An Overview on Peer-to-Peer Information Systems

Karl Aberer, Manfred Hauswirth
Swiss Federal Institute of Technology (EPFL), Switzerland

Abstract

The limitations of client/server systems become evident in an Internet-scale distributed environment. P2P systems offer an 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 briefly introduce the key concepts and properties of the P2P paradigm and overview commercial systems and research approaches.

Keywords: Peer-to-Peer Systems, Information Systems

1 Introduction

In large Internet-scale distributed systems such as the World-wide Web the shortcomings of the standard client-server model become evident: Individual resources are concentrated on one or a small number of nodes and in order to provide 24/7 access with acceptable response times sophisticated load balancing and fault-tolerance algorithms have to be applied. The same holds true for network bandwidth which adds to this bottleneck scenario.

These two main problems motivated researchers to come up with approaches for distributing processing load and network bandwidth among all nodes participating in a distributed information system. This solves the bottleneck problems but has to be "paid" with considerably higher algorithmic complexity and additional security problems.

This article provides am overview of P2P systems. In Section 2 we provide a definition of what P2P is, i.e., the properties that distinguish the P2P paradigm from other paradigms. Then we present the basic concepts and algorithms of well-known P2P systems (Napster in Section 3, Gnutella in Section 4, Freenet in Section 5), briefly sketch commercial systems in Section 6 (FastTrack and JXTA), and overview research systems (OceanStore, Chord, Pastry, Can, P-Grid, Random Walker, and Farsite).

2 What is P2P?

In a P2P every participating node acts both as a client and as a server ("servent") and "pays" its participation by providing access to some of its resources, most frequently, processing power and/or disk space. Although this idea is common to all P2P systems they differ considerably in their underlying architecture. A coarse but intuitive definition of P2P is given by Clay Shirkey (The Accelerator Group) [12]:

> *Peer-to-peer is a class of applications that take advantage of resources storage, cycles, content, human presence available at the edges*

of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, peer-to- peer nodes must operate outside the DNS and have significant or total autonomy of central servers.

Thus a P2P system can be viewed as an application-level Internet on top of the Internet. To quickly decide whether a given system is P2P in Shirkey's sense the following "litmus test" can be applied:

- Does the system give the nodes at the edges of the network significant autonomy?
- Does the system allow for variable connectivity and temporary network addresses?

More conceptually we can identify several principles underlying P2P systems:

- The principle of sharing resources: all P2P systems involve an aspect of resource sharing, where resources can be physical resources, such as disk space or network bandwidth, as well as, logical resources, such as services or different forms of knowledge. By sharing of resources applications can be realized which could not be set up by a single node. This was the driving motivation behind a P2P system such as Napster.
- The principle of decentralization: this is an immediate consequence of sharing of resources. Parts of the system or even the whole system are no longer operated centrally. Decentralization is in particular interesting in order to avoid single-point-of failures or performance bottlenecks in the system. Examples of fully decentralized systems are Gnutella and Freenet.
- The principle of self-organization: when a P2P system becomes fully decentralized then there exists no longer a node that can centrally coordinate it's activities or a database to store global information about the system centrally. Therefore nodes have to self-organize themselves, based on whatever local information is available and interacting with locally reachable nodes (neighbors). The global behaviour then emerges as the result of all the local behaviours that occur.

In addition P2P system often take into account other qualitative requirements such as the interest of participants of remaining anonymous or the fact that nodes in a P2P system are usually unreliable.

2.1 Types of P2P Systems

The P2P approach existed even before file sharing systems like Napster, Gnutella, or Freenet made the idea of P2P popular. In fact many well-established systems are based on similar foundations. In the networking domain the Internet (Arpanet) itself was one of the first P2P systems: Each Internet host could telnet/ftp any other Internet host. More recently mobile ad-hoc networks again take the P2P approach. In distributed databases systems such as the Mariposa system [20] exhibit P2P properties as well as many e-commerce systems, for example, eBay, B2B market places or B2B integration servers. As can be seen from these various examples the P2P approach in fact may be applied at several system levels as shown in Figure 1.

As mentioned above the network layer, i.e., the Internet, essentially is P2P. At the information management layer directories and databases can be either centralized or P2P. E-Commerce systems, for example, which would be at the application

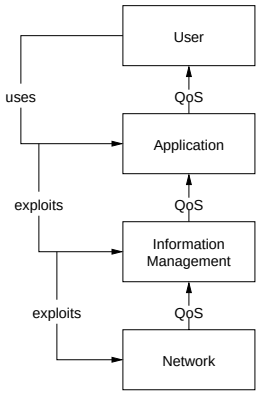

Figure 1: System layers that may exploit the P2P approach

layer can be centralized or P2P alike. The user layer, i.e., commerce, society, etc., is P2P as everyone experiences. So a careful analysis is required for each system that is supposed to be P2P, at which system level the P2P paradigm is applied. Table 1 shows this analysis for four well-established systems, including Napster, Gnutella and Freenet that we will discuss in detail.

	P2P user interaction	P2P application	P2P info mgmt.
eBay	yes	no	no
Napster	yes	yes	no
Gnutella	yes	yes	yes
Freenet	yes	yes	yes

Table 1: How much P2P is involved?

2.2 P2P System Inflation

Since the emergence of the P2P paradigm a growing number of P2P systems and systems exploiting the P2P approach has come into existence: Akamai, FastTrack, DFSI, Gnutella, E-Donkey, Intermemory, iMesh, Alpine, Aimster, Napster, Freenet, Gnutmeg, OFSI, Farsite India, OceanStore, Chord, CAN, JXTA, Gridella, Tornado, etc. Though only few of them are actually in wide-spread use, this shows that P2P is a very important issue both among researchers as well as users.

In the following sections we will first go into the details of three widely used systems, that may be considered as the main proponents of the P2P "phenomenon". (Napster in Section 3, Gnutella in Section 4, Freenet in Section 5) We will see that they base on quite different technical foundations. Then we will discuss some commercial approaches in the field (Section 6) and give overview on a number of research activities in the area (Section 7).

3 Napster

Napster [11] is the best-known P2P system by far. It became famous as a music exchange system that additionally offered many add-on services users liked. Napster was sentenced to go out of business due to copyright infringement and though taken over by Bertelsmann, still has not gained its original popularity.

Technically speaking Napster is a pretty simple P2P system: The Napster server holds a central database of all MP3/WMA music files that are offered by the currently participating peers. Clients (Napster terminology!) login to this server and send the list of files they offer (C/S style). New users must first establish an account with the Napster server for this purpose. Each client can send search requests to the Napster server and receive a list of clients who offer a file that matches the query (C/S style). The requester can choose from this list and request a download of the actual file (P2P style). Thus Napster is no pure P2P system but a combination of C/S and P2P. The interaction pattern of Napster is shown in Figure 2.

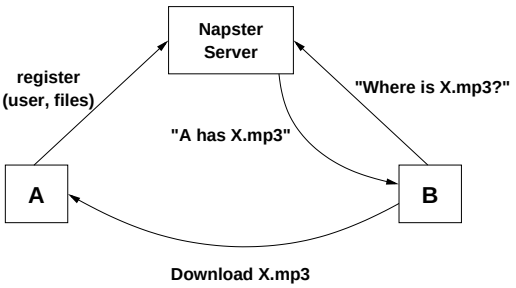

Figure 2: Napster interaction style

The centralized database of Napster avoids many of the problems of other P2P systems regarding query routing and indices. Also the querying is rather fast because the Napster server is run on a server farm. Due to the simplistic approach an upper bound for the duration of queries can be given and modelling of Napster is easy since its topology and features are known all the time. However, it limits the scalability of the system and is a single point of failure which in case it fails renders the system completely dysfunctional.

The protocol that Napster uses in these interactions was never published. There exist protocol definitions such as [17] that have been re-engineered from protocol traces. The protocol is complicated and inconsistent and exhibits may ad-hoc features. Unlike other systems Napster uses a simple structured schema for queries but does not address the reputation of peers.

4 Gnutella

Gnutella is a typical Internet success story. It was originally developed in a 14 days "quick hack" by Nullsoft (winamp) and intended to support the exchange of cooking recipes. The developer published it on the Nullsoft web server under GPL (GNU General Public License). After a couple of hours AOL, the owner of Nullsoft, realizing its potential demanded it to be taken off the web server again. However, this short online time was enough for many downloads. The protocol was reverse-engineered and soon after that third-party clients were available and Gnutella had entered the staged.

Technically speaking the Gnutella system is a decentralized file-sharing system whose participants form a virtual network communicating in a P2P fashion via the Gnutella protocol [6], which is a simple protocol for distributed file search. To participate in Gnutella a peer first must connect to a known Gnutella host. Specialized servers return lists of hosts to get started; this is outside the Gnutella protocol specification. The protocol itself consists of 5 basic message types shown in Table 2.

Type	Description	Contained Information
Ping	Announce availability and probe for other servents	None
Pong	Response to a Ping	IP address and port number of the responding servent; number and total kB of files shared
Query	Search request	Minimum network bandwidth of responding servent; search criteria
QueryHit	Returned by servents that have the requested file	IP address, port number, and network bandwidth of responding servent; number of results and result set
Push	File download requests for servents behind firewalls	Servent identifier; index of requested file; IP address and port to send file to

Table 2: Message types in the Gnutella protocol

These messages are routed by all servents using a constrained broadcast mechanism: Upon receipt of a message the servent decrements the message's time-to-live (TTL) field. If the TTL is greater than 0 and it has not seen the message's identifier before (loop detection) it resends the message to all the peers it knows. Additionally the servent checks whether it should respond to the message: If it receives a *Query* message it checks its local file store and if it can satisfy the request, responds with a *QueryHit* message. Responses are routed along the same path as the originating message.

A simplified Gnutella "session" works as follows: Servent A connects to servent B and sends a *Ping* message. B responds with a *Pong* and forwards the *Ping* as described above (e.g., it forwards it to its peers C and D who respond with another *Pong*, and so on). After some time A knows a number of servents and vice versa. Figure 3 shows how this announcement/routing mechanism works.

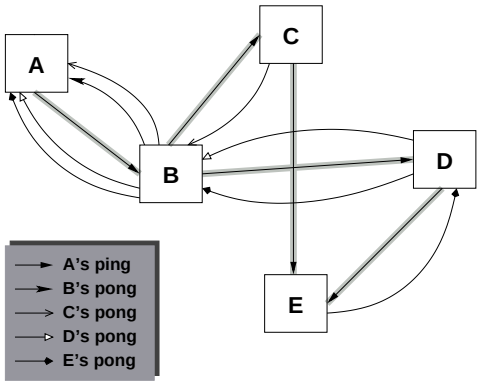

Figure 3: Gnutella: meeting peers (Ping/Pong)

A routes messages as described above and may now initiate queries. When it receives a *QueryHit* for one of its queries it tries to connect directly to the servent specified in the *QueryHit* and runs a simplified HTTP GET interaction to retrieve the file. In case the requested servent is behind a firewall it may send a *Push* message (along the same way as it received the *QueryHit*) to the firewalled servent.

The *Push* message specifies, where the firewalled servent can contact the requesting servent to run a "passive" GET session. If both servents are behind firewalls then the download is impossible. Figure 4 shows a search interaction.

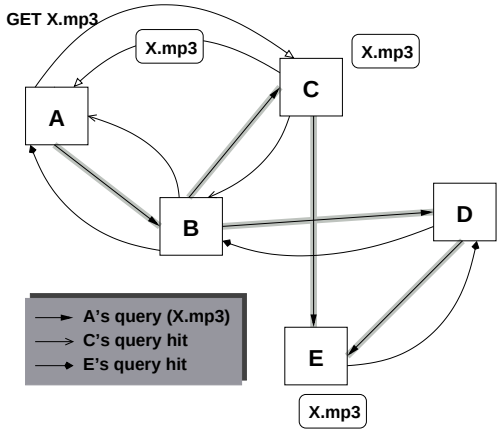

Figure 4: Searching in Gnutella (Query/QueryHit/GET)

4.1 Discussion of Gnutella

From a user's perspective Gnutella is a simple yet effective protocol: Hit rates for search queries are reasonably high, it is fault-tolerant towards failures of servents, and adopts well to dynamically changing "peer populations." However, from a networking perspective, this comes at the price of very high bandwidth consumption: Search requests are "broadcast" over the network and each node receiving a search request scans its local database for possible hits.

For example, assuming a typical TTL of 7 and an average of 4 connections C per peer (i.e., each peer forwards messages to 3 other peers) the total number of messages originating from one Gnutella message (including the responses) can be calculated as:

$$2 * \sum_{i=0}^{TTL} C * (C-1)^i = 26240 \tag{1}$$

4.2 Distribution of Queries and Data

Recent experiments [18] showed that in a real-world setting the Gnutella network traffic accumulates up to 3.5Mbps (or 353,396 queries in 2.5 hours as another experiment's result in [18]). To remedy this [18] suggests to apply caching which reduces the traffic considerably. Their experiments show that very popular documents are approximately equally popular and that accesses to less popular documents follow a Zipf-like distribution, i.e., the probability of seeing a query for the i^{th} most popular query is proportional to $\frac{1}{i^\alpha}$. This is similar to the access probabilities of web documents, which also follow a Zipf-like distribution. So the author postulates that caching may also be applicable to Gnutella and verifies this assumption using an enhanced Gnutella-client which uses caching.

4.3 Network Topology

Also, no estimates on the durations of queries and no probability for successful search requests can be given. Very little is known about the topology of the Gnutella network which would provide the foundation for an accurate mathematical model and would aid the development of efficient algorithms that exploit structural properties. As a first step a recent study [8] investigated Gnutella's topology. It shows that Gnutella exhibits strong small-world properties [9] (we will discuss small world properties in the next section in more detail) and a power law distribution of node degrees and that the topology of Gnutella is converging to a "backbone + outskirts"-like topology.

4.4 Free Riding

Besides these technical problems non-technical ones such as "free riding" challenge Gnutella [2]. "Free riding" means that most Gnutella users do not provide files to share and if, only a very limited number of files is of interest at all. [2] shows that 66% of Gnutella users share no files and nearly 47% of all responses are returned by the top 1% of the sharing hosts. This social problem starts to transform Gnutella into a C/S like system which soon may have to face the technical (degradation of performance, vulnerability, etc.) and legal problems of Napster. Two results of this study are given below.

Of 33,335 hosts:

- 22,084 (66%) of the peers share no files
- 24,347 (73%) share ten or less files
- Top 1 percent (333) hosts share 37% (1,142,645) of total files shared
- Top 5 percent (1,667) hosts share 70% (1,142,645) of total files shared
- Top 10 percent (3,334) hosts share 87% (2,692,082) of total files shared

Of 11,585 hosts:

- Top 1% of sites provide nearly 47% of all answers
- Top 25% of sites provide 98% of all answers
- 7, 349 (63%) never provide a query response

Another "social" issue which is not addressed by Gnutella is reputation: In a P2P system peers frequently have to "meet" unknown peers and have no possibility to judge their reputation, i.e., to what extent they can trust the peers and the data provided by them.

5 Freenet

Freenet [4, 5] is a P2P system for the publication, replication and retrieval of data files. Its central goal is to provide an infrastructure that protects the anonymity of authors and readers of the data. It is designed in a way that makes it infeasible to determine the origin of files or the destination of requests. It is also difficult for a node to determine what it stores, since the files are encrypted when they are stored and sent over the network. Thus following the reasoning of the designers of Freenet, nobody can be sued, even if he or she, for example, stores and distributes illegal content. Besides the aspect of anonymity protection the Freenet system implements another interesting concept: an adaptive routing scheme for efficiently routing requests to the physical locations where they are most probable to appear. In order to improve search efficiency Freenet maintains routing tables that are dynamically updated as searches and insertions of data occur. Thus the Freenet

graph evolves dynamically over time as implied by the routing tables. In order to further improve search efficiency, Freenet also uses dynamic replication of more popular files such that files can migrate to peers where they are more likely to be found. Thus Freenet does not require a central server as Napster, and compared to Gnutella Freenet avoids inefficient message broadcasts.

When a peer joins a Freenet network it has to know some existing node in the network (as in Gnutella alike). By interacting with the network it will fill its routing table, which is initially empty, and thus the Freenet network structure will evolve. The routing tables in Freenet (Table 3 shows an example), store the addresses of the neighbouring peers and additionally the keys of data items that this peer stores and the corresponding data. So when a search request arrives it may be that the peer stores the data in its table and can immediately answer the request. Otherwise it has to forward the request to another peer. This is done by selecting the one that has the most similar key in terms of lexicographic distance. When an answer arrives the peer stores the answer in its data store. If the data store is already full this might require to evict other entries. Here the strategy is to evict the least recently used. As can be seen in Table 3 the peer may also decide to evict the data corresponding to a key before it evicts the address of the peer in order to save space while maintaining the routing information. Since peers route search requests only to the peer with the closest key, Freenet implements a depth-first search strategy rather than a breadth-first strategy as in Gnutella. Therefore also the time-to-live of messages is substantially longer, typically 500. As in Gnutella Freenet messages carry identifiers in order to detect cycles.

Key	Data	address
8e47683isdd0932uje89	ZT38hwe01h02hdhgdzu	tcp/125.45.12.56:6474
456r5wero04d903iksd0	Rhweui12340jhd091230	tcp/67.12.4.65:4711
f3682jkjdn9ndaqmmxia	eqwe1089341ih0zuhge3	tcp/127.156.78.20:8811
wen09hjfdh03uhn4218	erwq038382hjh3728ee7	tcp/78.6.6.7:2544
712345jb89b8nbopledh		tcp/40.56.123.234:1111
d0ui43203803ujoejqhh		tcp/128.121.89.12:9991

Table 3: Sample Routing Table in Freenet

Figure 5 illustrates the Freenet search mechanisms and reorganization of the network. Peer A is sending a search request to B for file X.mp3. First B forwards the request to C. Since the TTL is 2 and C does not hold the data this request failed. Therefore B next forwards the request to C where the data is found. While sending back the response containing the file X.mp3, the file is cached at all peers, i.e., at B and A. In addition A learns about a new node C and a new connection is created in the network.

Next we describe the update mechanism in Freenet. When a file is inserted into the Freenet network, first a key of the file is calculated. Keys are represented as Uniform Resource Identifiers (URIs) of the form freenet:keytype@data. Freenet distinguishes different kinds of keys: Keyword Signed Keys (KSK), Signature Verification Keys (SVK), and Content Hash Keys (CHK). Keyword Signed Keys use a short descriptive text, provided by the user, to generate a public/private key pair. The public key part is hashed and used as the file key, whereas the private key part is used to sign the file. The file itself is encrypted using the user-defined descriptor as key. For finding the file the user must know the descriptive text. This way of encrypting and signing files provides a moderate level of security as dictionary attacks are possible. Signature Verification Keys allow to to create a subspace in Freenet, i.e., a controlled set of keys. It works similarly as Keyword Signed Keys only that

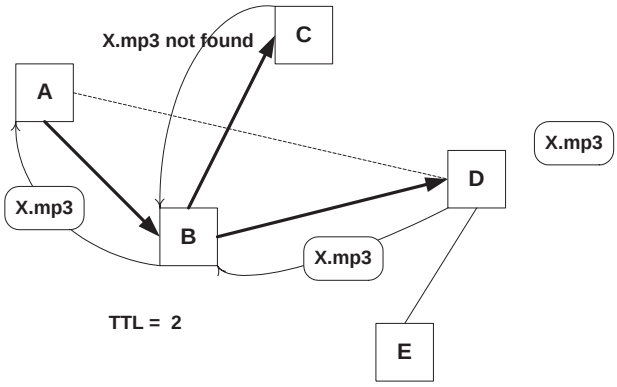

Figure 5: Searching in Freenet

the key pair is generated randomly. Users who trust the owner of a subspace will also trust documents in the subspace because inserting documents requires knowing the subspaces private key. Content Hash Keys are derived from hashing the contents of the file, and thus are a means to verify file integrity. After a key is generated an insert message with this proposed key and a hops-to-live value is sent to the neighbours. Then every peer checks whether the proposed key is already present in its local store. If yes, it returns the stored file and the original requester must propose a new key. If no, it routes the insert message to the next peer for further checking. The routing uses the same key similarity measure as searching. The message is forwarded until the time-to-live has reduced to 0 or a failure occurred. If the time-to-live is 0 and no collision was detected, then the file is inserted along the path established by the initial insertion message.

5.1 Discussion of Freenet

As the caching strategy for the routing tables is designed in a way that tends to cluster similar data (keys) at nodes in the network, the assumption is that nodes get more specialized over time. This assumption turns out to hold when performing simulations of Freenet. We describe one exemplary result: The simulation was done on a network of 1000 identical nodes, where each node stores maximally 50 data items and references at most 200 other peers. The initial topology of the network was a ring topology where each node is connected to 4 neighbours with the closest identifiers as illustrated in Figure 6.

In each time-step of the simulation an insert operation was performed randomly, with a time-to-live of 20. Every 100 steps 300 queries were executed with a TTL=500 from random nodes. One observes that the median path-length of the queries drops sharply from initial values close to the maximal TTL of 500 to a value of around 6. This and similar simulations show that the number of hops required for a search actually is close to logarithmic in the number of peers in the network.

An explanation why the search performance improves so dramatically is found in the properties of the graph structure of the network. Analyses of the resulting Freenet networks reveal that it has the characteristics of so-called small-world graphs. The notion of small-world phenomenon has been originally introduced by Stanley Milgram in 1967. He conducted an experiment where he instructed randomly chosen people in Nebraska to pass letters to a selected target person in Boston. The persons for forwarding the letter were required to be known on a first-name basis. Interestingly the number of steps required to reach the receiver had a

median of only 6 steps.

The work was extended to a computer science context by Duncan Watts and Steven Strogatz in 1998. They identified classes of graphs that exhibit a strong clustering property, but contain short paths between any nodes, the so-called small-world graphs. Strong clustering means that nodes that are linked to a given node, have also a high probability to be linked to each other, as it is the case for example in human relationships. Short paths on the other hand are a known property of random graphs, where graph theory shows that always paths of a length logarithmic in the number of nodes exist.

However, assuming that social networks are random graphs is fairly unrealistic. Small world graphs have short paths as they scatter few long-distance links into a graph structure that is mostly locally clustered. The Freenet algorithm achieves this structure through its tendency to cluster nodes storing similar data, but since this clustering is not perfect, long-distance links are always likely to occur. Figure 6 illustrates of how small world graphs relate to regular and to random graphs.

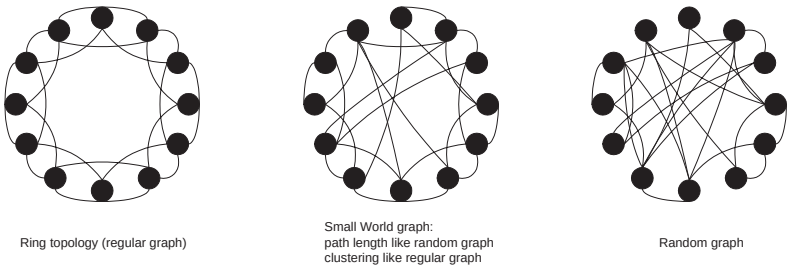

Figure 6: Small World Graphs

6 Commercial Systems

6.1 FastTrack

FastTrack (http://www.fasttrack.nu/) is a P2P library that powers several of the currently most successful P2P search engines such as KaZaA or Grokster. It uses a proprietary, encrypted protocol which is not published. Also no information on the system architecture is available. FastTrack's presumed architecture is that data nodes hold the data to be made available and so-called supernodes index this data and provide search capabilities for a set of data peers. Queries are also forwarded to other superpeers. However, it is unclear how this query routing works in detail. Additionally, a central super-superpeer exists that is responsible for special administrative tasks. Such, FastTrack can be viewed as a modern version of Napster that uses a hierarchical server structure instead of a centralized server. The giFT group (http://gift.sourceforge.net/) attempts to reverse engineer the protocol to make available third-party clients and software.

6.2 JXTA

JXTA [7] is a proposal for a uniform interface to P2P systems and to facilitate interoperability of P2P systems. JXTA defines a three layer P2P software architecture (Figure 7), a set of XML-based protocols, and a number of abstractions and concepts such as peer groups, pipes, and advertisements to provide a uniform platform for applications using P2P technology and for various P2P systems to interact.

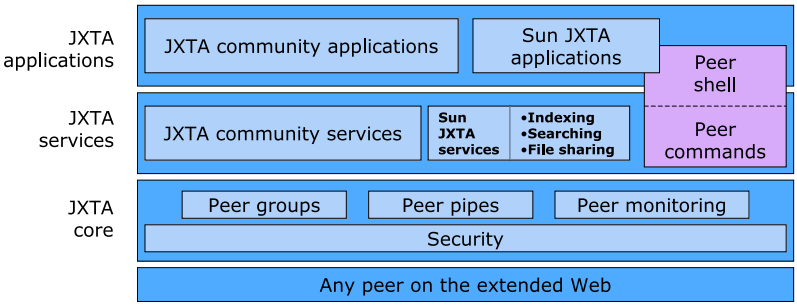

Figure 7: JXTA architecture

7 Research Systems

Besides the file sharing applications and commercially driven approaches a substantial amount of research has been recently devoted to P2P information systems. In this section we present some of the approaches with particular emphasis on how they address the problem of search.

7.1 OceanStore

OceanStore [15] is an Internet-based, distributed, global storage infrastructure. It consists of possibly millions of cooperating servers. Each participating server is a client and a server, therefore OceanStore is a true P2P system. The data is split up in fragments which are stored redundantly on the servers. Sophisticated replication mechanisms are devised in order to ensure data availability and performance. High fault tolerance is achieved by self-monitoring, the use of erasure coding and automatic repair. Updates are performed by using Byzantine consensus protocols. For search OceanStore provides the Tapestry subsystem, a self-organizing routing and object location system. For object location it uses hashed suffix routing. An example of how routing is performed is given in Figure 8. It illustrates the path a message originating at node 0325 takes that is destined for node 4598. Tapestry routes the message through nodes **** → ***8 → **98 → *598 → 4598, where asterisks represent wildcards. When the search starts at 87CA only two hops are required as multiple bytes are matched at node 7598.

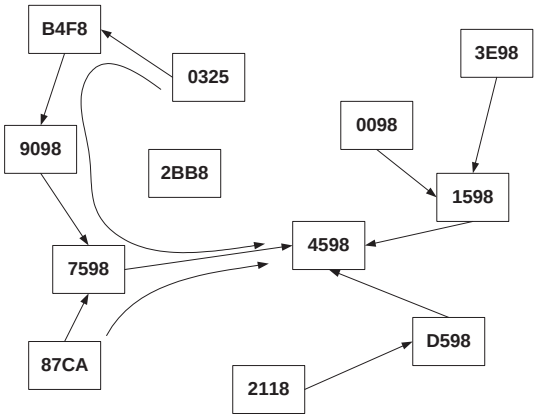

Figure 8: Searching in Tapestry

7.2 Chord

Chord [19] is a decentralized P2P lookup service that stores key/value pairs for distributed data items. Given a key, the node responsible for storing the key's value can be determined using a hash function that assigns an identifier to each node and to each key (by hashing the node's IP address and the key). Each key k is stored on the first node whose identifier id is equal or follows k in the identifier space. Each node maintains a routing table with information for only about $O(\log N)$ nodes. With a high probability, the number of nodes that must be contacted to resolve a query in an N-node network is $O(\log N)$.

7.3 Pastry

An approach similar to Tapestry and Chord was also used in Pastry [16]. Pastry and Tapestry differ in their approach to achieve network locality and to support replication. Existing applications built on top of Pastry include Past, a storage utility, and Scribe, a publish/subscribe system. Tapestry, Pastry and Chord derived their approach to routing from the landmark work of Plaxton [13].

7.4 CAN

CAN [14] is another proposal for a decentralized P2P lookup service. It differs from the previous ones as it uses a multidimensional space as underlying abstraction for routing. Keys are mapped into regions and peers split regions while memorizing peers responsible for neighbouring regions. Routing is then performed by forwarding requests to the regions closest to the position of the key. In this way the expected number of hops for a search is $O(dN^{1/d})$, where d is the chosen dimensionality.

7.5 P-Grid

P-Grid [1] is a self-organizing access structure that uses a prefix tree for routing search requests. More details on it can be found in these proceedings. It combines the properties of efficient routing as they are exhibited by the structured P2P search systems like OceanStore, Chord and CAN, with the ability to join partitioned networks, which is found in real-world systems such as Gnutella and Freenet. The structured P2P search systems have been lacking this ability, as they supported only operations for single nodes to join and leave the network.

7.6 Random Walker

A different strategy is taken in [10]. There the gossiping approach is applied as in Gnutella. However, gossiping is performed in a much more careful way. The query algorithm is based on multiple random walks that resolve queries almost as quickly as Gnutellas flooding method while reducing the network traffic by orders of magnitude. Distributed replication strategies are devised that yield close-to-optimal performance.

7.7 Farsite

Farsite [3] is designed as a serverless, distributed file system that does not assume mutual trust among the client computers on which it runs. Logically the system tries to provide the illusion of a central file server. Physically a group of desktop client computers collaboratively establishes this virtual file server that can be accessed by any of the clients. Farsite will offer a global name space for files, location-transparent access to files, reliability through multiple encrypted replicas of files,

and a hierarchical directory structure via a distributed directory service. So far only simulations are published but neither system architecture and design nor algorithms are publicly available.

8 Concluding Remarks

These are the tutorial notes for our tutorial on peer-to-peer (P2P) information systems held at WDAS2002. The long version of this tutorial was last held at ICDE2002. The slides of the ICDE version are available at http://lsirwww.epfl.ch/. The first version of this tutorial was given at ESEC2001.

References

[1] K. Aberer, M. Hauswirth, M. Punceva, and R. Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6(1), Jan./Feb. 2002.

[2] E. Adar and B.A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000. http://firstmonday.org/issues/issue5_10/adar/index.html.

[3] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2000)*, pages 34–43, 2000.

[4] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, 6(1), Jan./Feb. 2002.

[5] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymit y and Unobservability*, number 2009 in Lecture Notes in Computer Science. Springer Verlag, Berlin, 2001.

[6] Clip2. *The Gnutella Protocol Specification v0.4 (Document Revision 1.2)*. http://www.clip2.com/GnutellaProtocol04.pdf.

[7] L. Gong. JXTA: A Network Programming Environment. *IEEE Internet Computing*, 5(3):88–95, May/June 2001.

[8] M. A. Jovanovic, F. S. Annexstein, and K. A. Berman. Scalability Issues in Large Peer-to-Peer Networks - A Case Study of Gnutella. http://www.ececs.uc.edu/~mjovanov/Research/paper.ps.

[9] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. Technical Report Technical report 99-1776, Cornell Computer Science, Oct. 1999. http://www.cs.cornell.edu/home/kleinber/swn.pdf.

[10] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of 16th ACM International Conference on Supercomputing(ICS'02), New York, USA,*, June 2002.

[11] Napster homepage. http://www.napster.com/.

[12] A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, March 2001.

[13] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *Proceedings of the 9th Annual Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM SIGCOMM*, 2001.

[15] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz. Maintenance-free Global Data Storage. *IEEE Internet Computing*, 5(5), Sep./Oct. 2001.

[16] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany*, November 2001.

[17] Napster protocol specification, April 7 2001. http://opennap.sourceforge.net/napster.txt.

[18] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. http://www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html.

[19] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM*, 2001.

[20] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A Wide-Area Distributed Database System. *VLDB Journal*, 5(1):48–63, 1996.

Karl Aberer is with the School of Computing and Communication Science Science at the Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland. E-mail: karl.aberer@epfl.ch

Manfred Hauswirth is with the School of Computing and Communication Science Science at the Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland. E-mail: manfred.hauswirth@epfl.ch