Peer-to-Peer Technology

Lindgren, Per
Olsson, Christian

Chalmers University of Technology
Göteborg, 2006-04-30

Table of Contents

INTRODUCTION **2**

HISTORY **2**

PEER-TO-PEER CHARACTERISTICS **3**

- THE ASPECT OF DECENTRALIZATION 3
- A SELF RE-ORGANIZING NETWORK 4
- SCALABILITY 4
- PERFORMANCE AT LOW COST 4
- THE FREEDOM OF ANONYMITY 5
- A NEW TYPE OF FAILURES 5
- RELIABILITY BY REPUTATION 5

PEER-TO-PEER COMPONENTS AND ALGORITHMS **6**

- COMPONENTS 6
  - *Connection* *6*
  - *Group Management* *6*
  - *Robustness* *6*
  - *Class-Specific* *7*
  - *Application-Specific* *7*
- ALGORITHMS 7
  - *Centralized Directory Model* *7*
  - *Flooded Request Model* *8*
  - *Document Routing Model* *8*

PEER-TO-PEER SYSTEMS **9**

- DISTRIBUTED COMPUTING 9
- FILE SHARING 9
  - *Application Examples* *10*
- COLLABORATIVE SYSTEMS 10

REFERENCES **11**

Introduction

Peer-to-peer technology has increasingly become an important technique in a wide variety of areas, such as distributed and collaborative computing with special regards to the Internet. Both industry and academic interest have been put into development of new areas for peer-to-peer technology.

The definition of peer-to-peer technology is rather loose and there are many different views of how to best describe it. In general; peer-to-peer is about sharing: giving to and obtaining from the peer community. A peer shares some resources, and in return, it gets other resources. Peer-to-peer could also be about only giving resources, like sharing computational power to help solve a computational-heavy task.

Peer-to-peer is also a way of thinking when trying to decentralize systems or algorithms. The world gets more and more connected and distributed. Based on the number of users around the world, centralized systems have a difficult time serving the amount of data needed; thinking in peer-to-peer terms solves future load-balancing issues and scales very good in rapidly growing user-based systems.

Traditionally, a “peer” is defined as “like each other”, hence; a peer-to-peer system must then imply a system where independent peers depend on other independent peers. We say that a peer is independent, since it is not fully controlled by one user or system. Because of this fact, a peer cannot fully trust another peer, or rely on the behavior of other peers. It is rather a “brick” in a bigger “whole”.

Peer-to-peer is a technology along the lines of centralized and server/client-models; where in those systems there is typically one server and many clients involved. Peer-to-peer got in its definition no need to involve a server in the process; instead all participants are individual peers. The differences between these systems are far more complex than this; but conceptually this is the fundamental difference.

Peer-to-peer technology is most widely known for file-sharing applications, where the most famous example is the Napster application. There are many predecessors to Napster which is some of the most downloaded software applications on the Internet up to date.

This report will try to explain the main advantages of peer-to-peer technology, characteristics and in depth explanations on the techniques used in different peer-to-peer applications.

History

Peer-to-peer technology arose from the research and development of decentralizing systems. As Internet and users to systems began to grow, there was a need to improve current systems to let them scale better and broader over the user-group. This thinking was however held back by the fact that centralized systems had a clear advantage concerning the management of the systems and security-measures.

Peer-to-peer however began a new trend all across the world with famous applications; such as Napster, Gnutella and others alike. These programs had such an impact that the words “peer-to-peer technology” today is associated almost solely with file-sharing applications. However, these were not the first programs to take advantage of the peer-to-peer paradigm.

Two early decentralized networks called USENET and FidoNet were using peer-to-peer thinking early before Napster and such programs were invented.

USENET provides newsgroup-service in a distributed manner. USENET was started as early as 1979. The first version of it was the work of two graduate students Tom Truscott and Jim Ellis. At this time information wasn’t based “on-demand” as it is literally forced to be today. At this time, information was transferred in a slow and time-consuming manner; which usually resulted in downloads during the night when the distance-rates over the phone-line were the lowest. To accommodate this, a highly decentralized structure of USENET was developed.

FidoNet is also a decentralized system for exchanging messages. It was created in 1984 by Tom Jennings.

Both of these systems overcome the obvious problem of scalability among its users; it also introduced security-measures within the new decentralized manner, and both of these systems are still functioning and are still being used.

Advancing a few year and we reach the stepping-stone to peer-to-peer applications, namely Napster. Napster was created in 1999 by an 18-year old student, Shawn Fanning. Shawn was frustrated about how hard it was to swap and share digital music online. He however believed that there were many people connected that had the music-files stored on their hard-drives, just waiting to be copied. Shawn then focused on creating a program which would allow people to swap and share files with each other. Napster was born. Napster was an immediate success, and over a very short span of time, users were sharing millions of files with each other world wide. The key to the success was the fact that Napster brought free peer-to-peer technology to every user around the world. With this new availability to share and copy audio, video and other media-formats, the media-industry was not as happy about Napster as its user base. On April 13th 2000, Napster was sued by the rock band Metallica and a lengthy process began. In September 2001, an agreement was made that forced Napster to pay music and songwriters 26 million dollars.

The enormous media-coverage of this new phenomenon however had a tremendous impact of the new technology at hand. Many predecessors followed Napster and spawned several new applications that took advantage of the ideas from Napster.

Peer-to-Peer characteristics

The aspect of decentralization

One of the most important characteristic of the peer-to-peer model is the decentralization aspect. Centralized systems usually carry a server/client paradigm which relies heavily on the centralized system to help carry the load of computation needed for all the connected clients.

While this type of system works great for many applications and has a very robust way of handling with security-issues, it introduces inefficiency with several bottlenecks and wasted resources. Large-scale centralized systems are often very costly, and there is a limit on the load a centralized system can handle.

The most powerful decentralized peer-to-peer systems require all participants to be equal and handle the distributed load. This is however very hard to accomplish since it is hard to distribute the information equally when there is no centralized authority that has a global view of the different peers. Most peer-to-peer implementations therefore rely on a hybrid of centralization and decentralization, where there is at least one peer that has a more global view of the connected peers; while the direct transfer of information is handled peer-to-peer.

A Self Re-organizing Network

The entire network of peers in a peer-to-peer environment is constantly self-organizing and changes the form of the network based on number of peers and amount of information. A peer-to-peer network must always handle the rapid change in peers; as peers regularly "come-and-go". Information must be reachable from every other peer, nodes and so called super-nodes (in hybrid peer-to-peer systems) must constantly be re-organized to handle the changes. As opposed to centralized systems where managers need to re-organize systems to handle changes, in peer-to-peer technology, this task is fully left to the peers involved.

Scalability

Scalability is a major aspect of peer-to-peer technology; this is where the concept of peer-to-peer technology really shines. Napster could at its peak handle up to 6 million users world-wide, and there were no stopping there. The main advantage of peer-to-peer is that every peer only needs to know about a small number of nodes in the system which limits the amount of state that needs to be maintained and therefore increase scalability. A calculation of lookup-cost-restrictions among the peers for information reveals that peer-to-peer systems can scale to billions of simultaneous users.

Performance at Low Cost

The cost involved in maintaining and "owning" a peer-to-peer system is interesting; especially in regards to traditional distributed systems. Large-scale computational peer-to-peer software can handle more computational power than the fastest super-computer up to date; but at only a fraction of the cost. Hence, the cost related to the power of a peer-to-peer system is very cheap. Maintenance-cost is also a fraction of the cost in comparison to centralized systems, since information is stored among the different peers.

The cost of maintaining a peer-to-peer system is in regards to one of the most beneficial parts of peer-to-peer systems, namely the performance. Performance is usually affected by three different resources; processing, storage and networking. In regular centralized systems, bottlenecks in network capacity and network-delays can have a significant impact of the performance of a system. The decentralized nature of a peer-to-peer system better utilizes bandwidth and distributes resources among the peers. In order to try to reduce the impact of network congestion in peer-to-peer systems, the use of centralized information is limited and information is kept redundant to try achieving improved performance of the system. Other important techniques used to achieve optimized performance includes replication, caching and intelligent routing.

Replication puts copies of the content on nodes closer to the recipient, which reduces the connection distance between the sender and receiver. Problem with this is handling changes in the content and propagating of the newer and changed versions.

Caching can dramatically reduce the path-lengths to retrieve the information/object, and also the number of messages that is needed to communicate between the connected peers. A big problem in peer-to-peer technology involves latency-issues between the different peers; and how that can serve as a severe bottleneck in the system. Locally stored cache in the peers between the sender and receiver can help reduce latency-issues when retrieving objects or information.

Smart techniques to build connections between peers are a major research-area within peer-to-peer technology. Intelligent routing can "make or break" the performance of a peer-to-peer system. This could be putting peers with similar interests closer to each other or let "reliable" peers serve as organizing nodes and interconnect less reliant nodes. This can severely reduce the number of messages that is needed to be sent within a system to reach a desired result, hence increasing the performance of the system.

The Freedom of Anonymity

A well-debated characteristic of peer-to-peer systems is anonymity. Many peer-to-peer systems benefit from the fact that the user is using the system without knowing from where the information is gathered. In many cases the information could come from many different peers and only re-assembled at the user's computer. While some industries fight hard to prohibit this aspect, others propose this to be a form of "freedom of speech". Information can freely flow and be spread without explicitly revealing its source. This has many legal aspects, since information can be published and available without the author's consent or approval.

A New Type of Failures

One other strong aspect of peer-to-peer thinking is to avoid central point of failures. Since peer-to-peer technology tries to limit the use of centralized information; there is instead a strong need to avoid failures caused by failing nodes or peers, disconnections or unreachability. In order to avoid failures because of an unreachable host, replication of content, spanning over several nodes is a technique used to not interrupt the process; should a peer disconnect.

Reliability by Reputation

Peer-to-peer systems share many security-concepts with traditional distributed programming schemes, such as trust chains between peers, session key exchange, encryption and signatures. However, instead of "trust" between the peers, peer-to-peer technology often uses "reputation" among the peers to determine the trust. The more reliable a node is, the more it can be incorporated in the system. Some systems let peers with a reliable "up-time" benefit from this co-operation by granting it more bandwidth and other benefits within the system, hence; a peer must build a sense of "reputation" to gain benefits from the system.

Peer-to-Peer Components and Algorithms

Components

The components of a peer-to-peer system usually consist of five layers which consist of one or more blocks. The layers that will be discussed in this section are:

- Connection
- Group Management
- Robustness
- Class-Specific
- Application-Specific

Connection

As a base we have the communication layer which handles the communication between the peers. Handling the communication is one of the biggest challenges in peer-to-peer communication since it has to consider all types of Internet connections, such as dial-up connections, wireless connections and fast optical connections.

Group Management

Just above the communication block we have the group management layer. The group management layer handles discovery of other peers in the community and location and routing between them. The discovery algorithms can use several different approaches, such as using a centralized directory or using the communication range for mobile or wireless devices. The location and routing algorithms basically try to optimize the path of a message traveling from one peer to another.

Robustness

Above the management layer is the robustness layer. The robustness layer handles areas including security, resource aggregation and reliability. Security is a big issue in peer-to-peer communication and a big challenge since the peers will act as both a client and a server. Running a server with the same security parameters as for the client can have huge impact on the security of the system.

The basic idea of the peer-to-peer model is for the interacting peers to aggregate resources available on their systems. To classify the architecture of the peer-to-peer resource aggregation is not trivial since the resource can be of many different types.

To handle the reliability problem in peer-to-peer networks one can use many different solutions depending on what the task at hand is. For instance, if the task is to perform a heavy computation and if a peer goes down the task can be restarted on a different peer or it can be initially started on several peers simultaneously. In a file sharing network, data can be replicated across many peers.

Class-Specific

In the class-specific layer there are four blocks, scheduling, meta-data, messaging and management. The previous layers can be applied to almost every application using peer-to-peer technology but here it is far more application specific. The scheduling block is used for computational-intensive applications. Computational-intensive tasks are broken down to pieces and are solved in parts on different peers. Meta-data is used for content and file management applications where data is stored across the peers. Messaging is widely used wherever messages need to be sent between the peers. Management is used to control the underlying structure.

Application-Specific

This layer has three blocks; tools, applications and services. All these implement application specific functionality and use the underlying class-specific layer where it is needed.

Algorithms

In this section we will discuss three common peer-to-peer algorithms for finding specific information in a peer-to-peer network. The algorithms we will look closer at is

- Centralized directory model
- Flooded request model
- Document routing model

Centralized Directory Model

The peers of the peer-to-peer community connect to a central directory where they can publish information about the content they will offer to others. When the central directory gets a search request from a peer it will match the request with the peer in the directory and return the result. The best peer could be the closest (though this could be hard to determine), the fastest, cheapest or the most available. When a peer has been selected the transaction will follow directly between the two peers.

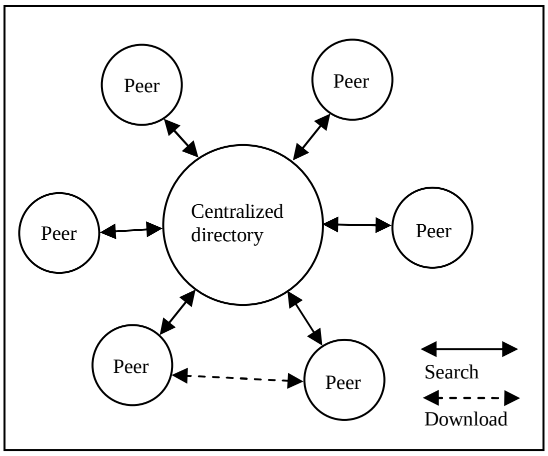


Figure 1

This algorithm requires a central server which is a drawback in peer-to-peer systems. It is a single point of failure and it can produce scalability problems. However, history shows that this model is quite strong and efficient even in larger systems.

Flooded Request Model

Unlike the centralized directory model this model is a pure peer-to-peer model without any central parts and without any advertisement of available resources. Instead, the flooded requests model floods each search request on the network, much like a broadcast, where each peer forwards the request to its directly connected neighbors until the request is answered or a preset maximum number of flooding steps has been reached.

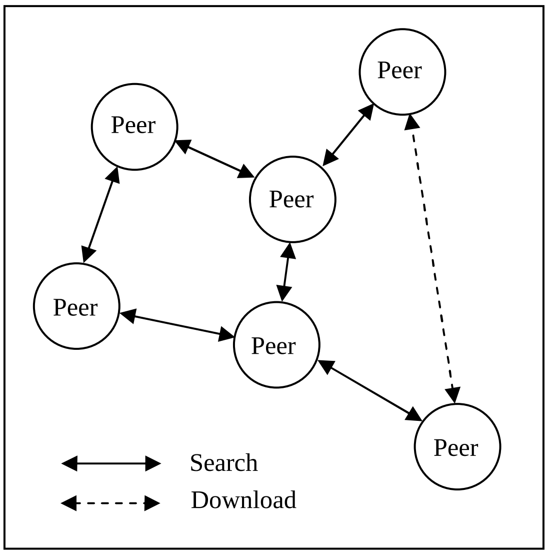
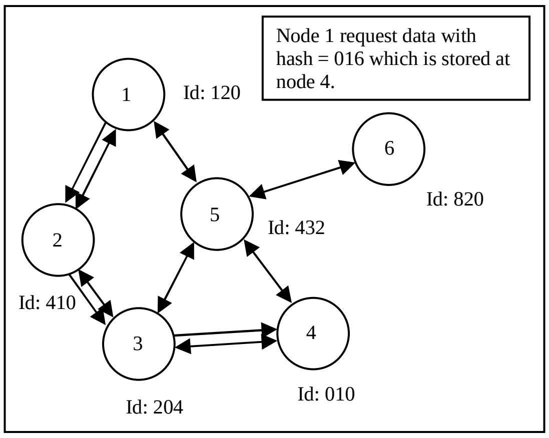


Figure 2

The obvious drawback of this model is that it requires a lot of bandwidth and that it therefore scales badly. However, in small networks it has been proved to be very efficient. To improve scalability, development has been made to the model. Some improvements involve super-peers that concentrate lots of the search requests to the super-peers. One may also consider caching the requests to improve performance.

Document Routing Model

The document routing model uses a different approach that has been used more recently. When a peer connects to the community it is assigned a random ID and each peer also knows a number of other peers. When data is stored or shared on a system using this model, an ID is assigned to the data based on a hash of the data in question. Each peer will then forward the data to the peer with the same ID (or the one closest to it) that it knows of. This is repeated until it reaches the peer with the same ID or the closest peer ID is the current peer's ID. During this process each peer that is involved in the routing also keeps a copy of the data. When a peer requests the data from the system, the request will be forwarded to the peer with the ID that is closest to the ID of the requested data. This is repeated until a copy of the data is found and the data will be transferred back to the requester the same way. In this transfer process, each peer that is involved will store a copy of the data as it is transferred.


Node 1 request data with hash = 016 which is stored at node 4.
1
Id: 120
6
Id: 820
5
Id: 432
2
Id: 410
3
Id: 204
4
Id: 010
Figure 3

This approach has several advantages and drawbacks. The biggest advantage is that this model scales very well. It is very efficient in larger, even global, communities. However, there are some drawbacks. The biggest one is that it is more complex to perform a search than with the flooded request model since the requesting peer must know the hash of the requested data.

Several algorithms have been implemented that make use of the document routing model. They all differ but the goals of each are quite similar. All of them try to reduce the number of hops that must be taken to locate the data and to reduce the amount of routing information that must be stored at each peer.

Peer-to-Peer Systems

In this section we will discuss three categories of usage of peer-to-peer systems. The categories to be presented are

- Distributed computing
- File sharing
- Collaborative systems

Distributed Computing

To use peer-to-peer systems in distributed computing has been a success and has been in use for quite some time. Successful projects have shown that it is possible to get high performance by using solely standard machines. The main focus right now is to use the idle time of internet connected standard computers, to let them perform some calculations while not being used. This requires a large number of Internet connected standard computers that run specific client software and a centralized controller. The central controller is responsible for splitting the problem into smaller parts and to process the results. To make the performance optimal the task to be solved need to be very large or very hard. To solve it, the client software on the standard computers fetches a small part of the task, solve it and report the result to a results database.

Unfortunately, the tasks have to be of the kind that it is possible to split them into smaller individual parts that do not require communication between the peers. This rules out super-computing like processing such as linear algebra problems or matrix computations. Distributed computing is nowadays instead used for problems that need to be processed with a high number of different input parameters, such as simulations, validations and biotechnical research.

File Sharing

File sharing is probably the area where peer-to-peer systems have had the most success. File sharing applications like Napster where among the first widely used applications that use peer-to-peer technology. The benefits of using peer-to-peer instead of traditional models are mainly the lower network bandwidth consumption, security and search capabilities that peer-to-peer offer. File sharing applications use one of the models described in the previous section (centralized directory, flooded request, document routing), but there also exist variations to these models. It is also important to realize that these models were based on the assumption that the files that are transferred are small, like a document or a picture, but now the trend is leaning towards larger file transfers such as music, movies and software. To deal with this situation, several modifications and additions have been made to the models. Software makes more intelligent decisions regarding where to download from and which way should it be transferred.

Application Examples

Napster was the first peer-to-peer file sharing application and was the application that started the whole peer-to-peer file sharing revolution. Napster used a centralized directory model to store information about shared music files on the peers. When a user connects, that users shared files will be added to the directory and when the user disconnects, the entries will be removed. Search requests are given to the centralized directory which will answer with the peers that are in possession of the requested file. However, the actual file transfers will not go through the centralized server. Instead the files will be transferred directly between the peers.

Morpheus is a file sharing application that is similar to Napster but the developers has made some improvements where the Napster model was weak. Instead of only searching for music, Morpheus can search for all media files. The results of the search are also easier to overlook. The actual file transfer mechanism has also been improved in several aspects. Morpheus is able to handle broken connections as well as increasing the download speed by using multiple sources.

File sharing application Kazaa was among the first to introduce supernodes. The protocol behind Kazaa is called FastTrack. Supernodes are fast nodes on fast connections and these nodes are assigned automatically. The supernodes are responsible for keeping track of local nodes contents and handle searches. Kazaa uses an intelligent download system that automatically chooses the best connection and uses multiple sources. To make sure that the downloaded file segments all match, Kazaa uses MD5 hashes. Unfortunately, the company behind Kazaa decided that Kazaa should be financed by advertisements and is therefore bundled with adware/spyware. Kazaa was blacklisted in March 2006 by the organization www.stopbadware.org.

Collaborative Systems

The idea of collaborative peer-to-peer systems is to allow collaboration between users on the application level. The span of the type application is large, ranging from applications of enjoyment such as instant messaging, chat programs and online games to more business oriented applications. Most collaborative systems are based on events. That is, each time an event occur at a peer, that will send an update message to all the other peers in the group and then update the application’s view (can be done either before or after an acknowledgement has been received).

Unfortunately, there are some difficulties involved in implementing collaborative systems. The biggest challenges are location, fault tolerance and real time constrains.

The location problem involves solving the location of other peers. The easiest way to solve this is to keep a centralized directory that each peer can consult. This is the most common solution but variants exist for smaller groups.

The fault tolerant problem is another challenge. In many applications it is essential that all messages sent is received properly by all peers. In some systems, the order of the received messages is also important. Most systems have solved this problem by queuing up the messages that could not be sent. These messages will then be delivered when the peer comes back online.

The real time constraints are harder to resolve since we cannot know how long it will take a message to be delivered (or if it has been lost on the way and will not be delivered at all). How to deal with this is up to each application. In early implementations, applications like network games didn’t update the screen until all other peers had acknowledged the messages. In small groups that were located on the same local network this solution was sufficient but with longer distances and larger groups it fails. Newer applications have more of a client-server solution for communication.

References

KaZaa, http://www.kazaa.com/.

MILOJICIC, D. KALOGAREAKI, V. LUKOSE, R. NAGARAJA, K. PRUYNE, J. RICHARD, B. ROLLINS, S. XU, Z. 2002. Peer-to-Peer Computing, HP Laboratories Palo Alto.

MORPHEUS, The Morpheus P2P homepage, http://morpheus.com/.

NAPSTER, The Napster Homepage, http://www.napster.com/.

STOPBADWARE, StopBadWare.org, http://www.stopbadware.org/.

SUNDSTED, T. 2001. The Practice of Peer-to-peer computing: Introduction and History, PointFire Inc, http://www-128.ibm.com/developerworks/java/library/j-p2p/.