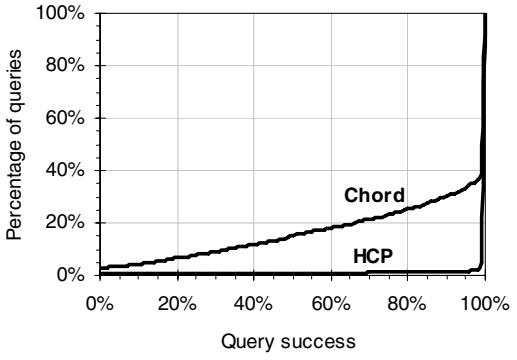

Fig. 7. Scenario 3: CDF of query success rates

our point of view, content availability is an important aspect when developing DHT-based services, as query success is a major criterion for user acceptance and hence the number of customers.

5 Conclusion

In this paper we presented a hybrid DHT design, which we applied to Chord to obtain the Hybrid Chord Protocol (HCP). We then evaluated its advantages in comparison to the conventional Chord algorithm. The introduced design aims primarily at the extension of structured DHT based P2P protocols to mobile environments, where a major part of the overlay network consists of resource-constrained mobile participants such as PDAs or mobile phones. By defining two different types of participating nodes, static and temporary nodes, the design allows disburdening of mobile participants, significantly decreased maintenance traffic and increased availability of provided content.

We performed multiple simulations of Chord and HCP in different scenarios. The simulations proved our theoretical analysis that HCP reduces the maintenance traffic by a factor of $1/\alpha$ in comparison to Chord, with α as the ratio of the mean online time of static nodes in HCP and the mean online time of all nodes in Chord. Moreover, our simulations verify the increased availability of provided content, and they show that it is sufficient for the evaluation of maintenance traffic to regard only the overlay network, without considering the underlying physical topology.

References

1. Zoels, S., Schollmeier, R., Kellerer, W., Tarlano, A.: The Hybrid Chord Protocol: A Peer-to-Peer Lookup Service for Context-Aware Mobile Applications. In: ICN 2005 (2005)
2. Aberer, K., Alima, L., Ghodsi, A., Girdzijauskas, S., Hauswirth, M., Haridi, S.: The Essence of P2P: A Reference Architecture for Overlay Networks. In: P2P 2005 (2005)

3. Rowstron, A., Druschel, P.: Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In: IFIP/ACM DSP 2001 (2001)
4. Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: SIG-COMM 2001 (2001)
5. Kaashoek, M., Karger, D.: Koorde: A Simple Degree-Optimal Distributed Hash Table. In: SODA 2004 (2004)
6. Malkhi, D., Naor, M., Ratajczak, D.: Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In: PODC 2002 (2002)
7. NS-2, The Network Simulator NS-2 Homepage, `http://www.isi.edu/nsnam/ns`
8. Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE: Universal Topology Generation from a User's Perspective, Technical Report BU-CS-TR-2001-003 (2001)

Appendix: Theoretical Evaluation of Maintenance Traffic

Assume an overlay network with N nodes and a total number of R references to shared objects. In this case, each node is responsible for storing $r = R/N$ references in average. Thus r object references have to be shifted when a node joins or leaves the overlay network. The total number of join and leave events e in a simulation scenario is determined by the number of nodes, their mean online time T and the simulation duration D:

$$e = N \cdot (2/T) \cdot D$$

Resulting, the total number of object references transferred to a joining or from a leaving node during a simulation (i.e., the maintenance traffic) is given by

$$m = e \cdot r = 2 \cdot D \cdot R/T$$

From this equation we can now evaluate the reduced maintenance traffic in an HCP system. Since HCP stores object references only on static nodes with mean online time $T_{\mathrm{HCP,static}}$, the total number of transferred references in an HCP simulation is

$$m_{\mathrm{HCP}} = 2 \cdot D \cdot R/T_{\mathrm{HCP,static}}$$

whereas Chord generates a total number of

$$m_{\mathrm{Chord}} = 2 \cdot D \cdot R/T_{\mathrm{Chord}}$$

transfers. The total number of references R and the simulation duration D remain constant in both cases. By definition, the mean online time of static HCP nodes is significantly higher than the mean online time of all nodes in a Chord system. Under the assumption that $T_{\mathrm{HCP,static}} = \alpha \cdot T_{\mathrm{Chord}}$ the generated maintenance traffic in HCP is decreased by a factor of

$$m_{\mathrm{HCP}}/m_{\mathrm{Chord}} = T_{\mathrm{Chord}}/T_{\mathrm{HCP,static}} = 1/\alpha$$

in comparison to the conventional Chord protocol.