An Overview of Peer-to-Peer Computing

R. O. Oladele

Department of Computer Science

University of Ilorin

P.M.B. 1515, Ilorin, Nigeria.

roladele@yahoo.com

Abstract

Peer-to-peer (P2P) computing is a technological concept which is applicable at varied levels of system architecture. Its typical characteristic is that there is symmetric communication and data exchange between the peers; each peer is both a client and a server. It is the basis for decentralized distributed computing. The concept is widely deployed in different contexts and no formal definition exists. Some of the benefits of P2P architecture include: improving scalability by avoiding dependency on centralized points; eliminating the need for costly infrastructure by enabling direct communication among peers; and enabling resource pooling. This paper presents an overview of the main characteristics of P2P systems and discusses the different application areas of P2P technology. Some issues and problems involved when deploying P2P products (systems and applications) are equally reviewed.

Keywords: peer, distributed computing, resource pooling, protocol, client, and server.

1 Introduction

The term ņpeer-to-peerorestation refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized fashion [1]. The resources encompass computing power, data (storage and content), network bandwidth, and presence (computers, human, and other resources). The critical function can be distributed computing, data/content sharing, communication and collaboration, or platform services. Decentralization may apply to algorithms, data, and meta-data, or to all of them. This does not prevent retaining centralization in some parts of the systems and applications if it meets the requirements. Typical P2P systems reside on the edge of the Internet or in ad-hoc networks. P2P is not a particular initiative, nor architecture or specific technology; it rather describes a set of concepts and mechanisms for decentralized distributed computing and direct peer-to-peer information and data interchange [2]. P2P mechanisms can be found at different levels of the system architecture. Most prominent at present are P2P applications for content and information exchange in the Internet such as Napster, Gnutella and eDonkey [1], [2], [3], [4].
Although the term peer-to-peer refers to a number of ideas and generic idea and mechanisms, the idea of peer-to-peer is applied in different contexts hence P2P systems do not necessarily have many characteristics in common; neither do they have to adhere to a determined set of attributes. A formal definition P2P computing does not exist. Infact, various definitions are being used by the P2P community. However, there are features many P2P systems share such as the lack of fixed hierarchical client-server relationship, the autonomy of the peering (sub)-systems, the fact that they have independent lifetimes, and the direct information exchange among peers. The peering structure allows resource sharing among participating entities. Conceptually, P2P computing is an alternative to the centralized and client-server models of computing, where there is typically a single or small cluster of servers and many clients. In its purest form, the P2P model has no concept of server; rather all participants are peers.

The P2P technology can help to reduce system costs and allows cost sharing by using existing infrastructure and bundling resources from different sites. Resource aggregation adds value beyond the mere accumulation of resources. P2P systems are for instance more resilient because of their distributed and non-hierarchical nature. It is pertinent to note that P2P technology and its potential are still being investigated. However, there is a clear indication that P2P approach can have considerable impact on applications and distributed systems. The concept
of P2P has been around for a long time. For example, it has been used in the Internet since its invention, in telephony, users talk to each other once the connection is established, in a computer networks the computers communicate peer-to-peer, and in games, such as Doom, players also interact directly [1]. However, the concept gained a larger visibility and became more widely known only with the advent of P2P applications. Therefore P2P is often associated with applications rather than with technical concepts.

This paper is organized as follows. In section 2, a number of characteristics that are conventionally attributed to P2P computing are discussed. In section 3, a general description of P2P systems categories is presented. Subsequently a few issues and problems related to the deployment of P2P technology are highlighted. Finally, in the last section, the paper is summarized and concluded.

2 Characteristics

This section discusses the often-cited features of P2P architecture namely: decentralization, scalability, anonymity, self-organization, cost-of-ownership, ad-hoc connectivity, performance, security, transparency, usability, fault-tolerance and interoperability. These features play a major role on the effectiveness and deployment of P2P systems and applications.

2.1 Decentralization

One main goal of decentralization is the emphasis on users' ownership and control of data and resources. In a fully decentralized system, all peers assume equal roles. This makes the implementation of the P2P models difficult in practice because there is no centralized server with a global view of all the peers in the network or the files they provide. This is the reason why many P2P file systems are built as hybrid approach as in the case of Napster, where there is a centralized directory of the files but the nodes download files directly from their peers.

In fully decentralized file systems such as Freenet and Gnutella, just finding the network becomes difficult. In Gnutella, for instance, new nodes must know the address of another Gnutella node or use a host list with known IP addresses of other peers. The node joins the network of peers by establishing connection with at least one peer currently in the network. Then, it can begin discovering other peers and cache their IP addresses locally. One way to categorize the autonomy of a P2P system is through the řpure P2Př versus řhybrid P2Př classification. This categorization has a direct effect on the self-organization and scalability of a system, as the pure systems can be loosely coupled to any infrastructure.
2.2 Scalability

An immediate benefit of decentralization is improved scalability. Scalability is limited by factors such as the amount of centralized operations (e.g., synchronization and coordination) that needs to be performed, the amount of state that needs to be maintained, the inherent parallelism an application exhibits, and the programming model that is used to represent the computation.

Napster attacked the scalability problem by having the peers directly download music files from the peers that possess the requested document. Consequently, Napster was able to scale up to over 6 million users at the peak of its service [1]. In contrast, SETI@home focuses on a task that is embarrassingly parallel. It harnesses the computer power that is available over the internet to analyze data collected from its telescopes with the goal of searching for extraterrestrial life forms. Systems like Avaki address scalability by providing a distributed object model. Ensuring good scalability should not jeopardize other desirable features, such as determinism and performance guarantees. To address this problem, hybrid P2P systems such as Napster, intentionally keep some amount of operations and files centralized.

Early P2P systems such as Gnutella and Freenet are ad-hoc in nature [3], [5], [6]. A peer has to "blindly" send its requests to many other peers, causing the rest of the peers to search for the requested document. This can cause the time to retrieve a document to be unbounded. In addition, searching may fail even when an object exists, making the behavior of the system non-deterministic.

Recent P2P systems, represented by CAN, Chord, Oceanstore, and PAST, dictate a consistent mapping between an object key and hosting node. Therefore, an object can always be retrieved as long as the hosting nodes can be reached. Nodes in these systems compose an overlay network. Each node only maintains information about a small number of other nodes in the system. This limits the amount of state that needs to be maintained, and hence increases scalability. The logical topology of the overlay provides some guarantees on the lookup cost. These systems are designed to scale to billions of users, millions of servers and over $10^{14}$ files [1].

In the future, as the bandwidth and computation power continue to grow, platforms will be able to take advantage of this power, which should become interesting to more applications. The net effect is that these architectures will enable more automated scaling, as much resources can be provided, the applications could scale.

2.3 Anonymity

An important goal of anonymity is to allow people to use systems without concern for legal or other ramifications. A further goal is to guarantee that censorship of digital content is not possible. The authors of Free Haven [1] identify the following forms of anonymity:
Author: A document’s author or creator cannot be identified.

Publisher: The person who published the document to the system cannot be identified

Reader: People who read or otherwise consume data cannot be identified

Server: Servers containing a document cannot be identified based on the document

Document: Servers do not know what documents they are storing

Query: A server cannot tell what document it is using to respond to user's query.

Basically, three different kinds of anonymity could be enforced between a communicating pair namely: sender anonymity hides the sender's identity; receiver anonymity hides the receiver's identity and mutual anonymity, whereby the identities of the sender and receiver are hidden from each other and other peers. Besides, there exists various degree of anonymity. A spectrum of anonymity degrees that cover absolute privacy, beyond suspicion, probable innocence and provably exposed are presented in the literature.

2.4 Self-organization

In cybernetics, self organization is defined as a process where the organization (constraint, redundancy) of a system spontaneously increases, i.e., without this increase being controlled by the environment or an encompassing or otherwise external system [7].

Self-organization is required in P2P systems because of scalability, fault tolerance, intermittent connection of resources, and cost of ownership. P2P systems can scale unpredictably in terms of the number of systems, number of users, and the load. It is very difficult to predict any one of them, requiring frequent re-configuration of centralized system. The significant level of scale results in an increased probability of failures, which requires self-maintenance and self-repair of the systems. Similar reasoning applies to intermittent disconnection; it is hard for any predefined configuration to remain intact over a long period of time. Adaptation is required to handle changes caused by peers connecting and disconnecting from the P2P systems. Finally, because it would be costly to have dedicated equipment and (or) people for managing such a dynamic environment, the management is distributed among the peers.

2.5 Cost of Ownership

One of the promises of P2P network is shared ownership. Shared ownership reduces the cost of owning the systems and the content, and the cost of maintaining them. This is applicable to all classes of P2P systems. It is probably most obvious in distributed computing. For instance, SETI@home is faster than the fastest supercomputer in the world, yet at only a fraction of its cost i.e 1%. The whole concept of Napster music sharing was based on each member contributing
to the pool of music files. Similar assumptions for peers are used in other file systems, such as OceanStore.

In P2P collaboration and communication systems, and in platforms, elimination of centralized computers for storing information also provides reduced ownership and maintenance costs. A similar approach is taken in wireless communication in the United States. A so-called 'parasitic grid' wireless movement, enables sharing of the existing home-installed 802.11b bandwidth among the users [8]). These networks compete with the companies installing wireless infrastructure at the fraction of the cost.

2.6 Ad-hoc Connectivity

The ad-hoc nature of connectivity has a strong effect on all classes of P2P systems. In distributed computing, the parallelized applications cannot be executed on all systems all of the time; some of the systems will be available all of the time, some will be available part of the time, and some will not be available at all. P2P systems and applications in distributed computing need to be aware of this ad-hoc nature and be able to handle systems joining and withdrawing from the pool of available P2P systems. While in traditional distributed systems, this was an exceptional event, it is regarded usual, in P2P systems.

In content sharing P2P systems and applications, users expect to be able to access content intermittently, subject to the connectivity of the content providers. In systems with higher guarantees, such as service-level agreements, the ad-hoc nature is reduced by redundant service providers, but the parts of the providers may still be unavailable.

In collaborative P2P systems and applications, the ad-hoc nature of connectivity is more evident. Collaborative users are increasingly expected to use mobile devices, making them more connected to internet and available for collaboration. To handle this situation, collaborative systems support transparent delay of communication to disconnected systems. This can be accomplished by having proxies delegated on networks to receive messages, or by having other sorts of relays on the sending system or somewhere in the network that will temporarily hold communication for an unavailable system.

Furthermore, not everything will be connected to the Internet. Even under this circumstance, ad-hoc groups of people should be able to form ad-hoc networks in order to collaborate. The supporting ad-hoc networking infrastructures, such as 802.11b, Bluetooth, and infrared, have only a limited radius of accessibility. Thus, both P2P systems and applications need to be designed to tolerate sudden disconnection and ad-hoc additions to group of peers.
2.7 Performance

Performance is a significant concern in P2P systems. P2P systems aim to improve performance by aggregating distributed storage capacity (e.g., Napster, Gnutella) and computing cycles (e.g., SETI@Home) of devices spread across a network. Owing to the decentralized nature of these models, performance is influenced by three types of resources namely: processing, storage, and networking. In particular, networking delays can be significant in wide area networks. Bandwidth is a major factor when a large number of messages are propagated in the network and large amounts of files are transferred among many peers. This limits the scalability of the system. Performance in this context does not put emphasis in the millisecond level, but rather tries to answer questions of how long it takes to retrieve a file or how much bandwidth will a query consume. The three key approaches for optimizing performance are: replication, caching, and intelligent routing.

2.8 Security

P2P systems share most of their security needs with common distributed systems: trust chains between peers and shared objects, session key exchange schemes, encryption, digital digests, and signatures. However, new security requirements appeared with P2P systems. Some of these requirements are multi-key encryption, sandboxing, digital rights management, reputation and accountability and firewalls.

2.9 Transparency and Usability

In distributed systems, transparency was traditionally associated with the ability to transparently connect distributed systems into a seamlessly local system. The primary form of transparency was location transparency, but other forms include transparency of access, concurrency, replication, failure, mobility, scaling, etc. Over time, some of the transparencies were further qualified, such as transparency for failure, by requiring distributed applications to be aware of failures, and addressing transparency on the Internet and Web.

Another form of transparency is related to security and mobility. Automatic and transparent authentication of users and delegation to user proxies can significantly simplify users' actions. Supporting mobile users and disconnection in particular, can enable users to work independently of whether and how they are connected to the Internet or intranets. A user can use P2P applications; as a user of services typically through web interfaces (e.g., content sharing, information gathering), wrapped around non-P2P applications typically on a P2P platform (e.g., Groove, .NET) or as locally installed P2P software (e.g., distributed computing screensavers and Napster)
2.10 Fault Tolerance

One of the primary goals of a P2P system is to avoid a central point of failure. Although most P2P systems (pure P2P) already do this, they nevertheless are faced with failures commonly associated with systems spanning multiple hosts and networks: disconnections/unreachability, partitions, and node failures. It would be desirable to continue active collaboration among the still connected peers in the presence of such failures. An example would be an application, such as genome@home executing a partitioned computation among connected peers. Will it be possible to continue the computation if one of the peers were to disappear due to a network link failure? If the disconnected peer were to reappear, could the completed results (generated during the standalone phase) be integrated into the ongoing computation? Questions like these would have to be addressed by P2P systems aiming to provide more than just "best effort" Internet service.

In the past, client-server disconnection has been studied for distributed systems that consider mobile clients e.g. Coda, and a common solution is to have application specific resolvers to handle any inconsistency on reconnection. Some current P2P systems such as Groove handle this by providing special nodes, called relays, that store any updates or communication temporarily until the destination (in this case another peer) reappears on the network. Others like Magi queue messages at the source until the presence of the destination peer is detected.

Another problem related to disconnection is non-availability of resources. This may occur either because the resource is unreachable due to network failure or because the peer hosting the resource has crashed (or gone offline). While the former may be resolved by routing around the failure and is already supported by the Internet, the latter requires more careful consideration. Replication of crucial resources helps alleviate the problem. P2P networks such as Napster and Gnutella represent systems having both a passive and an uncontrolled replication mechanism based solely on the file's popularity. Depending on the application running over these networks, it may be necessary to provide certain persistence guarantees. This requires a more active and reliable replication policy.

Anonymous publishing systems such as Freenet and Publius ensure availability by controlled replication. Oceanstore maintains a two-layered hierarchy of replicas and through monitoring of administrative domains avoids sending replicas to locations with highly correlated probability of failures. However, because a resource in the P2P system could be more than just a file - such as a proxy to the Internet, shared storage space, or shared computing power - the concepts of replicated file systems have to be extended to additional types of resources. Grid computing solutions such as Legion provide tolerance against node failures by restarting computations on different nodes.
A challenging aspect of P2P systems is that the system maintenance responsibility is completely distributed and needs to be addressed by each peer to ensure availability. This is quite different from client-server systems, where availability is a server-side responsibility.

### 2.11 Interoperability

Although many P2P systems already exist, there is still no support to enable these P2P systems to interoperate. Some of the requirements for interoperability include: How do systems determine that they can interoperate? How do systems communicate, e.g., what protocol should be used, such as sockets, messages, or HTTP? How do systems exchange requests and data, and execute tasks at the higher level, e.g., do they exchange files or search data? How do systems determine if they are compatible at the higher protocol levels, e.g., can one system rely on another to properly search for a piece of information? How do systems advertise and maintain the same level of security, Quality of Service (QoS), and reliability?

In the past, there were different ways to approach interoperability, such as standards IEEE, (e.g., IEEE standards for Ethernet, token ring, and wireless); common specification (e.g., Object Management Group) [11], common source code (e.g., OSF DCE); opensource (e.g., Linux); and de facto standards (e.g., Windows or Java).

In the P2P space, some efforts have been made towards improved interoperability, even though interoperability is still not supported. The P2P working Group [12] is an attempt to gather community of P2P developers together and establish common ground by writing reports and white papers that would enable common understanding among P2P developers. The P2P Working Group gathers developers from both ad-hoc communication systems and grid systems. The Grid Forum is a similar effort in the grid computing space. Both efforts represent an approach similar to OMG, in defining specifications and possibly reference implementations.

The JXTA effort [13] approaches interoperability as an open-source effort, by attempting to impose a de facto standard. A number of developers are invited to contribute to the common source tree with different pieces of functionality. Only a minimal underlying architecture is supported as a base, enabling other systems to contribute parts of their own implementations. A number of existing P2P systems have already been ported to the JXTA base.

### 3 Peer-to-Peer Systems

This section describes a coarse taxonomy of P2P systems. The four classes of P2P systems namely file and content sharing, collaboration support, distributed computing, and P2P platforms are discussed. Historical systems that predated the recent notion of P2P systems are discussed
3.1 File and Content Sharing

Among the systems that feature most prominently in the P2P domain are user applications running on top (or at the edge) of the Internet allowing a large group of users to interact and share resources. Most popular are file or content sharing applications such as Napster, Gnutella, Mojo Nation, eDonkey and Freenet. Napster was the first major system enabling direct exchange and sharing of content. While the actual exchange of content in Napster is between peers, the discovery of the peers, however, is highly centralized (i.e. it is stored in a central directory). Gnutella provides a purely distributed file sharing solution without a central node.

In the strict sense, Gnutella is not an application but a protocol used to search (for), and share files. To find content and other peers a user has to know the IP address of at least one other Gnutella node. A node issues a query for a file by sending it to all other nodes known to it. If a node cannot serve a request it can forward it to other nodes. The query travels the Gnutella network until the file has been found or its time-to-live has been reached. Effectively this discovery mechanism floods the network, which can cause scalability problems. Another issue in Gnutella is free riders, i.e. users do not contribute but just take content from other users. This has been confirmed in a study where measurements showed that 66% of the users do not offer content at all and 73% offer less than 11 files for download. Mojo Nation is a P2P content exchange application that introduces a virtual currency (so called Mojos) to counter free riders. This currency is also used as incentive to contribute resources (such as storage space and content). The peers in Mojo Nation can have different roles, i.e. Block Server (provides storage), Content Tracker (content search services), Publishing Agent (content publishing service), and Relay Service (forwarding service). The content is split into blocks and distributed throughout the Mojo Network. Hence, a block server only hosts part of the content but not the entire file.

Another popular file sharing system is eDonkey. The special feature of eDonkey is that it identifies the files using an MD4 based hash value and the file size. This method allows identifying files with the same content but slightly different names. It also enables the download of the content from different source files and hence increasing the net download rate.

Freenet is also a file/content sharing system. The primary goal of Freenet is to make its use completely anonymous. Neither the users requesting nor those placing files in Freenet can be identified. Further, an operator of a Freenet node is not able to determine what data is stored on its local disk. Freenet is completely decentralized and presents peer-to-peer in its purest form.

3.2 Collaboration

P2P collaboration and communication support is provided by systems such as Centrespan, Jabber, AIMster, Magi and Groove. These systems allow collaboration between users without the use of messaging server. The simplest systems also allow joint authoring of documents, graphics, slides, etc. Games using P2P technology are also regarded as collaborative application
since they also support interaction amongst users. Groove is a system that provides a variety of applications for communication, content sharing (files, images, and contact data), and collaboration (i.e., group calendaring, collaborative editing and drawing, and collaborative web browsing). Magi is a P2P infrastructure platform that supports the implementation of secure, cross-platform collaborative systems. The core services of Magi include a Communication Service, an Event Service, a Buddy Manager and an Access Controller. It relies on DNS (Domain Name System) as a directory for the IP addresses of the Magi instances.

In general, events and message exchange taking place in a P2P collaborative system are relayed instantly to all other members of the P2P group. Issues to be considered are fault tolerance and real-time constraints. The former is linked to reliable group communication whereas the latter refers to interaction constraints. Group and multi-peer communications research have actually addressed these problems in the mid 1990s.

Collaborative P2P systems share the problem of locating and addressing peers (respectively the P2P group) with all other P2P application areas. In contrast to others, however, communication here is mostly synchronous amongst an identifiable group although specific users might remain anonymous. Hence the addressing problem is related to addressing issues in group multicast communication.

3.3 Distributed Computing

A distributed system has been defined as a computer system in which several interconnected computers share the computing tasks assigned to the system as a single entity. Such systems are for instance clusters or the GRID. Generally, systems that aggregate resources from a number of networked computers to achieve better processing and scalability benefit most from using P2P technology. However, while the resource usage in this case is P2P, there is often a central instance that manages and coordinates the computational process. A prerequisite for distributed computing is that tasks can be split into sub-tasks, which can be processed independently from each other. Interaction between peer systems processing sub-tasks should be kept to a minimum. Suitable processes are Single Process-Multiple-Data and multiprogramming problems where a given job has to be run on many different input data sets. This mostly includes simulation and model validation tasks. Typical application area examples are physics, finance and biotechnology. Because specific applications have to be developed using very specific constraints, paradigms, and environments, their development cost is prohibitive to most users, and the scope remains limited to highly visible research domains, such as human genome project, alien seeking, cancer research, and weather model validation. One of the most popular distributed computing applications is SETI@home, a project that aims to find proof of extraterrestrial intelligence. Computers connected to the SETI@HOME system process data collected from a radio telescope. Their task is to find and identify any possible signals from intelligent populations outside the solar system. A database server controls the operations; the
peers operate effectively as clients that get their jobs from this server. Hence peers in the context of this project refer to computing systems that make their resources available to others but are still centrally controlled.

3.4 Platforms

There is an increasing trend away from native operating systems towards middleware platforms as application hosting environments such as Java Virtual Machine or Web browsers. These platforms provide a largely operating system independent development environment and also offer high-level functions to system developers. Some of these platforms provide P2P support and use P2P mechanisms. P2P components used in this context are for instance naming, communication, security and resource aggregation. The two most comprehensive platforms that currently support P2P are Sun's JXTA and Microsoft's .NET. JXTA can be considered as successor of Jini, an early attempt of Sun to introduce the concept of P2P platform. It provides an open general purpose network programming and computing platform for distributed applications running on a variety of devices such as handheld, set-top boxes, special household equipment, etc. It takes a layered approach and provides basic mechanisms as well as higher-level services. The three components associated with JXTA are JXTA Core, JXTA Services and JXTA applications. The services include discovery, authentication and management. A more high-level service is the Content Management Service that allows JXTA applications to share, retrieve and transfer content within peer groups. Further services are naming, routing and indexing. JXTA provides some applications such as the command line interface JXTA Shell, and Instant P2P which has chat capabilities.

The third major platform using P2P technology is MS .NET. The goal of .NET is to enable users of multiple devices ranging from relatively powerful PC to mobile devices to access web services using existing web standards and protocols such as XML(Extensible Markup Language), SOAP(Simple Object Access Protocol), UDDI(Universal Description Discovery and Integration protocol) and WSDL(Web Services Definition Language). A central design goal of .NET is the de-componentization and decentralization of distributed services. Service discovery is done through UDDI whereas the service itself can be accessed as web services. MS .NET is not a pure P2P platform but incorporates some of the principles and concepts of P2P computing in its design.

4 P2P Related Issues

P2P technology has a number of advantages over the client-server systems. They are extensible and inherently fault-tolerant. In the application context the potential of P2P systems to ensure user control, anonymity and privacy is important. This is possible since peering systems have a greater degree of autonomous control over their data and resources. However, there are a number
of issues that have not been resolved. There is for instance no global view at the system level and it is difficult to maintain a consistent system state. Some systems therefore have a central instance that manages the peers and impose policies. However central control might not be required. In the communication domain the idea of maintaining a common state in a peer group by applying policies that govern and co-ordinate the characteristic of P2P systems has been discussed. This concept could be explored further for P2P systems that require a consistent state. Policies can also be extended to cover performance and QoS parameters. Hence it would be ensured that a minimal quality is available making the system more reliable and deterministic.

Addressing, is a problem that affects P2P systems at all levels. At the communication layer, concepts such as multicast and any-cast support the communication among peers to some extent. The identity of individual peers does not necessarily have to be disclosed in this context. At the application level other concepts such as document routing are used to find content. The addressing problem depends on the context and further work is required to find the best solution for a particular addressing problem.

Protection and security is another area that requires further research. Current security mechanisms such as firewalls often inhibit the exchange between P2P systems. Nevertheless, new security mechanisms are required in the context of peer interaction, data exchange, and remote processing and program execution.

Issues at the application level are related to trust, responsibility and liability. P2P applications can ensure a high degree of privacy and anonymity. If such a system is not policed, anybody can do anything without the possibility to catch the offender. This also raises the question of who might be liable in the case of serious offences. Certain mechanisms are required to protect P2P applications from offending users. Concepts such as reputation mechanisms are being discussed in this context.

5 Summary and Conclusion

In this paper, an overview of P2P computing is presented. Among other things, the characteristic features of P2P systems, P2P systems categories, and issues related to P2P systems deployment are discussed. In the rest of this section, the description, importance and the prospect of P2P are summarized. Finally a concluding remark is given.

5.1 Final Note on P2P

P2P is many things to many people and it is not possible to come up with a simplified definition. It is a mind set, a model, an implementation choice, and property of a system or an environment.

A mind set: - As a mind set, P2P is a system and/or application that either takes advantage of resources at the edge of the system or supports direct interaction among its users. Such a systems
and /or application remains P2P regardless of its model or implementation. Examples include SETI@home, which is considered to have a client-server model, but displays the first mind set property, and Slashdot which enables the second mind set property, but really has a centralized implementation.

**A model:** - A system and/or application supporting the model presented in Figure 1(a) is P2P. In its purest form, P2P is represented by Gnutella, Freenet, and Groove. According to this, SETI@home does not have a P2P model, whereas Napster has a hybrid model. An implementation choice: - P2P systems and applications can be implemented in a P2P way such as JXTA or Magi. However, a non-P2P application can also be implemented in a P2P way. For example, application layer multicast can have a P2P implementation, and parts of the ORBs or DNS servers are also implemented in a P2P way.

**A property of a system or an environment:** - .NET as well as environments, such as small device sensor networks, may require P2P implementation solutions while not necessarily supporting a P2P application. P2P solutions may be required for scalability, performance, or simply because of the lack of any kind of infrastructure, making P2P the only way to communicate. This is similar to looking at P2P as an implementation choice, however in this case P2P is the forced implementation choice.

### 5.2 The Importance of P2P

As P2P becomes more mature, its future infrastructures will improve. There will be increased interoperability, most connections to the (Internet) world, and more robust software and hardware. Nevertheless, some inherent problems will remain. P2P will remain an important technology for the following reasons:-

Â Scalability will always be a problem at certain levels (network, system, and application), especially with global connectivity, much of it wireless. It will be hard to predict and guarantee all service-level agreements. P2P can contribute to each area.

Â Certain parts of the world will not be covered by (adequate) connectivity, requiring ad-hoc, decentralized groups to be formed. P2P is a well-suited option when there is a lack of infrastructure.

Â Certain configurations of systems and applications will inherently be P2P and will be amenable to P2P solutions.
5.3 P2P and the Future

There are at least three ways in which P2P may have impact in the future:

1. P2P algorithms probably have the greatest possibility of making impact. As the world becomes increasingly decentralized and connected, there will be a growing need for P2P algorithms to overcome the scalability, anonymity, and connectivity problems.

2. P2P applications are the next most likely to succeed in the future. Examples, such as Napster are a convincing proof of such a possibility.

3. P2P platforms are the third possible scenario for P2P. Platforms such as JXTA may be widely adopted, in which case many other P2P systems can also gain wide adoptions.

5.4 Concluding Remark

P2P is an elegant technology that has already found its way into existing investments, products and research. It will remain an effective panacea for certain inherent problems of distributed systems. It may not be the only solution and may not be fit for all problems. However, it will remain a strong option for implementing scalability, anonymity, and fault resilience requirements. P2P algorithms, applications, and platforms have an opportunity for deployment in the future. From the economy viewpoint, cost of ownership may be the driving force for P2P. The strong presence of P2P products indicates that P2P is not only an attractive research technology but also a promising product base.

References


[16] Rowstron, A. and Druschel, P. Storage management and caching in PAST, large-scale, persistent, peer-to-peer storage utility in Proceedings of SOSP, 2001, pp.188-201.


