
Peer-to-Peer Networking

Configuring Issues and Distributed Processing

by Odd Arild Skaflestad and Nina Kaurel

PREFACE This article is written in relation to the subject SIE50AC, *Self Configuring Systems*, a colloquium at NTNU autumn 2001 supervised by Professor Rolv Bræk. This article serves as a final assignment. The article is based upon a SIE50AC lecture given on the same topic.

Abstract

Peer-to-peer networking has become the hottest new thing in networking, and a lot of companies have been tempted to give this new networking paradigm a try. However, few solutions satisfy the actual peer-to-peer definition, and hybrid peer-to-peer solutions dominates the market. Central elements are included to control management and configuration. Specially configuring issues are discussed in this article, but also some elements of distributed processing is included.

Introduction

Inspired by the questions raised through this semesters colloquium and the following discussion of what is truly self configuration of a system and what is not, it is interesting to look into the concept of peer-to peer networking and study how the configuration of these networks is done. The popularity of peer-to-peer networks and networks that bear strong similarity with peer-to-peer networks is growing. Networks and applications aiming to solve tasks such as file sharing, instant messaging and sharing of resources like cpu and storage seems to be a good candidate for more rational networking. For instance, lending free capacity on a computer to ideal organizations seems to be a smart way of processing enormous amounts of data. The Ants concept includes some of the sharing of cpu and distributed processing properties. However, there is a potential down side of the matter. Serious security issues have to be solved when lending capacity or files to other and unknown peers, and legal issues when sharing files that are copyrighted has already been dis-

cussed as a major drawback of the peer-to-peer networking concept.

“To peer or not to peer”

The definition of a pure peer-to-peer network [1] is stated to be a network where no kind of centralized router is present, and every peer node has equal rights. As an example, in the pure form of peer-to-peer network there is no centralized server for routing. This means that there has to be some kind of distributed routing mechanism in the peers which supports the peer-to-peer network. This might be the biggest problem for peer-to-peer networking to overcome, even bigger if the peer-to-peer network is to scale up to a global user mass. Generally peer-to-peer applications have to deal with centralized services as authorization and certificates to become of use in a global environment. Hence, following this strict definition for truly being a pure peer-to-peer network, many of today's solutions and applications would find the answer to be “not to peer”.

Routing Issues

One idea of how to make the routing in a pure peer-to-peer network is based on a distributed catalogue structure [1]. This demands dynamic measures for distributing the load between peers when it comes to lookups and storage (if only one peer has the whole catalogue, then it is the same as a centralized naming server). It also demands some kind of pre-knowledge of where to search. See FIGURE 1

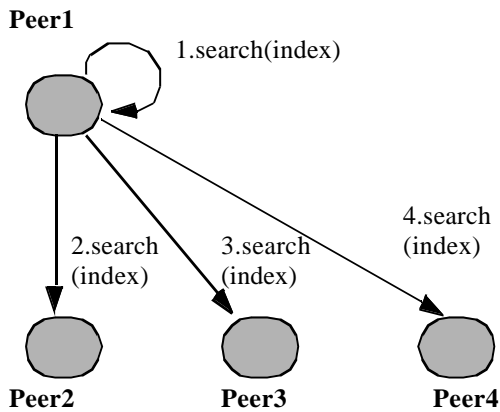


FIGURE 1 Distributed Catalogue Search

Routing seems to be a common problem when dealing with peer-to-peer networks. How can a peer gather sufficient pre knowledge, and how is it possible to be self configuring, when a certain amount of pre knowledge has to exist? The second suggestion of peer-to-peer routing solution[1] is a “broadcast/flooding” technique.

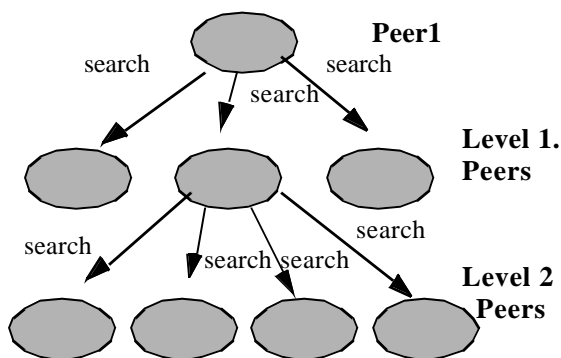


FIGURE 2 Message Relay

To avoid propagating the network and its peers with too much traffic, the search messages are restricted by a “limited horizon” (FIGURE 2 shows a 2 level horizon). This corresponds to the IP package hop limit. The problem of needing knowledge of the first level peers is present in this routing solution too. It might be possible to solve the routing problem with random broadcast messages on the network or by air (blue-tooth, IR), but there will be no guarantee of the presence of a peer, allowing the searching peer to connect. In the Internet world a DHCP offers a dynamic pool of available addresses. DHCP

has however a centralized design, maintaining a central address pool. Peer-to-peer networks are in its pure definition implemented by for example some ad-hock networks. The insecurity of the presence of a fellow peer will diminish the system operability. And reliable and critical applications are not helped with a unreliable access network of this kind. Hence, the notion of peer-to-peer has been extended to cover a range of protocols and solution that does not fully satisfy the pure peer-to-peer definition.

Hybrid peer-to-peer networks.

To be able to offer a consistent connection, many of the peer-to-peer protocols have introduced a central element in the peer structure. This central element may be introduced as a static routing table, or as a dynamic combined group manager and routing table. However, to be consistent with the idea of peer-to-peer networking, the management functions should be assigned to the peers. A central routing table with no further functionality than routing is accepted as a necessary element in hybrid peer-to-peer networks, and some of these hybrid solutions are described in more detail in this article.

Hybrid Routing Issues

The central router in a hybrid peer-to-peer network may be configured to play different roles. One of two typical configurations[2] are router as a router in a traditional sense (e.g. Internet routers), while the other solution is more of a look up service (e.g. DNS in Internet).

Model 2 in FIGURE 3 is rated as the most peer-to-peer like solution. The router does not forward messages as a router in the traditional sense, but contains an address resolution table to assist peers in finding fellow peers. The central server has been downgraded to a simple lookup service, and all communication is not directed through the central router (if it can be called a router). Since all peer traffic is not directed through the router, congestion problems are avoided through the relatively small amount of processing when performing lookup [2]. It is possible to distribute this address resolution table into a distributed catalogue, and hence be able to define the system as a pure peer-to-peer protocol. The reoccurring problem will however be; how gather the sufficient pre-knowledge to register and communicate with other peers, and

at the same time be guaranteed reliable communications?

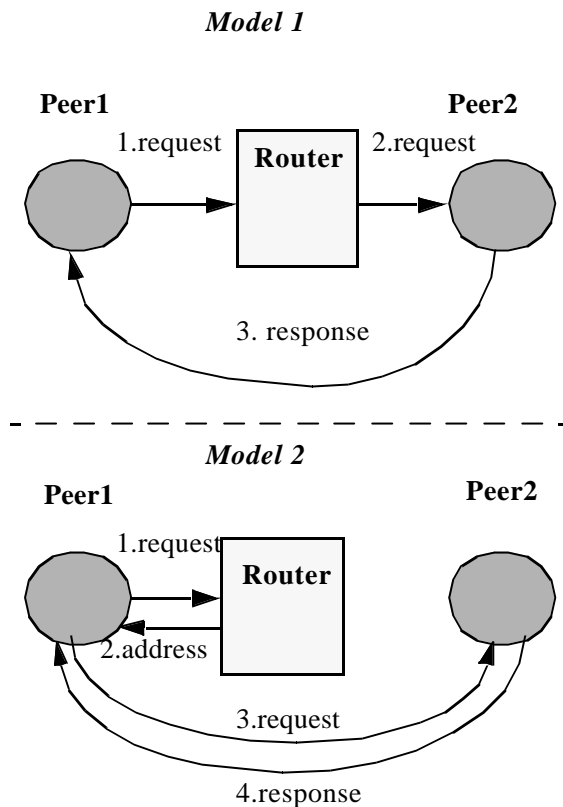


FIGURE 3 Routers in Hybrid Peer-to-Peer networks

Peer-to-Peer usage

The concept of peer-to-peer computing is not a new idea. Asynchronous messaging was used in the signaling process of the first telephone networks, signaling between equal and independent peers (telephones). Examples of radio relay between radio antennas is an example of peers working together (sharing resources). The computer world has more and less been dominated by the synchronous function call paradigm, and the integration of the two worlds in the Internet was based on the client/server paradigm. However, the peer-to-peer notion is rediscovered by the integrated computing world, which makes use of some of the advantages of peer-to-peer networking [3]. The client/server programming paradigm will probably not be rejected since it is a fundamental foundation of the existing infrastructure. There are, however, reasons to believe in a hybrid service combining both client/server and peer-to-peer programming tech-

niques. Major companies offering programming tools and commercial software products (Sun, Microsoft) have seen the value in peer-to-peer computing, and tries to integrate the concept into their solutions. However, the companies that have got the doubtful honour of (re)introducing peer-to-peer networking have been “small” companies, that found a niche in file sharing. The product that jump-started the peer-to-peer programming paradigm, Napster[4], has been sentenced for copyright piracy.

The next sections introduce three different parts of the peer-to-peer programming world (Napster, Gnutella, JXTA) with a special weight on how the three solutions solve the configuring problem stated earlier in this article.

Napster

Napster seems in the first place to be nothing like a peer-to-peer networking application. The Napster protocol is composed by the traditional part in data communication, client and server. The reason that Napster is introduced to be the originator of the peer-to-peer paradigm, is that it is the first service taking advantage of the enormous amounts of free storage placed in the Internet clients. Assuming an ordinary computer with a 100 Mhz processor and 100Mb of storage space, and assuming that there are 100 million PC connected as clients on the Internet, this adds up to a total capacity of ten thousand terabytes of storage, and ten billions Mhz processing power [4]. This capacity was unused until Napster came around. The problem was however that most clients did not have a static IP address. This was a result of the ISP’s dynamic IP address assignment, which made client address resolution through DNS lookup impossible. In the strictest sense, servers in the Internet communicate as peers, since network servers have static IP addresses. Napster took advantage of a DNS work around introduced by ICQ (Peer-to-Peer Instant messaging protocol) in 1996[4], introducing a proprietary protocol that made it possible to bypass the DNS problem, and update the dynamic client IP addresses at real time. The Napster protocol works on top of the existing Internet, and utilizes a proprietary naming service, linking the dynamic client user addresses to a specific Napster name. To be able to maintain the specific Napster naming technique, the Napster server has to be included in every network transaction. The Napster server offers a

naming server, a search engine (for MP3 files), and the Napster client application using the services offered by the server.

Configuration

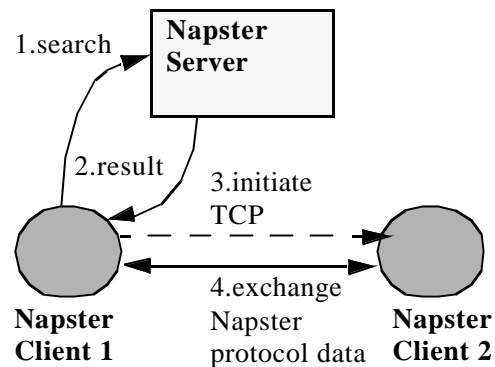
The Napster client and the Napster server is configured as any other client/server application over a TCP/IP connection. The downloaded Napster client has a built in DNS entry for the Napster server (for example www.napster.com). The Napster server is usually configured to listen to ports 8888 or 7777[6]. When the TCP three way handshake is successfully completed, the Napster client will start sending the proprietary Napster protocol when communicating with the server.

<length><type><data>

FIGURE 4 The Napster Protocol Format

The Napster protocol defines a “length” field, stating exactly how much data the message contains. The Napster protocol [6] also includes a “type” field, maintained by a proprietary type index list used in the Napster client and server. The “data” field contains the message data. The actual download and sharing of files does not include the server in the message transaction. When using the Napster search engine, the result will return the current IP address of a client that possesses the file the client asked for. A TCP connection will be initiated between the two clients. In this case, the clients act as peers, and are able to act as both client and server. If the file possessing Napster client is screened behind a firewall, a message is sent through the Napster server to this client, which in turn will tell the client to initiate the TCP connection. The Napster protocol data is sent on the established TCP connection. The Napster server will receive status messages from the Napster client and thus play an important role in the Napster service. The active part of the Napster server is one of the main issues when discussing the legal issues of sharing MP3 files.

Open Download



Firewalled Download

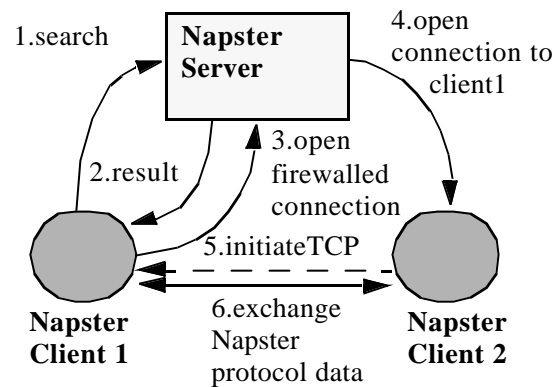


FIGURE 5 Napster File Download

Legal Issues

A short note concerning Napster’s legal problems has to be included in this article. Napster maintain that they have just invented an application, a search engine and a special naming service [8]. Separately neither of this is illegal. Napster says that it is the users responsibility to be aware of what they share. Napster offers central registration and caching of client file names, but the actual exchanged content is never passed through the central Napster server. The music industry, being the main opponent to Napster and MP3 files sharing, see no interest and value in suing single users for sharing copyrighted files since it would lead to a myriad of expensive law suites (At its peek, Napster had 70 million subscribers[7]). By appointing Napster as the distributor and organizer of illegal music sharing, the music companies had nominated a

single foe. The RIAA (Recording Institute Association of America) has been pulling Napster in to court at several occasions [8]. Some ISPs have also installed Napster filters to exclude Napster clients using their access net. However, the Napster technology is out in the open, and there exists a range of different implementations imitating the Napster client/server applications (OpenNap, jnerve, jnap, xnap, snap, gnap...etc), continuing the file sharing introduced by Napster.

Gnutella

Unlike Napster, the Gnutella protocol does not maintain any form of central caches and does not offer a new naming policy to deal with the dynamic client IP addresses. This is possible because Gnutella peers may be connected to different peers for every single connection that is initiated. Hence a central IP address table is not necessary, since the peer IP address is broadcast when a peer connects to the Gnutella network. The Gnutella protocol is more consistent with the peer-to-peer idea, rejecting the central server offered by the Napster protocol. However, the configuration of a Gnutella peer faces the same problems: who to contact to get up and running.

Configuration

To be able to start communicating, a “rendezvous point” has to be known to the Gnutella peers. In the childhood of this protocol, the users had to find this point on their own, since there was no pre configured addresses in the downloaded clients [9]. This problem has been solved by an application called gnuCache (some similarities with the DHCP functionality), which ran at some Gnutella enthusiasts server. It was possible to find stable connection points on maintained web sites which were also reachable through the Gnutella peer (all communication is performed on the TCP/IP network protocol). The first connection is a TCP three way handshake, also used in the Napster protocol. When the TCP connection is up and running, the connected Gnutella peers will confirm the connection with a specific handshake [10].

```
GNUTELLA CONNECT/0.4\n\n
GNUTELLA OK\n\n
```

FIGURE 6 Gnutella Handshake

The initiating peer sends a connect request on the established TCP connection, and the receiving peer replies with OK. Since most Gnutella clients are pre configured with a contact address the connection will be initiated automatically when the Gnutella peer is started. The user is passive throughout the configuration of the peer-to-peer setup. The setup needs some help from a central service, delegating a “first” contact for the peer. But what happens next is very much in accordance with the message relay technique suggested earlier [1].

How does Gnutella Work

Gnutella is not a file transfer protocol, but more a host and their files finder. Most peers maintain a direct address to between 6-10 other peers. These peers are situated at the connecting peer’s horizon level 1. There is no relation of peer geographic closeness and the assigned level 1 peers.

```
// [HEADER] <DATA>
[ <MessageID><Payload
ID><TTL><Hops><Payload length> ] <DATA>
```

FIGURE 7 Gnutella Message Format

The Gnutella header reflects its originating peer by the “MessageID” field. This is a unique id created at the different peers, based on a unique seed (for example IP addresses). “Payload

ID” is a identifier for what kind of message is being sent (PING, PONG, PUSH, QUERY, HITS). The “TTL” field declares the horizon level of the message being sent, usually set to 7. “Hops” is increased by one in every hop the message is passing, and the message is dropped when “TTL=HOPS”. “Payloadlength” declares the length of the data following the header. The peer will continuously send PING to connect all level 1 peers and afterwards PING will be sent to maintain the connection. Every peer is a message router, maintaining a fixed address to the connected level 1 peers, and a temporary table which maps a incoming message “MessageID” to the “IP address” of the sending peer. The PING reply, PONG, will contain the same “MessageID” and back trace the PING message through the peer routers to the originating sender. QUERY messages are sent to all the 1. level connected peers, which in turn will increment the “HOPS” field and forward the message to all its 1. level peers (except the originating

peer). A QUERY will be replied with a HITS message in the same manner as the PONG message returns a PING request. Gnutella also provides a PUSH message as a work around for firewalled peers. The PUSH message back trace the HITS message, and ask the HITS originating peer to PUSH the message to the initiating peer. In figure 8 the number of connecting hosts is 3, and horizon level is 3. Peer A and Peer B replies HITS, and there is a cyclic message between C and D. Since there is no hierarchy between the peers, there have to be checks for cyclic messages. A lookup in the temporary table for every message will discover if a message with the same "MessageID" arrives from different peers. When downloading the actual file found at some remote peer, a HTTP connection is established to the remote peer. This implies that the Gnutella peer also implements basic web server functionality. The HITS message contains enough information (IP address of client and file structure of the file) to collect the file through a HTTP GET request. This proves that Gnutella is more a file advertising protocol.

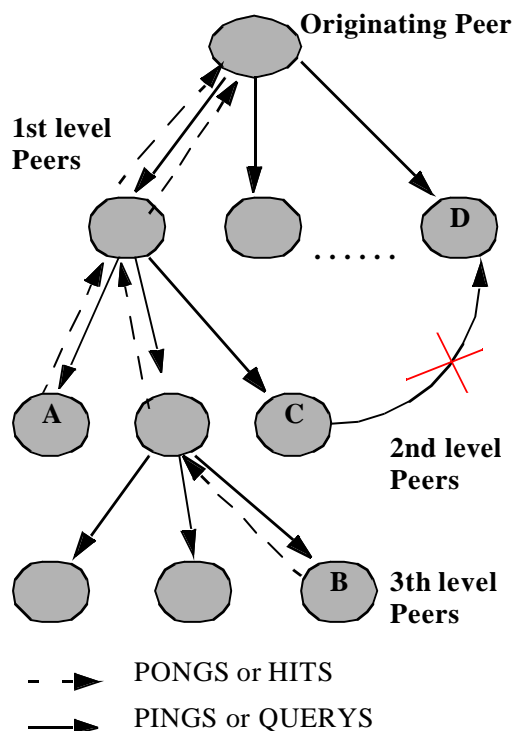


FIGURE 8 The Gnutella Protocol

A Gnutella Problem

One single QUERY message in Gnutella, with an ordinary setup of 10 connected hosts and horizon 7 will in worst case produce $10 \cdot (9^6)$ messages (approximately 5.3 million messages). The Gnutella protocol produces a lot of network traffic, and since every peer is treated as equal, this also implies treated as equal message routers. By this assumption, peers connected through slow modems are assigned the same routing load as a high speed network connected peer. Modem peers thus became overloaded with network traffic, and this resulted in a mesh of dead ends in the Gnutella peer network [11]. The network became slow. The solution to this problem was to introduce a reflector application[11] in some of the high speed connected peers. The reflectors act as proxies for the modem connected peers. Hence, routing of Gnutella messages does not include modem connection peers. The most famous Gnutella peer applications (Gnotella, LimeWire) use this principle.

Peer to Peer?

Gnutella is strongly based on peers sharing storage resources. However, to be able to get the protocol work smoothly, some central elements have been added. These central elements are implemented in "enthusiastic" peer nodes as with the first connection problem, or in "high capacity" nodes as with the reflector, but are not a part of a centrally offered service. This separates Gnutella from the Napster server. Even if both services are primarily used to share music files (Gnotella may be used to share a range of different file types), the lack of a central element will challenge the RIAA and the music industry lawyers more than Napster did.

JXTA

The first companies taking advantage of peer-to-peer like services were not market leaders, and pretty much made their name by pioneering the new technology. However, the big software companies have gained interest in the peer-to-peer concepts, and there is no lack of motivation and ambitions. Sun has introduced JXTA as a research project, and Microsoft.NET also makes usage of peer-to-peer technology [14]. As mentioned earlier, the integration of traditional web services and peer-to-peer networking is a common motivation for the major software companies to put some effort into the peer-to-peer

concept. JXTA is mentioned as a group communication tool with a large resemblance with the Gnutella protocol. The effort in standardizing a peer communication tool to take care of group management in peer systems is one of the objectives of JXTA. Some JAXTA critics [13] states that it is too early for peer-to-peer system to have a standardized substructure. The idea of JXTA is to offer a JXTA core running on any platform, and to use XML on top of any kind of network (IP, Bluetooth, GSM,...). Critics state that XML is a too verbose protocol, and does not fit into the broadcasting environment of JXTA, since it does not scale well beyond small peer groups (similarity to the Gnutella problem). The JXTA core is a Java library, but the technology is supposed to be implemented on platforms running Java, C/C++, Perl and Python giving JXTA the platform independent quality. This has to be done since handheld units might have problems running a Java VM, and at the same time be responsible for passing the verbose XML protocol. A reminder should be the Gnutella network that had problems with modem connections, because they were too slow and hence created a lot of network dead ends. Is a 9.6 kbps GSM connection in a peer-to-peer network going to do any better?

JXTA structure

JXTA is composed as a three layer "cache". In this structure, the core is implemented on all peers on the JXTA network. It also has a Service layer offering some standardized methods like network services, including search and indexing, directory, storage systems, file sharing, distributed file systems, resource aggregation and renting, protocol translation, authentication and PKI services. This is not mandatory, but offers some help functionality often used in peer-to-peer communication. On top of the second layer, the peer applications are placed.

While Napster and Gnutella were more of a specific service implemented protocol, JXTA is meant to become a foundation for any peer application. This is done by offering the core JXTA services and the JXT protocols implemented in XML as a tool for the applications. JXTA describes their protocols as follows [12]:

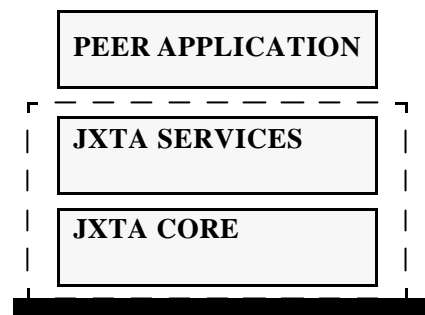


FIGURE 9 The JXTA Stack

“The JXTA protocols are easy to implement and integrate into P2P services and applications. Thus service offerings from one vendor can be used transparently by the user community of another vendor's system. The JXTA protocols are defined to be independent of programming languages, so that they can be implemented in C/C++, Java, Perl, and numerous other languages. Heterogeneous devices with completely different software stacks can interoperate with the JXTA protocols. The JXTA protocols are designed to be independent of transport protocols. They can be implemented on top of TCP/IP, HTTP, Bluetooth, HomePNA, and many other protocols.”

The motivation is clearly to give a generic tool for peer-to-peer programming, maybe even a standard for programmers utilizing the concept. The 6 JXTA protocols are also independent of each other, and have different functionality

PDP=Peer Discovery Protocol,
 PRP=Peer Resolver Protocol,
 PIM=Peer Information Protocol,
 PMP=Peer Membership Protocol,
 PBP=Peer Binding Protocol,

Configuring

The 6 protocols are independent, but since the aim in this article is to discuss the configuration issues, or the self configuration of different peer-to-peer solutions, the PDP and the PRP is the protocols of interest. The other management protocols are described in the JXTA Protocol specification [12]. However, when trying to discover a new service, an attribute describing the

quality the searching peer is looking for is appended to the PDP message. The specific value of these attributes may also be added to specify the search further. The discovery message advertises its sender attributes as well, when sending a discovery message into the network. This is in accordance with the caching possibilities in the JXTA service layer. By caching every attribute from each passing message, the caching peer might optimize his next discovery search. The protocol advertisement is passed as XML values, and is independent and understandable independent of what platform and peer implementation language. There are given no guarantees that the PDP message will be replied, and even if a peer matches the requested attributes, the requesting peer is not obliged to answer the message. This is similar to the IP “best effort” service.

This is a first step broadcast, and the peer accepting the message will search its cache, and if successful return a peer advertisement of a matching peer. If not successful, the PRP protocol is activated to search the next step of peers for a match.

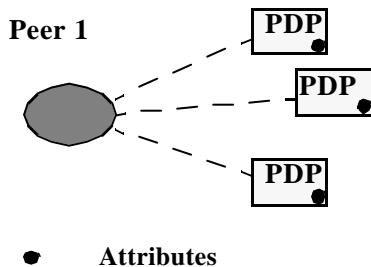


FIGURE 10 Peer PDP Broadcast

It is possible to configure the PRP specifically, to limit its range, but by default, the PRP is set to cover the “world peer group”. The PRP has the same IP like qualities, and through a rendezvous service message are propagated into the network in a manner very similar to the Gnutella protocol implementation. All data sent on these specific protocols are XML, giving a understandable way of communication between the different peer programming languages and platforms. However, XTA has the same configuration problem as encountered in the other peer-to-peer protocols. Even if JXTA is meant to be a general peer-to-peer enabling service, there is no guarantee that the first step (PDP) is going to be successful.

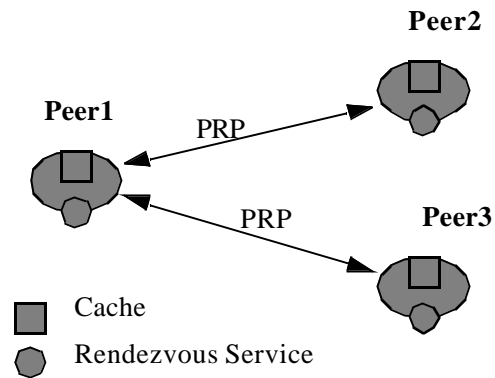


FIGURE 11 Peer Communication Through PRP

Distributed computation

One of the big advantages of peer-to-peer networks is the possibility to share unused computing resources. There is several applications available that desires to use the untapped resources of computers in the Internet(ex[15][16]). The next section will use the Seti@home project[17] [18]to illustrate the concept of distributed computation.

Seti@home

The SETI@home project[17], managed by a group of researchers at the Space Sciences Laboratory of the University of California, Berkeley, is the first attempt to use large-scale distributed computing to perform a sensitive search for radio signals from extraterrestrial civilizations. SETI@home conducts its observations at the National Astronomy and Ionospheric Center's 305-meter radio telescope in Arecibo, Puerto Rico. The project concentrates on detecting very narrow band signals from possible extraterrestrial life, and it is covering about 25% of the sky.

The process of analyzing the amounts of data the SETI@home project generates would require more computational power than is available in the largest existing supercomputer. But because of the structure of the required calculations it is easy to divide the job into smaller tasks that can be solved independent of each other. That makes it very suited for distributed computing.

The SETI@home server

Each data unit covers 10 kHz of bandwidth for 107 seconds. The units are transferred into a temporary storage with a capacity of holding

750.000 data units. From the temporary storage it is distributed to the contributing PC through a data server(see FIGURE 12). The main SETI@home server consists of three computers. One holds the user database, containing information on each of the SETI@home volunteers (including the number of work units completed, time of last connection, and team membership).The second server system holds the science database. The science database contains information on the time, sky coordinates, frequencies, and so forth for each work unit generated, as well as information about how many times the work unit has gone to SETI@home users and how many results have been received. The third server system contains the work unit storage, handling distribution of work units and storage of returned results. Communications between the server and clients use the hypertext transfer protocol (HTTP).

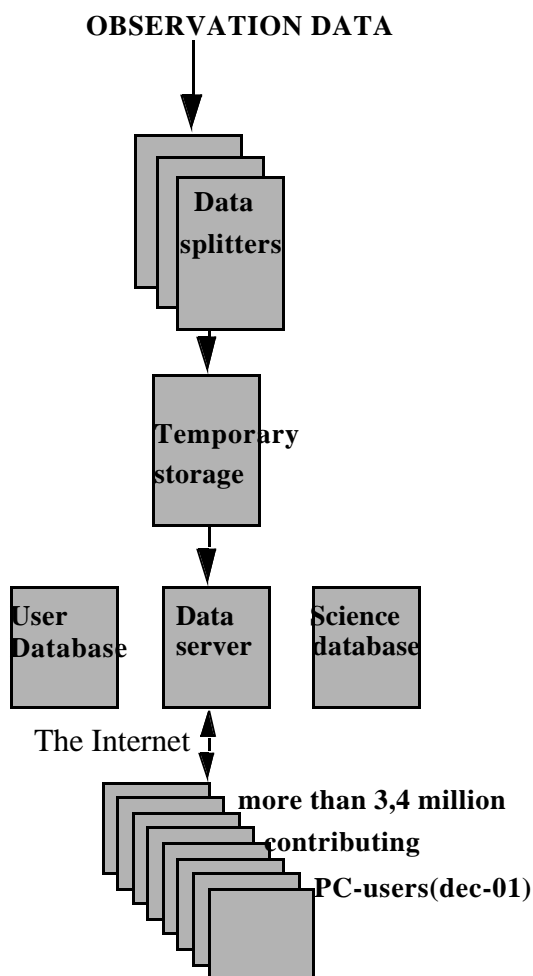


FIGURE 12 Structure of SETI@home data server.

The SETI@home client program

The SETI@home client program can run on various combinations of CPUs and operating systems. Users download the software from the SETI@home Web site [17]. For Microsoft Windows and Apple Macintosh, the software installs itself by default as a screen saver, only processing data when the screen saver is active. After the client program has scanned the data for signs of unpredicted signals, a few potential signals are returned to the server for further analyses.

Is this pure peer-to-peer?

The use of central control makes the SETI@home project in conflict with the idea of pure peer-to-peer computing. Still this is recognized as peer-to-peer, because it supports the idea of contributing autonomous peers working together as one system. Most distributed computing projects(all the big ones) are today based on centralized coordination.

Grids

One way of organizing peers is in computational grids[19][20], analogous to electric power grids. Grid computing allows to couple geographically distributed resources and offers consistent and inexpensive access to resources irrespective of their physical location or access point. It enables sharing, selection, and aggregation of a wide variety of geographically distributed computational resources. This makes it possible to address distributed computing resources as a single computer. The parallel processing of applications on distributed systems provide scalable computing power. This enables exploration of large problems with huge data sets, which is essential for creating new insights into the problem. A grid can be viewed as a seamless, integrated computational and collaborative environment. Grids are an improvement of what is now recognized as distributed computing. Today's distributed computing initiatives are highly centralized. The idea of computational grids is that coordinated use of distributed computing power shall be possible without central coordination. The lack of central organization will also make it more scalable than today's solutions. There are several initiatives to make standards for organizing of Grids [21][22]. But grids are not only for distributed computation, it is intended to be a framework for developing of all sorts of peer-to-peer applications.

Ants

The ant notation mentioned in relation with peer-to-peer computing is, as one might guess, inspired by the way real ant colonies solve complicated problems. Ants might be described as agents with simple resources. Alone they can only solve very limited tasks, but their advantage is their ability to work together and learn from each other. Since there is no standards yet there is several different solutions to how this can be done. This article will concentrate on describing one approach, The Anthill Project [23].

The Anthill Project

The Anthill Project is a framework for peer-to-peer application development, deployment and testing. The project has to goals [23]:

- 1) To provide an environment for simplifying the design and deployment of new systems
- 2) To provide a testbed for studying and experimenting with peer-to-peer systems in order to understand their properties and evaluate their performance

The system is based on the multi agent systems (MAS) [24] paradigm. A MAS is system of agents that can observe their environment and perform simple computations leading to actions based on these observations. MAS is different from other agent systems in the fact that there is no central coordination of activity. The agents in a MAS system work together to solve tasks that are beyond the capabilities and knowledge of individual agents.

A system based on Anthill is composed of interconnected “nests”. The nests is peer computers that can host and generate ants. The ants move between nests in a network, and interact indirectly with each other by updating the information in the nests. This is similar to the way real ants communicate. The Anthill project has also adopted other properties from real ants, by implementing genetic algorithms in the evaluation framework. By giving the ants a set of parameters that define their behavior (a “genetic code”), the evaluation framework admits selection of those ants that is best at solving a specific problem.

Anthill is a framework for developing all sorts of peer-to-peer applications. It takes care of communication, security and scheduling and

lets the programmer concentrate on writing ant algorithms and defining the structure of the system. It is still under development.

Security

This is a little outside the scope of this article, but there has to be an awareness of that security issues is raised when allowing peers access to your personal computer unit.

There is several security issues connected to distributed computing. Contributors can have incentives to adjust the results of the work carried out on their computer(several of the contributors to the SETI@home project have tried to manipulate the data the client program returns to the server), and there is also the possibilities of letting virus into your computer when you make it available for others to use. A proposed solution to some of these issues is presented by Golle/Mironov [25].

There is also examples of file sharing protocols as Gnutella that have been utilized to spread data viruses.

Some programers and security managers [26] categorizes peer-to-peer programs as potential security hazards, while the pro peer-to-peer programmes [27] states that peer to peer computing does not have to be any more insecure than running a Java Applet in your local browser (Sandbox principle). Anyhow, the peer-to-peer architectures should motivate the peer users for more thorough security checks.

Conclusion

The organization of computers in networks as peers instead of clients and servers seems like a good way of meeting new demands for network functionality. Since all nodes in the network has equal rights and possibilities it is also well suited for self configuration.

Central administration is contradicting the peer-to-peer notion of independent peers. By examples it has been shown that some sort of central coordination is included in most peer-to-peer “like” applications. Specially if secure and reliable peer-to-peer networks are wanted.

As an intermediate solution hybrid networks are useful. They will still be used for file sharing, distributed processing and instant messaging, maybe integrated with the existing network paradigm (client/server).

References

- [1] Ross Lee Graham, Linköpings university, Intelligent Information Systems LAB, <http://www.ida.liu.se/conferences/p2p/p2p2001/pure.html>
- [2] Ross Lee Graham, Linköpings university, Intelligent Information Systems LAB, <http://www.ida.liu.se/conferences/p2p/p2p2001/hybrid.html>
- [3] Convergence of Peer and Web Services, Jeff Schneider, CEO of Resilient Edge, <http://www.openp2p.com/pub/a/p2p/2001/07/20/convergence.html>
- [4] What is P2P...and what isn't, Clay Shirkey, Partner at The Accelerator Group, <http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html>
- [5] Napster: Popular Program Raises Devilish Issues, Erik Nilsson, President of POP-head.com, <http://www.oreillynet.com/lpt/a/network/2000/05/12/magazine/napster.html>
- [6] Napster Protocol Specification, <http://open- nap.sourceforge.net/napster.txt>
- [7] Next Step for P2P? Open Services, Gene Kan, lead developer of the Gnutella project, <http://www.openp2p.com/lpt/a/p2p/2001/08/02/openservices.html>
- [8] Why the RIAA is fighting a Loosing Battle, Steve McCanell, 5/12/2000, <http://www.openp2p.com/pub/a/network/2000/05/12/magazine/riaa.html>
- [9] Gnutella: Alive, Well, and Changing Fast, Kelly Truelove, CEO of Clip2, <http://www.openp2p.com/lpt/a/p2p/2001/01/25/truelove0101.html>
- [10] Gnutella Protocol Specification, <http://gnutelladev.com/protocol/gnutella-protocol.text>
- [11] Gnutella: To the Bandwidth Barrier and Beyond, 6/11/2000, <http://www.clip2.com/gnutella.html>
- [12] JXTA Protocol Specification, <http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html>
- [13] The Trouble with JXTA, Adam Langley, http://www.openp2p.com/pub/a/p2p/2001/05/02/jxta_trouble.html
- [14] JXTA Takes Its Position, O'Reilly researcher and developer, http://www.openp2p.com/lpt/a/p2p/2001/04/25/jxta_position.html
- [15] Cancer Research project, <http://members.ud.com/projects/cancer/index.htm>
- [16] Entropia, <http://www.entropia.com/>
- [17] The SETI@home homepage, <http://setiathome.ssl.berkeley.edu>
- [18] SETI@home: Massively Distributed Computing for SETI, Eric Korpela, Dan Werthimer, David Anderson, Jeff Cobb and Matt Lebofsky, <http://www.computer.org/cise/articles/seti.htm>
- [19] <http://www.gridcomputing.com/>
- [20] Grids and Grid Technologies for Wide-Area Distributed Computing, Mark Baker, Rajkumar Buyya and Domenico Laforenza
- [21] The anatomy of the Grid, Ian Foster, Carl Kesselman and Steven Tuecke, <http://www.globus.org/research/papers/anatomy.pdf>
- [22] The Grid P2P Topology Protocol, Ben Houston, <http://www.excortex.org/p2p/grid.html>
- [23] The Anthill Project, project description, <http://www.cs.unibo.it/projects/anthill/description.htm>
- [24] Towards the Standarsization of Multi-Agent System Frameworks, Roberto A. Flores-Mendez, <http://turing.acm.org/crossroads/xrds5-4/multiagent.html>
- [25] Uncheatable Distributed Computations, Philippe Golle and Ilya Mironov, <http://>

crypto.stanford.edu/~pgolle/papers/
distr.html

[26] Peer-to-peer network security may depend soon on the strength of your 'reputation', P.J. Connolly ,<http://www.infoworld.com/articles/op/xml/01/03/19/010319opswatch.xml>

[27] Security Concerns Miss the P2P Point, Jon Orwant, Editor in Chief of The Perl Journal, http://www.openp2p.com/pub/a/p2p/2001/03/27/orwant_security.html