

# Survey of Peer-to-Peer Systems

Kaoutar El Maghraoui  
*Computer Science Department,*  
*Rensselaer Polytechnic Institute, Troy, NY, USA*

**Abstract.** Over the past decade, peer-to-peer systems (P2P) have gained an increasing popularity among the Internet community. Although the P2P concept is as old as the creation of the Internet itself, it has been given recently a paramount attention by researchers from various computer science fields. P2P systems provide a paradigm shift from the classical client server approach to a completely decentralized approach where central coordination is no longer needed. A P2P network provides a fault tolerant, scalable, and cost effective solution to a large number of Internet applications such as file sharing, and distributed computing. In this paper, we introduce key concepts and properties of P2P systems and survey main commercial and research P2P systems.

**Keywords.** Decentralized coordination, Distributed computing, Information sharing, p2p, self-organizing.

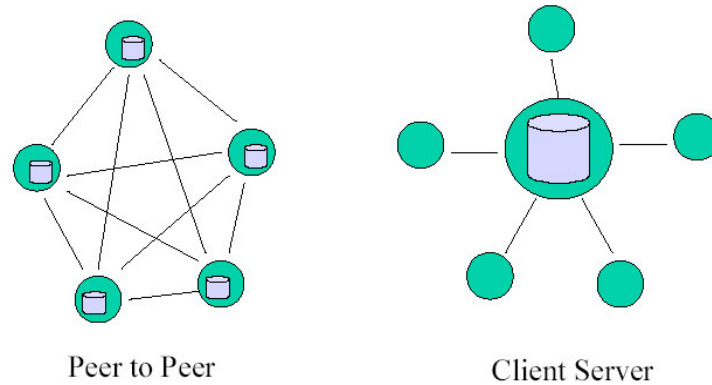
## Introduction

With the constantly increasing participation in the Internet and the high dynamic nature of its nodes, the traditional client/server model poses severe limitations:

- Scalability is hard if not impossible to achieve.
- Complex load balancing and fault-tolerant techniques are needed to provide robustness and reliability to large-scale Internet applications.
- Servers easily become overloaded when facing a large number of requests and hence easily become single points of failures and a source of bottleneck.
- Substantial administration and maintenance are needed.
- This model fails to take full advantage of the large number of available resources in the Internet. A lot of resources still remain idle at the edge of the network

Given all of these limitations, researchers have been highly motivated to come up with new techniques that address the limitations of the traditional client/server model in large scale Internet environments. In P2P systems, there are no central servers. Every node acts both as a client and a server. This approach solves efficiently the scalability issue and distributes the load and the network bandwidth among all participating nodes or peers. This strategy solves also the bottleneck issue since no central server is responsible for handling all the incoming requests. Any peer in the system could respond to user queries given the necessary resources and computational capability. It is important to mention that the idea of peer-to-peer is not a new one. Even the first telephone systems and Arpanet can be viewed as P2P systems since they have many of the features existing in new P2P systems. This idea however has been emphasized more recently due to the increasing need of scalable and fault-tolerant strategies over large –scale systems.

This paper provides a survey of P2P systems. Section 1 introduces the technology underlying P2P systems and discusses some of its main benefits and limitations. It also presents the different types of P2P applications and discusses some current efforts towards making this technology standardized. Section 2 discusses main P2P systems. Section 3 concludes with ongoing research efforts and some of the open research questions that still need to be addressed.



**Figure 1.** The Peer-to-Peer versus the Client/Server approach

## 1. P2P Technology

### 1.1 What is P2P?

P2P systems have not been fully standardized yet. Therefore there is still no formal definition of P2P systems as it is the case with most new emerging technologies. However most of the people who tried to define P2P agreed on its features. Clay Shirkey from The Accelerator Group has given a very intuitive definition that has captured most of the important features of P2P systems [1]:

*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 must operate outside the DNS and have significant or total autonomy of central servers.*

A P2P system forms an overlay network where resources are shared among all participants. Information is also exchanged directly without the involvement of a third party and without the need of a centralized coordination. Another key feature of P2P systems as mentioned in Clay's definition is the volatility of the network connections. Peers operate outside the DNS, which is mainly characterized by its static nature, where nodes rarely change their topology. Peers can join and leave the P2P network at any time in a flexible manner without harming the functionality of other peers. This is one of the most important features that have contributed to the popularity of these systems. It has allowed the Internet to be the environment of choice for a wide range of applications without having to worry about its constantly changing connectivity and availability.

Several principles characterize P2P applications:

- *Resource sharing*: Any peer is involved somehow by sharing some of its resources. These resources could be physical resources such as CPU cycles, memory, storage capacity, and network bandwidth or logical resources such as services or some other form of knowledge.
- *Decentralization*: There is no central coordination in a P2P system. A consequence result of this is the avoidance of a central point of failure. The load of the system is shared among all participating peers. This improves the utilization of the system resources and renders the system scalable and more robust.
- *Autonomy and self-organization*: Due to the absence of a central coordinator, peers are responsible for organizing themselves autonomously. P2P protocols provide peers with the ability to join an existing system, share and exchange resources with the rest of the system, and leave the system when desired. A peer can be looked at as a completely independent entity that has the flexibility to interact with its neighbors with a lot of flexibility.
- *Emerging global behavior*: The behavior of the entire system emerges as an accumulation of the behavior of the individual peers and how they interact with each other. It is usually difficult to determine the global behavior due to the high dynamicity of the P2P system and the unpredictable behavior of the participating peers. The topology is also constantly changing.
- *Dynamicity*: This principle is one of the key features of P2P systems. The Internet is inherently dynamic and constantly changing. P2P applications fit perfectly with the Internet model. It is more intuitive as many applications exhibit this dynamic behavior. It is much easier for nodes to join and leave the system without affecting or harming the entire system as a whole. In traditional client/server architectures, this was not possible or was not an easy feature to provide. Client/Server applications are mainly characterized by their static nature.

They are several benefits and advantages to P2P systems. They provide better price/performance, scalability, fault tolerance, resource sharing, and deployment flexibility. However, these added values come with a certain price. Several challenges need to be addressed as well to meet the requirements of these systems. Table 1 summarizes the different benefits and their challenges.

**Table 1.** Benefits and Challenges of P2P Systems

Benefits	Challenges
<ul style="list-style-type: none"> <li>• Efficiency</li> <li>• Price/performance</li> <li>• Resource Sharing</li> <li>• Massive Resources</li> <li>• Fault tolerance</li> <li>• Scalability</li> <li>• Deployment Flexibility</li> </ul>	<ul style="list-style-type: none"> <li>• Security and building reputation and trust systems</li> <li>• Efficient Load balancing algorithms</li> <li>• Search</li> <li>• Data integration</li> <li>• Data availability</li> <li>• Replication strategies</li> <li>• Resource discovery and peer formation protocols</li> </ul>

## 1.2 P2P Applications

P2P technology has been widely deployed in different areas. Several applications have been developed and are currently being used in the areas of business, e-commerce, distributed computing, and communication.

The most popular applications have been file-sharing applications such as Napster, Gnutella, Morpheus, Freenet, and Kazaa. Napster [2] was one of the first P2P applications that have gained wide popularity. It was used for sharing mp3 files between registered users. It also provides several services such as chat rooms, and querying Napster servers (a farm of servers). Though Napster has been considered as one of the first P2P applications, it is not a pure P2P system. Napster maintains a central database that indexes all the participating users and the files that they are sharing. Users query the Napster server (see Figure 2), which returns a list of users having the requested files. The user can then contact directly any participant from the returned list. The file exchange happens directly between the two users. This is really where the P2P concepts come to play. Napster uses both concepts from the Client/Server model and the P2P model. Scalability is therefore very limited due to the Centralization of Napster servers. Napster uses also a very simplistic structured schema for the queries and thus does not address the issue of reputation of participating peers. Many similar applications emerged after Napster with pure P2P features such as Gnutella, Freenet, and Kazaa. They rely on distributed search mechanisms to address the limitations of Napster's central servers and to provide fault-tolerance and scalability. More details will be given about these systems in section 2.

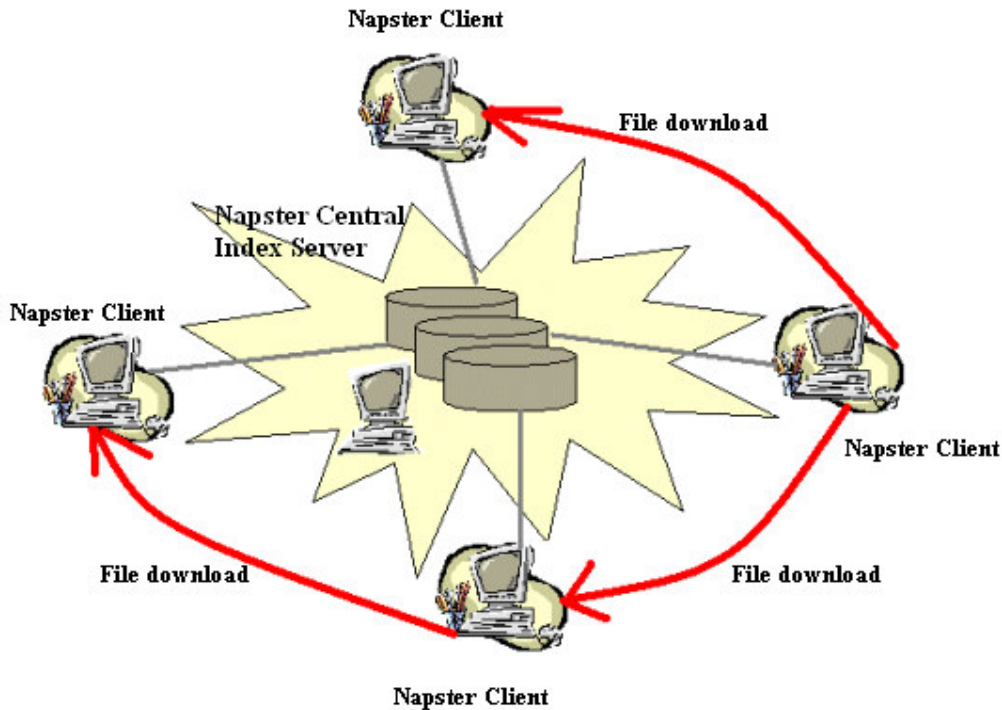


Figure 2. Centralized architecture of Napster

P2P concepts have been also very suitable and successful in the area of distributed computing. They are especially useful for large computations that have a high degree of parallelism and a coarse correlation between their different components. A considerable Speedup can be achieved if large calculations could be divided into completely independent and parallel parts that can run on separate nodes. Typical applications are [SETI@home](#) [3], [Folding@home](#) [4], and [FightAids@home](#) [5] (see Table 2). All these applications require a lot of computational power and try to exploit the Internet's idle resources. Users participate in these projects by downloading and running some client software that runs on a low priority

and do not interfere with the user's own work. The installed software gets triggered only when the machine is idle and performs part of the huge computation. Unlike file sharing P2P applications, computations are run on peers that do not communicate with each other.

**Table 2.** Popular P2P Distributed Computing Applications

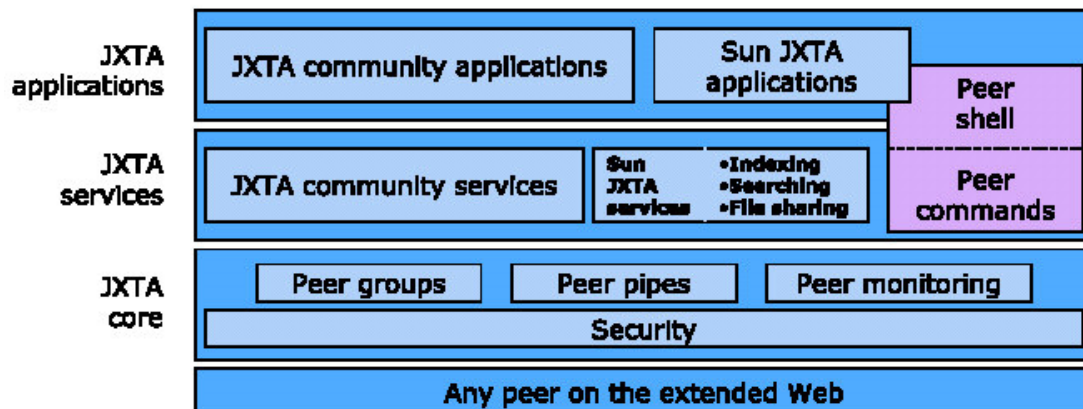
Distributed Computing Projects	Goals
<a href="#">SETI@home</a>	Search for extraterrestrial intelligence in the outer space
<a href="#">Folding@home</a>	Study of the protein folding, misfolding, aggregation, and related diseases
<a href="#">FightAIDS@home</a>	Assist fundamental research to discover new drugs for AIDS

They are other P2P applications used for communication such as ICQ, AOL Instant messaging, MSN Messenger, and some implementations of Internet radio broadcasts and videoconferencing.

### 1.3 Standardization efforts

Most of the P2P applications have been developed in isolation and are not compatible with each other. Although they rely on the same concepts, they use different protocols and strategies for peers' formation, information exchange, and control. To address this issue, SUN proposed JXTA [6], a proposal for a uniform interface to P2P systems to allow interoperability between different P2P systems. JXTA stands for Juxtapose. It provides a platform with basic functionalities necessary for P2P networks, a common set of open protocols, and an open source reference implementation. JXTA protocols do not depend on any specific programming language, network topology, or particular security model. They are all specified as a set of XML messages that are exchanged between the participating peers. JXTA standardizes the following P2P features:

- Peer discovery
- Peer self-organization into groups
- Resource advertisement and discovery
- Inter-peer communication
- Resources and peers monitoring



**Figure 3.** JXTA Architecture

JXTA defines a three-layer P2P software architecture (Figure 3). The first layer consists of all the high level P2P applications. The second layer defines different P2P services such as Indexing, searching, and file sharing. JXTA protocols are only responsible for publishing and discovering these services. To invoke a specific service any mechanism could be used such as SOAP, RMI, CORBA, etc. However all peers should perform the same steps when invoking a given service. There are two categories of services: services pertaining to a peer, and services pertaining to a group of peers. The third layer defines the core functionalities of JXTA: peer groups, peer pipes, and peer monitoring.

Pipes are unidirectional, and asynchronous virtual communication channels. They are responsible of sending and receiving messages between endpoints. They support two modes of communications: point-to-point and one-to-many communication. Therefore any pipe has one input line and one or many output lines. Pipes are dynamically bound to endpoints at runtime. They can be connected to more than one peer at the same time. JXTA defines the abstractions of pipes to peers and peers to endpoints. End users can implement reliable, secure, and bi-directional pipes on top of the cores pipes that JXTA provides.

Given the dynamic nature of P2P networks and the large participation these networks usually experience, security becomes very important. Trust is one of the key issues that need to be addressed in such environments. JXTA does not provide a well-defined authentication. However it does provide hooks to implement security. JXTA architecture shows only one security layer in the third layer. This is not an accurate representation since security can manifest itself in all the layers of the architecture. More specifically, the transport and service layers should support also secure communication channels (Ipssec, TLS, etc.)

## **2. Discussion of Main P2P Systems**

### *2.1 Gnutella*

Gnutella [7] is a file-sharing P2P system. It is one of the main successful successors of Napster. Unlike this latter, it is not limited only to the exchange of music files. It supports the exchange of any file types. The Gnutella relies on a decentralized file search protocol through which peers communicate between each other and exchange files in a complete decentralized manner without having to refer to a centralized indexing database. To participate in the Gnutella file exchange, a peer must first know an already existing Gnutella host. Specialized servers exist for such purposes: they return a list of well know peers as a bootstrapping mechanism. Each peer maintains a list of neighboring peers. To search for a file, a peer asks its neighbors for the files of interest. If one of the neighbors has the files, it responds and the search stops. Otherwise a constrained form of broadcast is used to find the files: each request has a time to leave field (TTL). Upon receipt of a request message, the neighbor first decrements the TTL value. It checks first if it can satisfy the request. If it can, it responds with a hit message. Otherwise, if the TTL value is greater than zero and it has not seen the same request before (for loop detection), it forwards the message to its neighbors. The responses are routed along the same path as the request messages (see Figure 4). The Gnutella protocol consists of 5 main messages as shown in table 3.

The Gnutella protocol is a simple and efficient protocol. It is completely decentralized and hence there is no single point of failure. It is fault-tolerant towards failures and not as

susceptible to denial of service attacks as Napster. However the protocol could flood the network with query messages due to its broadcast mechanism. For example, with a TTL Of 7 and an average of 4 connections per peer (4), [8] shows that the total number of messages that could originate from one Gnutella message can be calculated as:

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

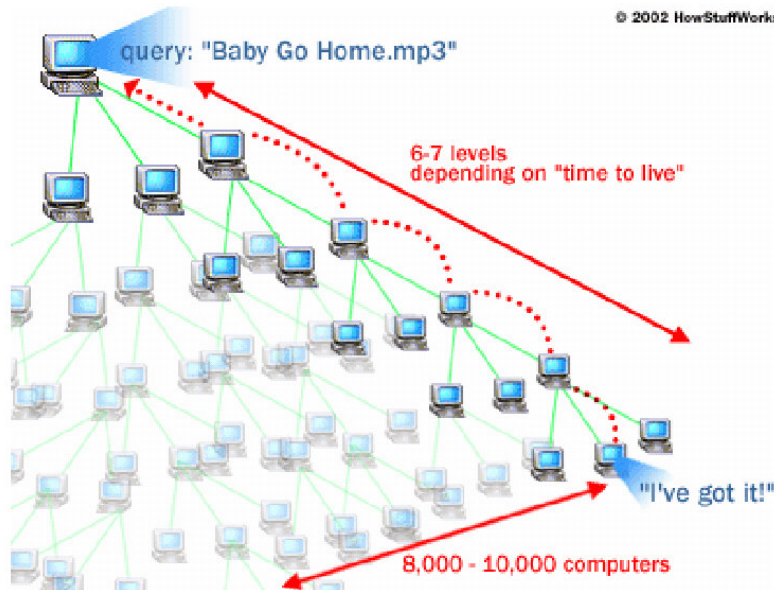


Figure 4. Searching in a Gnutella network

Table 3. The Gnutella protocol message types

Gnutella Message	Description
Ping	Used to announce the presence of a peer and to check the status of other peers.
Pong	A message sent a response to Ping. The message contains the IP address, port number, and the size of the files shared by the pinged peer.
Query	Used to query a search for a file. The message contains the required minimum bandwidth of the target peer and the search criteria.
QueryHit	The peer, who can satisfy the Query messages, responds with a QueryHit message. This message contains the IP, port number, network bandwidth of the target peer, and the number of results, and the results.
Push	This message is used to push the Query messages behind a firewall. It contains the Server identifier, the index of the requested file, the IP address and port number of the requesting peer.

Some real-world experiments [9] have shown that the network traffic in a Gnutella network could reach up to 3.5Mps. To alleviate this problem, [9] suggests using caching to reduce the traffic. This paper has argued that popular documents have a Zipf-like distribution, which is similar to the access of web documents. So caching would reduce significantly the amount of high traffic Gnutella networks experience. Another problem that challenges Gnutella is “free riding” [10]. Free riding describes the fact that only a small fraction of the

peers in the Gnutella network share files and respond to most of the queries. [10] shows that 66% of the peers do not share any files and almost 47% of all the query hits are returned by only 1% of peers that share files. This problem tends to transform Gnutella into a centralized system and hinders significantly its performance. Another issue that still challenges Gnutella is the reputation of the participants.

## 2.2 Kazaa

Kazaa [11] is a hybrid of the centralized Napster and the decentralized Gnutella. It introduces a very powerful concept: “Super Peers”. Super peers act as local search hubs. Each super peer is responsible for a small portion of the network. It acts as a Napster server. These special peers are automatically chosen by the system depending on their performance (storage capacity, CPU speed, network bandwidth, etc) and their availability. Normal peers upload all their files to their assigned super peer, which periodically exchange the lists of files they share. All requests are sent to super peers. Kazaa is powered by FastTrack (<http://www.fasttrack.nu/>), a P2P library that powers many of the P2P search engines.

## 2.3 Freenet

Napster, Gnutella, and Kazaa do not provide user anonymity. Everyone knows who they are downloading from or who sent them queries. Freenet has been designed to allow such anonymity of users among other features. Freenet [12, 13] is a P2P system for the publication, retrieval and replication of files. Besides the anonymity of users, it also ensures that nodes do not know the contents of what they are storing exactly since all stored file are encrypted. Freenet has also improved the search efficiency by using an adaptive routing mechanism that routes requests to the locations where they are most probable to appear. Freenet maintains routing tables that dynamically change to reflect the new search and new data insertions that happen. It uses also dynamic replication of the more popular files to improve the search efficiency. The system is completely decentralized unlike Napster and avoids unnecessary broadcasts unlike Gnutella. There are two main operations that govern the Freenet operations: search and update. The following sections will give more details about these two operations.

Similar to Gnutella, a peer needs to know an existing Freenet host to be able to join the Freenet network. The newly joining peer starts building its routing table through its interaction with the other peers. The routing table stores the IP address of the neighboring peers, the keys of the items this peer stores, and the items themselves. Upon the receipt of a search query, the peer searches its own data store, if it finds a match it responds with a hit. Otherwise, it forwards the request to the peer who has the most similar key in terms of lexicographic order. When a response arrives, the peer stores some relevant information in its data store to help routing new requests. Freenet implements the depth first search strategy for routing, while Gnutella implements the breadth first search strategy.

Besides maintaining routing tables, peers maintain also data stores. Data stores have been designed with the following requirements:

- Finding rapidly a given document given a certain key
- Finding rapidly the closest key given a certain key
- Prioritising the most popular documents and selecting the right data store entry to delete when under pressure

Keys are used as a way to locate quickly documents. They are modelled according to the structure of Uniform Resource Identifiers (URIs). Different kinds of keys are supported:

Keyword Signed Keys (KSK), Signature Verification Keys, and Content Hash Keys (CHK). Table 4 describes the functionality of each one of these keys. After a key is generated, an insert message is sent to all the neighboring peers with the generated key and a TTL value. Every peer checks if it has this file in its data store. If so, it responds with the stored file to the requesting peer, who must generate a new key. Otherwise it routes the message to the next peer using the same key similarity criteria like in search. The routing will stop if the TTL value becomes 0 or if a failure occurs. If no collisions have been detected and the TTL is 0, the file is inserted along the same path that has been established with the routing messages.

**Table 4.** Freenet Keys

Key Types	Format	Functionality
KSK	Short descriptive text to describe the document Example: Classes/Algorithms/P2P	Uniquely identifies a document. A public/private key pair is generated from the descriptive text. Public part is hashed and used as the file key The private part is used to sign the file
SVK	User info is added to avoid name space conflict Private/public key pair	Allow the creation of a subspace in Freenet: a set of controlled pairs The key set is generated randomly
CHK	Hashing of the content of the file	Use of message digest algorithm

Freenet is considered as one of the Internet success stories. It is a scalable distributed model that has succeeded in keeping the anonymity of participating users. It relies on adaptive routing and searching strategies that cause the search to converge very fast. It uses also several strategies to reduce the amount of bandwidth consumed by its network. However there are still some challenges that need to be addressed by Freenet. Anonymity inserts some additional overhead. There is a trade-off between anonymity and searching efforts. Freenet does not guarantee if a search failed because the file didn't exist or because it didn't find it. Moreover, it doesn't guarantee a document that you submit today will exist tomorrow.

### 3. Ongoing Research and Future Directions

P2P applications have found a lot of popularity among individual users mainly for sharing music, video, and picture files. It is currently building its way up to enterprise users. However before being able to meet all the requirements of business applications, there are still a lot of challenges that need to be addressed. Security is one of the key issues that have not been enforced properly in this technology. QoS is another difficult issue that have to be addressed. Scalability needs also to be improved to be able to accommodate a worldwide participation of users especially in the business world.

A second generation of P2P systems have recently emerged that are referred to as structured P2P systems. They are self-organizing, load balanced, and fault-tolerant systems. One their major difference with unstructured P2P systems like Gnutella, Freenet, and Kazaa is that they have scalable guarantees on a number of hopes to answer a given query. They are all based on a distributed hash table interface. Examples include Chord [14], Pastry [15], Content Addressable Networks (CAN) [16], P-Grid [17], and Farsite [18]. All these

systems are still in their research phases. They rely on high probabilistic guarantees and they focus mainly on one of the key problems in P2P systems: efficient search.

Some ongoing work addresses also the problems of P2P with DBMS. Data sharing P2P systems generally provide a coarse granularity of data. Different problems exist when trying to provide a finer granularity of data mainly: clear semantics for querying and updating (e.g. SQL), dynamic schema mapping, and replication and consistency. PeerDB [19] is a research effort that is trying to address some of these issues.

Security is one of the main recent research topics in P2P systems. Different possible attacks and vulnerabilities have been identified such as Attacks by self-replication, Man in the middle attack, denial of service attacks, routing attacks, Partition attacks. Research efforts have suggested different measures to increase the security of P2P systems that range from cryptography, to replication, to reputation protocols [20, 21]. Still security remains a very challenging problem in P2P systems given the diverse and dynamic nature of these systems. Security models need to be intelligent enough to cope with the constantly changing environment of the P2P network.

## References

- [1] A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, March 2001.
- [2] Napster homepage. <http://www.napster.com/>.
- [3] SET@home homepage. <http://setiathome.ssl.berkeley.edu/>.
- [4] Folding@home homepage. <http://www.stanford.edu/group/pandegroup/folding/>.
- [5] FightAids@home homepage. <http://fightaidsathome.scripps.edu/>.
- [6] L. Gong. JXTA: A Network Programming Environment. *IEEE Internet Computing*, 5(3): 88-95, May/June 2001.
- [7] Clip2. *The Gnutella Protocol Specification v 0.4 (Document Revision 1.2)*. <http://www.clip2.com/GnutellaProtocol04.pdf>.
- [8] K. Aberer, M. Hauswirth. *An Overview on Peer-to-Peer Information Systems*, 2002, <http://citeseer.ist.psu.edu/aberer02overview.html>
- [9] K.Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. <http://www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html>.
- [10] E. Adar, M. Hauswirth, M. Runceva, and R. Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6 (1), Jan./ Feb. 2002.
- [11] Kazaa homepage. <http://www.kazaa.com/us/index.htm>.
- [12] 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.

- [13] 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 Anonymity and Unobservability*, number 2009 in Lecture Notes in Computer Science. Springer Verlag, Berlin, 2001.
- [14] 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.
- [15] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. *In Proceedings of the 18<sup>th</sup> IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany*, November 2001.
- [16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. *In Proceedings of the ACM SIGCOMM*, 2001.
- [17] K. Aberer, M. Hauswirth, M. Puceva, and R. Schmidh. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6 (1), Jan./ Feb. 2002.
- [18] 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 on the International Conference on Measurements and Modelling of Computer Systems (SIGMETRICS 2000)*, pages 34-43, 2000.
- [19] W.S. Ng, B.C. Ooi, K.L. Tan and A. Zhou. PeerDB: A P2P-based System for Distributed Data Sharing. *In Proc. of ICDE*, 2003.
- [20] D. S. Wallach. A Survey of Peer-to-Peer Security Issues. [dwallach@cs.rice.edu](mailto:dwallach@cs.rice.edu)
- [21] E. Sit, and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. *Laboratory for Computer Science, MIT*.