AGENT EXTENSIONS FOR PEER-TO-PEER NETWORKS

Kalyan Valiveti, B.E.

Thesis prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

December 2003

APPROVED:

Paul Tarau, Major Professor

Robert P. Brazile, Committee Member

Rada Mihalchea, Committee Member

Sandra L. Terrell, Interim Dean of the Robert B. Toulouse School of Graduate Studies

Valiveti, Kalyan, Agent extensions for peer-to-peer networks, Master of Science (Computer Sciences), December 2003, 51 pp., 10 figures, references, 29 titles.

Peer-to-Peer (P2P) networks have seen tremendous growth in development and usage in recent times. This attention has brought many developments as well as new challenges to these networks. We will show that agent extensions to P2P networks offer solutions to many problems faced by P2P networks. In this research, an attempt is made to bring together JXTA P2P infrastructure and Jinni, a Prolog based agent engine to form an agent based P2P network. On top of the JXTA, we define simple Java API providing P2P services for agent programming constructs. Jinni is deployed on this JXTA network using an automated code update mechanism. Experiments are conducted on this Jinni/JXTA platform to implement a simple agent communication and data exchange protocol.

ACKNOWLEDGEMENTS

I would like to thank my major professor, Dr. Paul Tarau for his excellent guidance and support in completion of this work. I thank Dr. Krishna Kavi for his suggestions at various stages of this work. I express my gratitude to my committee members, Dr. Rada Mihalchea and Dr. Robert Brazile, for their invaluable suggestions. I take this opportunity to express gratitude to my parents Murthy Valiveti and Lakshmi Valiveti for giving me needed encouragement and support. I would also like to thank Prasanna Vemuri and other friends who encouraged me in doing this work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Problem Statement

Sizeable part of population using Internet uses P2P networks for file transfer. P2P network applications are becoming more than one-to-one file sharing software. Existing Peer-to-Peer (P2P) platforms provide many attractive features. Of all, scalability is one of the best-tested features of existing P2P networks with increasing number of simultaneous users. Other recently developed features in P2P networks include ability to control over the network without the use of centralized server, ability to get a resource from simultaneous online peers, load balancing in peers and ability to share information from any computer irrespective of security firewalls [16].

There are many challenges faced by these systems. They are:

- Search terms are limited to user knowledge over the subject
- More emphasis is given to sharing files than sharing or providing web services
- Poor search techniques flood the network with query messages
- No version control of published resources on the network
- Search criterion is limited to "meta-data" rather than the content or any other parameter
- Security issues are often not addressed by most of the P2P systems

Agent based Peer-to-Peer systems provide answers to current challenges faced by these systems. Agents provide automatic execution of certain tasks without the user intervention. The search query data, results and shared resources form valuable database that is often not seen as a powerful resource. This can be cached to improve search mechanism. Such an approach when adopted in Gnutella systems is evaluated in the past [21]. In this research, we define an architecture in which Jinni Agents are used to access this database in JXTA to improve the abilities of P2P system. JXTA is considered along with Jinni, Prolog based agent engine to provide such a platform.

1.2 Overview

Second chapter discusses about different aspects of P2P systems. It also explains different approaches of P2P networks like Napster, Gnutella, Fasttrack, and JXTA. Third chapter gives introduction to JXTA and why it is chosen in this context. It also explains JXTA architecture, different terms and concepts. Configuration of a peer to get connected to JXTA is also discussed in this chapter. Agent based systems and how they can change the present scenario are discussed in chapter four. This chapter gives an overview of why and how agent based systems help Peer-to-Peer networks. Basic agent architecture is laid out in chapter five. This chapter gives an overview of how Jinni and JXTA can be used together. It also describes the Jinni/JXTA API that describes the implementation of simple JXTA procedures. Agent container, communication principles are also given. An example of how this proposed system can work towards improvement of P2P systems is given out at the end of fifth chapter. Conclusions, details of how far we have achieved

our targets, are given in chapter six. Directions for future work and possible enhancements are discussed in chapter seven.

CHAPTER II

PEER-TO-PEER SYSTEMS

2.1 Introduction to Peer-to-Peer Systems

Peer-to-Peer systems can be defined as systems that are result of convergence of traditional client and server roles into one. [11]. Peer-to-Peer (P2P) systems are capable to communicate among themselves without the need of a central server. P2P systems have seen a surge in acceptance in recent years. This can be attributed to its wide range of advantages they offer. With computers becoming powerful day-by-day, most of the computers connected to Internet have server capabilities and bandwidth abilities to become a server. P2P offers a wide range of services including sharing files, resources like printers, online gaming, and instant messaging. This arena has widened due to the efforts to provide web services in P2P networks.

For example, if you consider online game by 3 persons at 3 different locations. Using Client-Server mechanism all 3 nodes act as clients and there is a central server, which replicates the moves of each of the players and sends them simultaneously to these 3 different locations. No processing is done at the client systems. On other way, if you consider a P2P scenario, there is no central server. All 3 nodes participating in game act as both server and client. They send their data to others and get data from others in P2P network. This will enable faster, efficient and reliable system. Figure 2.1 represents client-server architecture.

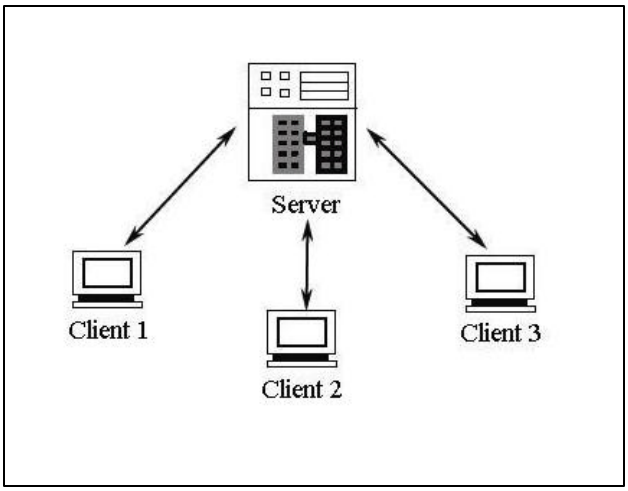

Figure 2.1: Client – Server architecture

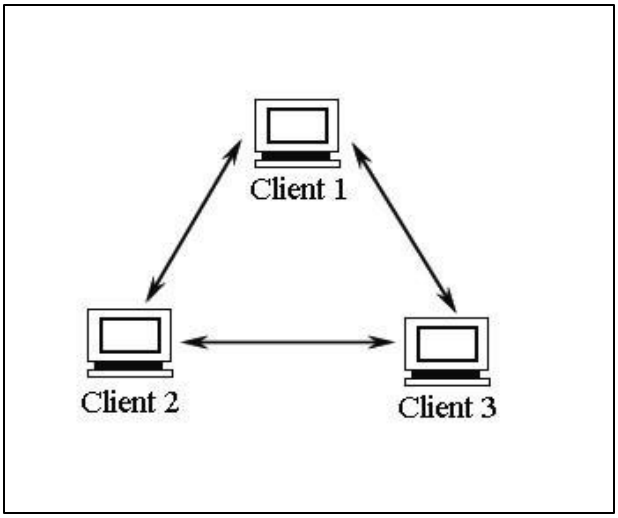

Figure 2.2: Peer-to-Peer architecture

Figure 2.2 shows a basic Peer-to-Peer architecture. In above example, the P2P system eliminates the use of a centralized server. Eliminating centralized server produces these following advantages:

- Eliminating single point failures
- Eliminating delays, overhead of having a server

- Reducing Internet traffic in case of high bandwidth consuming online games
- Distribution of content on network
- Efficient use of resources

A basic P2P system comprises of different properties.

2.2 Underlying Network Architecture

A P2P system is either a hybrid P2P or pure P2P or centralized P2P depending on the network architecture they follow. Centralized P2P systems have a centralized means of passing queries and finding resources. Once resources are found, the resource sharing is done in a P2P fashion. The centralized P2P system can manage a central database of resources on the network and respective address to look for on the network. A centralized P2P system is represented in Figure 2.3. Peers direct their queries to this central server and get required peer information from them. After they get information on which peer has the required resource, they connect to that resource and communicate in a P2P way. The advantage of such a system is that control over the system is more when compared to Pure P2P systems. Napster File sharing system uses this kind of architecture. This was more prone to legal problems due to presence of control over the systems and content shared in these networks. Example of such a system is Napster [12] file sharing network. This kind of resource sharing architecture depends on a central server making it easy to search and find a resource. However, it deviates from a key concept of P2P system to be able to be autonomous and independent of a single source.

The network architecture will decide on the level of control on transactions in networks. In case of centralized systems, controlling network is possible. In a pure P2P scenario, the level of control is very limited.

In Pure P2P scenario, the search request for resources is distributed from one peer to others in the network. The discovery and publishing features occur in a one-to-one fashion. The advantage of such systems is that it is completely P2P and can have all the above advantages put forth by P2P networks. However, these systems require more time to discover nodes in a search process and as there is no centralized server, route discovery, resources sharing, load balancing and network management have to be done in a dynamic way. This will decrease throughput and increase turn around time.

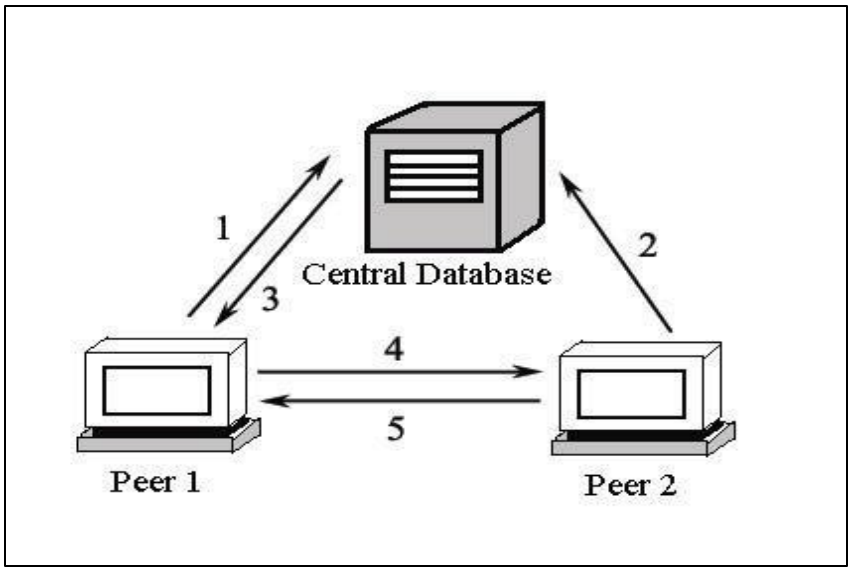

Figure 2.3: Operations in central server based Peer-to-Peer architecture

A compromise between both the above designs is hybrid P2P networks. Hybrid P2P networks are represented in Figure 2.4. Concept of "super node" is introduced here. The central database in Napster is brought down to few levels and maintained at several nodes

having information about few sub nodes under it. This kind of architecture provides partial control over the network. Super nodes maintain information on resources shared and peers sharing them. Most of query traffic occurs between these powerful super nodes only. Super nodes are elected on the basis of their high bandwidth capabilities. Most of the present day P2P systems are designed in this way. JXTA, Gnutella [25] and Fasttrack P2P engines work in this way. Gnutella calls them "super peers" and JXTA calls them "Rendezvous". Virtual control over the network can be achieved on these partial P2P systems using agent-based techniques.

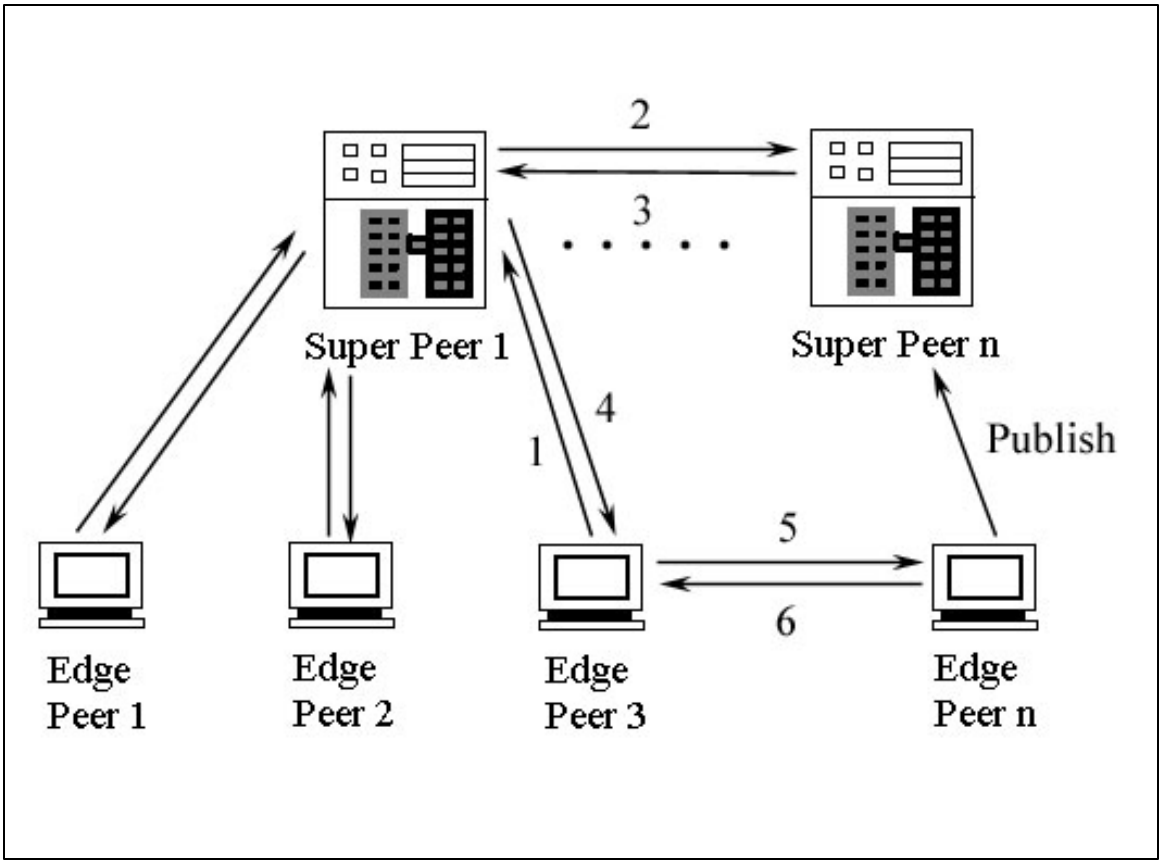

Figure 2.4: Super peer concept in peer-to-peer networks

2.3 Joining a network

Whenever a node comes up, the first operation is to find peers that belong to same network. This is done in different ways in different P2P networks. In Centralized P2P system, peers connect to central server and register themselves that they are available. In Hybrid P2P systems, each node has a cache of "super nodes". Whenever the nodes want to be a part of the network, all they do is try to connect to one of the super peer addresses in the cache. Similar approach is also used in Gnutella P2P system.

Pure P2P system does not need a centralized server to locate resources. Searching and Sharing of resources are done in a P2P model. Example of such a system is Fasttrack P2P network. Clients like Grokster, Kazaa operate on this network. Fasttrack uses "Super nodes" concept to search for nodes and resources. Super nodes are peers with more bandwidth capabilities. They connect to a small subset of nodes which route search requests, help in connecting to resources and provide an entry point into the network. In JXTA, discovery of nodes and resources is done using Rendezvous nodes. These nodes are analogous to super nodes in Gnutella network.

2.4 Search Mechanisms in P2P

The main aim of a P2P file-sharing system is to locate and exchange files between peers using decentralized servers [12]. When a peer requires a file it sends a query to the peer network in order to locate the requested file. There are different ways in which a peer locates the file that matches the query request at the other peer.

Search for resources in systems using central server is by performed by going through a central index. Search depends on Meta data in Fasttrack systems. This is advantage in a

sense that any of the Meta data contents can be matched to find the required resource. However, this also is a disadvantage when the Meta data is not relevant to resource information or no way describes the resource. No intelligence to know if the file has valid information is available. This is one of the disadvantages of Gnutella system. Also, updating files that are found is difficult, which means that it is difficult to know if a file update can be downloaded without having to download the entire file. Search in JXTA uses advertisements. In JXTA version 1.0, Rendezvous peers contain all the advertisements generated by edge peers in a JXTA network. Peers searching for the advertised resource would contact rendezvous peers, get the required information of the edge peers and connect to them. In JXTA version 2.0, only the index of advertisements is contained with the rendezvous peers rather than the advertisements themselves.

In general the search protocol in a P2P network works as soon as a node sends message among peers requesting for data. This search message is called "query" and it is propagated to all his peers. The peer after receiving the query request starts searching locally for relevant matches. If it finds a resource, the peer generates a query-hit message, which includes the address of the peer and its network connectivity. This gives the capability to choose the best network connection if the end-user receives data from more than one peers [13].

2.4.1 Flooding Broadcast of Queries

In this method, when a peer makes a query, the query is sent to all the neighbor peers. If its neighbor peer does not have the match for the query, the query is then sent to the neighbor's neighbor peer. It continues searching for peers, which has the solution for the

query, when a resource is found that peer will send a message to the end-user indicating that it has a match for the query, and then establishes a peer-to-peer connection.
Each query has a time-to-live (TTL) counter, which is set to a particular value. The advantage of this method is flexibility in processing the queries. Raising the number of peers in the system will cause the network to reach bandwidth saturation quickly thus limiting it to a small networks.

2.4.2 Selective Forwarding Systems

In this system, query is selectively sent to specific peers who are considered likely to locate the resource. Peers will become super peer automatically if they have sufficient bandwidth and processing power. This type of systems use flow control algorithm, where peers with low bandwidth will make queries to super peers. This algorithm applies a sensible priority scheme to drop queries that won't fit into the connections.

2.4.3 Decentralized Hash Table Networks

In decentralized hash table networks, each file stored with in the system is given a unique id, which is used to identify and locate a resource. A solution can be obtained quickly despite the size of the network when the correct id is given. If a peer is looking for a file from another peer, it must obtain this key first in order to retrieve the file.

2.4.4 Centralized Indexes and Repositories

In this system, query is sent to a main server that has the index of all peers and their resources. The server will look-up the index to see if it can find the match for the query and then it sends a message to the end-user of where he could get the file. These systems

provide the best performance for resource discovery but if the server fails it brings down the whole network.

2.4.5 Distributed Indexes and Repositories

In this approach, each content broker in the network keeps an index of local files as well as some files stored in the neighboring content broker. When it receives the query from the peer, it checks locally first and if no matches were found it uses local index to decide upon where to send the request. The index on each server changes as files move through the system. This system provides the best performance if designed properly.

2.5 Get Resource

Once the search results are retrieved and required resource is found, the next operation would be to get the resource. Some of the systems may be behind a firewall/NAT. In such cases, contacting a peer directly and getting the resource is not possible. Tunneling is operations of a third peer helping the two peers that are involved in a sharing process. The third peer would facilitate data communication between these peers behind firewall and help them in sharing files.

Early P2P systems like Napster did not have such a function in them. However, Fasttrack has tunneling mechanism implemented in it. Super nodes/super peers do this job of providing service as a third peer in Fasttrack systems. JXTA has concept of relay peer using, which one would connect to network if it is behind a firewall. User configuration before connecting to the network will allow one to choose different relay peers available at a particular time. Fasttrack systems have the capability to download same file from different peers.

2.6 Publish Resource

Most of the P2P systems use a "shared folder" and sandboxing approach towards publishing a resource. Any resource in such a folder will be available for sharing. In Peermetrics system, if you save a folder in p2p/files/binary would allow it to be published under search items p2p, files, and binary. This is a folder dependant approach and may cause some problems in case of sharing too many files with varied interests. JXTA creates an advertisement for every resource that is published and an index to it is stored in rendezvous peer. Depending on user, it may also be sent to all peers in its group.

2.7 Send Resource

Limits to computer cycles that can utilize bandwidth are initialized to make sure that peers do not over overwhelm a system.

In sending resources, different systems approach in a different way. In Fasttrack protocol, it is done using laying out a file system layer on the top of existing file. This will allow breaking up of file into smaller parts and sending out it onto network. This will also enable peers to retrieve different parts of a file from different peers simultaneously. This also requires good file encryption techniques at client and server ends to ensure that a file has been completely transferred without any security breach in the network. In JXTA, this approach is not completely implemented as this functionality is more described to be application specific.

2.8 Digital Rights Management

Digital rights management is one of the most sought after features in P2P networks. This will allow users to reap the benefits of P2P systems without violation of copyright law or

ownership laws. A digital rights management system in P2P networks can be implemented at different stages of transfer of resources in P2P networks. It can be done at the time of transfer of resource in P2P network as a handshake mechanism. This simple procedure allows secure way of transfer. The implementation of digital rights can also be done in following way. A peer on the network can download the content but needs an authorization key to enable and use the resource. This process is similar to public key and private key mechanisms in Cryptography. In this mechanism, the user has to register a particular resource and a corresponding key is generated. This key has machine specific value so that it is not valid for any other machine. This key is attached to the particular resource each time the resource is used. This procedure though very efficient in protecting digital ownership rights is more cumbersome process. Similar system can be found in Microsoft media player [23]. The other disadvantage is that this process requires an Internet connection to obtain the digital key. Trust is another vital factor that should be achieved between the provider and consumer. Audio and video watermarking [26] are also pursued as operations for protecting digital rights in some cases. A third party that is acceptable to parties involving in the transaction can be introduced to improve security and trust among peers sharing resources [24].

CHAPTER III

JXTA

JXTA (pronounced as Juxta) is a set of protocols that Sun Microsystems has put forward to facilitate the development on Peer-to-Peer (P2P) networks. The set of protocols are programming language independent thus making it easy for developing applications. JXTA J2SE, java implementation of JXTA has become more popular due to its virtues of interoperability, easy deployment and impetus from Sun Microsystems.

The main virtue of JXTA lies in the abstraction of certain functionalities and systematic approach that facilitates network programming. Key terms in JXTA are Peers, IDs, Pipes, Advertisements, Services, Rendezvous peers and Relay peers. Basic functions of a P2P network are to publish, discover, receive and send resources. These can be achieved easily by utilizing the protocols and JXTA J2SE implementation. Pipes are used as channels to send messages or binary data. Advertisements are used to publish peer and service information. Using these concepts, JXTA provides vital functions in P2P networks. JXTA defines terms like Rendezvous Peers and Relay peers to serve as super nodes and offer tunneling support nodes respectively.

JXTA has a 3-layer network structure, which is as shown in figure 3.1.

JXTA network structure is divided into three layers with varying levels of abstraction. The lowest level consists of basic services like peer group creation, advertisements, pipes, Id structure and security. This will ensure that all services or applications that are built on this layer will get the services of JXTA Core.

JXTA Services forms the second layer in the network layer stack. Services include search, discover, and publish processes that are built upon lower levels in JXTA. JXTA Applications are on the top in the network architecture. Examples of JXTA applications are file sharing, resource sharing and other kinds of P2P applications. These applications are built upon the two layers whose services are sought by this applications layer.

JXTA has a 3-layer network structure, which is shown as follows:

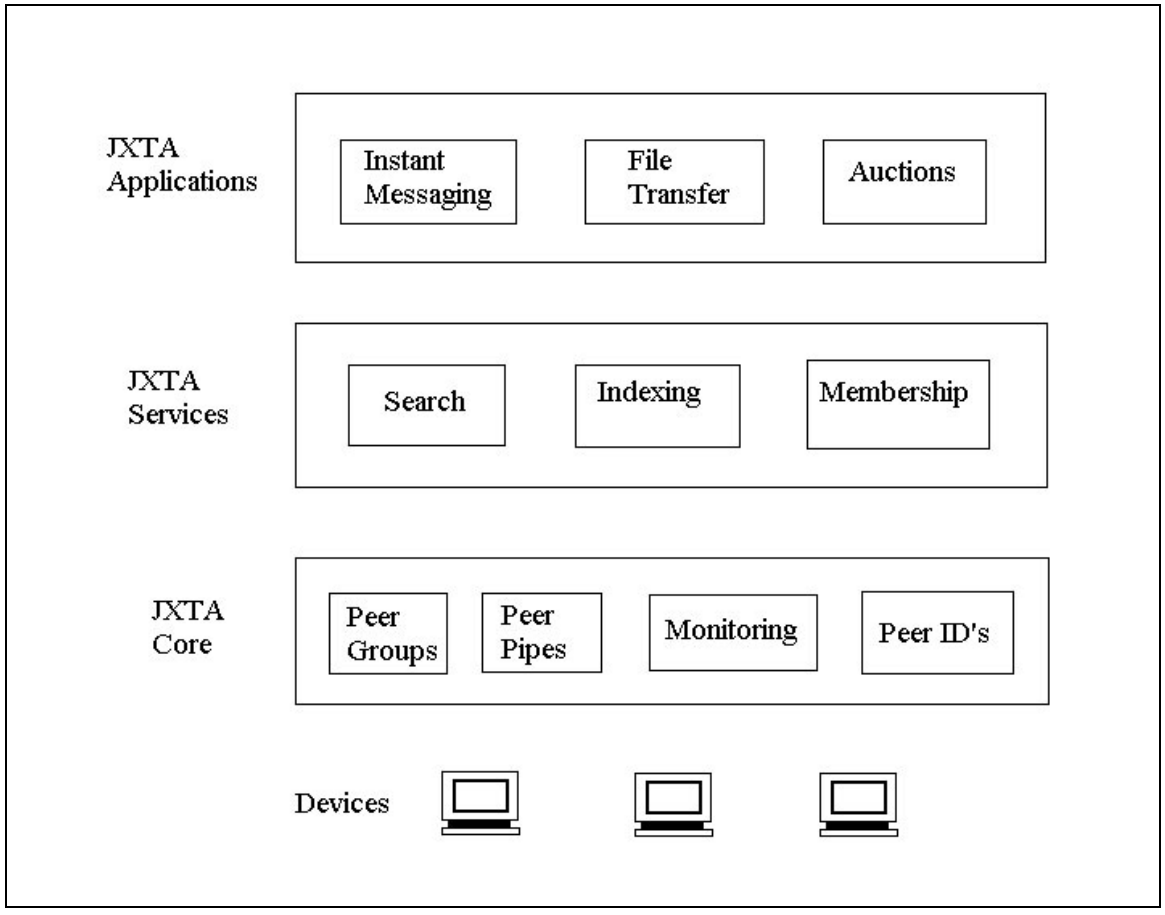

Figure 3.1: JXTA architecture showing different layers

Figure 3.1 is adapted from "JXTA Programming Guide", http://www.jxta.org/docs/JxtaProgGuide_v2.pdf, Project JXTA, May 2003

3.1 Protocols in JXTA

The different protocols in JXTA are:

Peer Discovery Protocol: This is used by peers to advertise their resources and discover resources from other peers.

Peer Information Protocol: This protocol is used by peers to obtain status information from other peers.

Peer Resolver Protocol: It is used by peers to send a generic query to multiple peers and receive response.

Pipe Binding Protocol: This protocol is used by peers to establish a communication channel in way of pipe,

Endpoint Routing Protocol: It is used by peers to find routes to destination ports.The information includes all the valid relay peers and other intermediate peer information.

Rendezvous Protocol: This protocol describes a mechanism by which peers can subscribe to a propagation service. Rendezvous Protocol allows a peer to send messages to all listening instances of the service [8].

JXTA J2SE is implementation of protocols in Java version. Here are the descriptions of various terms in JXTA protocols and how they are implemented in J2SE. The term JXTA is used interchangeably with JXTA J2SE in this document.

3.2 JXTA Peer

"A JXTA peer is any device that implements core JXTA protocols" [17]. They may consist of any device that can provide basic services like discovery and publishing of services. Peer groups are formed by these peers and a peer can be a member of any

number of peer groups at a time. These peers can be any device that is capable of processing power and networking abilities. Peer groups have their own membership policy and admission of a peer into their group may require certain qualifications. Peer groups offer services to their members. By default, all peers joining the network will join Net peer group. However, it is possible to create and join new groups. Net peer group offers basic services like peers, advertisements in JXTA P2P. Hence, whatever group they create or belong to at a later stage, they have to be a part of Net peer group.

3.3 Rendezvous Peer

A seeding peer should be able to help any peer that tries to join the network. Rendezvous peers provide that functionality of helping peers get onto network. Peers in a network have a cached list of rendezvous peers, which is consulted to connect to one of the rendezvous peers. All the advertisements that are produced and requested by edge peers are indexed at Rendezvous peers. This will enable to reduce traffic. JXTA protocol requires atleast one peer with static address. Within a peer group, a number of edge peers may elect themselves to be offer rendezvous service and become rendezvous peers. However, they should consider the constraints and responsibilities of doing so. Some of the constraints would be if it has high bandwidth capability, not being behind any firewall and be reliable to other peers to connect to.

3.4 Relay Peer

Relay Peer is one that will enable peers behind firewall or NAT to communicate in the network. They provide tunneling feature that can surpass any network obstacles like firewalls or NAT. JXTA message contains routing information, which will provide

adaptability to the message itself. If a Relay peer fails, the message gets routed in an alternative path due to the peer information contained in the message. By doing this, the peers using the relay feature will always get required messages irrespective of Relay peer failure.

3.5 JXTA ID's

JXTA ID's are assigned to entities in the protocol specification. Pipes, Peers, Advertisements. Services are uniquely identifiable/addressable by this ID. This Unique ID system allows a peer to address services or other peers. Unique Ids are unique due to the fact that they are dependant on time and machine they are generated.

3.6 JXTA Pipes

They are mapped to given space and time there by making them unique. JXTA pipes can be classified as:

- JXTA Unicast Pipes
- JXTA Propagate Pipes
- JXTA Secure Pipes.

Figure 3.2 shows pipe as visible to the user and its network abstraction. This figure depicts the fact that a pipe may connect thorough different nodes before it connects to destination peer and still do not need any user involvement in doing so.

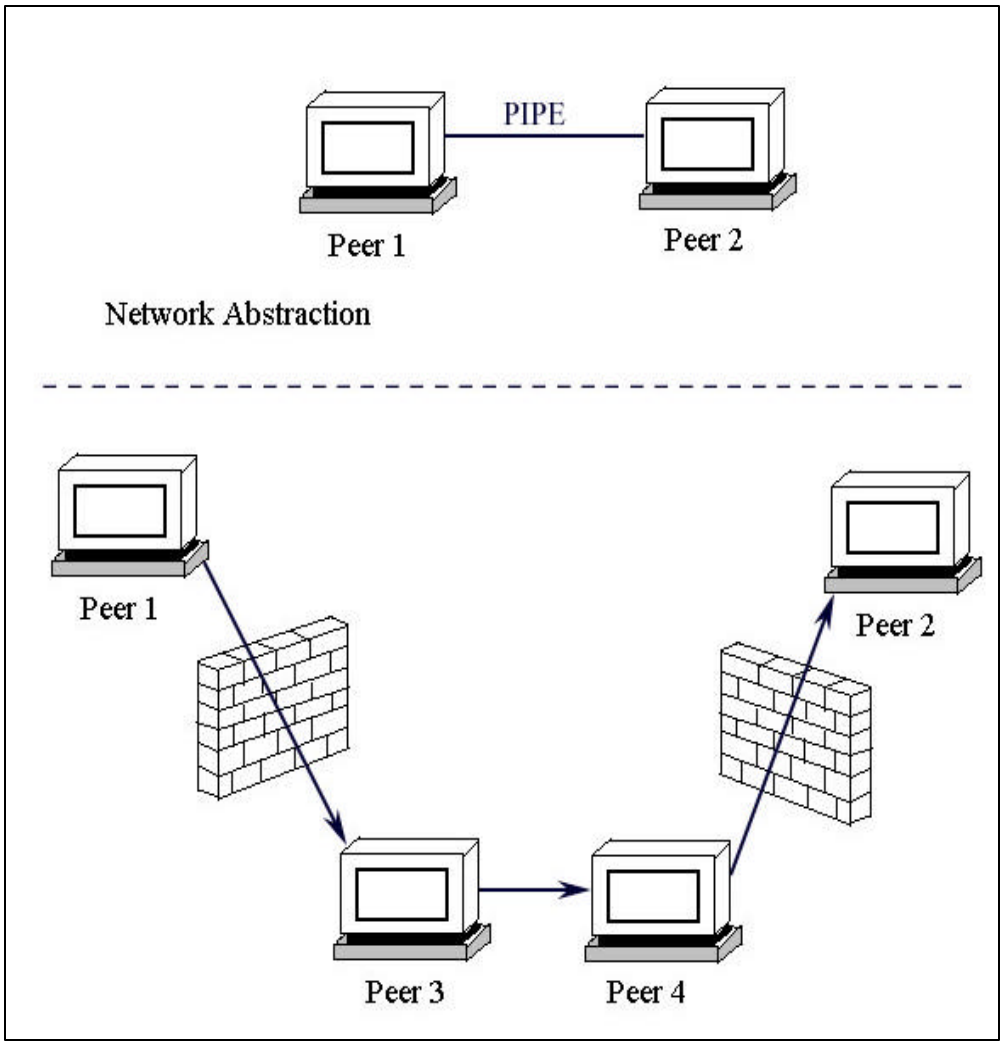

Figure 3.2: Network abstraction in JXTA pipes

Figure 3.2 is adapted from "JXTA Protocols 1.2", Project JXTA,

http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf, May 2003

3.6.1 JXTA Unicast Pipes

JXTA Unicast pipes are unreliable channels that facilitate one-to-one connection process. The transport level operations are abstracted from the user.

3.6.2 JXTA Propagate Pipes

JXTA Propagate pipes allow multicast functionality by sending message to many peers simultaneously. Again these kinds of pipes are insecure and unreliable.

3.6.3 JXTA Secure Pipes

For reliability one has to go for JXTA Secure pipes which support synchronous transfer of messages and high security in transfer.

JXTA pipes are uni-directional in design, however java based implementation provides bi-directional pipes and security layers also on it.

Pipes can be uni-directional or bi-directional. One end of the pipe is known as input pipe and the other is called output pipe. Input Pipe is one where the message starts and output pipe is the other end where the message reaches. Communication via pipes occurs as follows: A peer that wants to get connected opens an input pipe and sends out the pipe Id to open connection. Any other peer that is interested in connecting to it will have to use that pipe ID to open an output pipe and communicate via pipe.

Pipes in physical space can connect through many peers with different physical addresses. All these details are hidden from the user. Pipes provide an abstraction for sockets hiding all the complexity of IP addresses and sockets.

3.7 JXTA Sockets

JXTA sockets act as bi directional pipes, which are more reliable and secure than Bi directional pipes. They are built on pipes and provide all advantages pipes offer. Invoking a socket will in turn create a pipe, which either listens on a pipe or sends something on a pipe. Multicast functionality is also available on these pipes. The pipes are bound at run time if the pipe Id's match in value. For example, a socket with a pipe id X looks for a

pipe id X and once it finds it, the pipes are bound. The other advantage of a JXTA socket is availability of IO streams. In pipes, all the communication takes place by sending JXTA Messages.

3.8 JXTA Messages

A basic unit of data exchange that is sent between two peers in a JXTA network is called a Message. Messages can be sent using pipes in JXTA. A message has routing information so that it can reach the destination independently. The JXTA protocols are represented as a set of messages exchanged between peers [8]. Messages can be either XML files or binary data. Expression of messages in XML allows peers to participate in a protocol.

3.9 JXTA Advertisements

Advertisements are used to publish information about peers, pipes, services and peer groups. They are represented as XML files that contain information like name, ID and other attributes depending on the type of the advertisement. For example, a pipe advertisement has its ID, type of pipe and name as its attributes. A peer has a unique ID that is not dependant on its physical address. A service has an ID and each service includes a pipe advertisement for communication purposes. A peer on other end to opens an output pipe on that pipe ID extracts pipe advertisement. When each peer comes up, these advertisements are transported from the Rendezvous peer. Each of these advertisements has a definite time to live, after which they become obsolete and gets deleted automatically.

Here is an example of a Pipe advertisement:

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
        <Id>
                urn:jxta:uuid-
0228252A33B34AC3A60583641E45556B9EC78AA959094863937995914FA3302C04
        </Id>
        <Type>
                JxtaUnicast
        </Type>
        <Name>
                JINNIJXTA
        </Name>
</jxta:PipeAdvertisement>
```

Here JINNIJXTA is the name of the pipe and id contained in <ID> tags is the id needed for another pipe to connect to this pipe.

3.10 JXTA Services

A peer group offers different kinds of services. For example, a Net peer group offers basic services of JXTA architecture. Services may vary from transfer of a file to providing a web service to all peers in a peer group. These peer group services are

published in form of XML advertisements. Peer group services involve pipe Id's and once the pipe Id is retrieved, any peer in the group can invoke and get services from that particular peer [8].

3.11 JXTA Security

Security layer exists in lower level of JXTA platform architecture to ensure security exists so that applications built on it have inherent security features. Security features provided in JXTA are:

- TLS (Transport layer security): JXTA provides SSL (Secure shell layer) to ensure secure communications. Secure pipes, sockets are built on this layer. Presence of this layer would provide safety from eavesdropping.
- Root certificate: A root certificate is generated as a part of peer advertisement during configuration. This is used to sign any service that peer supports.
- Peer level security: ID and password are assigned to each peer to ensure security at peer level. The password is stored along with configuration files. Encrypted password is saved in configuration files [9].

3.12 Configuration of JXTA in J2SE

Configuration is a 4-step process. Step-by-step process is shown in the figure 3.3. The first step is to provide a username and give a proxy address if you are behind a firewall. The second step is to define which transport protocol is being used. (TCP or HTTP). Third step is to choose Rendezvous and Relay peer information and select them. Fourth would be to provide a username and password to start the application. Once the user completes this process, a. jxta folder is created in the working directory. This directory

has cm - the cache manager, platformconfig - a file that contains platform configuration information and folder pse - that contains password information. Cache manager has a cache of advertisements that are result of discovery calls to Rendezvous and other peers. Once an advertisement is requested for, the peer searches for it in the local cache and then issues a discover command to get advertisements. This process is repeated until the required advertisement is available. Platformconfig file has information regarding the protocol used, the message cache size, the peer ID, name, root certificate and ports to communicate. A configuration is specific to one folder and another instance of the same application can be invoked by working in instantiating another instance from another directory.

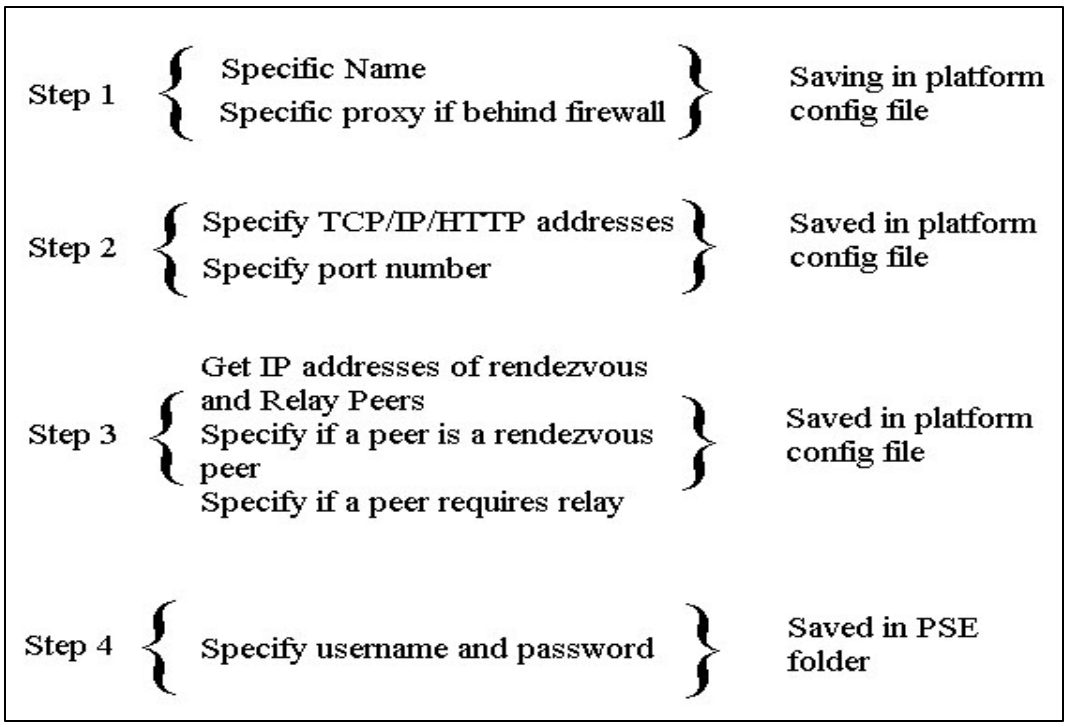

Figure 3.3: Step by step process of configuration of a peer in JXTA

CHAPTER IV

AGENT BASED P2P SYSTEMS

4.1 Agents

Agents can be defined as "Self-contained, interactive and concurrently executing object, which had an internal state and could respond to messages from similar objects" [2]. Agents can be "used to improve and enhance the performance of systems, and they are provided with attributes of intelligence, communication and adaptability to this end" [3]. Agents comprise of either mobile code that moves between different computers or code that executes periodic defined functions on a single computer. For example, an algebraic computation that is started on a node can take all its present values and complete the computation on a different node. The state of the computation is carried onto the other node to facilitate computation.

4.2 Overview of agent-based P2P systems

Agent based methodologies are introduced in P2P systems more than a couple of years ago. Some of such systems provided an infrastructure to agent-based P2P systems, where as others tried to address specific issues like routing requests and improving search mechanisms in P2P networks. Anthill [27] is one of the early agent frameworks designed to improve P2P systems. This framework allows application development in P2P networks and also provides a good testing environment. Based on ideas of multi-agent systems and evolutionary programming, this framework aims at interaction among agents and with nodes in the network. A Java prototype based on JXTA is also developed [27].

JXTA based agent architecture with emphasis on developing efficient network layer is conceived by Rita Yu Chen and Bill Yeager [10]. This focuses on using JXTA core P2P protocols and agent technology to facilitate the mobile agent deployment. This architecture uses TLS layer in JXTA to provide security feature. An agent-based protocol, SMAP, is defined to define agents and their interactions. Poblano, a reputation based distributed trust model for accessing and evaluating content on P2P networks is used for testing purposes in this case. Demonstrating the ability of JXTA to support an agent platform and ability of this agent architecture to adapt to quick changes in P2P network are detailed in this work [10].

An agent based P2P architecture is designed and implemented by Hooman Homayounfar [4]. In this work, an agent-based architecture is built around Jini. In this implementation, network layer issues are taken care by Jini [29] and Java RMI. The detailed architecture in this work is very useful in building agent based P2P systems. Concept of v-server is used to provide peer-to-peer communication. This implementation however has taken less attention towards practical problems like firewalls or NAT in implementation of P2P networks [4].

Efforts were also made to improve particular functionality of P2P networks by introducing agents into the network. Andrew Smithson and Luc Moreau discussed resource discovery in context of agent-based systems [27]. This paper provides answers to improving both synchronous and asynchronous search mechanisms. A search agent is implemented along with a GUI to improve search mechanism in P2P networks.

4.3 Advantages of Agent Based P2P Systems

P2P network comprises of different nodes running on different platforms. The network itself is dynamically changing and lot of network traffic is involved in query messages and resource transfer. By applying agent-based architecture to these networks, many of the existing features can be improved and some features like offering web services over P2P networks can be provided.

Some of the existing features that can be improved are:

- Knowledge based search mechanism
- Formation of virtual peer groups
- File encryption and security
- Content-based search

4.3.1 Knowledge Based Search Mechanism

All the P2P engines depend on user knowledge than on a knowledge base. That could explore more possibilities than a user with limited abilities [4].

4.3.2 Form Virtual Peer Groups

Virtual peer groups enable peers to join or leave certain set of peers with same properties. This would result in improvement of search mechanism and decrease of network traffic. This also solves query message overhead problem by scoping the query message transfer.

4.3.3 File Encryption

Security features are not implemented in an efficient manner in present P2P systems. File Encryption and decryption mechanisms on client side of a peer would solve this problem.

4.3.4 Virtual Control Over the Network

A trade off between Centralized P2P systems and Pure P2P systems is control over the network. If a network is decentralized and can still control the network, that would be a combination of advantages in Centralized P2P systems and Pure P2P systems. This can be achieved with the help of agents in P2P systems. Deployment of mobile agents over network can control the content on network by verifying digital signatures of every resource shared on the network.

4.3.5 Content Based Search

Another feature which most of the P2P engines is search mechanism using key words or Meta data rather than the content. Content can be given an hash key or a digital signature of that particular file which will improve search mechanism and prevent invalid Meta data/ key words.

4.4 Why JXTA?

JXTA has these advantages over other P2P networks to consider developing agent based P2P network

- Built in complete Java environment, it provides better interoperability than others
- Network layer abstraction provides needed support to develop an agent platform without considering firewalls, NAT or DHCP hindrances.
- Systematic breakdown of entities and properties like pipes, advertisements makes it easier to develop applications
- Deployment of agent based mechanisms is easy due to easy interface that can be created using JXTA J2SE API

4.5 Jinni, a Prolog Agent Engine

Jinni (Java Inference engine and Networked Interactor) is a lightweight, multi-threaded, Prolog based scripting tool. It has Reflection based architecture that allows reflecting Java classes into Jinni and Jinni to Java. Because of this functionality, Java's merits can be combined with fast computational power of Jinni to prove to be a good network platform as wells as provide good platform for agent based architecture [1].

Jinni is built on simple basics of Prolog terms, Agents and Places. Prolog terms are constants and variables. Places are processes with atleast one server and a shared blackboard allowing Linda and Remote Predicate Call transactions. Blackboard contains Prolog terms, which can be accessed by agents. Agents are collections of threads that work towards achieving various goals.

CHAPTER V

AGENT COMMUNICATIONS AND DATA EXCHANGE PROTOCOL

5.1 Basic Agent Architecture

P2P Agent architecture has these three components: Agent container, JXTA network layer and Database. Agent container comprises of different processes that co ordinate, communicate, take decisions and improve their knowledge with time. Agents are also generated in this agent container. Agent communication and basic agent architecture is shown in figure 5.1. JXTA provides the network layer with valuable abstractions like pipes, advertisements and services. Database consists of search requests, search results and shared resources.

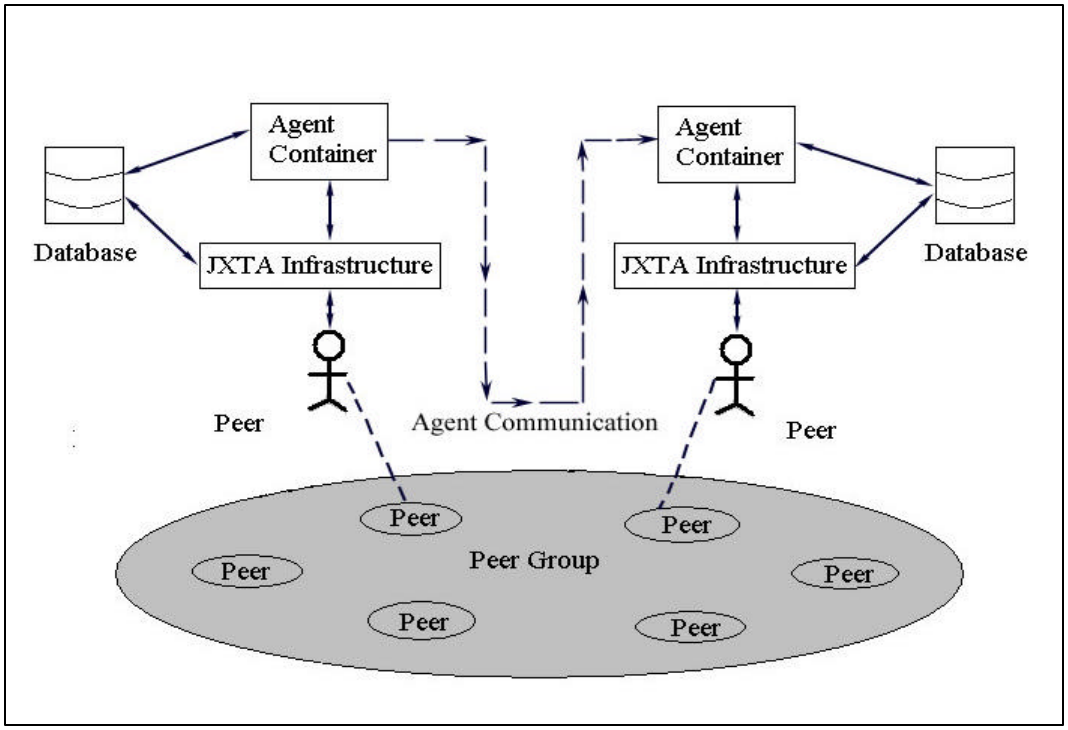

Figure 5.1: Basic JXTA/Agent architecture

Agent architecture relies on JXTA for all its network mobility and computation. This is made possible by using a JXTA adapter that has basic JXTA functionalities abstracted into simple functions. For example, JXTA agent environment can issue a command like invoke_jxta (username, password) and this would configure JXTA environment in current peer to become a valid JXTA peer. During configuration, various parameters like HTTP configuration, port number, available rendezvous peers, relay peers are pre defined and these values are assumed to be valid at the time of invocation. Using these functions, the agent layer communicates with the JXTA. Jinni, the Prolog based agent engine, used in this case has reflection capabilities that will allow mapping the same functions into Prolog functions using reflection.

5.2 Jinni/JXTA API

An easily extendable API is created in Jinni that maps certain important functionalities in JXTA. Some of them are:

invoke_jxta(uname, pword) - will create platform configuration for JXTA environment

discover_file(filename) - generates a resource request advertisement and sends it to rendezvous and other peers in group

publish_file(filename) - publishes an resource advertisement to other peers in the group and to rendezvous peer

get_file(filename) - actual transfer operation once the required resource is found at client side of a peer

put_file(filename) - actual transfer operation once the required resource is sought for at the server side of a peer

5.3 Implementation of JXTA API

Here are different steps in a sample application using some of the Jinni/JXTA API:
All the peers belong to "Jinni agent" group that is a sub group of Net peer group. All the advertisements that are sent into Jinni agent group are also sent to Net peer group. All these peers apart from the services they receive from Jinni agent group also receive messages from Net peer group. If a new node joins this group, an automatic update mechanism allows the peer to have agent infrastructure.

5.3.1 Initialize

For a peer to be a part of JXTA network, it needs to become a part of Net peer group as a first step. The rendezvous peer is to be chosen and if behind a firewall then a proxy needs to be selected apart from selecting a relay peer. This will enable a peer behind a firewall to share resources with anyone on the network. Username and password are to be supplied. A peer ID is created based on different parameters like username, time of creation. A HTTP port/TCP port has to be chosen. That particular group issues a root certificate. All configuration is saved to a local directory called ".jxta". Deleting it will enable reconfiguration of settings. Cache manager, Pse and Platformconfig folders are created after this configuration is completed. Cache manager contains all the group advertisements that are published in the same group and parent group in the hierarchy. PSE folder contains username and password information of the peer configuration. Platformconfig contains root certificate, TCP/IP, HTTP information, name of the peer and peer ID. Running configurator from another folder will allow creating another

environment to start a peer in JXTA network. This will allow easy simulations and convenient research platform.

Configuration is abstracted as a function that would create a peer configuration and join the group. If username and password are supplied at run-time, the peer is automatically configured to be a part of Net peer group. This will create the ".jxta" folder, which is consulted for all configuration information.

5.3.2 Invoke Server/Client Operation

Depending on whether it is a discovery request or a publish request, a Server or a Client process is started. A discovery request will allow server to create a pipe ID and listens on it. The client that wants to publish this resource would extract the pipe ID and creates an output pipe. Then the required file is sent though this pipe to the server. Server would save the file in a local folder. If it already has the resource and is trying to update, the last updated date of the file is considered before the transfer of file takes place.

5.3.3 Send File

File transfer can achieve many things depending on the nature of the file. Files can be java executable jar files; class files or exe executable files themselves. JXTA provides such a mechanism of sending messages in a secure and reliable way. Jinni, an agent engine comes as a lightweight jar file that is easy to transfer. Once the jar file is transferred from one peer to another, it is saved to a local folder. Jinni jar file comprises of class files that may be used for various agent activities. These class file methods are extracted by using Java Reflection API. Java provides a very strong reflection API that

enables to extract method, variable and class information in general from the class file. This will allow peer to create an agent container.

5.4 Agent Container

An agent container has an agent environment that facilitates communication, coordination and mobility of agents. Different approaches may be defined to develop the agent knowledge base and learning mechanisms [4].

Agent container has access to search requests, queries and resources published. Agents process the information in the database and can conclude in performing certain actions. For example, the processor may conclude that a file is last updated more than x number of days or so, it will generate a search request to notify user that a newer version is available.

Agent Container - Database interactions:

Agent Container and Database interactions can be observed in the figure 5.2. The three components JXTA layer, agent container and database are part of each peer in P2P network.

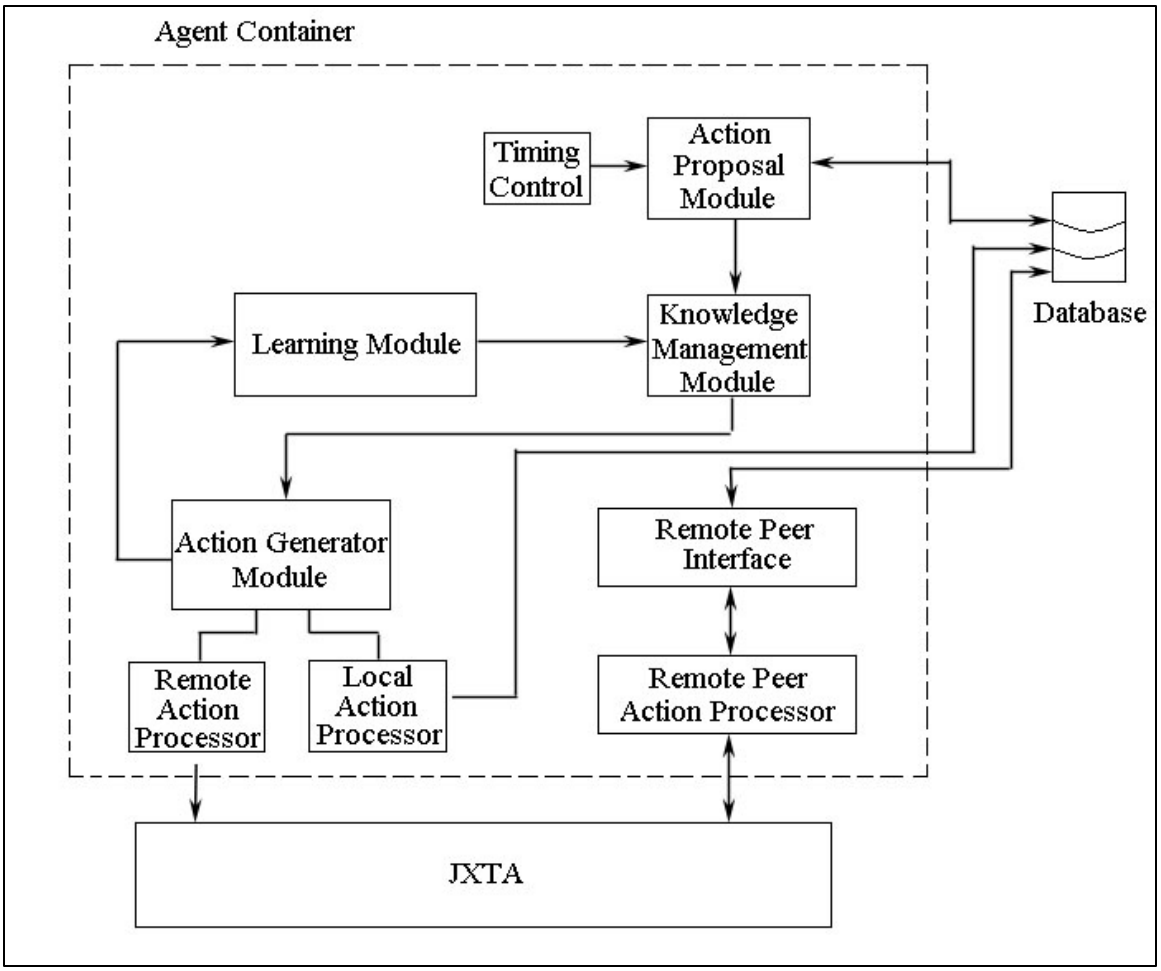

Figure 5.2: Block diagram showing logical modules in agent container

Agent Container has different logical modules that are implemented in a particular order. They are:

- Action proposal module
- Knowledge management module
- Action generator module
- Learning module
- Local action processor

- Remote action processor
- Remote listener

5.4.1 Action Proposal Module

Depending on the information from the database or mere timing parameters, an action proposal is made in this module. For example, if you consider an automatic file update mechanism, the Action proposal module collects data from database at certain intervals of time. Depending on the time last updated parameter, it would a request to knowledge management module to update a particular file.

5.4.2 Knowledge Management Module

Knowledge management module decides over if the action is to be performed or not. It would take its past experience using the learning module and decide if the action is necessary. In this case the action would be to update file. The criterion may be if the file is least used and has been updated many times, the proposal may be dropped. Else, the proposal with knowledge information is passed to Action generator module.

5.4.3 Action Generator Module

This module decides if the action required is local or remote. In this example, the action of updating file is remote, as it has to go onto network and search for availability of that file. A possible local action would be to check for a file checksum or ensuring security for local shared files. The feedback of the operation and log information is sent to Learning module.

5.4.4 Learning Module

This module supports knowledge management by providing the role of a repository of past events and responses. In update file mechanism, a learning module may perform actions like what kinds of files are requiring frequent updates and frequency of the operation.

5.4.5 Local Action Processor

After the action processor decides the action can be performed locally, it is sent to local action processor. The division between local action processor and remote action processor will allow to decide if a connection to JXTA network is required or not. The local action is performed and if there any update is to be made to the database it is made so.

5.4.6 Remote Peer Interface

This module listens for any requests using JXTA network layer. It also sends out responses to those requests. Example for a request would be a request for a file or last updated parameter of a file.

5.4.7 Remote Action Processor

Depending on the requests that are received by the remote listener, remote action processor contacts local database and then sends out the response.

5.4.8 Remote Listener

Remote listener listens to requests from other peer agents and upon getting one processes it. Often, such requests may also result in updating of database. Security should be a necessary factor in implementation of remote listener, as allowable actions on local

database should be defined before any remote process could do any damage to local machine.

5.5 Agent Communication

Apart from these agents can perform operations like accessing cache information of JXTA folder, processing it and generating appropriate actions. Different ways of agent communication are through Shared Blackboards, Message passing, remote procedure calls and Associative broadcast mechanism

5.5.1 Blackboard Mechanism

In the previous example of database interaction, synchronization and issues are addressed by Jinni's Blackboard mechanism.

Blackboards are shared entities, which look for certain patterns/objects on it and if found instantiate an action [2]. In other words, they will be able to trigger an action if a particular event occurs. They also can be viewed as thread coordinator and controller. These properties make it a good communication place for agents [1].

5.5.2 Associative Broadcast

Associative broadcast is used to target messages to processes in specific states, which are in turn used to form composition. Associative broadcast enables coordination based on every process in an interacting set locally maintaining common state necessary for collective decision procedures. Associative broadcast enables targeting of messages to processes in specific states and enables each process to select the properties of messages it will receive [5]. Associative broadcast is so-called because the model is similar to content addressing of associative memory. A target set specification travels with each

message that is broadcast onto the network. The target set is determined for each message by the set of local recipients whose local states match that of the target specification. Therefore, the sender does not need to know the identities of the targets in order to broadcast a message; the cardinality of the target set is completely transparent to the sender, even when the set is empty [6].

5.6 Application of Agent Design in P2P Networks

This section will discuss an application of how agent communication, database interaction and JXTA can bring more advanced features to P2P networks.

As an example, the search process can be improved in P2P networks by using P2P infrastructure in JXTA and agent capabilities in Jinni.

Peer groups can be created in JXTA network and each peer group offers services to peers in its group [3]. Essential part of a peer group is that it should at least have one rendezvous peer; at least one peer should not be behind a firewall and accept to the terms of group membership. A peer can be part of one or more peer groups simultaneously. In present day P2P systems, the propagation of search request is not efficient as it either searches for the entire network by flooding or searches in a particular local view of "super peers". This can be improved by scoping the search request propagation or publish request propagation a particular selection of groups. Not only this reduces the network traffic, but also improves the search result relevance as the search request in propagated in few selected groups.

Here is an example of such a system:

Peer groups are used to classify peers that are interested in a particular kind of resources. For example, a group may be interested in getting sports related or science related or world cultures related material. The peers have to find such groups and join or leave them accordingly. However, this process can be automated by using agent-database interactions, JXTA network layer and assuming a special peer group that provides indexing services.

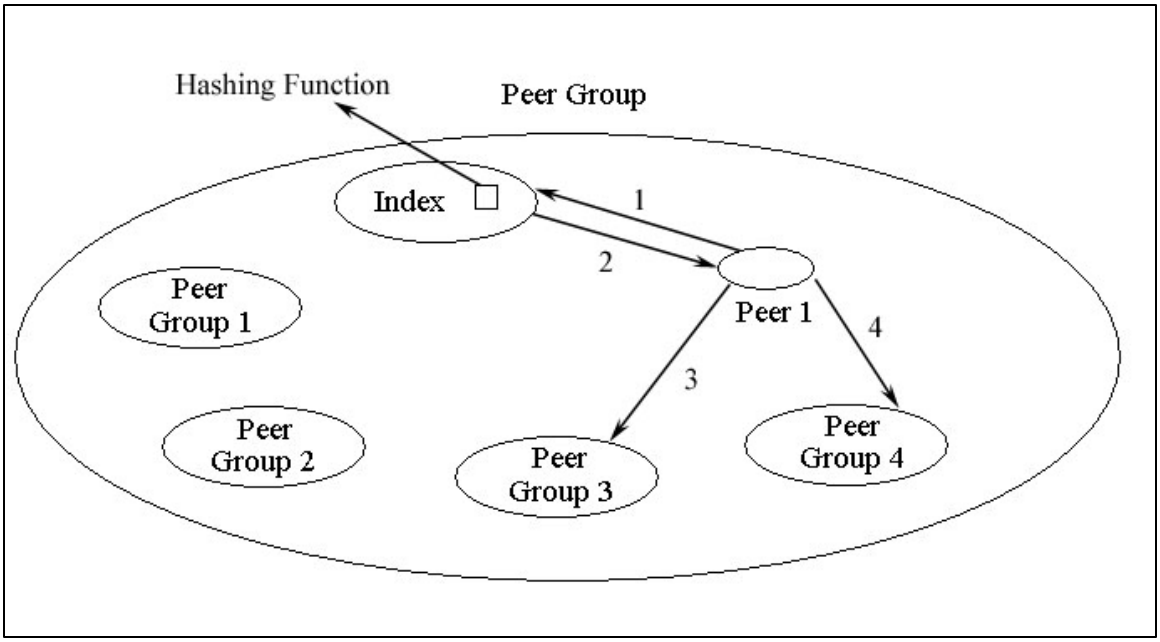

Figure 5.3: Application showing an example of proposed architecture

Figure 5.3 illustrates different steps in which agent communicates with index peer group (1), receives the list of related peer groups (2) and join peer groups (3) and (4). The figure does not show a case in which there may be overlaps in peer group formations. For instance, Peer group 1 and peer group 2 can fall into same category.

Agent database interactions are used to extract the searching behavior from the cache of the peer. All the search queries and results are cached in the database as a part of every

search process in JXTA. After a peer accumulates enough search terms, say X, the action proposed module invokes an agent to start process of optimization. The cache is searched for relevant search data and is passed to Knowledge management module. Knowledge management is assumed to be an intelligent entity with sufficient resources to know if a term occur in possible areas of groups. It is possible that a search term happens to be a valid terms in two or more valid areas. For example, a term like "jxta" may find matches in groups, which are interested in learning jxta, and also groups that are trying to build applications in jxta. Apart from that it may also have other groups that are not related to project JXTA. All these relevant possible search groups are retrieved by knowledge management module. This module will come up with finite number of possible group names the peer is interested to join in. After that, the action generator module generates a remote action of joining in Index services group and sends the possible group names to it. The request for searching in those group names is sent to this group. Index services group is any other peer group that offers indexing of all possible peer groups existing at that time as a group service. Using a hash key the Index services maps the available peer groups into groups. All these groups have description that helps in grouping the groups. These groups may be hierarchical and some times are combinations of existing groups. The Index services peer group sends the possible peer groups that match with the group name terms sent to it to the peer that requested them. Upon receiving the peer group names, the peer may join them using the peer group advertisement that is obtained from Index services group.

The peer receives all the advertisements that are published in that particular group. Depending on the relevance of the advertisements received by the peer, the peer may later decide to be or not to be part of a peer group. Filtering mechanism is adopted at each peer to ensure if the advertisements reaching the peer are the ones that are sought for. Basing on past search terms, the membership to groups also changes dynamically improving the efficiency of P2P networks. Publishing resources will also form a similar cycle. The only difference would be using the publish message cache rather than search request cache. It has always been found that using caching for such data yields efficient results.

The main factor that determines the efficiency of this system is the overhead in creating peer groups, joining and leaving them. This process is verified to be possible in JXTA. However, the performance issues are to be studied in detail.

The advantages of such a system are:

- Peer can join, leave groups and get only relevant information without human knowledge or intervention.
- Peer may filter the results, so that the groups they join are ones they are interested in.
- Efficient scoping environment for peers to publish or discover resources

Disadvantages of such a system are:

- Knowledge based mechanisms at each node have to have enough intelligence on how they decide relevant groups to join

- User behavior may not be dependent on past search criterion, in which case the number of groups he joins and leaves is very high. In such a case, there will be no positive outcomes of having the additional overhead of joining and leaving a peer group.

CHAPTER VI

CONCLUSIONS

Different P2P networks, their approaches towards different issues are compared and evaluated. Most of the P2P applications available concentrate more on distributing resources at faster rate, scalability issues, they ignore security issues and efficient search algorithms. After evaluating different features of existing P2P networks, JXTA is found to be more effective and reliable than others. The simple design, architecture and implementation in Java provide the best combination of platform to develop agent-based systems. Features of JXTA that enable it to be choice worthy P2P infrastructure are laid out. Advantages of an agent based P2P system and enhancements that can be made to P2P systems are discussed. Agent based database interaction environment is devised on top of JXTA platform.

Different agent communication mechanisms are discussed. Shared blackboards that are an integral part of Jinni, message passing, and associative broadcast mechanism are discussed. An agent based P2P network based on Jinni and JXTA is designed. Three important steps in doing so are discussed.

1. An Adaptor API that communicates between Jinni and JXTA is put in place. This API is used by Jinni to communicate with JXTA and perform its operations. All the communication features and data transfer features use this API to move from one peer to another. This API is used to devise an automated code update mechanism.

2. A Jinni/JXTA agent engine with database transactions is defined. Logical sections are defined to discuss the different processing blocks in agent container. Also, different communication ways in this agent architecture are discussed.

3. An example showing how this system would improve the efficiency of a P2P system is devised. This example shows the current architecture to provide a scoping region for search requests to reach different groups. Due to the presence of a mapping Index peer group, this design would decrease number of queries in the system. Different constraints of the model and its advantages are discussed.

Finally, P2P agent extensions are devised to improve the operation of P2P networks.

CHAPTER VII

FUTURE WORK

Implementation constraints, Performance issues are to be addressed in this Jinni/JXTA protocol. As the JXTA project is yet in its developing stage, more resources are required to discuss the issue of performance and scalability of these networks. As not all the enhancements that can be done to agent based systems are designed, this architecture can be applied to other problems faced by P2P networks. The Jinni/JXTA API can be improved to provide scope for different messaging systems to transparently pass though JXTA layer and communicate with agents on other peers. Also, different approaches towards network messaging like Associative broadcast can be implemented. JXTA builds the P2P infrastructure from network basics and will prove to be a promising place for developing agent based architectures.

BIBLIOGRAPHY

[1] "Extensions to Jinni Mobile Agent Architecture" Satyam Tyagi, Master Thesis, University of North Texas, 2001

[2] "A Logic Programming Based Software Architecture for Reactive Intelligent Mobile Agents", Paul Tarau, Editors: Van Roy P. and Tarau P., Proceedings of DIPLCL'99, http://www.binnetcorp.com/wshops/ICLP99DistInetWshop.html, Nov 1999

[3] “Project JXTA 2.0 Super-Peer Virtual Network", Bernard Traversat, Ahkil Arora, Mohamed Abdelaziz, Mike Duigou, Carl Haywood, Jean-Christophe Hugly, Eric Pouyoul, Bill Yeager, May 2003

[4] "An Advanced P2P Architecture using autonomous agents ", Hooman Homayounfar, Master Thesis, The University of Guelph, Jan 2002

[5] "Associative Broadcast and the Communication Semantics of Naming in Concurrent Systems”, B.Bayerdorffer, PhD. Dissertation, Dept. of Computer Sciences, Univ. Of Texas at Austin, Dec. 1993

[6] "Distributed Programming with Associative Broadcast", Bayerdorffer, B, Proceedings of the 28th Hawaii International Conference on System Sciences, January 1995.

[8] "JXTA Programming Guide", http://www.jxta.org/docs/JxtaProgGuide_v2.pdf, Project JXTA, May 2003

[9] "JXTA and Security", Navneeth Krishnan, Ch. 8 JXTA: Java P2P Programming, June 2002

[10] “Java Mobile Agents on Project JXTA Peer-to-Peer Platform”, Rita Yu Chen, Bill Yeager, 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9, 2003

[11] "Peer-to-Peer: Peering into the future", Jon Crowcroft, Ian Pratt, 2002

[12] “A Measurement study of Peer-to-Peer File Sharing Systems”, Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble, Technical Report # UW-CSE-01-06-02, Jan 2002

[13] “Improving Search in Peer-to-Peer Networks”, Beverly Yang, Hector Garcia-Molina, In Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), 2002

[14] "A Local Search Mechanism for Peer-to-Peer Networks", Vana Kalogeraki, Dimitrios Gunopulos, D. zeinalipour-Yazti, In Proceedings of the eleventh international conference on Information and knowledge management, pages 300—307, ACM Press, 2002

[15] "P2P based knowledge source discovery on research support system papits", Tadachika Ozono, Shoji Goto, Nobuhiro Fujimaki, Toramatsu Shintani, International Conference on Autonomous agents 2002.

[16] "Peer to Peer: Peering into the future", Jon Crowcroft, J J Thomson, May 2002

[17] "JXTA Protocols 1.2", Project JXTA, http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf, May 2003

[18] “Free riding on Gnutella”, Eytan Adar and Bernardo A. Huberman, First Monday journal, Sep 2000

[19] “Fasttrack Network”, http://www.fasttrack.nu, 2002

[20] “Peermetrics”, http://www.peermetrics.com/peer_system0.8/docs/guide/technical_overview.html, 2002

[21] "Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella", Evangelos P. Markatos, 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002

[22] "Designing Peer-to-Peer Applications: an Agent-Oriented Approach", D. Bertolini, P.Busetta, A. Molani, M. Nori and A.Perini, 2002

[23] "Windows Media Programming Reference", http://msdn.microsoft.com/library/default.asp?url=/library/ens/wmrm/htm/programmingreference.asp, 2003

[24] "Escrow Services and Incentives in Peer-to-Peer Networks", Bill Horne, Benny Pinkas, Tomas Sander", 2001

[25] “Peer-to-Peer Architecture Case Study: Study Gnutella Network”, Matei Ripeanu, Technical Report, Computer Science Department, The University of Chicago, 2001

[26] "Digital Watermarks for Audio Signals", Laurence Boney, Ahmed H. Tewfik, Khaled N. Hamdy, International Conference on Multimedia Computing and Systems, pp 473-480, 1996

[27] “Anthill: A framework for the development of agent-based peer-to-peer systems”, Ozalp Babaoglu, Hein Meling, Alberto Montresor, Technical Report UBLCS-2001-09, 2001

[28] “Engineering an Agent-Based Peer-To-Peer Resource Discovery Systems”, Andrew Smithson, Luc Moreau, 2002

[29] “Jini network technology”, Sun Microsystems, http://wwws.sun.com/software/jini/whitepapers/architecture.html, Jan 1999