Self-organized construction of distributed access structures: A comparative evaluation of P-Grid and FreeNet*

Karl Aberer, Manfred Hauswirth, Magdalena Punceva
École Polytechnique Fédérale de Lausanne (EPFL)

Abstract

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

1 Introduction

The P2P approach circumvents many problems of client-server systems but results in considerably more complex searching, node organization, and security. Napster, which made the P2P idea popular, avoids some of this complexity by employing a centralized database with references to files on peers. However, a premier goal in the design of a P2P system is to support a global search functionality without using central directories. Two fundamental approaches exist to achieve this:

- Unstructured: The data is distributed randomly over the peers and broadcasting mechanisms are used for searching. Examples are Gnutella [6] and [11].
- Structured: A distributed, scalable access structure is built up to route search requests. Examples are FreeNet [4], Chord [7], CAN [12], Pastry [14], Tapestry [13] and P-Grid [1, 3].

In systems following the first approach peers can manage their data completely independently, i.e., the approach is fully decentralized. The peers are free to choose which data they store. The types of search predicates are not limited, and no update dependencies exist (unless replication is employed and mechanisms to ensure consistency among the replicas exist). However, these advantages are paid with high search costs in terms of excessive bandwidth consumption (Gnutella) or additional delay ([11]).

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

- Peers are constrained in their autonomy of deciding on their role in the distributed access structure. This may not be acceptable for autonomous peers both for reasons of resource consumption and for reasons related to application aspects, such as dealing with illegal content.
- Peers may have changing IP addresses (DHCP) or may not even have routeable addresses if NAT is used.
- Existing, independent networks (i.e., access structures) may not be merged or separated easily because each join or leaving of even a single peer requires careful reorganization of the access structure.

In contrast to that, FreeNet, CAN and P-Grid follow a different approach. The decision on the role of a peer within the access structure, i.e., the part of the search space a peer is associated with, is determined by bilateral interactions among the peers. The interactions are initiated by some randomized process, typically search requests issued in the network. Thus the use of global knowledge for identifi-

*The work presented in this paper was supported (in part) by the Swiss National Fund grant 2100-064994, "Peer-to-Peer Information Systems."

cation of peers is replaced by employing a randomized process.

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

In P-Grid the access structure, a binary trie, is constructed as the result of random bilateral interactions (so-called exchanges) in which the search space is successively partitioned. These interactions could be driven by search requests or by any other kind of mechanism creating randomized communications. In both approaches independent networks can be joined into one access structure.

In principle also CAN would be in this category. In CAN peers may select any point in the search space (a d-dimensional torus) to take over responsibility for the corresponding region. However, only operations for adding and leaving of individual nodes are defined at the moment.

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

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

To better understand the trade-offs among the different approaches we performed a comparative simulation study of different approaches for randomized construction of structured P2P networks. In this paper we present some of our results achieved from comparing P-Grid and FreeNet. We were specificly interested in a comparison with FreeNet since it is most similar to P-Grid regarding its qualitative characteristics: no global identification scheme is exploited in the access structure construction, networks may freely join and split, and a considerable degree of replication both of routing information and data is used. Since FreeNet is based on a heuristics it is on the other hand by no means clear that it works efficiently, since there exist no theoretical results on this aspect. Thus our results also provide performance characteristics of FreeNet which so far are not available.

We developed a simulation environment which provides exactly comparable conditions for both approaches, in terms of available resources, initial settings and query and communication patterns. The results achieved from our study confirmed our expectations on the performance of P-Grid and revealed that FreeNet achieves comparable results only, if it is allowed to construct routing tables of considerable size, which might render the system unscalable.

In this paper we will first introduce the P-Grid access structure and the randomized construction algorithm that has been used in the simulation study. For the corresponding information on FreeNet we refer the reader to [4]. Then we will describe the experimental setup and some key results from the simulations. We will conclude by summarizing related developments we are currently pursuing for creating a P-Grid based P2P data management infrastructure and first applications we have studied for P-Grid.

2 P-Grid in a Nutshell

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

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

sponsible for the other side of the binary tree at that level.

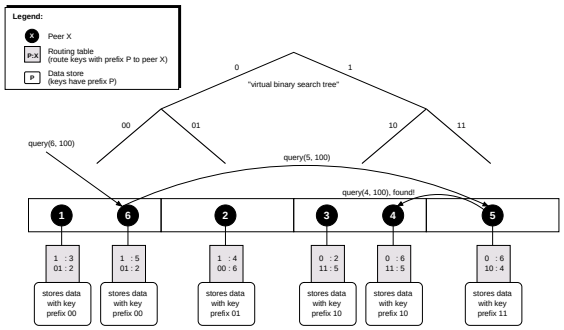


Fig. 1. Example P-Grid

Thus, if a peer receives a query string it cannot satisfy, it forwards the query to a peer that is "closer" to the result. In Figure 1, Peer 1 forwards queries starting with 1 to Peer 3, which is in Peer 1's routing table and whose path starts with 1. Peer 3 can either satisfy the query or forward it to another peer, depending on the next bits of the query. If Peer 1 gets a query starting with 0, and the next bit of the query is also 0, it is responsible for the query. If the next bit is 1, however, Peer 1 will check its routing table and forward the query to Peer 2, whose path starts with 01.

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

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

Informally the algorithm used in the simulations is given in Figure 2 (a slightly optimized version of the algorithm presented in [3]).

Initially, all peers are responsible for the entire search space, that is, all search keys. When two peers responsible for the same path meet, they divide the search space and each peer takes responsibility for one half and stores the other peer's address to cover the other half. If one peer has a path that is a prefix of the other peer's path, only the peer with the shorter path extends its path by one bit. If peers whose paths share a common prefix meet, they can initiate new exchanges by forwarding each other to the peers in their routing tables. However, only the peer with the shorter path is taking advantage of this. This has proven to be more effective than the approach that both peers try to find new peer to perform exchanges with. Such, P-Grids can be constructed efficiently in a self-organizing way without central control. Simulation results also show that the number of peers responsible for the same keys is distributed uniformly with a low deviation from the expected average number of peers responsible for a key [1].

3 Experimental results

In the setup of our experiments we assume that each system consists of N peers. The initial topology is assumed to be a random graph with fixed minimal and maximal degrees. In addition to the initial neighbors, each peer has a routing table of size t_{max}, where peers keep information about the addresses of other peers together with information relevant for routing (paths in P-Grid, keys in FreeNet).

We assume that a total of d data objects are stored in the system and peers have a data store of size s. Data objects are identified by binary keys. A necessary condition is that $d < s * N$, but in general we will assume that $rf * d < s * N$, such that rf replicas can be kept on average. Initially all peers have empty routing tables. The experiments consist of sets of randomly chosen queries sent to random peers. At the beginning, the peers rely only on the initial topology for communication to forward queries. The forwarding of queries is used to construct the routing tables and replicate the data objects. Details regarding the bootstrapping of FreeNet can be found in [5]. We briefly describe the bootstrapping algorithm used for P-Grid.

To bootstrap P-Grid each peer initiates random walks to forward the queries. The random walk is limited by a time-to-live value, which is cho-

```
exchange(a1, a2, r) {
  randomly swap the roles of a1 and a2; (* to prevent bias *)
  determine the common prefix of path(a1) and path(a2) and its length lc;
  exchange references at all levels where the paths match;
  (* uniformly distributes references over the network *)
  li = length of remaining path of ai;
  (* Case 1: both paths empty, introduce new level *)
  CASE l1 = 0 AND l2 = 0 AND lc < maximum possible path length
    extend path(a1) with 0 and path(a2) with 1;
    add mutual references for future search;
  (* Case 2: path differ in length by 1: split shorter path *)
  CASE l1 = 0 AND l2 = 1 AND lc < maximum possible path length
    extend path(a1) by one bit different to the corresp. bit in path(a2);
    update references of a1 with a2;
  (* Case 3: analogous to case 2 with roles exchanged *)
      ...
  (* Case 4: use references to find other peers if no refinement possible *)
  OTHERWISE IF r < maximum recursion depth
     (* assume a1 has the longer paths, otherwise exchange their roles *)
     take a reference from the peer a1 at the level of the common prefix;
     a2 performs a new exchange with the referenced peer
    (* which shares with a1 a longer common prefix *);
}
```

Fig. 2. P-Grid exchange algorithm

sen randomly between 1 and ttl_{max}. In each step the query message is forwarded to a randomly chosen neighbor. Each peer who receives the query, checks its data store. If the key is not found and the time-to-live is not reached yet, the peer forwards the query message to a random neighbor. When the time-to-live is reached the random walk process stops and an attempt for an exchange is made between the peer who initiated the query and the peer that was reached last. This is how we initiate exchanges between randomly chosen peers, assuming that ttl_{max} is sufficiently large. Depending on the peers' paths one or both peers may refine their paths or if the paths are in a prefix relation, the peer with the shorter path initiates an exchange with a peer with the longest common prefix found in the other peer's routing table. This recursive process is limited by a maximal recursion depth of 2.

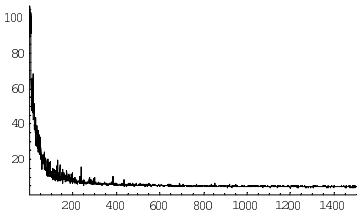
As mentioned earlier we use the version of the exchange algorithm, which constructs a balanced tree of depth $path_{max}$. If a peer extends its path to $path_{max}$, it stops to initiate random walks and thus stops to extend its path. Further it uses the P-Grid routing mechanism to route the queries.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

3.1 Simulation Results

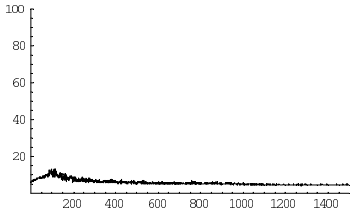
To illustrate the performance of the two systems we provide some exemplary simulation result. All our results will be available at http://lsirpeople.epfl.ch/punceva/project/accessp2p.htm.

The total number of peers N is 1000. Each peer has at least 3 and at most 6 neighbors in the initial topology. The total number of inserted data objects d is 5000. They are identified by randomly chosen binary keys of length 16. Each peer initially stores $\frac{d}{N} = 5$ data objects. First we give an example where the replication factor for both systems is 20, but maximal sizes of routing tables t_{max} are 250 for FreeNet and only 35 for P-Grid. For P-Grid we used $path_{max} = 7$ and also $ttl_{max} = 7$. The experiment consisted of 150000 random queries (for data that is already present in the system) sent to a random peer. We assumed that all peers are online all the time. As expected, we observed that for both systems there are two phases: a bootstrap phase when each system is building up its routing tables and a stable state when performance remains constant. In the stable state both systems achieved a very high query success rate of more than 99%. The average number of messages per query generated in the stable state was 4.58 messages for FreeNet and 4.54 for P-Grid. This means that we got slightly better results for P-Grid with much less resources spent (smaller routing table and less messages). Another interesting figure are the costs required to get the system into a

stable state: P-Grid required 771625 messages and FreeNet 785413 which is approximately the same effort. The following two graphs show the average number of messages per query averaged over 100 queries.



(a) Freenet



(b) P-Grid

Fig. 3. Average number of messages per query

Another interesting simulation result for FreeNet is given below. All parameters are the same as in the previous simulation except for t_{max}, which is set to 35 and corresponds to the maximal size of routing tables used for P-Grid. The simulation consisted of 82000 queries. As can be observed the number of messages per query is much higher than in the previous experiments. For the last 100 queries the average number of messages generated per query was 154 and the success rate was 70%. Thus the good query performance of FreeNet depends heavily on the fact that it stores a considerable number of addresses in its routing tables.

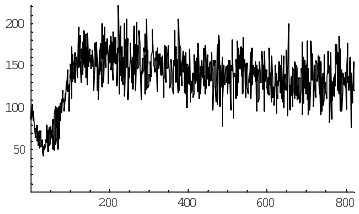


Fig. 4. FreeNet: Average number of messages per query

4 P-Grid Development

In our P-Grid project we pursue the goal of gradually evolving it into a general-purpose distributed infrastructure. We have implemented P-Grid in Java and are currently in the final test phase. The software will be released as open source via the P-Grid webserver (http://www.p-grid.org/) which also provides detailed information on all aspects of P-Grid. Two important technical developments for turning P-Grid into a generally applicable infrastructure are:

To address the issue of updates in a decentralized way we have designed an update algorithm [8] based on rumor spreading which provides probabilistic guarantees for consistency. It was inspired by the fundamental work on randomized rumor spreading presented in [10]. The update algorithm is efficient (analytically proven) and based on a generic push/pull gossiping scheme for highly unreliable, replicated environments, dealing with the realistic situation that peers are mostly off-line.

To handle the problem of changing IP addresses of peers we have designed a completely decentralized, self-maintaining, light-weight, and sufficiently secure peer identification service [9] that allows us to consistently map unique peer identifications onto dynamic IP addresses in environments with low online probability of the peers constituting the service. The basic idea is to store the mappings in P-Grid itself: Peers store their current id/IP mapping in P-Grid and update it if the IP address changes (for example, if they come online again). Although at first sight this may look as an unsolvable, recursive "hen-egg problem," we demonstrate in [9] that not only most of the original queries will be answered successfully, but also, that the recursions triggered by failures will lead to a partial "self-healing" of the whole system.

References

[1] Karl Aberer. P-Grid: A self-organizing access structure for P2P information systems. In *Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS 2001)*, Trento, Italy, 2001.

[2] Karl Aberer. Scalable Data Access in P2P Systems Using Unbalanced Search Trees. In *Proceedings of Workshop on Distributed Data and Structures (WDAS-2002), Paris, France*, 2002.

[3] Karl Aberer, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6(1), Jan./Feb. 2002.

[4] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, 6(1), Jan./Feb. 2002.

[5] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies: International Workshop*

on Design Issues in Anonymity and Unobservability, number 2009 in LNCS, 2001. http://freenetproject.org/cgi-bin/twiki/view/Main/ICSI.

[6] Clip2. The Gnutella Protocol Specification v0.4 (Document Revision 1.2), Jun. 2001. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[7] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with chord, a distributed lookup service. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, 2001.

[8] Anwitaman Datta, Manfred Hauswirth, and Karl Aberer. Updates in Highly Unreliable, Replicated Peer-to-Peer Systems. Technical Report IC/2002/47, École Polytechnique Fédérale de Lausanne (EPFL), 2002. http://www.p-grid.org/Papers/TR-IC-2002-47.pdf.

[9] Manfred Hauswirth, Anwitaman Datta, and Karl Aberer. Handling Identity in Peer-to-Peer Systems. Technical Report IC/2002/67, École Polytechnique Fédérale de Lausanne (EPFL), 2002. http://www.p-grid.org/Papers/TR-IC-2002-67.pdf.

[10] Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized rumor spreading. In *IEEE Symposium on Foundations of Computer Science*, 2000.

[11] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *International Conference on Supercomputing*, 2002.

[12] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM*, 2001.

[13] Sean Rhea, Chris Wells, Patrick Eaton, Dennis Geels, Ben Zhao, Hakim Weatherspoon, and John Kubiatowicz. Maintenance-free global data storage. *IEEE Internet Computing*, 5(5), 2001.

[14] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.