Self-organized Construction of Distribu Access Structures: A Comparative Evaluation of P-Grid and Freenet*

K. ABERER
Swiss Federal Institute of Technology Lausanne (EPFL), Switzerla

M. HAUSWIRTH
Swiss Federal Institute of Technology Lausanne (EPFL), Switzerla

M. PUNCEVA
Swiss Federal Institute of Technology Lausanne (EPFL), Switzerla

Abstract

This paper presents a quantitative comparison of the efficiency of t self-organized construction processes of the P-Grid and Freenet peer-to-p systems. Starting from a defined, realistic network topology we simulate t construction of their access structures and measure the incurred message lo and memory usage for routing tables. Besides these results our experimen setup may also be used as a starting point for defining a standard test a evaluation suite for P2P systems.

Keywords

Peer-to-Peer Systems, P-Grid, Freenet.

1 Introduction

The P2P approach circumvents many problems of client-server systems sults in considerably more complex searching strategies, node organizatio anisms, and security issues. Napster, which made the P2P idea popula some of this complexity by employing a centralized database with refer files on peers. However, a premier goal in the design of a P2P system is to a global search functionality without using central directories. Two fund approaches exist to achieve this:

*The work presented in this paper was supported (in part) by the Swiss National Fund 064994, "Peer-to-Peer Information Systems."

- Unstructured P2P systems: The data is distributed randomly over t and broadcasting mechanisms are used for searching. Examples are [6] and [9].

- Structured P2P systems: A distributed, scalable access structure is to route search requests. Examples are Freenet [4], Chord [7], CA Pastry [12], Tapestry [11], and P-Grid [1, 3].

In systems following the first approach peers can manage their da pletely independently, i.e., the approach is fully decentralized. The peers to choose which data they store. The types of search predicates are not lim no update dependencies exist (unless replication is employed and mecha ensure consistency among the replicas exist). However, these advantages with high search costs in terms of excessive bandwidth consumption (C or additional delay [9].

The second approach is clearly superior in terms of search efficiency need to establish a distributed access structure requires some form of c tion. We can distinguish two fundamentally different ways of how this c tion can be achieved. In distributed hash tree (DHT) approaches such as Pastry and Tapestry, a global identification scheme for the peers is exploit ally a pseudo-unique ID generated by extending the IP address of the order to decide which part of the search space the peer is associated with ing this kind of global knowledge implies the following drawbacks:

- Peers are constrained in their autonomy of deciding on their rol distributed access structure. This may not be acceptable for autc peers both for reasons of resource consumption and for reasons re application aspects, such as dealing with illegal content.

- Peers may have changing IP addresses (DHCP) or may not even hav able addresses if NAT is used.

- Existing, independent networks (i.e., access structures) may not be or separated easily because each join or leaving of even a single quires careful reorganization of the access structure.

In contrast to that, Freenet, CAN and P-Grid follow a different approa decision on the role of a peer within the access structure, i.e., the part of th space a peer is associated with, is determined by bilateral interactions an peers. The interactions are initiated by some randomized process, typicall requests issued in the network. Thus the use of global knowledge for ident of peers is replaced by employing a randomized process.

In Freenet peers maintain the keys of peers that could answer earlier successfully for future routing. The most similar key is chosen in a de strategy. The routing tables are constructed as by-product of query answe

P-Grid the access structure, a binary trie, is constructed as the result of ran
lateral interactions (so-called exchanges) in which the search space is succ
partitioned. These interactions could be driven by search requests or by a
kind of mechanism creating randomized communications. In both approa
dependent networks can be joined into one access structure. In principle al
would be in this category. In CAN peers may select any point in the sear
(a d-dimensional torus) to take over responsibility for the corresponding
However, only operations for adding and leaving of individual nodes are
at the moment.

In our work we are interested in the question whether approaches r
the knowledge on global identification with randomization work as ef
as the approaches that rely on a global identification scheme, both with
to constructing the distributed access structure and with respect to usi
searches. If this is the case, it would be feasible to combine the increase
of decentralization achieved by avoiding any use of prior global knowle
the advantage of controlled complexity of search.

In fact, in settings with a sufficient degree of replication, which is
unavoidable in a practical P2P system where peers are frequently una
randomized approaches for access structure construction and search have
successful. For Freenet this has been shown by simulation studies [4]. Fo
we have shown that index construction and search are efficient. For sea
we have demonstrated that the expected cost is $\log n + 1$ messages, where
number of peers, even in cases where the search tree is unbalanced [2].

To better understand the trade-offs among the different approaches
formed a comparative simulation study of different approaches for ran
construction of structured P2P networks. In this paper we present som
results achieved from comparing P-Grid and Freenet. We were specifica
ested in a comparison with Freenet since it is most similar to P-Grid rega
qualitative characteristics: no global identification scheme is exploited i
cess structure construction, networks may freely join and split, and a cons
degree of replication both of routing information and data is used. Since
is based on a heuristics it is on the other hand by no means clear that
efficiently, since there exist no theoretical results on this aspect. Thus ou
also provide performance characteristics of Freenet which so far are not a

We developed a simulation environment, which provides exactly con
conditions for both approaches, in terms of available resources, initial
and query and communication patterns. The results achieved from our st
firmed our expectations on the performance of P-Grid and revealed that
achieves comparable results only, if it is allowed to construct routing t
considerable size, which might be a scalability problem.

2 P-Grid in a Nutshell

P-Grid [1, 3] is a peer-to-peer lookup system based on a virtual distribute
tree (a binary trie): Each peer only holds part of the overall tree, namely
from a leaf to the root together with the corresponding routing informat
construction of a P-Grid is based on a distributed, randomized algorithm a
not rely on properties of the peers that are given a-priori. Searching in P
efficient and fast even for unbalanced trees [2]. We assume peers to fail fr
and to be online with low probability. Therefore routing information and d
to be replicated. Figure 1 shows a simple P-Grid.

Every participating peer's position is determined by its path, that is, th
bit string representing the subset of the tree's overall information that th
responsible for. For example, the path of Peer 4 in Figure 1 is 10, so
all data items whose keys begin with 10. For fault-tolerance multiple p
responsible for each path, for example, Peer 1 and Peer 6. P-Grid's query
approach is simple but efficient: For each bit in its path, a peer stores a r
to at least one other peer that is responsible for the other side of the binar
that level.

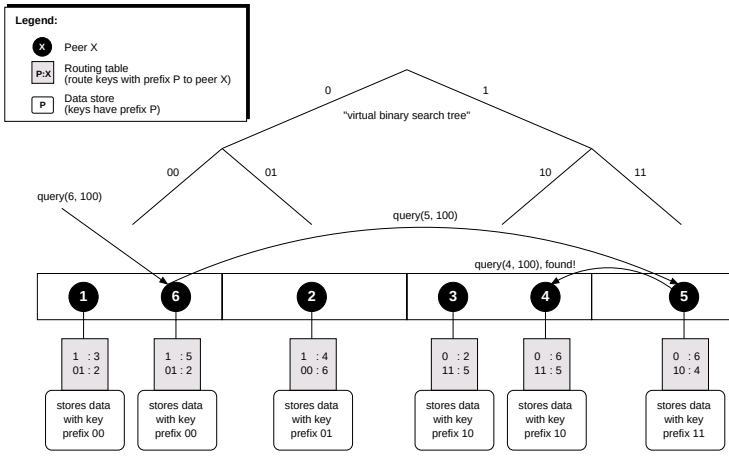


Figure 1: Example P-Grid

Thus, if a peer receives a query that it cannot satisfy, it forwards the
a peer that is "closer" to the result. In Figure 1, Peer 1 forwards queries
with 1 to Peer 3, which is in Peer 1's routing table and whose path starts
Peer 3 can either satisfy the query or forward it to another peer, dependin
next bits of the query. If Peer 1 gets a query starting with 0, and the next b
query is also 0, it is responsible for the query. If the next bit is 1, howev

1 will check its routing table and forward the query to Peer 2, whose pa
with 01.

The P-Grid construction algorithm [3] is based on purely randomiz
struction and guarantees that peer routing tables always provide at least
from any peer receiving a request to one of the peers holding a replic
path's data so that any query can be answered regardless of the peer
Additionally, it leads to an approximately uniform replication of data a
ing information, such that searches are successful with high probability
situations where peers are frequently off-line [1].

We are currently studying two versions of the algorithm. The first,
lier, version is designed to construct balanced search trees of a given
depth. It is the algorithm we used for our simulation study. Since balanc
are not suitable for situations in which the data distribution is skewed,
developed a variant of the algorithm, that adapts the tree shape to the cur
distribution. As a by-product this makes any assumptions on the depth of
unnecessary. For this case we have in particular shown that the search
terms of message exchanges remain logarithmic even if the constructed t
non-logarithmic depth.

Informally the algorithm used in the simulations is given in Figure 2 (a
optimized version of the algorithm presented in [3]). Initially, all peers
sponsible for the entire search space, that is, all search keys. When tw
responsible for the same path meet, they divide the search space and ea
takes responsibility for one half and stores the other peer's address to c
other half. If one peer has a path that is a prefix of the other peer's path,
peer with the shorter path extends its path by one bit. If peers whose pat
a common prefix meet, they can initiate new exchanges by forwarding ea
to the peers in their routing tables. However, only the peer with the shorte
taking advantage of this. This has proven to be more effective than the a
that both peers try to find new peer to perform exchanges with. Such, P-G
be constructed efficiently in a self-organizing way without central contro
lation results also show that the number of peers responsible for the sam
distributed uniformly with a low deviation from the expected average nu
peers responsible for a key [1].

3 Freenet

Freenet is a P2P system that supports the publication, replication and ret
data files while protecting the anonymity of authors and readers of the d
also difficult for a node to determine what it stores, since the files are e
when they are stored and sent over the network. Freenet uses an adaptive
scheme for efficiently routing requests to physical locations where they a
probable to appear. Freenet maintains routing tables that are dynamically

```
1     exchange(a1, a2, r) {
2       randomly swap the roles of a1 and a2; (* to prevent bias
3       determine the common prefix of path(a1) and path(a2)
4         and its length lc;
5       exchange references at all levels where the paths match;
6       (* uniformly distributes references over the network *)
7       li = length of remaining path of ai;
8       (* Case 1: both paths empty, introduce new level *)
9       CASE l1 = 0 AND l2 = 0 AND lc < maximum possible path le
10        extend path(a1) with 0 and path(a2) with 1;
11        add mutual references for future search;
12      (* Case 2: path differ in length by 1: split shorter pat
13      CASE l1 = 0 AND l2 = 1 AND lc < maximum possible path le
14        extend path(a1) by one bit different to the correspond
15          bit in path(a2);
16        update references of a1 with a2;
17      (* Case 3: analogous to case 2 with roles exchanged *)
18            ...
19      (* Case 4: use references to find other peers if no refi
20         possible *)
21      OTHERWISE IF r < maximum recursion depth
22          (* assume a1 has the longer paths, otherwise exchange
23             their roles *)
24          take a reference from the peer a1 at the level of the
25            common prefix;
26          a2 performs a new exchange with the referenced peer
27        (* which shares with a1 a longer common prefix *);
28    }
```

Figure 2: P-Grid exchange algorithm

as searches and insertions of data occur. Thus the Freenet graph evolves
cally over time as implied by the routing tables. In order to further improv
efficiency, Freenet uses dynamic replication of files along the search path

When a peer joins a Freenet network it has to know some existing no
network. By interacting with the network it will fill its initially empty ro
ble. The routing tables in Freenet store the addresses of the neighboring p
additionally the keys of data objects that these peers store and the corres
data. So when a search request arrives at a peer it may be that the peer
stores the data and can immediately answer the request. Otherwise it ha
ward the request to another peer. This is done by selecting that peer in the
table that has the most similar key to the query in terms of lexicographic
(depth first search with backtracking). Search results are routed back t
way the query took and replicated along this path to provide faster hits i
queries. Search results are stored in the peer's data store. If the data sto
the least recently used entry is evicted.

Figure 3 shows the algorithm used in the simulation.

```
1     query(peer, queryID, key, ttl) {
2       (* check whether the query is still valid (expired or
3          already seen) *)
4       IF ttl = 0 THEN
5         return failure(queryID, ttl);
6       ELSE IF queryID already seen THEN
7         return failure(queryID, ttl-1);
8       (* check whether we can satisfy the query locally otherw
9          forward it *)
10      IF key is found in local store THEN
11        return success(queryID, data);
12      ELSE {
13        WHILE peers exist in the routing table that have not b
14          contacted
15
16          find lexicographically closest peer i in routing tab
17          and forward the query;
18
19          IF query(routingtable[i], queryID, key, ttl-1)
20             returns success THEN
21            IF local store is full THEN
22               evict least recently used entry from store;
23            store (queryID, data) in local store  (* replicati
24            return success(queryID, data);
25      }
26      return failure(queryID, ttl-1);
27    }
```

Figure 3: Freenet algorithm used in the simulation

4 Experiments

4.1 Simulation Environment

For comparing quantitative properties of the two P2P systems P-Grid and we first created a general simulation environment for P2P systems using matica 4.2. The purpose of the simulation environment is to provide a po for simulating protocols of different P2P systems and test their performa der various conditions. Moreover, the environment allows us to test two systems under exactly the same conditions and to compare their perfo directly. The simulation environment consists of general initial settings common for all P2P systems and set of specific functions which depen specific details of the protocols of each system.

We assume the following common settings any P2P system:

- Each peer is identified with a unique ID number.
- We assume that there is an initial topology for communication. The and maximal numbers of neighbors per peer are fixed and neigh chosen randomly. Thus, the initial topology is a random graph w minimal and maximal degrees.

- Each peer has a routing table of size t_{max}, where peers keep info about the addresses of other peers together with information rele routing (paths in P-Grid, keys in Freenet).

- We assume that a total of d data objects are stored in the system a have a data store of size s. Data objects are identified by binary necessary condition is that $d < s * N$ (N being the number of pee system), but in general we will assume that $r * d < s * N$, such that r can be kept on average.

In addition to the common settings, the simulation environment includ tions that implement specific details of each protocol. Here we briefly sur the most important functions for any P2P system to be simulated:

- *route*: This function takes at least two parameters: the peer who the query and the key. Additional parameters can be provided deper the protocol. In the case of a structured system the *route* function routing tables to select a peer to forward the query which is the c both P-Grid and Freenet. The strategy for selecting the peer from the table depends on the protocol. Alternatively, in the case of an unst system it is possible to use a function that performs broadcast searc initial random graph topology which is similar to Gnutella-like sea

- *deliver*: This function takes at least two parameters: the peer where object is stored and the peer who requested it. Depending on the this can be implemented as sending one message to the request directly which is the case for P-Grid or sending the message back the chain of peers who participated in forwarding the message whi case for Freenet.

- *update*: In P-Grid we use this function to speed up the synchro of the peers that are responsible for the same part of the data spa does not influence the simulation results but simply is done to spee simulation itself.

For the experiments presented in this paper, we assume that initially have empty routing tables. The process of constructing the routing table strapping phase) is based on exchanging routing information during the interactions between peers.

The experiments consist of sets of randomly chosen queries sent to peers. At the beginning, the peers rely only on the initial topology for co cation to forward queries. The forwarding of queries in P-Grid and the of query results in Freenet is used to construct the routing tables and repl data objects like in the real systems. We use this to allow the systems to ev der realistic conditions for some time in order to provide more realistic r

ments. Details regarding this process in Freenet can be found in [5]. W
describe the bootstrapping algorithm used for P-Grid.

To bootstrap P-Grid each peer initiates random walks to forward the
The random walk is limited by a time-to-live value, which is chosen ra
between 1 and ttl_{max}. In each step the query message is forwarded to a ra
chosen neighbor. Each peer who receives the query, checks its data stor
key is not found and the time-to-live is not reached yet, the peer forw
query message to a random neighbor. When the time-to-live is reached the
walk process stops and an attempt for an exchange is made between the p
initiated the query and the peer that was reached last. This is how we
exchanges between randomly chosen peers, assuming that ttl_{max} is su
large. Depending on the peers' paths one or both peers may refine the
or if the paths are in a prefix relation, the peer with the shorter path ini
exchange with a peer with the longest common prefix found in the othe
routing table. This recursive process is limited by a maximal recursion de

As mentioned earlier the exchange algorithm constructs a balanced
depth $path_{max}$. If a peer extends its path to $path_{max}$, it stops to initiate
walks and thus stops to extend its path. Further it uses the P-Grid routin
anism to route the queries. In addition, the maximum number of randor
is limited if a peer does not extend its path. We also use a simplified ve
the update algorithm from [8] to synchronize the data stores of peers th
reached $path_{max}$ and share the same path. If all peers have either extend
paths to $path_{max}$ or performed the maximal number of random walks,
system is considered to be in a stable state. There may be a small fraction
which did not succeed in extending their paths. They simply forward the
to a random neighbor.

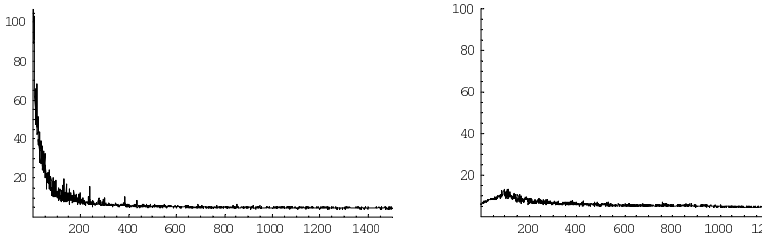
4.2 Simulation Results

To illustrate the performance of the two systems we provide some exampl
ulation result. All our results will be available at http://lsirpeople.epfl.ch/
project/accessp2p.htm.

4.2.1 Changing Peer Population

Here we present two experiments with different population sizes. For
experiment the total number of peers N is 1000. Each peer has at lea
at most 6 neighbors in the initial topology. The total number of inser
objects d is 5000. They are identified by randomly chosen binary keys o
16. Each peer initially stores $\frac{d}{N} = 5$ data objects. First we give an examp
the replication factor for both systems is 20, but maximal sizes of routin
t_{max} are 250 for Freenet and only 35 for P-Grid. For P-Grid we used $path$
and also $ttl_{max} = 7$. The experiment consisted of 150000 random queries

that is already present in the system) sent to a random peer. We assur
all peers are online all the time. As expected, we observed that for both
there are two phases: a bootstrap phase when each system is building up its
tables and a stable state when performance remains constant. In the sta
both systems achieved a very high query success rate of more than 9
average number of messages per query generated in the stable state v
messages for Freenet and 4.54 for P-Grid. This means that we got slight
results for P-Grid with much less resources spent (smaller routing table
messages). Another interesting figure are the costs required to get the sys
a stable state: P-Grid required 771625 messages and Freenet 785413, v
approximately the same effort. Figure 4 shows the average number of m
per query averaged over 100 queries.



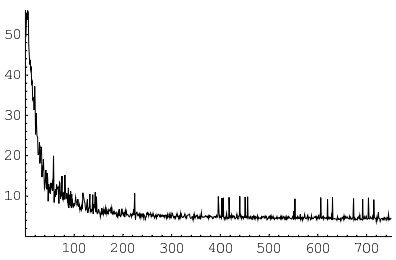
(a) Freenet (b) P-Grid

Figure 4: Average number of messages per query

During the bootstrapping phase in the simulation peers in P-Grid init
dom walks with a randomly chosen ttl between 1 and ttl_{max} to resolve
During this phase the query success rate is low and starts to increase when
tem starts to stabilize. When a peer's path reaches the length of $path_{max}$,
initiates a proactive update mechanism (in the simulation) that generat
tional messages which explains the peak in the graph for P-Grid at the be
Once all peers, except for a very small fraction, have extended their paths
tem is in a stable state where the query success rate is high (above 99%)
number of generated messages is low and stable.

In the next experiment we changed the number of peers to 500. The
and maximal number of neighbors in the initial topology are the same a
first experiment, 3 and 6, respectively. The total number of inserted data
d is 2500, thus each peer stores $\frac{d}{N} = 5$ data objects as in the previous c
replication factor for both systems is 20. The maximal size of the routin
t_{max} for Freenet is 125 and for P-Grid it is 30. For P-Grid the parameter
is set to 6. The experiment consisted of 75000 queries that were sent to a

peer. Again we assumed that all peers are online all the time. We obser
ilar behavior as in the first experiment. The query success rate was hig
99%. The average number of messages per query generated in the stable s
4.63 for Freenet and for P-Grid was 4.04. Figure 5 shows the average nu
messages per query averaged over 100 queries. These two experiments
that the performances for both systems is not affected by changing the po
size.

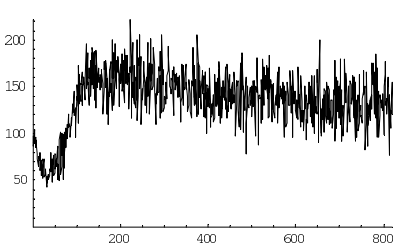


(a) Freenet (b) P-Grid

Figure 5: Average number of messages per query

4.2.2 Impact of Changing the Routing Table Sizes in Freenet

These two experiments illustrate the impact of the sizes of routing tables in
For the first experiment all parameters are the same as in the previous sim
with 1000 peers except for t_{max}, which is set to 35 and corresponds to th
mal size of routing tables used for P-Grid with 1000 peers. Thus the set
both systems are exactly the same. The simulation consisted of 82000
As can be observed the number of messages per query is much higher
the previous experiments for Freenet. For the last 100 queries the avera
ber of messages generated per query was 154 and the success rate was
the second experiment we tested Freenet with 500 peers under exactly t
conditions as P-Grid with 500 peers, which means we limited the maxima
the routing tables t_{max} to 30. Note that in this case each peer stores approx
twice as much routing information than before but still much less than
population size. Again we observed that the number of messages per qu
still very high. For the last 100 queries it was 109.75 and the success
80%. Figure 6 shows the number of messages averaged over 100 queries
experiment that consisted of 24000 queries. As can be seen from this expe
the good query performance of Freenet depends heavily on the fact that
a considerable number of addresses in its routing tables.

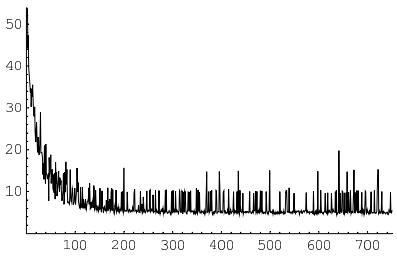


(a) Freenet: 1000 peers (b) Freenet: 500 peers

Figure 6: Average number of messages per query

4.2.3 Impact of the Replication Factor

In order to see the impact of the replication factor we performed the r experiments. The replication factor for both systems was decreased to total number of peers was 500 for both systems. The maximal size of tables t_{max} was 125 for Freenet and 35 for P-Grid. The experiments in bo consisted of 75000 queries. Figure 7 shows the number of messages pe averaged over 100 queries. We observed that the average number of mes a stable state for P-Grid was 4.62 but for Freenet increased to 5.91. Also be observed from the figure, this parameter is pretty stable for P-Grid an much more for Freenet. The query success rates were above 99%.



(a) Freenet (b) P-Grid

Figure 7: Average number of messages per query

5 Conclusions

As a result from the experiments with the two systems we can conclude Grid can achieve similar performance as Freenet in terms of messages pe

but with much less resources spent in terms of size of routing tables. If co
under exactly the same conditions which means relatively small sizes of
tables compared to the whole peer population, the performance of Freer
riorates significantly. The good performance of Freenet depends strongl
rameters such as the size of the routing table and the replication factor. Ch
these parameters affect performances of Freenet while P-Grid retains mo
behavior.

References

[1] ABERER, K. P-Grid: a Self-organizing Access Structure for P2P I
tion Systems. In *Proceedings 6th International Conference on Coo
Information Systems (CoopIS)*, (Trento, Italy, 2001), pp. 179–194.

[2] ABERER, K. Scalable Data Access in P2P Systems Using Unb
Search Trees. In *Proceedings 4th Workshop on Distributed Data an
tures (WDAS)*, (Paris, France, 2002), pp. 107–120.

[3] ABERER, K., HAUSWIRTH, M., PUNCEVA, M., AND SCHMIDT,
proving Data Access in P2P Systems. *IEEE Internet Computing 6*, 1
58–67.

[4] CLARKE, I., MILLER, S. G., HONG, T. W., SANDBERG, O., A
LEY, B. Protecting Free Expression Online with Freenet. *IEEE
Computing 6*, 1 (2002), 40–49.

[5] CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T. W. Fr
Distributed Anonymous Information Storage and Retrieval System.
*ceedings International Workshop on Design Issues in Anonymity an
servability*, (Berkeley, CA, 2001), pp. 40–66.

[6] CLIP2. The Gnutella Protocol Specification v0.4 (Document Revis
Jun 2001. http://www9.limewire.com/developer/gnutella_protocol_1

[7] DABEK, F., BRUNSKILL, E., KAASHOEK, M. F., KARGER, D., M
R., STOICA, I., AND BALAKRISHNAN, H. Building Peer-to-Peer
with Chord, a Distributed Lookup Service. In *Proceedings 8th Work
Hot Topics in Operating Systems (HotOS)*, (Oberbayern, Germany,

[8] DATTA, A., HAUSWIRTH, M., AND ABERER, K. Updates in High
liable, Replicated Peer-to-Peer Systems. In *Proceedings 23rd Inter
Conference on Distributed Computing Systems (ICDCS)*, (Rhode Isl
2003), pp. 76–87.

[9] LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. Search an
cation in Unstructured Peer-to-Peer Networks. In *Proceedings Inter
Conference on Supercomputing (ICS)*, (New York, NY, 2002), pp. 8

[10] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R
SHENKER, S. A Scalable Content-Addressable Network. In *Proc
ACM Conference on Applications, Technologies, Architectures an
cols for Computer Communications (SIGCOMM)*, (San Diego, CA
pp. 161–172.

[11] RHEA, S., WELLS, C., EATON, P., GEELS, D., ZHAO, B., WE
SPOON, H., AND KUBIATOWICZ, J. Maintenance-free Global Data
IEEE Internet Computing 5, 5 (2001), 40–49.

[12] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, Distribute
Location and Routing for Large-scale Peer-to-Peer Systems. In *Proc
ACM International Conference on Distributed Systems Platforms
ware)*, (Heidelberg, Germany, 2001), pp. 329–350.

Karl Aberer is with the School of Computer and Communcation Sciences, Swis
Institute of Technology Lausanne (EPFL), 1015 Lausanne. E-mail: karl.aberer@

Manfred Hauswirth is with the School of Computer and Communcation
Swiss Federal Institute of Technology Lausanne (EPFL), 1015 Lausanne. E-m
fred.hauswirth@epfl.ch

Magdalena Punceva is with the School of Computer and Communcation
Swiss Federal Institute of Technology Lausanne (EPFL), 1015 Lausanne. E-m
dalena.punceva@epfl.ch