Advanced Peer-to-Peer Networking: The P-Grid System and its Applications*

Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, Roman Schmidt, Jie Wu

Distributed Information Systems Laboratory
École Polytechnique Fédérale de Lausanne (EPFL)

The limitations of client/server systems become evident in an Internet-scale distributed environment. Peer-to-peer (P2P) systems offer an interesting alternative to traditional client/server systems: Every node acts both as a client and a server and "pays" its participation by providing access to its computing resources. Systems such as Napster and Gnutella have proven their practical applicability. In this article we give an overview of our P-Grid system which provides an advanced P2P infrastructure targeted at application domains beyond mere file-sharing. We present the conceptual foundations and outline some of the applications we are developing at the moment.

1 Introduction

In Internet-scale applications resources typically are concentrated on a small number of nodes, which must apply sophisticated load-balancing and fault-tolerance algorithms to provide continuous and reliable access. Network bandwidth to and from these nodes must be increased steadily if the system is successful and thus attracts lots of traffic. Caching and replication were introduced a posteriori to remedy these problems of the client-server setting when the World Wide Web, as the most successful Internet service, developed into a network bandwidth nightmare.

Peer-to-peer systems offer an alternative to such traditional client-server systems for a number of application domains. In P2P systems, every node (peer) of the system acts as both client and server (servent) and provides part of the overall resources/information available from the system. In a pure P2P system no central coordination or central database exists and no peer has a global view of the system. The participating peers are autonomous and self-organize the system's structure, i.e., global behavior emerges from local interactions. Peers will not always be available but still all existing data and services should be accessible which should be addressed by sufficiently high replication.

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 to achieve this exist:

- Unstructured: The data is distributed randomly over the peers and (un)constrained broadcasting mechanism are being used. Examples are Gnutella [6] and [12].
- Structured: A distributed, scalable indexing structure is built up to route search requests. Examples are FreeNet [5], Chord [8], and P-Grid [1, 4].

In systems following the first approach peers can manage their data completely independently, i.e., the approach is fully decentralized, the types of search predicates are not limited, and no update dependencies exist (unless replication occurs and is managed). However, these advantages are paid with high search costs in terms of messages (Gnutella) or additional delay ([12]).

The second approach is clearly superior in terms of search efficiency, but the need to establish a distributed index usually requires some form of central coordination or global knowledge. Chord is a representative of such systems. Additionally, many systems following the structured approach do not allow sub-networks to freely merge and split, but changes to the network topology are constrained to the joining and leaving of single nodes.

* The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 and the Swiss National Fund grant 2100-064994, "Peer-to-Peer Information Systems."

Notable exceptions within the second class of approaches are Freenet and P-Grid. They exhibit the advantages of the structured approach but try to maintain the beneficial characteristics of unstructured networks, such as complete decentralization and support for sub-networks. Essentially they achieve this by replacing any form of global coordination by randomization. P-Grid shares some common properties with Chord [8], CAN [13], and Tapestry [14] but requires much less tight coupling since it does not exploit global knowledge as these systems do and thus is more related to Freenet.
Thus we could also distinguish between partially decentralized P2P systems such as Chord, but also Napster, and fully decentralized P2P systems such as Gnutella, Freenet, and P-Grid. Since we believe that only completely decentralized systems will be able to fully exploit the advantages in terms of scalability and self-organization we focus our interest on the second class.
In the following we give an overview of our P-Grid system [1, 4] which provides an advanced P2P infrastructure targeted at application domains beyond mere file-sharing. We present the conceptual foundations and outline some of the applications we are developing at the moment on the basis of P-Grid.

2 P-Grid in a Nutshell

P-Grid [1, 4] is a peer-to-peer lookup system based on a virtual distributed search tree: Each peer only holds part of the overall tree, which comes into existence only through the cooperation of the individual peers. Searching in P-Grid is efficient and fast even for unbalanced trees [2] (*O(log(n))*, where *n* is the number of leaves). Unlike many other peer-to-peer systems P-Grid is a truly decentralized system which does not require central coordination or knowledge as, for example, Chord [8] and OceanStore [14] do. It is based purely on randomized algorithms and local interactions. Also we assume peers to fail frequently and be online with a very low probability. Figure 1 shows a simple P-Grid.

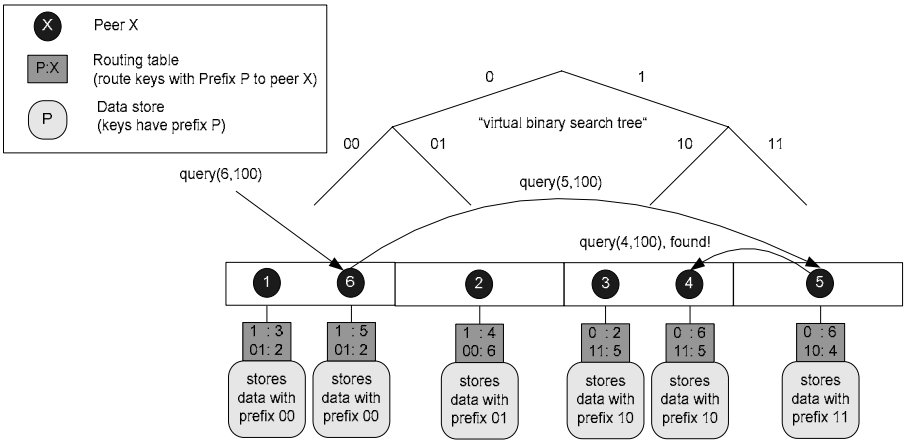

Figure 1: Example P-Grid

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 can be responsible for the same 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 responsible for the other side of the binary tree at that level. Thus, if a peer receives a binary query string it cannot satisfy, it must forward 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 [4] 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 so that any query can be satisfied regardless of the peer queried. Additionally, it guarantees that a sufficiently high number of replicas exists for any path and that the peers representing a certain path also

know some of their replicas. Similarly, the routing tables will hold also multiple references for each level which the routing algorithm selects randomly [4].

3 Updates in P-Grid

Until recently P2P systems were primarily used for sharing static, read-only files. Thus most systems did not provide update mechanisms that would work in the presence of replication which is used frequently in P2P systems to improve scalability and availability. For example, centralized (or hierarchical) P2P systems, such as Napster or FastTrack, maintain a centralized index of data items available at online peers. If an update of a data item occurs this means that the peer that holds the item changes it. Subsequent requests would get the new version. However, updates are not propagated to other peers which replicate the item. As a result multiple versions under the same identifier (filename) may co-exist and it depends on the peer that a user contacts whether the latest version is accessed. The same holds true for most decentralized systems such as Gnutella.

In other systems, updates are partially dealt with. In Freenet an update is routed "downstream" based on a key-closeness relation. Since the network may change during this and no precautions are taken to notify peers that come online after an update has occurred, consistency guarantees are limited.

To address the issue of updates in a decentralized way we have designed an update algorithm [9] based on rumor spreading that addresses these problems and provides probabilistic guarantees for consistency. It was inspired by the fundamental work on randomized rumor spreading presented in [11]. 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 offline. [9] provides an analytical model to demonstrate the significant reduction of message overhead using certain optimizations techniques (partial lists) and proper tuning of the gossiping (push) phase which in consequence improves the scalability of the algorithm. The efficiency of the pull phase depends solely on the efficiency of searches in the P2P system. The analytical model for the gossiping algorithm is a significant contribution in contrast to most of the literature in this area which relies solely on simulation results. Since our algorithm is generic the analytical model is valid for many of the other variants of flooding algorithms and so are the results of our analysis. Another major advantage of the algorithm is that it is totally decentralized and uses no global knowledge but exploits local knowledge instead. This makes it suitable for state-of-the-art systems in the P2P, mobility, and ad-hoc networking domains.

Some of the applications discussed in the following, such as dynamic address management (Section 4.1) and trust (Section 4.2) depend heavily on the provision of an update functionality.

4 Applications

This section presents some of the applications that use P-Grid as its underlying infrastructure: handling of dynamic IP addresses and peer identity, trust assessment and maintenance, and efficient, adaptive media dissemination.

4.1 Handling dynamic Addresses and Identity of Peers

As IP addresses have become a scarce resource most computers on the Internet no longer have permanent addresses. For client computers this is usually not a big problem but with the advent of P2P systems, where every computer acts both as a client and as a server, this has become increasingly problematic. In advanced P2P systems ad-hoc connections to peers have to be established, which can only be done if the receiving peer has a permanent IP address. To handle this we have designed a completely decentralized, self-maintaining, light-weight, and sufficiently secure peer identification service [10] 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 [10] 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 which improves the success probability. The analysis deduces the probability of success given the percentage of peers that are offline, and the involved effort (messages) to achieve success for queries. For security we apply a combination of PGP-like public key distribution and

a quorum-based query scheme. The public keys themselves are stored in P-Grid, and replication can provide guarantees that are probabilistically analogous to PGP's web of trust. This, however, requires some further work to fully justify our claims. The approach also can easily be adapted to other application domains, i.e., be used for other name services, because we do not impose any constraints on the type of mappings.

4.2 Trust as the Basis for E-commerce

The vast majority of interactions among peers in a P2P system are between complete strangers who do not have any prior knowledge about each other. Since peers are fully autonomous this leaves much room for exercising opportunistic behavior of various forms, ranging from "free riding" in file sharing P2P networks to fraud and deception in e-commerce related interactions. Researchers have recognized the importance of this problem and trust and reputation management has been accepted as an appropriate solution. An ideal trust and reputation management solution would provide a mechanism based on reporting, sharing and using reputation information (or, put another way, information about past interactions) in a society of agents in which cooperation among the agents is the dominant strategy. To the best of our knowledge, none of the numerous works on trust and reputation management in online communities, which have appeared in recent years, fulfills this strong requirement. In [3] we present our decentralized trust management model that analyzes past interactions among agents to make a probabilistic assessment of whether any given agent cheated in its past interactions. The emphasis is put not only on assessing trust but also on providing a scalable data management solution particularly suitable for decentralized networks. The problem in decentralized networks that many authors fail to see or try to solve in a wrong and inefficient way, is that reputation data is aggregated along the wrong dimension in the sense that each agent has information about its own past interactions with others but cannot easily obtain opinions of others about any other particular agent in the network. To achieve this necessary re-aggregation of reputation data we use P-Grid to store trust-related information. For any particular agent we designate a set of replicas to store the ratings of trust-related behavior of that agent (complaints filed by it about others and complaints filed by others about it) so that the reputation data can be accessed and collected in logarithmic time. As replicas may provide false data, an appropriate replication factor along with a proper voting scheme to identify the most likely correct reputation data set are applied in order to achieve accurate predictions. Trust assessments themselves are made based on an analysis of agent interactions modeled as Poisson processes. As it was shown by simulations, cheating behavior of the agents can be identified with a very high probability. The model is simplistic in the sense that, for any agent, it decides whether the agent cheated in the past or not, but it can be easily extended to give predictions of the agents' behavior in the future.

4.3 Adaptive Media Dissemination

Content delivery networks are typically composed of hundreds of machines interconnected in order to minimize the service time of users requesting large data files (usually media assets). The time and cost inherently associated with the installation and management of such networks are enormous, and may be reduced significantly by using peer-to-peer and self-organizing systems. The main problem is the efficient dissemination of (usually large amounts of) data while maximizing the throughput of the system which requires sophisticated caching and replication: Very popular assets have to be replicated conveniently to avoid bottlenecks.

In [7] we propose and evaluate a completely automatic and decentralized solution to disseminate content in a P2P network. Traditional systems would typically rank the different assets based on the number of requests they have generated, and would replicate the most popular files accordingly. This is obviously not suitable for decentralized systems, where information as well as decision processes are distributed among the peers. The key idea here is to take advantage of the structural properties of the P2P system to replicate data at strategic locations (peers) contacted while processing the query. In this way, we minimize the service time of the clients (by disseminating the content based on its popularity) while letting the access structure of the P2P system unaffected: the replication process is thus transparent and does not imply any consequence on the existing indexing scheme.

For P-Grid, we make use of the virtual tree structure for replicating the popular assets as shown in Figure 2.

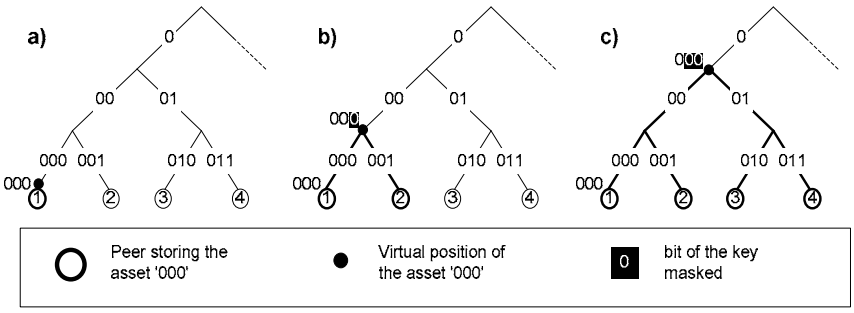

Figure 2: The replication mechanism

In a greedy fashion, a popular file will first be replicated at the peer next to its original location, before being copied to the other two, four and eight direct neighbors, etc., until it competes with other assets having approximately the same popularity. This approach obtains near optimal performance in terms of the mean time consumed for serving a client, while conserving the intrinsic decentralization properties of a P2P system.

5 Experimental results

To test P-Grid's performance and compare it with other P2P systems we performed several simulation experiments. Particularly, we wanted to compare P-Grid with FreeNet [5] which uses similar concepts. Both simulated systems consisted of 1000 peers. The initial communication topology was a random graph with a minimum degree of 3 and a maximum degree of 6. Each peer initially had 5 randomly chosen documents out of a set of 2^{16} uniformly distributed documents. The storage replication factor for documents was set to 10 for FreeNet and 20 for P-Grid. The maximum number of references in the routing tables was 250 for FreeNet and only 35 for P-Grid. The experiment consisted of 150000 random queries sent to randomly chosen peers. We were especially interested in comparing the average number of messages generated per query and the success rate of the queries. In our experiments we assumed that all peers are online all the time. 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 phase when performance remains constant. In the stable state both systems achieved a very high query success rate of 99%. We monitored the average number of messages per query generated in the stable state to be 4.97 messages for FreeNet and 4.55 for P-Grid. This means that we got better results for P-Grid with less resources spent. In particular, FreeNet achieves stable behavior only if the routing tables cover a substantial fraction of the complete network (in our experiment 25% of the total peer population were required). Another interesting metric is the cost for getting the system into a stable state. With respect to this, P-Grid required 781474 and FreeNet required 799987 messages in total which is approximately the same effort.

6 Conclusions

In many domains the P2P approach has proven to be a valid alternative to the currently dominating client-server approach. As has been shown by successful systems such as Gnutella, P2P systems can solve scalability problems and facilitate new ways of interaction among users which can be exploited for new application domains such as e-commerce. Having proven their basic applicability we believe that the time has come to push the P2P paradigm to higher levels of abstraction such as structured search, higher-level services, and introduction of semantics. In our P-Grid project we try to pursue this goal and gradually improve 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.

7 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 and Zoran Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the 10th International Conference on Information and Knowledge Management (2001 ACM CIKM)*, pages 310–317. ACM Press, 2001.

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

[5] 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.

[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] Philippe Cudré-Mauroux and Karl Aberer. A Decentralized Architecture for Adaptive Media Dissemination. In *Proccedings of the IEEE International Conference on Multimedia and Expo (ICME 2002), Lausanne, Switzerland*, 2002.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker. *A Scalable Content-Addressable Network*. Proceedings of the ACM SIGCOMM, 2001.

[14] 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.