ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

IMPLEMENTATION OF JXTA BASED PEER- TO-PEER COLLABORATIVE SYSTEM

A case on Improved Patient care Service at Jimma Hospital and the Surrounding Health Institutes

By
Fisseha Bayu

A Project paper submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science

June, 2008

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

IMPLEMENTATION OF JXTA BASED PEER- TO-PEER COLLABORATIVE SYSTEM

A case on Improved Patient care Service at Jimma Hospital and the Surrounding Health Institutes

By

Fisseha Bayu

Acknowledgement

First and foremost, I would like to express my deepest sense of appreciation to my advisor Dr. Dejene Ejigu for his unlimited guidance, valuable suggestions, encouragement, advice and support till the completion of this project.

I am thankful to my friends, who have helped me in proof reading this project and participating in the experiment, by providing their PC's for testing my project.

Last but not least, I take this opportunity to express my gratefulness to my families for their continuous moral support and patience during my study.

Table of Contents

<u>List of Tables</u>

List of Figures

Acronyms

API	Application Programming Interface
JBPTPCS	Jxta Based Peer To Peer Collaborative System
CMS	Content Management Service
DFD	Data Flow Diagram
IM	Instant Messaging
NAT	Network Address Translation
P2P:	Peer to Peer
PBP:	Pipe Binding Protocol
PDA	Personal Digital Assistance
PDP:	Peer Discovery Protocol
PEP:	Peer Endpoint Protocol
PIP:	Peer Information Protocol
PRP:	Peer Resolver Protocol
RVP:	RendezVous Protocol

Abstract

Communication among people is now becoming involved in day to day activities of people. On a daily basis, people use a wide range of electronic services to enhance their lives, to communicate with friends and colleagues, to make travel reservations, to exchange resources, to exchange messages, or to access information. To this end, the core objectives of this project work are to exchange messages and share files among JXTA peers. During this process, one does not need to have prior knowledge about the existence of the resource and the location where it exists.

JXTA technology provides the mechanism by which peers can search a shared content through its discovery service so that peers can locate and download the shared content easily. JXTA technology helps us to discover shared resources around the globe, even resources behind fire wall or NAT device.

Peers that need to share a resource with other peers publish advertisements locally and remotely. Other peers discover the name of the advertisement in order to open a pipe to communicate and download content among them.

In this work, we have developed a JXTA based peer to peer collaborative system (JBPTPCS). It implements a peer-to-peer system that helps peers to discover each other and each others' resources, share the resources and exchange messages within the local area setting. Among the functionalities of JBPTPCS are:

- Discover peers, peer groups, shared contents, and advertisements.
- Download shared contents.
- Exchange unicast, bidirectional, and multicast messages among peers.

Chapter One

Introduction

1.1 Background

The term peer-to-peer (P2P) refers to a class of systems and applications that utilize distributed resources to perform a critical function in a decentralized manner. The resources include computing power, data, network bandwidth, computers, and other resources. The critical function can be distributed computing, data/content sharing, communication or collaboration. A peer gives some resources and obtains other resources in return. Conceptually, P2P computing is an alternative to the centralized and client-server models of computing, where there is typically a single or small cluster of servers and many clients [1, 6].

The main advantage of P2P networks is that they distribute the responsibility of providing services among all peers on the network; this eliminates service outages due to a single point of failure and provides a more scalable solution for offering services.

As with any computing system, the goal of P2P systems is to support applications that satisfy the needs of users. Selecting a P2P solution is often driven by one or more of the following goals [3]:

• Cost sharing/reduction. Centralized systems that serve many clients typically bear the majority of the cost of the system. When that main cost becomes too large, a P2P architecture can help spread the cost over all the peers.

• Improved performance and interoperability. Each node in the P2P system brings with it certain resources such as computing power or storage space. By aggregating computing resources at thousands of nodes, P2P systems are able to perform computationally intensive functions.

• Improved scalability. Scalability is limited by factors such as the amount of centralized operations that needs to be performed and the amount of systems that needs to be maintained.

Since most operations in client/server architecture are done on the server, there is a limit on the number of processing to be formed on a server and hence this limits scalability. Therefore, with the lack of strong central authority, improved system scalability can be achieved.

• Collaboration. Collaborative P2P applications allow users to collaborate in real time, without relying on a central server to collect and relay information.

• Enabling ad-hoc communication and collaboration. Related to dynamism is the notion of supporting ad-hoc environments. By ad hoc, we mean environments where members come and go based perhaps on their current physical location or their current interest.

Specifically when we consider health institutes, according to [6], documentation and data exchange are some of the most time consuming activities. Data from different health institutes or professionals are not linked together, so it is not possible to collect all entries of a corresponding patient and to create a complete patient data set. Medical history including previous examination results and medications by other physicians are important for current patient treatment. Moreover, information from laboratories, hospitals, and pharmacies are desired. Hence, the aim is to share a patient information online using P2P architecture. The secure sharing of medical/clinical data including medication, laboratory results, test results etc helps to provide better treatment to the patient. The following are among the advantages of such systems.

- Data exchange among health institutes and physicians is possible.
- Encourages electronic patient record keeping tradition.
- Reduce the amount of hand written data exchange and minimizes possible manual exchange error.

1.2 Statement of the Problem

P2P systems are designed with the goals of decentralization, ad-hoc connectivity, and reduced cost of ownership. P2P has more decentralized control and data compared to client/server model; it supports systems whose parts can come and go and can communicate in an ad-hoc manner; and the cost of ownership is distributed among the peers. Compared to P2P, client-server systems have centralized points of control and data at the servers.

Most Internet services are distributed using the traditional client/server architecture. In this architecture, clients connect to a server using a specific communications protocol, such as the File Transfer Protocol (FTP), to obtain access to a specific resource. Most of the processing involved in delivering a service usually occurs on the server, leaving the client relatively unburdened. Most popular Internet applications, including the World Wide Web, FTP, telnet, and email, use this service-delivery model [1].

The client/server architecture has a major drawback. As the number of client's increases, the load and bandwidth demands on the server also increase, eventually preventing the server from handling additional clients. The client in the client/server architecture acts in a passive role, capable of demanding services from servers, but incapable of providing services to other clients. This architecture is, therefore, clearly unable to utilize the resources such as data, available disk space, and computation power of the client's machine.

On the other hand, although, there are a number of P2P solutions that will be explained in the next chapter, the proprietary and specialized nature of the existing solutions highlights the need for a standard set of protocols to address the particular requirements of the P2P domain.

In this work, we try to demonstrate sharing of resources and communication among peers using JXTA technology in a local area network. This work can be applied to peers in different health centers with a minimum configuration change.

1.3 Objective

1.3.1 General Objective

The main objective of this project is to develop and implement a per-to-peer collaborative system that uses JXTA technology and services.

1.3.2 Specific Objective

The specific objectives of the project include:

- Study the mechanism how peers are discovered in a network.
- Study the mechanism how peer groups are discovered in a network.
- Study the mechanism how peers join a certain peer group.
- Study the mechanism how peers exchange message using pipes and sockets.
- Study the mechanism how advertisements are discovered in a network.
- Study the mechanism how contents are transferred among peers using pipes.
- Study the capabilities of JXTA technology and its services to apply it to meet the objective of the project.
- Develop prototype.

1.4 Methodology

Patient data exchange in health institutes and information exchange among health professionals was explored. Additionally, a literature review was done on applications that were done on peer to peer systems.

1.4.1 Data collection

To determine system requirement, secondary data from different sites, which provide health information exchange, has been collected and reviewed. Additionally, observation was made on some health institutes.

1.4.2 Documentation tools and programming language

Microsoft word, Microsoft paint, and Rational Rose were used as a documentation tool. The application was developed using JXTA, java reference implementation.

1.4.3 Prototype Implementation

We have implemented a peer-to-peer system that helps peers to discover each other and each others' resources, share the resources, and exchange messages within the environment of local area network.

1.5 Significance of the Work

According to [12, 13], the exchange of medical documents help to minimize adverse drug interactions and allergies, prevent errors in dosing, speed up retrieval of patient information and reduce unnecessary paperwork. In conclusion, it stated that the more information a physician has at the time of decision making, the better decision he can make about the patient.

Since JXTA is a powerful tool for cross-organizational interactions and networking, it makes the exchange of medical documents among health institutes easier. Also, its grouping service improves the ability of the healthcare workers to exchange views and to cope with dynamic situation, which in turn makes it possible to offer more efficient medical services. As a result, professionals in different health organization and patients treated in different health centers can gain a lot. In general the benefit of the project can be seen from different perspectives:

- Patients:-
    - They get better medication by easily referencing their past history.
    - Are not required to take medication result from one health center to another health center, from one laboratory to health center etc.
- Physicians:-
    - They can easily refer history data of patients from different health centers to give better treatment.
    - They can exchange ideas, views and multimedia data with other physicians in other health centers.
- Institutes :-
    - Can get help from a specialized physician working in other health center.
    - Better capture records of patients.
- Developers:-
    - Learn more about the capability of this new technology.

1.6 Limitations

According to [1], JXTA pipes are not reliable. In JXTA technology almost all advertisements are obtained through discovery, by initiating discovery service. Advertisements are the way by which peers exchange status information, by which resources are shared, by which pipes are created to propagate or transmit information. Peers send discovery requests to obtain advertisements. Discovery response for the discovery request can be none, one or more. When there is no response it does not mean that the requested service is not there. But, it can be because of unreliable service the technology provides. Similarly, because of unreliable means of communication the response may not reach to the intended peer.

1.7 Organization of the Document

This document contains seven chapters including this chapter. Chapter two presents review of applications and related works developed on a P2P system. Chapter three defines and describes concepts with regard to JXTA technology and services. Besides, functions of the application are described to give a general view about the services the application provides. In chapters four, we present the analysis of the system. Chapter five discusses the design of the developed system. Chapter six deals with the implementation of the system. Finally, Chapter seven presents conclusions and future works

Chapter 2

Literature Review and Related Works

2.1 Definition

According to [9], P2P is "A communications model in which each party has the same capabilities and either party can initiate a communication session". According to [14], "In general P2P describes an environment where computers (hosts) connect to each other in a distributed environment that does not use a centralized control point to route or connect data traffic." So we could define P2P as direct communication or collaboration (mostly file-sharing) between computers, where none are simply client or server, but all machines are equals - peers. With this definition, communication between two servers is P2P.

Lots of P2P applications were developed to fulfill a specific requirement. Here we try to review some prominent P2P applications.

2.2 Usenet

According to [1], Usenet was created in 1979 to provide a way for two computers to exchange information in the early days. Their first iteration allowed a computer to dial another computer, check for new files, and download those files; this was done at night to save on long-distance telephone charges. The system evolved into the massive newsgroup system that it is today. Usenet has a few properties that help distinguish it as probably the first P2P application. It has no central managing authority—the distribution of content is managed by each node, and the content of the Usenet network is replicated (in whole or in part) across its nodes. Usenet is a way for machines to talk to each other to allow news messages to be posted and disseminated over a network. Since Usenet, the most popular P2P applications have fallen into one of three major categories: instant messaging, file sharing, and distributed computing.

2.3 ICQ

Mirabilis released ICQ [1], a P2P instant messaging (IM) application in November 1996, which gave its users a faster way to communicate with friends than traditional email. ICQ allows users to be notified when their friends come online and to send instant messages to their friends. In addition to its main capability of instant messaging, ICQ allows users to exchange files. Though classified as a P2P application, ICQ relies on a hybrid of the P2P and client/server architectures to provide its service. ICQ uses a central server to monitor which users are currently online and to notify interested parties when new users connect to the network. All other communication between users is conducted in a P2P fashion, with messages flowing directly from one user's machine to another's with no server intermediary.

2.4 Napster

Napster [1] burst onto the Internet stage in 1999, providing users with the capability to swap MP3 files. Napster employs a hybrid P2P solution similar to ICQ, relying on a central server to store a list of MP3 files on each user's machine. This server is also responsible for allowing users to search that list of available files to find a specific song file and its host. File transfer functionality is coordinated directly between peers without a server intermediary. In addition to its main file-sharing functionality, Napster provides a chat function to allow users to send text messages to each other.

According to [1, 8, 10] Napster doesn't provide a complete solution for bypassing firewalls; it is capable of traversing only a single firewall. Each peer acts as a simple router, capable of sending content to a fire walled peer when a request is made via HTTP. Napster provides no message-routing capabilities, meaning that simple peers on the network can’t act as router peers to enable other peers to perform double firewall traversal.

2.5 Gnutella

The Gnutella project [1, 4, 11] took the file-sharing concept pioneered by Napster one step further and eliminated the need for a central server to provide search functionality. The Gnutella network’s server independence, combined with its capability to share any type of file, makes it one of the most powerful demonstrations of P2P technology. Peers on the Gnutella network are responsible not only for serving files, but also for responding to queries and routing messages to other peers. Note that although Gnutella doesn’t require a central server to provide search and IP address resolution functionality, connecting to the Gnutella network still requires that a peer know the IP address of a peer already connected to the P2P network. For this reason, a number of peers with static or resolvable IP addresses have been established to provide new peers with a starting point for discovering other peers on the network.

In the Gnutella network, each peer acts as a simple peer, a rendezvous peer, and a router peer, using TCP for message transport and HTTP for file transfer. Searches on the network are propagated by a peer to all its known peer neighbors, which then propagate the query to other peers. Advertisements for content on the Gnutella network consist of an IP address, a port number, an index number identifying the file on the host peer, and file details such as name and size. Gnutella peers don’t provide full router peer capabilities, which mean that, as with Napster, Gnutella peers are capable of traversing only a single firewall [1, 8].

2.6 Freenet

Freenet [1, 2, 8] is a completely distributed decentralized P2P system. It has no notion of global coordination at all. Communication is handled entirely by peers operating at a global level. The system operates as a location-independent distributed file system across many individual computers that allow files to be inserted, stored, and requested anonymously. A node is simply a computer that is running the Freenet software, and all nodes are treated as equals by the network. Each node maintains its own local data store which it makes available to the network for reading and writing, as well as dynamic routing table containing addresses of other nodes and the keys that they are thought to hold. This removes any single point of failure or control. By following the Freenet protocol, many such nodes spontaneously organize them-selves into an efficient network. Freenet enables users to share unused disk space, but it does not have a search system.

2.7 HealthNet Ethiopia

According to [13] HealthNet Ethiopia is established in 1994 in collaboration with Addis Ababa University Medical School, which hosts the network. Initially, departments and units within the Faculty of Medicine were connected. Gradually, however, the Medical Library took over the operation and began to market the service throughout the country.

HealthNet is being used by health professionals for their own professional development, for teaching and enriching lecture notes, for research, and most importantly, for clinical service.

Through HealthNet Ethiopia, thousands of health practitioners throughout Ethiopia have access to a wide range of information services that are critical to health care delivery. Health workers benefit from these services in a variety of ways:

- They exchange disease related information, hold consultations, collect and send epidemiological data and ask questions in search of solutions to specific health problems.
- They schedule consultations and referrals, making it unnecessary for ill patients to travel long distances with no guarantee of seeing a physician.
- They subscribe to electronic medical and health publications and search for information through Get Web, a tool developed by SATELLIFE that allows users to download content from the World Wide Web with simple e-mail commands.
- Medical students and faculty exchange project documents, thesis and other scholarly articles with colleagues around the world.
- The NGO community uses HealthNet to exchange reports on activities and on-going projects with international headquarters.

HealthNet Ethiopia has continued to increase its community, penetrating all health settings where a computer with a modem and a telephone line is available. Today, there are 62 points that are connected and making use of HealthNet's services. These points cover a wide geographical area and include hospitals, medical schools, non-profit organizations, clinics, health research centers and individual health practitioners at their homes or offices.

Addis Ababa University has launched a university-wide networking initiative that will interconnect six campuses in Addis, including the Medical School where HealthNet Ethiopia is housed. This technical upgrade will enable HealthNet Ethiopia to significantly expand and upgrade its services to a larger community of health professionals throughout the country.

2.8 Smart Card

Currently Ministry of health is developing software that helps to record patient’s history in a smart card for government health institutes. A patient that gets medication from any governmental health institution will be given a smart card with all the information recorded on the smart card. Any government health institute can refer the medical history of the patient from the card. Currently, the project is under construction and is developed by a foreign company. Since it is not yet deployed, there is no document to refer at, and hence, this information is obtained from members of IT department of the ministry. With smart card software, clearly, we can see that there is still a burden on the patient side to carefully and securely hold the card. Besides, other health institutes except government ones are not supported by this project.

2.9 Summary

Unfortunately, the current applications of P2P tend to use protocols that are proprietary and incompatible in nature, reducing the advantage offered by gathering devices into P2P networks. Each network forms a closed community, completely independent of the others and incapable of leveraging their services [1].

In summary, according to [1], to evolve P2P into a mature solution platform, developers need to refocus their efforts from programming P2P network fundamentals to creating P2P applications on a solid, well-defined base. To do this, P2P developers need a common language to allow peers to communicate and perform the fundamentals of P2P networking. Realizing this need for a common P2P language, Sun Microsystems formed Project JXTA (pronounced *juxtapose* or *juxta*) to design a solution to serve all P2P applications.

With this motivation, we undertook this project to develop a P2P system using JXTA technology that can improve patient care service among health institutes that may use different platforms and operating systems and facilitates information exchange among health professionals and benefits any health institutes that is governmental or non governmental.

Chapter 3

Overview of JXTA

3.1 Introduction

According to [1, 13], JXTA is an open, network computing platform designed for P2P computing. It provides a base P2P infrastructure over which other P2P applications can be built. It was originally conceived by Sun Microsystems designed with the participation of experts from academia and industry.

JXTA allows flexible organization, distribution of role to participating peers, and the ability to share information with the other peers for group collaborations. JXTA is a P2P application development infrastructure that enables developers to easily create service-oriented software. The JXTA grouping service facilitates to share data in a secure manner under P2P environments. Besides, P2P architecture is being used to explore better the computing power and bandwidth of networks than client/server architecture.

According to [1], in the same manner that the Internet provides domain name lookup (DNS), World Wide Web, email, and other services by spreading responsibility among millions of servers, P2P has the capacity to power a whole new set of robust applications by leveraging resources spread across all corners of the Internet.

The JXTA platform defines a set of protocols designed to address the common functionality required to allow peers on a network to form robust pervasive networks (a network of connected small devices such as PCs, workstations, servers, personal digital assistants (PDA), sensors, mobile phones etc), independent of the operating system, development language, and network transport employed by each peer.

According to [1, 15, 16], JXTA is a set of six protocols for P2P systems as shown in Figure 3-1. P2P systems must implement at least one of these protocols to work with other JXTA systems.

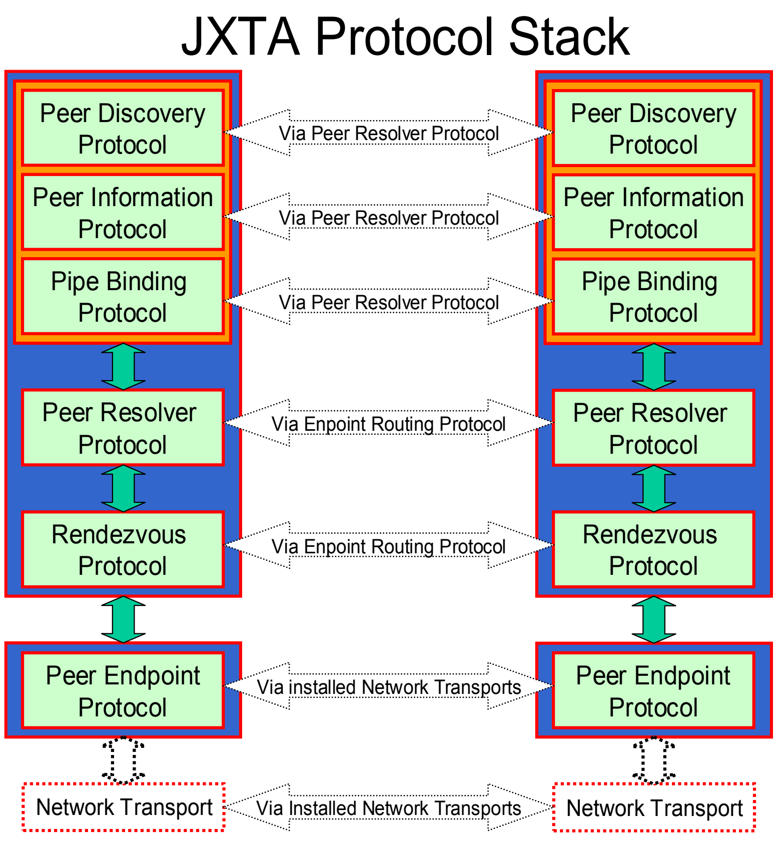

JXTA Protocol Stack
Peer Discovery Protocol
Peer Information Protocol
Pipe Binding Protocol
Peer Resolver Protocol
Rendezvous Protocol
Peer Endpoint Protocol
Network Transport
Via Peer Resolver Protocol
Via Enpoint Routing Protocol
Via installed Network Transports
Via Installed Network Transports

Figure 3-1 JXTA protocol stack [15]

3.2 JXTA protocols and services

3.2.1 The Peer Resolver Protocol (PRP)

- enables a peer to implement high-level search capabilities
- allows a peer to send and receive generic queries to find or search for peers, peer groups, pipes, and other information

3.2.2 The Peer Discovery Protocol (PDP)

- A peer uses the PDP to advertise and discover a JXTA resource
- resources are described by advertisements e.g. can be services, pipes, peers, peer groups, or any other advertisements
- Using this protocol, peers can advertise their own resources, and discover the resources from other peers
- Peer resources are published using XML-based advertisements

3.2.3 The Peer Information Protocol (PIP)

- PIP uses to gather information from a discovered peer.
- Allows peers to learn about the capabilities and status of other peers e.g. traffic load, capabilities, state etc e.g. one can send a ping message to see if a peer is alive.
- Also query a peer's properties where each property as a name and a value string.

3.2.4 The Pipe Binding Protocol (PBP)

- PBP is used by peers to bind a pipe advertisement to a pipe endpoint
- allows a peer to establish a virtual communication channel (i.e. a pipe) between peers
- allows the binding of the two or more ends of the pipe endpoints forming the connection
- a peer binds a pipe advertisement to a pipe endpoint thus indicating here messages actually go over the pipe

According to [1], a communication pipe is similar to a water pipe. Information gets inserted at one end and is received at the other. Pipes are described using a pipe advertisement. Currently there are three types of pipes: Unicast, UnicastSecure, and Propagate.

3.2.5 The Peer Endpoint Protocol (PEP)

- PEP uses to create a communication channel. The PBP uses this channel to communicate with other peers.
- allows a peer to find information about the available routes for sending a message
- allows the implementation of routing algorithms into JXTA
- Peers implementing the endpoint routing protocol respond to queries with available route information giving a list of gateways along the route.

3.2.6 The RendezVous Protocol (RVP)

- RVP is used to propagate JXTA messages through a rendezvous peers
- allows a Peer to send messages to all the listeners of the service
- The rendezvous protocol defines how a peer can subscribe or be a subscriber to a propagation service allowing larger communities to form
- A rendezvous nodes' scope is a peer group e.g. the rendezvous protocol is used by the peer resolver protocol and by the pipe binding protocol in order to propagate messages.

According to [1], generally JXTA technology enables

- Heterogeneous devices with completely different software stacks can interoperate through JXTA protocols
- Implementation over TCP/IP, HTTP, Bluetooth, etc

Chapter 4

System Analysis

System analysis is the preliminary phase of a software project. During this phase, the feasibility of the project is analyzed. According to [5], system analysis is the process of breaking down an entire system into modules, analyzing each module separately, and determining the relationships between them. System analysis involves examining a business activity to identify the problem domains and to suggest alternative solutions. In this chapter, the functional and non-functional requirements of the system were described and modeled using UML models.

4.1 Current System

Currently, there is no peer to peer application developed among health institutes surrounding Jimma hospital. When a patient takes medication in a new health center, there is no way of getting the patients medical history from the other health center, where the patient was already got medication previously. It is only during referral cases that a patient has his pervious history along with him. Whenever a patient moves from one health center to another he/she is getting treatment as a new patient.

4.2 Proposed System

4.2.1 Overview of the new System

The new system solved the problems described in the existing system. The new system made information exchange possible by creating a peergroup that incorporates interested health centers to exchange patient information among them. Besides, professionals working in different health centers can get a chance to exchange information. For example a health center that lacks a specialist in a certain field can communicate with another health center with that specialist to exchange views through bidirectional message exchange (chat room) functionality. In addition they may exchange multimedia data by using the content download functionality.

4.2.2 Functional Requirements

According to [7], functional requirements describe the activities and services an information system or an application needs to provide. Functional requirement is a function or service needed in an information system to satisfy business needs. Functional requirements correspond to the actions performed in the system. The JBPTPCS system provides the following functionalities:

- The system should be able to discover JXTA peers.
- The system should be able to provide service to discover and join a JXTA peer group.
- The system should provide unicast, bidirectional, and multicast message exchange service.
- The system should be able to discover and download shared contents from JXTA peers.

4.2.3 Non-Functional Requirements

According to [7], non functional requirements correspond to the process of explaining the features, characteristics, attributes, and constraints of the information system used to limit the boundaries of the proposed solution. The non functional requirements deal on performance, efficiency, reliability, flexibility, and expandability. The non-functional requirements of the JBPTPCS system are the following.

- The system must be easy to use and user friendly.
- The system should be readable (i.e. code readability).
- The system should be extendible.

4.3 Analysis Model

To produce a model of the system that captures all functionalities and that eliminates contradictory requirements, we need to construct the analysis model based on the requirements of the system. According to [5], system analysis contains three models: functional, object and dynamic models. The functional model is described by use case diagrams, object model is described by class diagrams, and the dynamic model is described by sequence, state chart and activity diagrams. In this project the analysis model is described with use case, sequence, class, and activity diagrams.

4.3.1 Use case Diagram

A use case diagram indicates the interaction of a user, or an external system that interacts with our system. The use case diagram is used to define the functions or services that makeup a system. Use case diagrams can be used to show all of its available functionality of the system. According to [7 and 18], a use case is written to show or represent all the interactions occurring between the user and the system. It enables customers and developers to clearly understand the purpose and scope of the system before it is developed. The following use cases are identified in the JBPTPCS application.

- Start JBPTPCS: - this functionality initiates the system.
- Discover peers: - this functionality helps to discover the JXTA peers connected to this peer.
- Discover peerGroup: - this use case provides the list of peer groups.
- Create peerGroup:-this use case provides a service to create a new peer group.
- Join peerGroup: - using this functionality a peer joins one of the peer groups discovered.

- Create chat: - this use case provides chat room service through which two peers are able to communicate online.
- Send propagate message: - this use case provide the functionality for a peer to send a multicast message to multiple peers.
- Send unicast message: - this use case provides a service to send a unicast message to a single peer.
- Discover sharedFile: - this use case provides a service by which a peer searches a list of contents shared by peers.
- Download file:- this use case provides a functionality that helps a peer to download a file obtained by discover sharedFile use case.

The use case diagram is depicted in Figure 4-1 below.

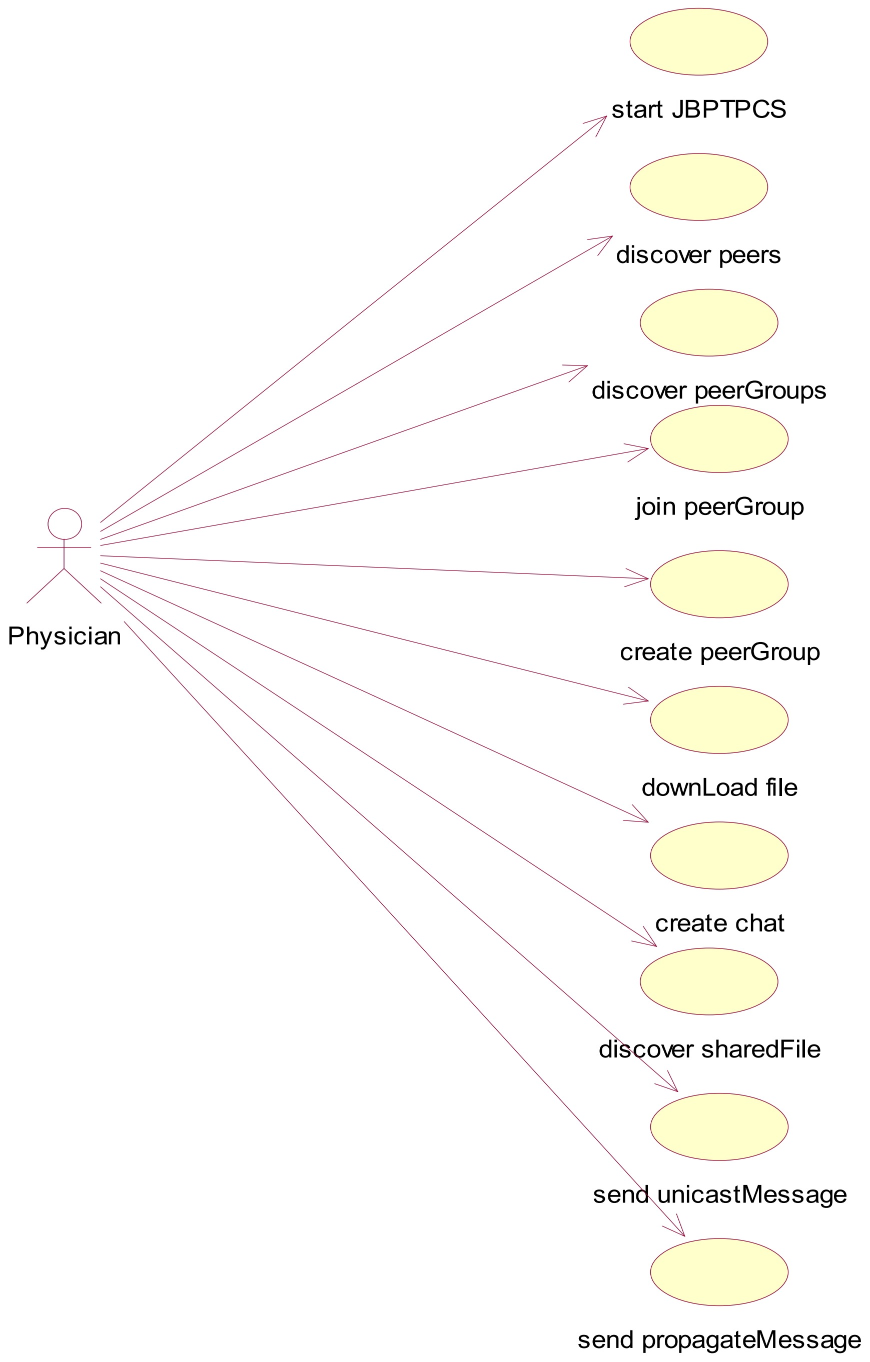


Figure 4-1UML use case diagram for the JBPTPCS system

4.3.2 Description of the identified use cases

The descriptions of the use cases are shown below from Table 4-1 through Table 4-8.

Table 4-1 Description of start JBPTPCS use case

1. Use Case Name :	Start JBPTPCS
Actor :	Physician
Pre condition :	There must be JBPTPCS icon on the desktop.
Flow of Events :	i. The use case starts when a Physician clicks on JBPTPCS icon from the desktop. ii. The system displays a Main Menu. iii. The system waits for the user to click any button to process.
Post condition	The system is ready for use.

Table 4-2 Description of discovers peers use case

2. Use Case Name :	Discover Peers
Actor :	Physician
Pre condition :	The start JBPTPCS must be started
Flow of Events :	i. The use case starts when a Physician clicks on discover peers button from the Main Menu. ii. The system displays discover peers form. iii. The Physician clicks on discover peers button from the form. iv. The JXTA discovery service launches a discovery service to get JXTA peers on the network. v. The system displays a list of found JXTA peers.
Post condition	List of JXTA peers are displayed.

Table 4-3 Description of discover peerGroups use case

3. Use Case Name :	Discover peerGroups
Actor :	Physician
Pre condition :	Start JBPTPCS use case must be started
Flow of Events :	i. The use case starts when a Physician clicks on discover peerGroup button from the Main Menu. ii. The system displays discover peer groups form. iii. The Physician clicks on discover peer groups button from the form. iv. The JXTA discovery service launches a discovery service to get JXTA peer groups on the network. v. The system displays a list of found JXTA peer groups
Post condition	List of JXTA peer groups are displayed.

Table 4-4 Description of Download files use case

4. Use Case Name :	Download file
Actor :	Physician
Pre condition :	Discover sharedFile use case must be started
Flow of Events :	i. The use case starts when a Physician clicks on download file button from the Main Menu. ii. The system displays a form with a set of instruction. iii. The Physician clicks on download file on the form. iv. The system displays a dialog box to enter yes/no. v. The Physician enters yes/no response. vi. The system displays a dialog box to enter a file name. vii. The Physician enters the file name to download. viii. The JXTA discovery service launches a search mechanism to search for shared file. ix. If found, it displays the name and number of found files and download it.
Post condition	The stated file is downloaded.

Table 4-5 Description of Join peerGroup use case

5. Use Case Name :	Join peerGroup
Actor :	Physician
Pre condition :	Discover peerGroup use case must be started
Flow of Events :	i. The use case starts when a Physician clicks on join PeerGroup icon from The Main Menu window. ii. The system displays a peerGroup list. iii. The Physician views and select a peer group. iv. The system displays a dialog box v. The Physician enters the peer group name. vi. The system joins the peer to the selected peer group.
Post condition	The peer becomes member of the selected peer group.

Table 4-6 Description of Create Chat use case

6. Use Case Name :	Create chat
Actor :	Physician
Pre condition :	Start JBPTPCS use case must be started
Flow of Events :	i. The use case starts when a Physician clicks on chat icon from The Main Menu window. ii. The system displays a chat form. iii. The Physician clicks on start chat button on the form. iv. The system displays a dialog box v. The Physician enters his chat name. vi. The system reloads the chat form. vii. The discovery service search for the advertisement. viii. If successful displays a success message on the chat form. ix. The Physician starts chat
Post condition	Chat room is created.

Table 4-7 Description of create peerGroup use case

7. Use Case Name :	Create peerGroup
Actor :	Physician
Pre condition :	Start JBPTPCS use case must be started
Flow of Events :	i. The use case starts when a Physician clicks on create peerGroup icon from The Main Menu window. ii. The system displays a create peerGroup form. iii. The Physician clicks on create peerGroup button from the form. iv. The system displays a dialog box. v. The Physician enters the peerGroup name. vi. The system creates a new peerGroup.
Post condition	New peer group is created.

Table 4-8 Description of Discover sharedFile use case

8. Use Case Name :	Discover sharedFile
Actor :	Physician
Pre condition :	Start JBPTPCS use case must be started
Flow of Events :	i. The use case starts when a Physician clicks on discover sharedFile icon from The Main Menu window. ii. The system displays a Shared File form. iii. The Physician clicks on discover shared file button from the form. iv. The discovery service searches shared file from JXTA peers. v. If successful, display list of discovered shared files. vi. The Physician views the shared files from the form.
Post condition	List of shared files are displayed.

4.3.3 Sequence Diagram

UML sequence diagrams model the flow of messages within our system in a visual manner, and are commonly used for analysis purposes. They're called sequence diagrams because of the sequential nature of the ordering of the messages (the horizontal arrows).

UML sequence diagrams are used as a dynamic modeling technique, as are collaboration diagrams and activity diagrams. According to [19, 20], sequence diagrams are used to:

1. Validate, describe and show the way that how our system is used.
2. Explore our design because they provide a way for us to visually step through invocation of the operations defined by our classes.
3. Detect bottlenecks within an object-oriented design. By looking at what messages are being sent to an object, and by looking at roughly how long it takes to run the invoked method, we quickly get an understanding of where we need to change our design to distribute the load within your system.
4. Identify new responsibilities for classes and objects, and, sometimes, even new classes. Its implication is that it helps us to update our class model appropriately.
5. Show the interactions between objects in the sequential order that those interactions occur.

The sequence diagrams for the identified use cases are shown below from Figure 4-2 through Figure 4-8.

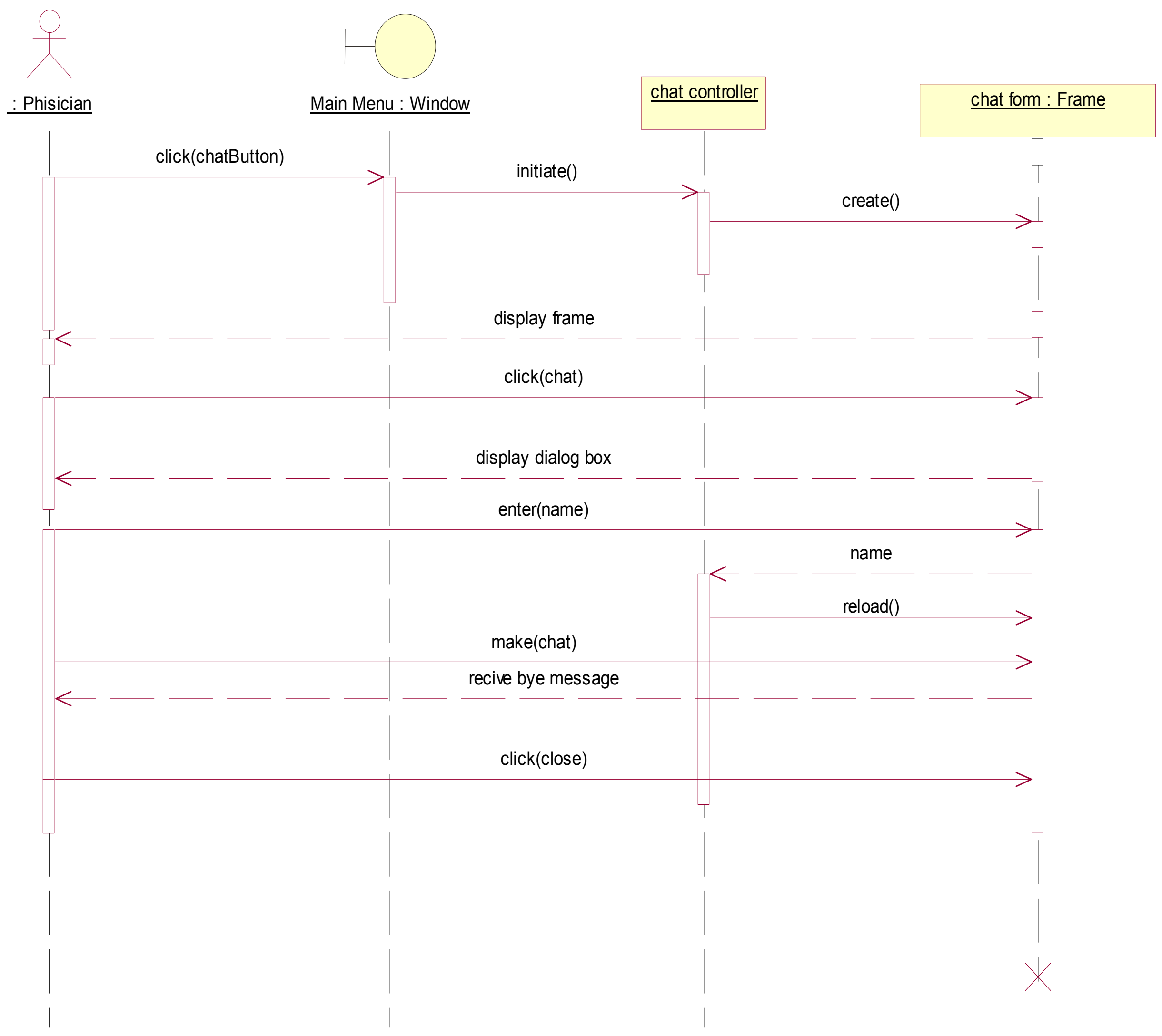


Figure 4-2 UML sequence diagram for create chat use case

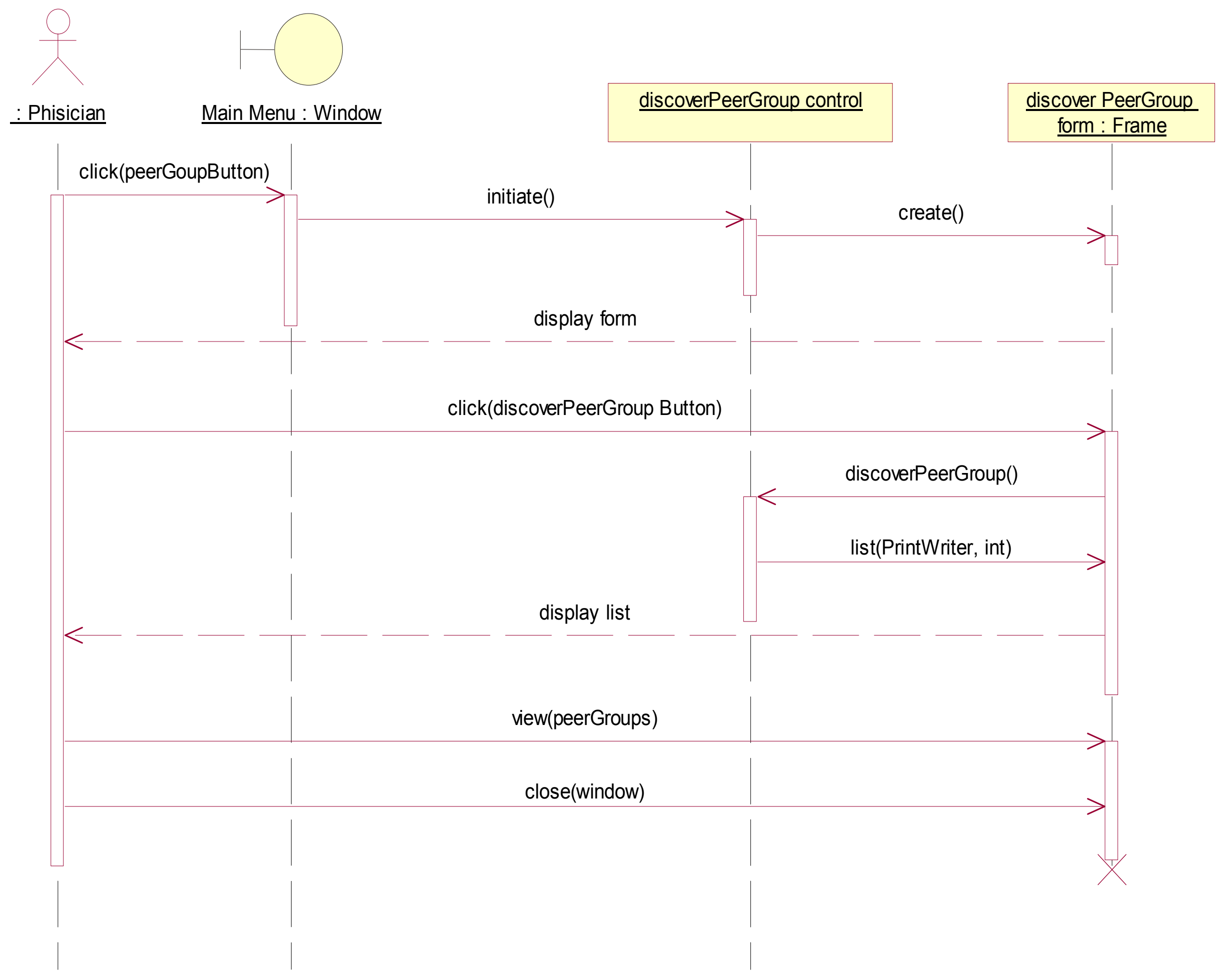


Figure 4-3 UML sequence diagram for discover peerGroups use case

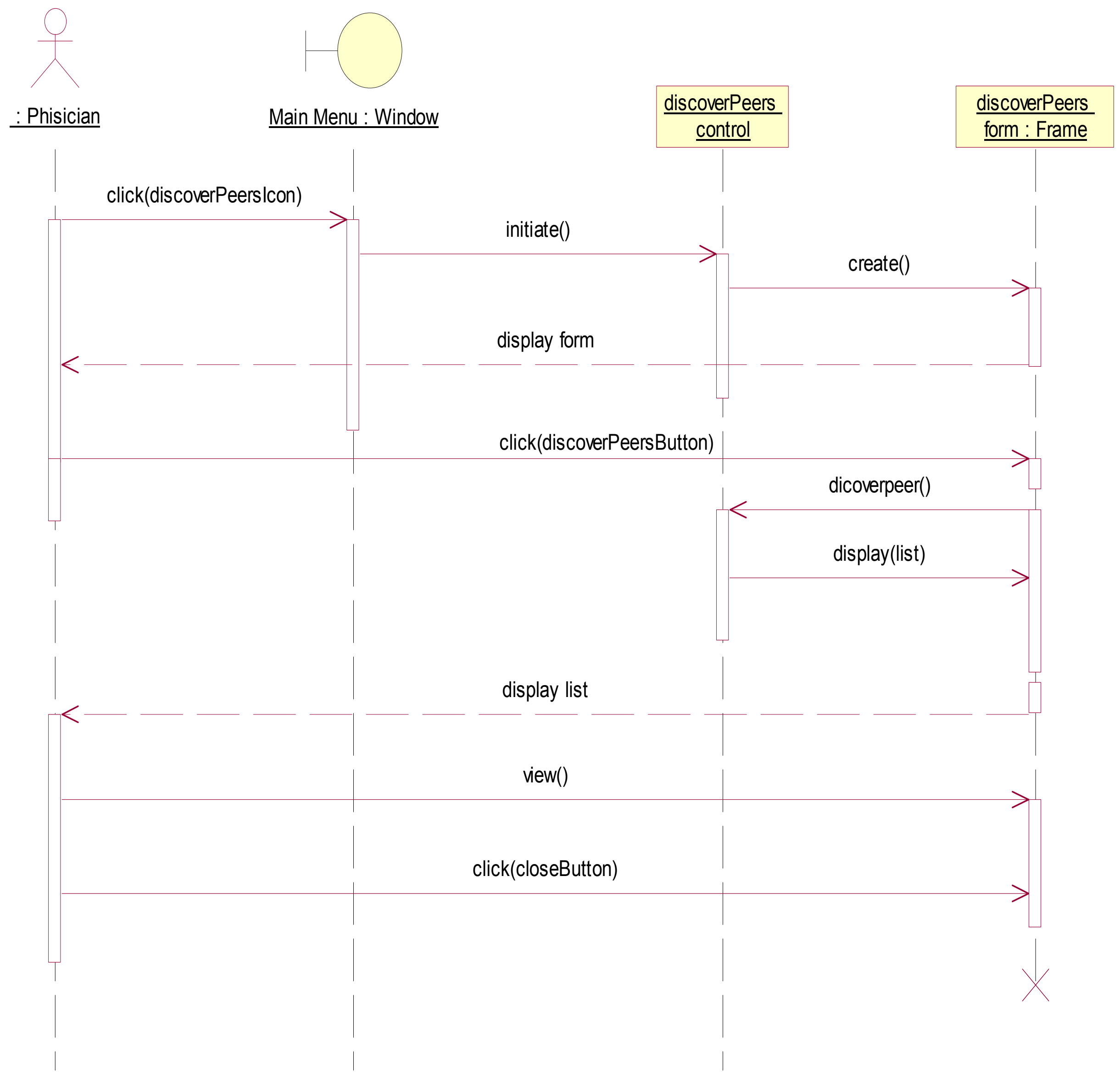


Figure 4-4 UML sequence diagram for discover peers use case

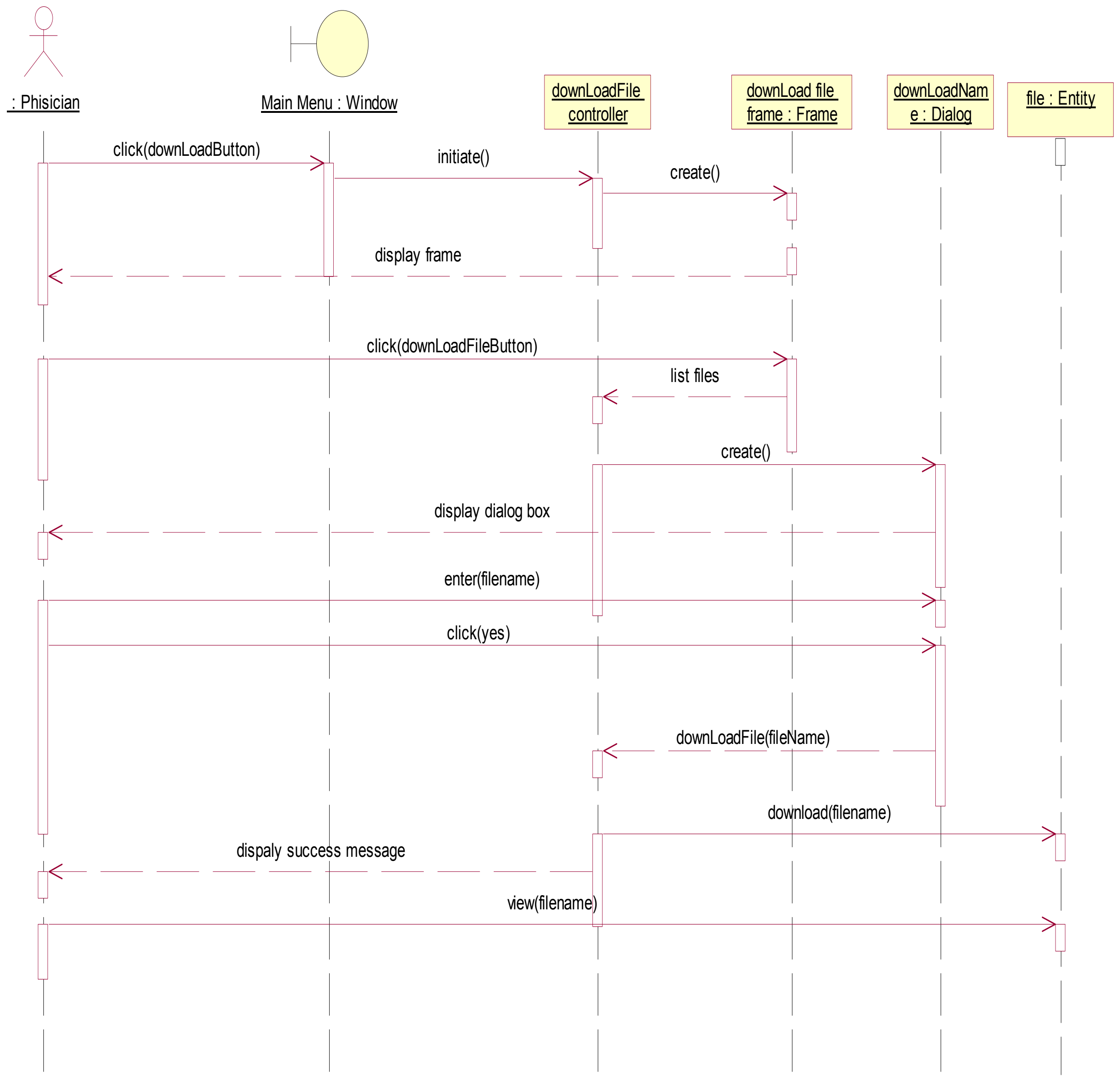


Figure 4-5 UML sequence diagram for download sharedFile use case

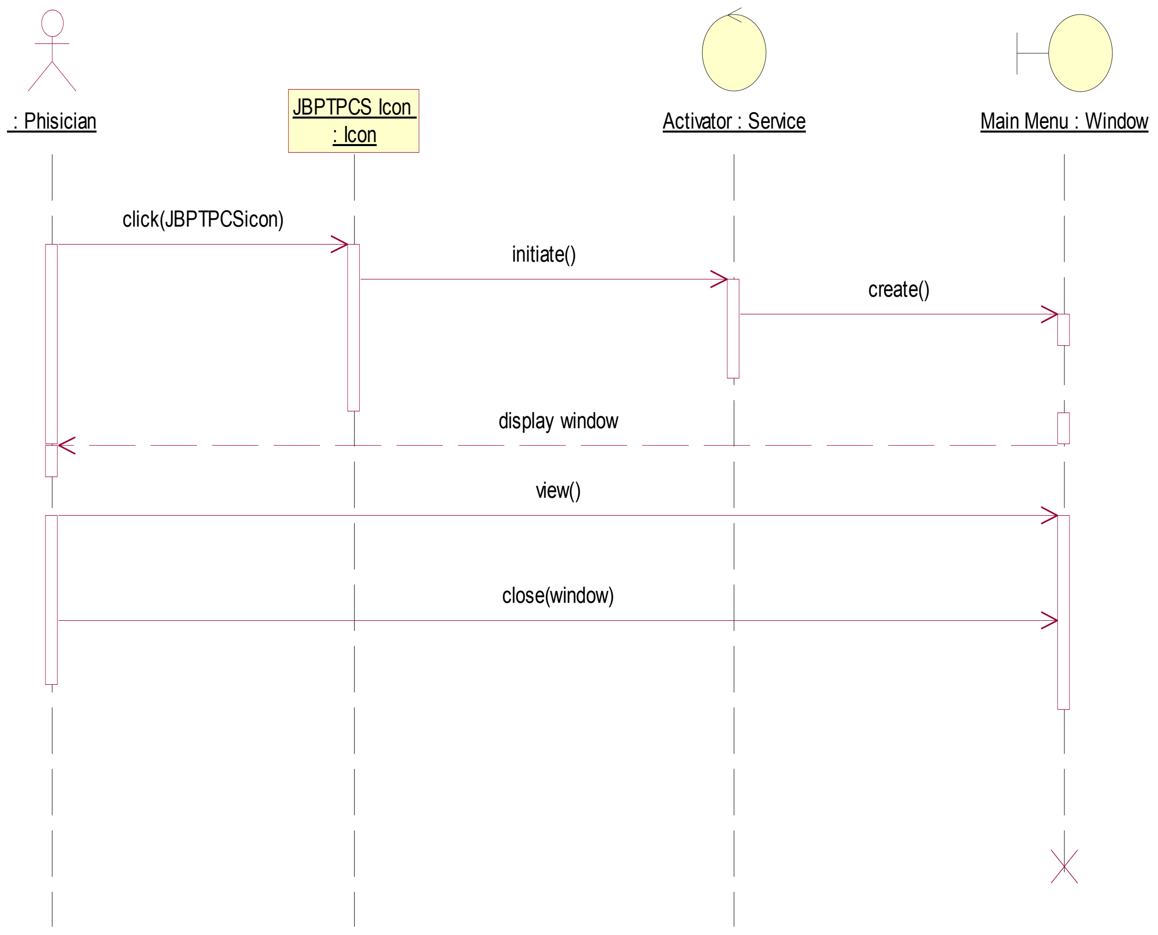


Figure 4-6 UML sequence diagram for start JBPTPCS use case

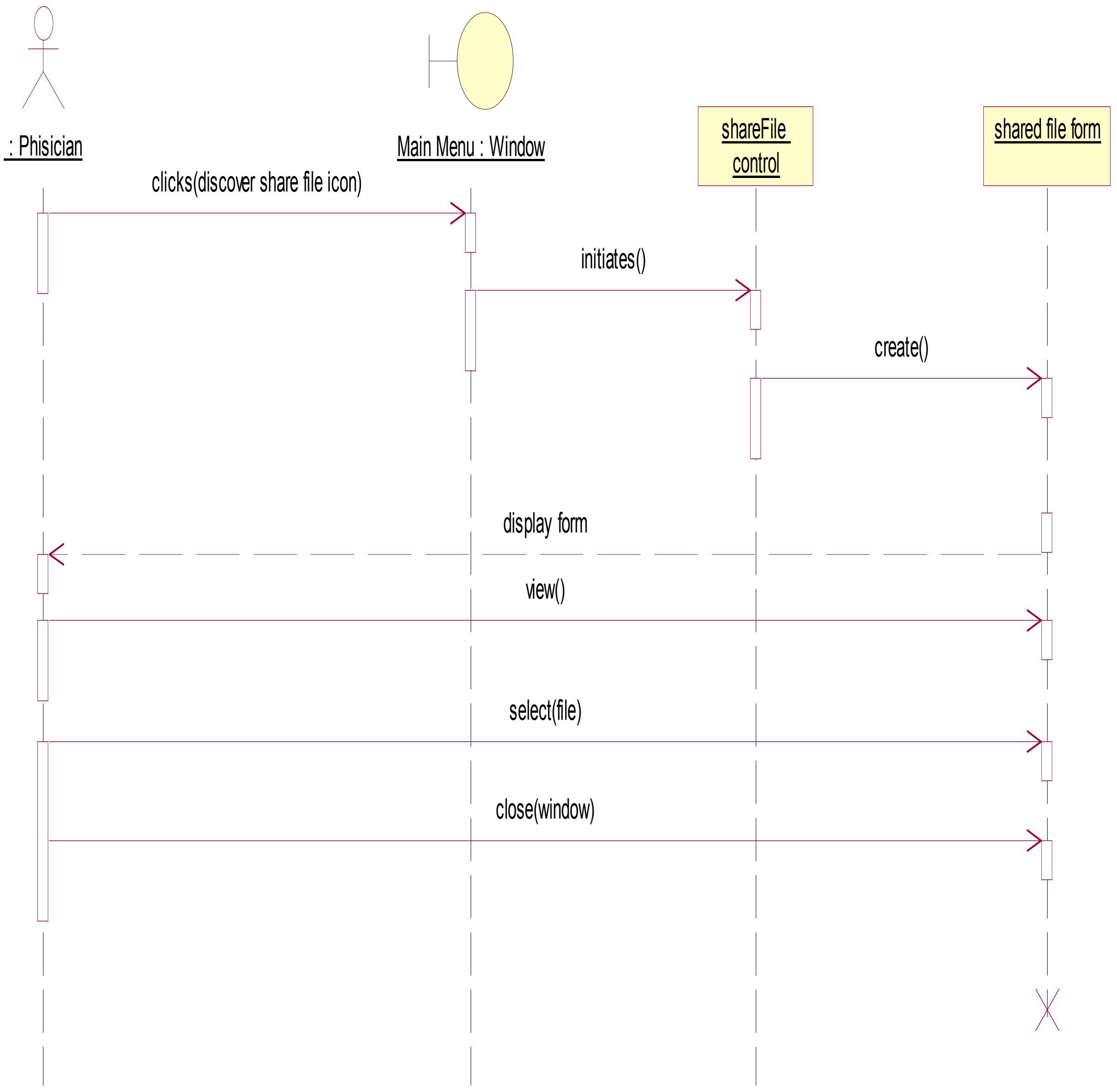


Figure 4-7 UML sequence diagram for discover sharedFile use case

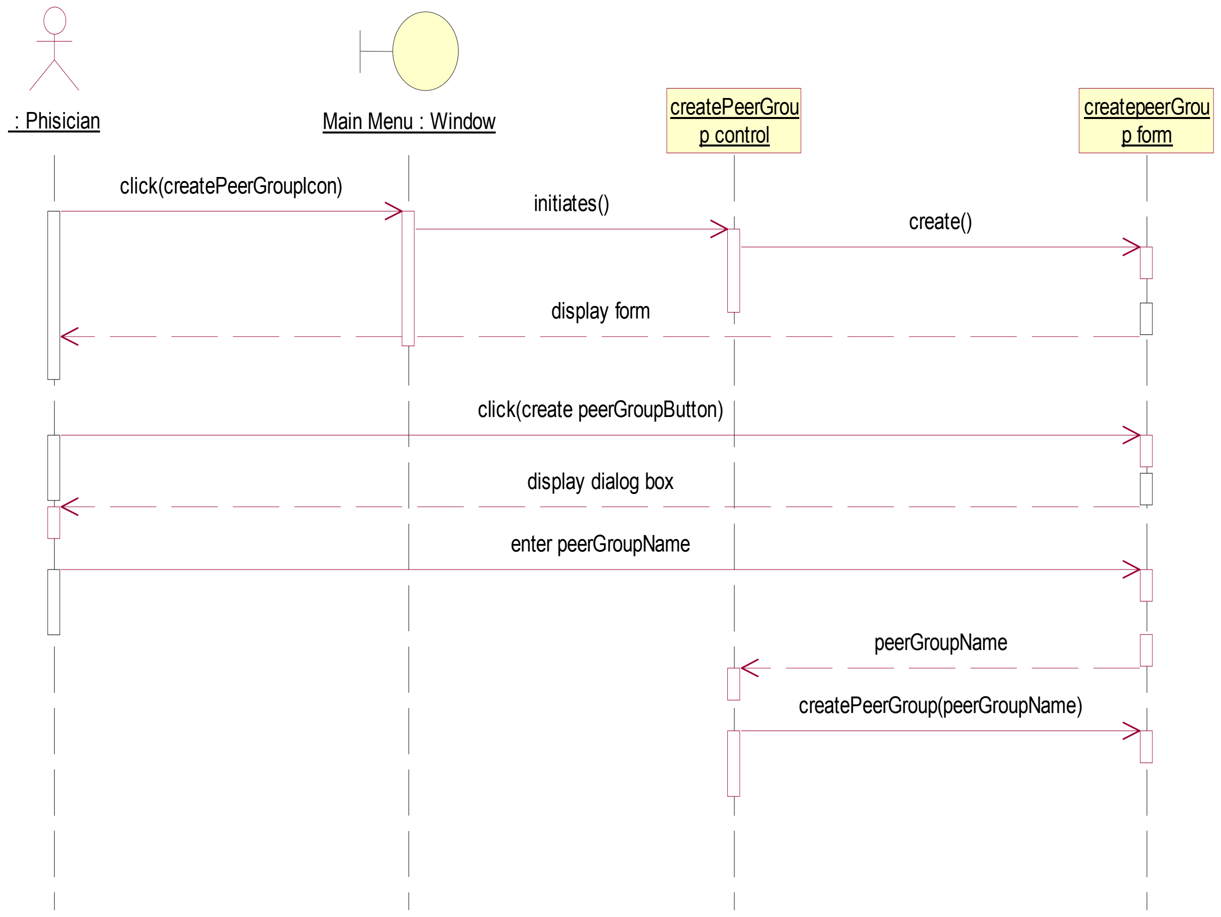


Figure 4-8 UML sequence diagram for create peerGroup use case

4.3.4 Class diagram

According to [21, 22], class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. A class diagram represents an overview of the system through the use of classes and their relationships. Classes enable us to visualize the interactions in a software system. Fig 4-9 represents the class diagram of our system.

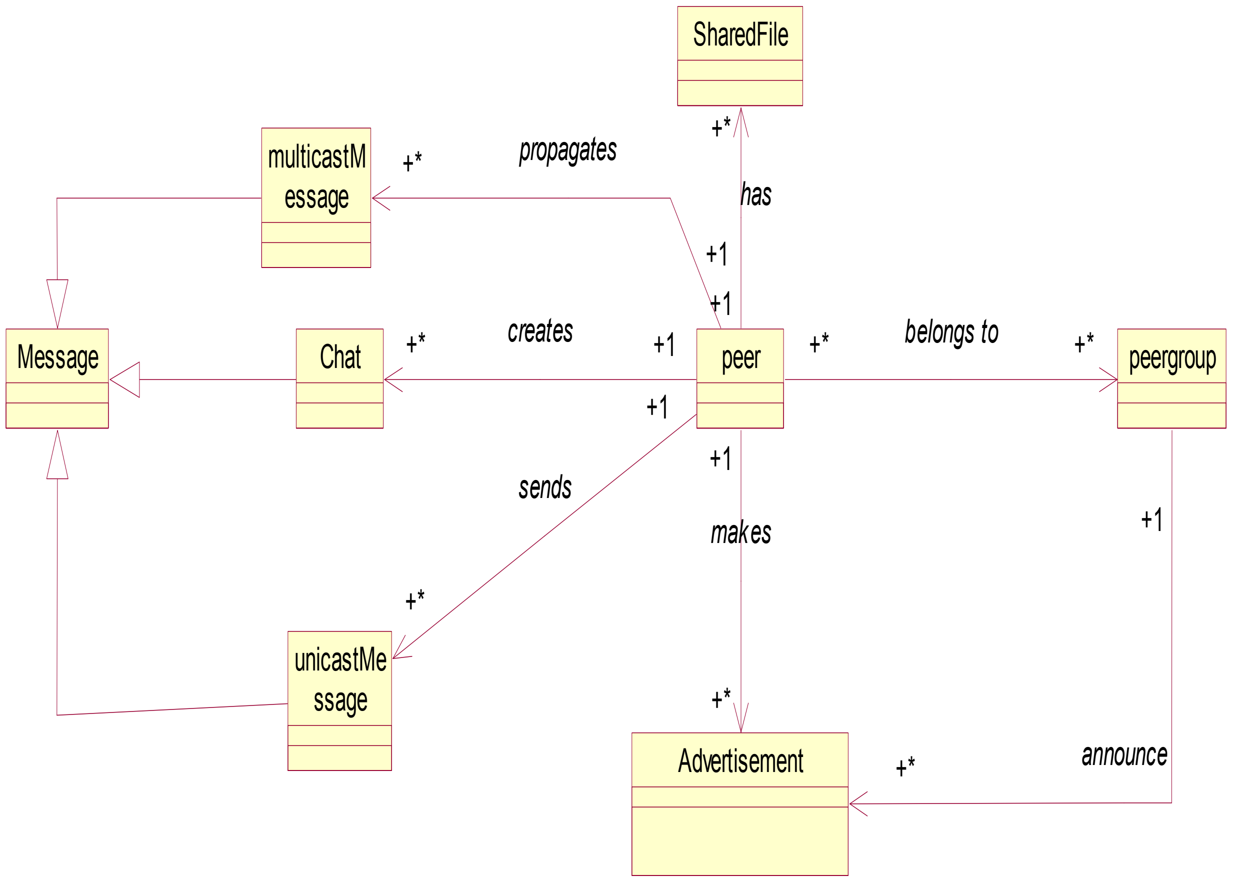


Figure 4-9 UML class diagram of JBPTPCS system

4.3.5 Activity Diagram

In UML, an activity diagram is used to display the sequence of activities. Activity diagrams show the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity.

According to [23 and 24], an activity diagram represents the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. In many ways UML Activity diagrams are the object-oriented equivalent of flow charts and data-flow diagrams (DFDs). They are used to explore the logic of:

- A single use case or
- Software processes as we show in our system in Fig 4-10 below.

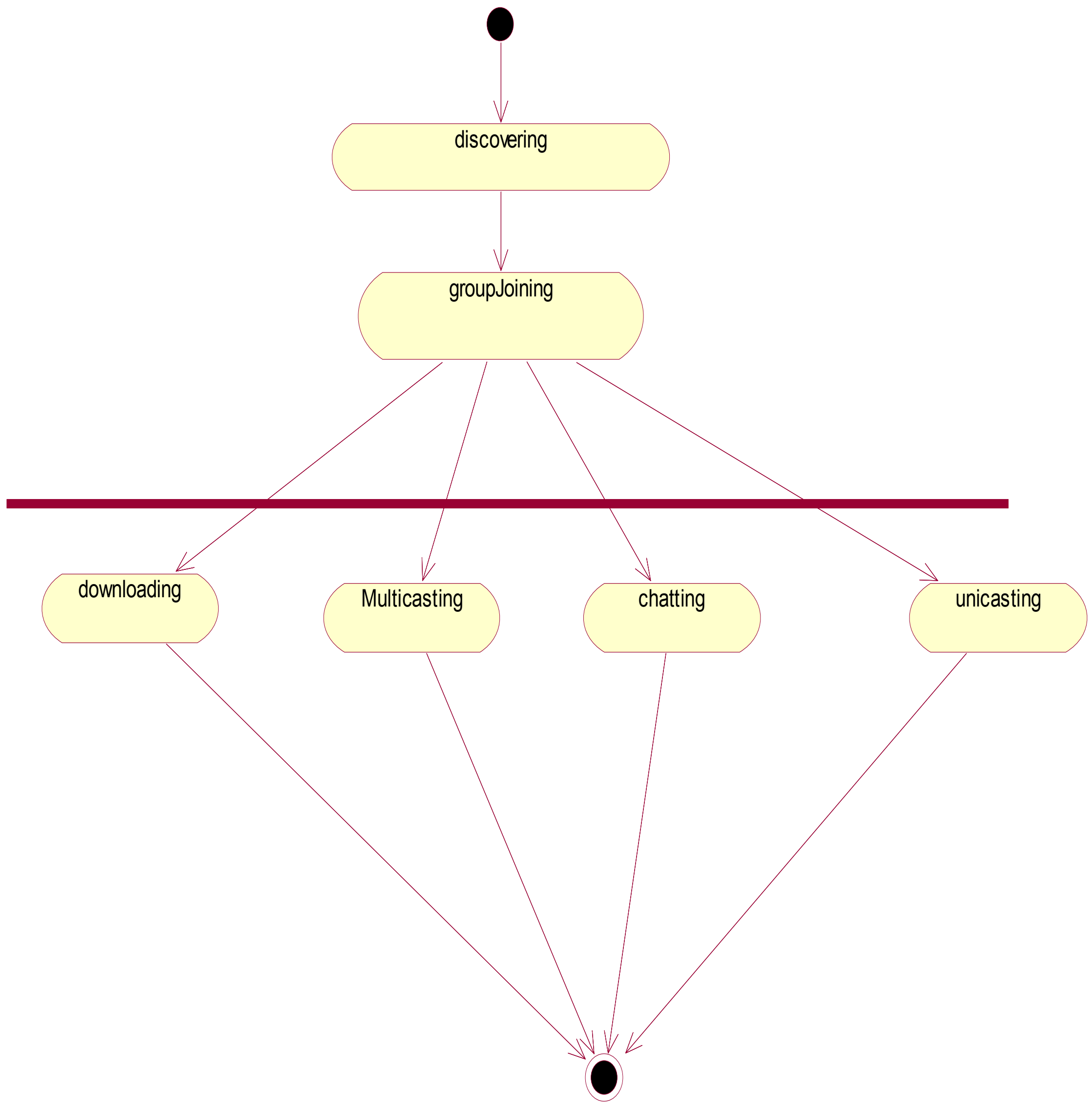


Figure 4-10 UML activity diagram of JBPTPCS system

Chapter 5

System Design

5.1 Introduction

In chapter 4 we have identified the functional and non-functional requirements of the system and produce the analysis model. Based on the analysis model, the design of the system is presented in this chapter. Design phase of software development deals with transforming the requirements described in the requirement analysis into a form implement able using a programming language.

According to [5], system design is concerned with how the system functionality is to be provided by the different software and hardware components of the system. It involves identifying which system capabilities are to be implemented in software and which in hardware. According to [7], a software system is broken down into subsystems, each of which is created individually. Hence, the design of the software system serves as a communication medium between these sub systems. Here in our system design, first we discuss the JBPTPCS application, and then set the design goals and following that the architecture of the system will be described in terms of its subsystem decomposition.

5.2 The JBPTPCS application overview

The JXTA based peer to peer collaborative system (JBPTPCS) is an application developed based on the JXTA technology. The JBPTPCS resides on the application layer in the JXTA architecture. The service layer provides network services for the application layer that are desirable but not necessarily a part of every P2P solution. These services implement functionality that might be incorporated into several different P2P applications, such as the following: searching for resources on a peer and sharing documents from a peer. The service layer uses the protocols from the JXTA core. The JXTA core layer provides the elements that are absolutely essential to every P2P solution. Ideally, the elements of this layer are shared by all P2P solutions. The elements of the core layer are peers, Peer groups, Network transport (pipes, endpoints), advertisements, protocols, and security. It also includes the six protocols provided by JXTA. The JBPTPCS application in relation to the JXTA protocol is seen in the figure 5-1 below.

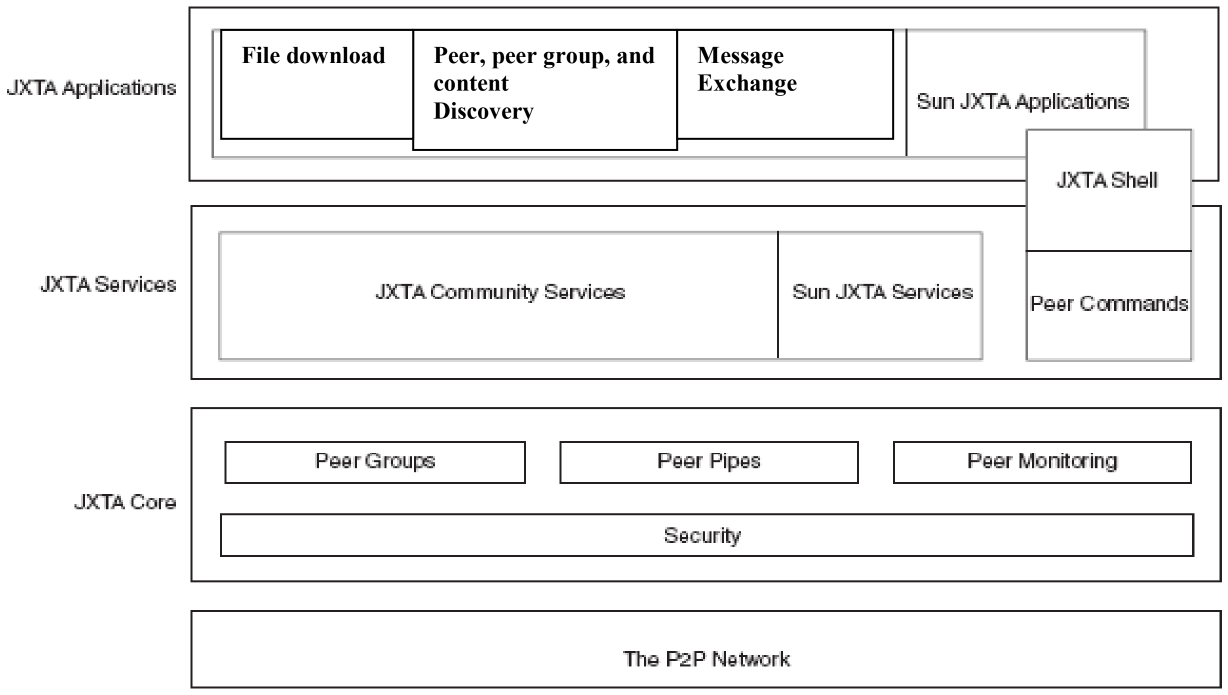


Figure 5-1The JBPTPCS functions on the JXTA three-layer architecture

JBPTPCS application comprises three basic functionalities

- Peer, Peer group, and content discovery.
- Peer to Peer multicast, unicast and bidirectional message exchange.
- Peer to Peer File Download.

5.2.1 Resource discovery

Resource discovery is the mechanism by which a peer is able to see what types of contents are shared among peers. A peer does not need to know in advance the existence as well as the location of shared resources. A peer first searches a resource to see its existence and downloads it, if it succeeds.

5.2.2 Message exchange

According to [17], online message exchange is a way of communicating by sending text messages among peers. The primary use of a chat is to share information via text with a group of other users. Email is great. It’s much faster than snail mail, and you can exchange message and get a response sooner. But, it can not let you to have instant communication with your contact in "real time”. Chat is an interactive service that lets you and your contact exchange messages in "real time”. Besides, chat has the following advantage, say over audio communication

- It can enable asynchronous communication: The text stays; it is easier to save the text and look back at it than saving the audio.
- It can lower the cost of interrupting others: It is a less disruptive to interrupt and ask a question during chat communication than audio.
- It can help audio problems, such as those that occur in public places.
- It can make it easier to communicate for someone who is shy or has difficulties expressing themselves verbally.

5.2.3 Peer to Peer File Download

Peer to Peer File Download is simply a transfer of files from one peer to another. File sharing occurs in networks which allow individuals to share, search for, and download files from one another. P2P file sharing eliminates the need for central servers, allowing all computers to communicate and share resources as equals. A P2P system would allow for files to be shared without relying on a server. But, P2P file sharing systems can be less reliable than server systems since most peers are personal computers which typically do not run 24 hours a day like a server.

5.3 Design Goals

Defining design goals is the first step of system design. Design goals are used to identify the qualities our system should focus on. The following design goals of the system are inferred from non-functional requirements stated in the previous chapter.

- Design issue 1: JBPTPCS should be easily usable and user friendly.
- Design issue 2: JBPTPCS should be readable i.e. the system should be easily understandable from the source code and should be maintainable.
- Design issue 3: JBPTPCS should be extensible i.e. easy to add new functionality.

5.3.1 Maintainability Criteria

JBPTPCS considers the following maintainability issues

- Modifiability: Easiness to change the functionality of the system.

Though in this project we scope the application to peers on local area network, the JXTA technology is boundless to create JXTA peers around the globe. Hence, emphasis is given on modifiability so as it should be easy modifiable to change its functionality to apply the application on JXTA peers around the globe.

- Extensibility: Easiness to add functionality or new class to the system.

Based on the same reasoning as modifiability, the emphasis is given to extensibility so that the application should be extendable by adding new functions to the system.

- Adaptability: Easiness to port the system to different application domain.

Also this application is done for health institutes; it is easily modifiable to apply to other application domains that require information exchange among them.

5.3.2 End User Criteria

According to [5], end user criteria include qualities that are desirable from a user point of view. These include usability (i.e. how difficult is the software to use and to learn) and utility (i.e. how well does the system supports the users work). As described in design issue 1 of section 5-3, the system was developed to be usable and supports the work of the user.

5.4 Proposed software Architecture

A peer to peer architecture was implemented here. A peer-to-peer architecture is a generalization of the client/server architecture in which subsystems can act both as client and as servers, in the sense that each subsystem can request and provide service. The architecture is shown in Fig 5-2 below.

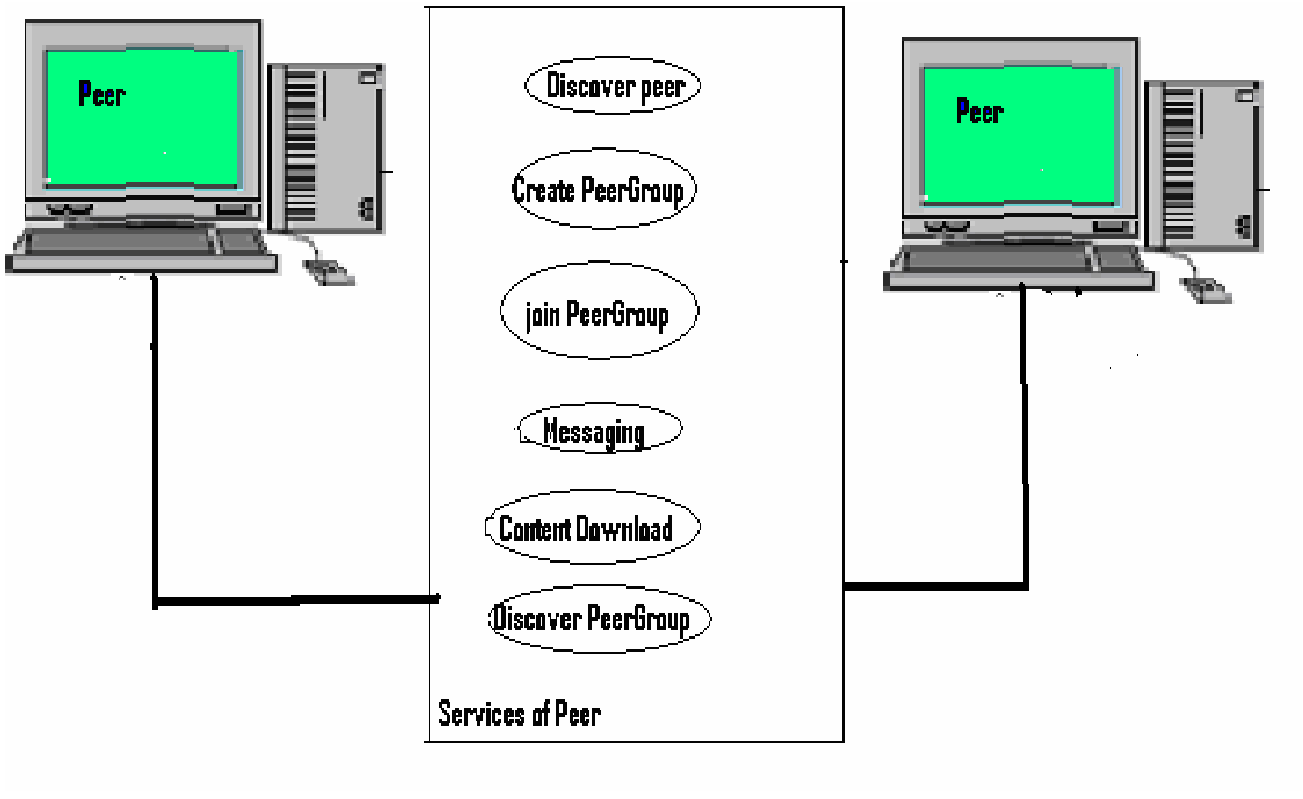


Figure 5-2 Architecture of JBPTPCS

5.4.1 Subsystem Decomposition

According to [7], a subsystem is characterized by the services it provides to other subsystems. Decomposing a system into manageable pieces reduces the complexity of the solution domain by minimizing dependencies among classes. Subsystem decomposition of the system is shown in Fig 5-3 below.

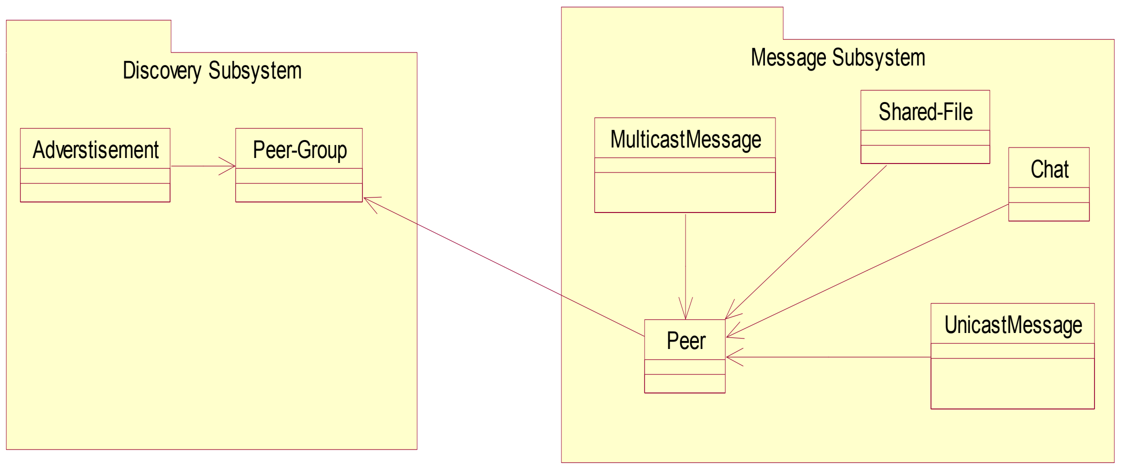


Figure 5-3 UML subsystem decomposition of JBPTPCS

5.4.2 Hardware/Software mapping

Each peer is allowed to respond for discovery request, exchange message, share content and download content. Communication among peers is done through discovery using the underlying TCP/IP protocol. The hardware/software (deployment) diagram is shown in Fig 5-4 below.

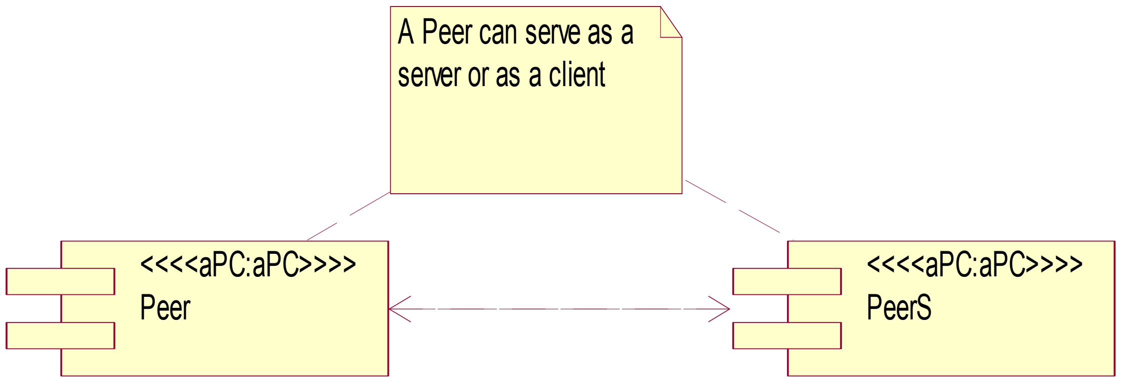


Figure 5-4 UML Deployment diagram of JBPTPCS

5.4.3 Persistent Data Management

The downloaded file will be saved as a Flat file using the operating system service.

.

Chapter 6

Implementation

6.1 Programming Tool

The programming tool is JXTA, Java reference implementation (i.e. JXTA API and Java classes are used). The java standard development Edition (J2SE), version 1.5 or latter is used.

6.2 Details of the implementation

The application mainly performs three functionalities.

6.2.1 Advertisement or content discovery

Advertisements are the basic unit of data exchanged between JXTA peers to provide information on available services, peers, peer groups, and pipes. All resources rely on the advertisement to represent themselves on the JXTA network. With advertisements, the problem of finding peers and all their different types of resources can be reduced to a problem of finding advertisements describing those resources.

The PDP defines a protocol for requesting advertisements from other peers and responding to other peers' requests for advertisements. Peer discover resources by sending a request to other peers, usually a rendezvous peers, and receive responses containing advertisements describing the available resources on the P2P network.

The Peer Discovery Protocol consists of only two messages that define the following:

- A request format to use to discover advertisements
- A response format for responding to a discovery request

These two message formats, define all the elements required to perform discovery transactions between two peers, as shown in Figure 6.1 below.

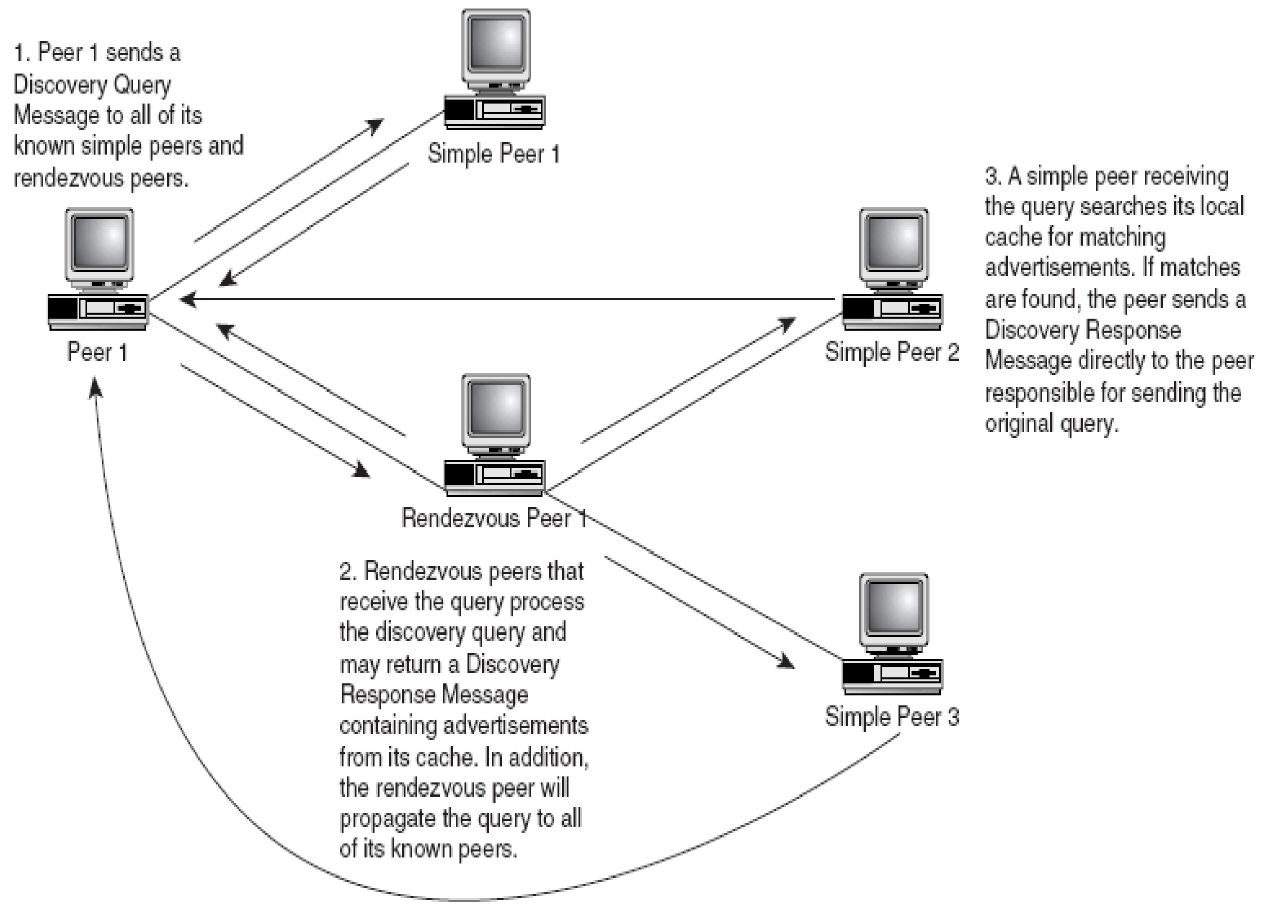

1. Peer 1 sends a Discovery Query Message to all of its known simple peers and rendezvous peers.
Simple Peer 1
3. A simple peer receiving the query searches its local cache for matching advertisements. If matches are found, the peer sends a Discovery Response Message directly to the peer responsible for sending the original query.
Peer 1
Simple Peer 2
Rendezvous Peer 1
2. Rendezvous peers that receive the query process the discovery query and may return a Discovery Response Message containing advertisements from its cache. In addition, the rendezvous peer will propagate the query to all of its known peers.
Simple Peer 3

Figure 6-1 Resource discovery mechanism among JXTA peers [1]

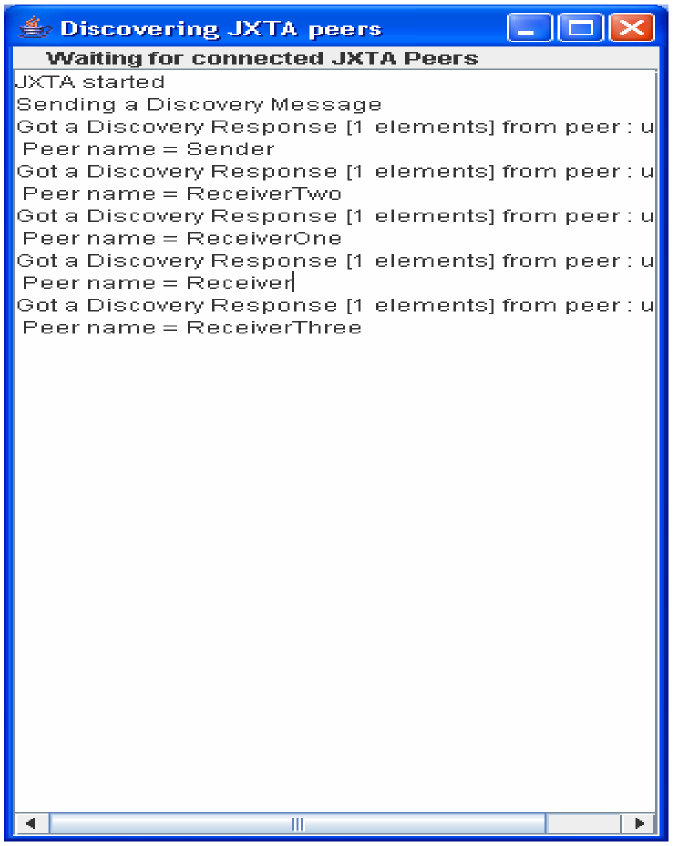
Every JXTA peer that wants to share its resource with other peers publishes its advertisement locally and remotely. Any JXTA peer that needs a resource discovers by using its PDP for the published advertisement in order to utilize it. This means that a JXTA peer is not required to know in advance the existence as well as the location of resources. What is expected from the peer is to send a request query among JXTA peers. A peer that has the requested resource responds with its peer advertisement. Based on the response the peer utilizes the resource. Hence, it is clear from the statements above that discovering a resource is the prime task among JXTA peers. For this purpose this application comprises three types of discovery services.

6.2.1.1 Peer discovery

The most common and widely understood component of any P2P system is the peer. A peer is simply an application, executing on a computer device that has the ability to communicate with other peers. For the entire system to work, it is fundamental that the peer have the ability to communicate with other peers. A single peer can function both as a “client” (to request information from other peers) and as a “server” (to answer requests from other peers). Peers have peer name and a unique peer ID. According to [14] JXTA peers are basically categorized into two; Edge Peer and super peer. Edge peers can be further classified as minimal edge peers and edge peers. Super peers are either rendezvous or relay peers.

- Minimal edge peer does not cache advertisements or route messages. PDAs and cell phones are examples of minimal edge.
- Edge peer caches an advertisement, replies to discovery requests, but it does not forward discovery requests. Personal computer (PC) is an edge peer.
- Rendezvous super peer caches advertisements and forwards discovery requests.
- Relay super peer maintains routing information about a peer that cannot directly connect to other peers.

Peer discovery component of this application provides the feature to search for the JXTA peers that are currently active and connected to the requesting peer. A JXTA peer can have the ability to know all other live JXTA peers using this functionality. A sample screen shot is shown in Fig 6-2 below.



```
JXTA started
Sending a Discovery Message
Got a Discovery Response [1 elements] from peer : u
 Peer name = Sender
Got a Discovery Response [1 elements] from peer : u
 Peer name = ReceiverTwo
Got a Discovery Response [1 elements] from peer : u
 Peer name = ReceiverOne
Got a Discovery Response [1 elements] from peer : u
 Peer name = Receiver
Got a Discovery Response [1 elements] from peer : u
 Peer name = ReceiverThree
```

Figure 6-2 Sample output for discover peers functionality

6.2.1.2 Peer group discovery

If several peers get together to share files or work on a large, computationally intensive problem, they form a group. The group is defined by one of the peers publishing the information necessary about the group. The JXTA network has a peer group called the WorldPeerGroup which is the largest peer group that consists of all JXTA peers. For Java implementation of JXTA the largest peer group is named NetPeerGroup. The NetPeerGroup is the default group that all new peers automatically join on the JXTA network as shown in Figure 6-3 below.

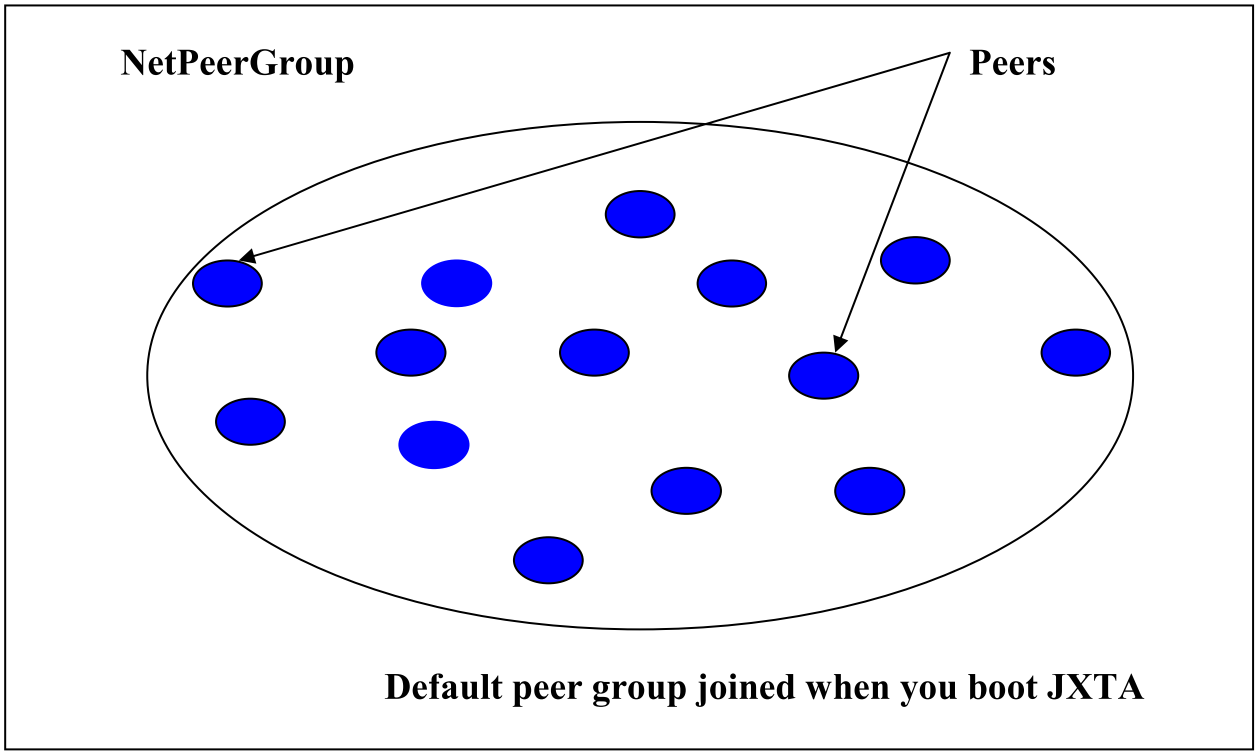


Figure 6-3 JXTA NetPeerGroup

Currently we can create and join a new peer group, which is a sub group of the NetPeerGroup, either in a public or private format as shown in Figure 6-4 below.

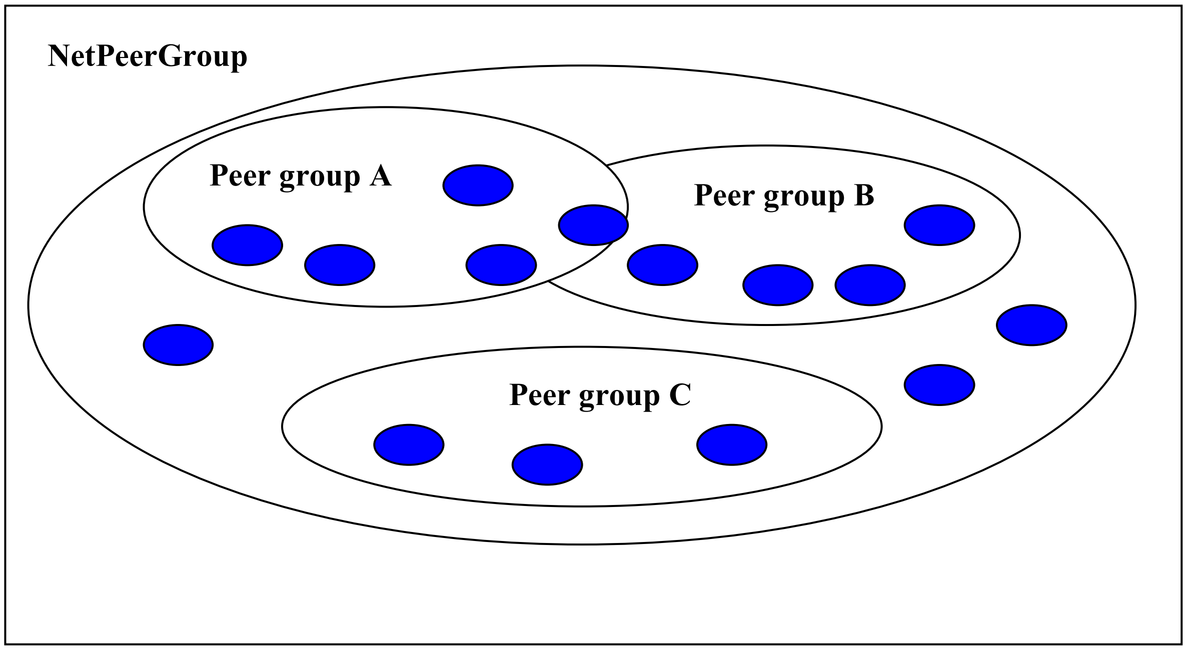


Figure 6-4 JXTA NetPeerGroup the largest peer group

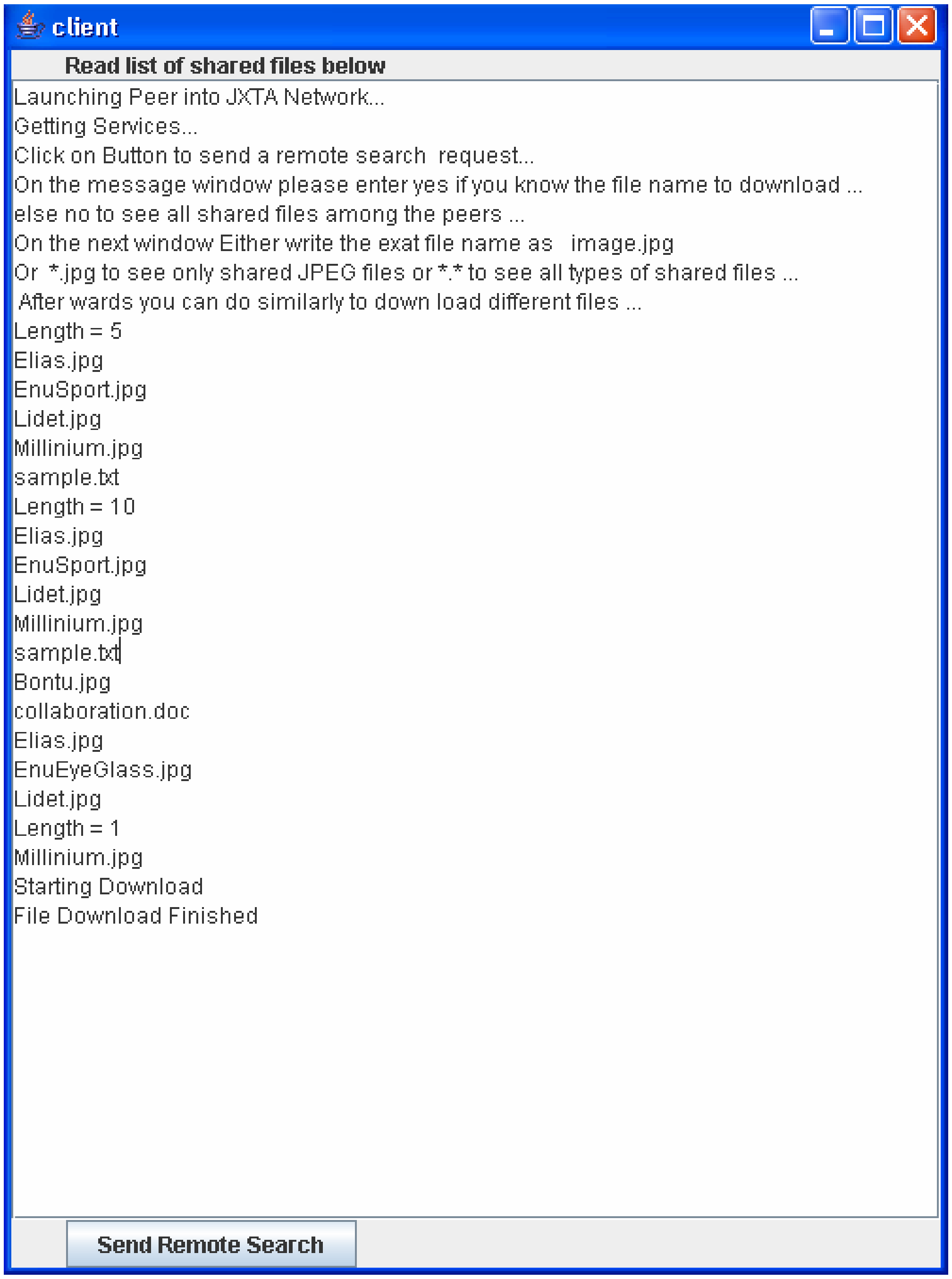
The public peer group doesn’t require a username or password, but a private one does to join and use services of the peer group. Any peer can create either type of peer groups for whatever purpose it desires.

Peer group discovery function of this project provides to search for peer groups currently active in the network. Hence, a peer discovers the existing active peer groups and joins to the group the peer is interested in.

6.2.1.3 Content discovery and Download

The Content Management Service (CMS) API of JXTA enables a peer to share data—such as text documents, graphics files, sound files, and other media—with remote peers. To maintain consistency with the specification, the CMS relies on advertisements to provide information about the media or files a peer is going to share, and relies on JXTA pipes to transfer the content. The JXTA pipes are used for receiving requests, queries, and content.

The content discovery component of this application helps a peer to search for shared files among all JXTA peers. So a peer can have information about the files that are shared among JXTA peers. Having this information a peer can download a file by providing the name of the file. Fig 6-5 below shows a screen shot of this service.



```
Launching Peer into JXTA Network...
Getting Services...
Click on Button to send a remote search  request...
On the message window please enter yes if you know the file name to download ...
else no to see all shared files among the peers ...
On the next window Either write the exat file name as   image.jpg
Or  *.jpg to see only shared JPEG files or *.* to see all types of shared files ...
 After wards you can do similarly to down load different files ...
Length = 5
Elias.jpg
EnuSport.jpg
Lidet.jpg
Millinium.jpg
sample.txt
Length = 10
Elias.jpg
EnuSport.jpg
Lidet.jpg
Millinium.jpg
sample.txt
Bontu.jpg
collaboration.doc
Elias.jpg
EnuEyeGlass.jpg
Lidet.jpg
Length = 1
Millinium.jpg
Starting Download
File Download Finished
```

Figure 6-5 A sample output of Discovered shared files

6.2.2 Message exchange

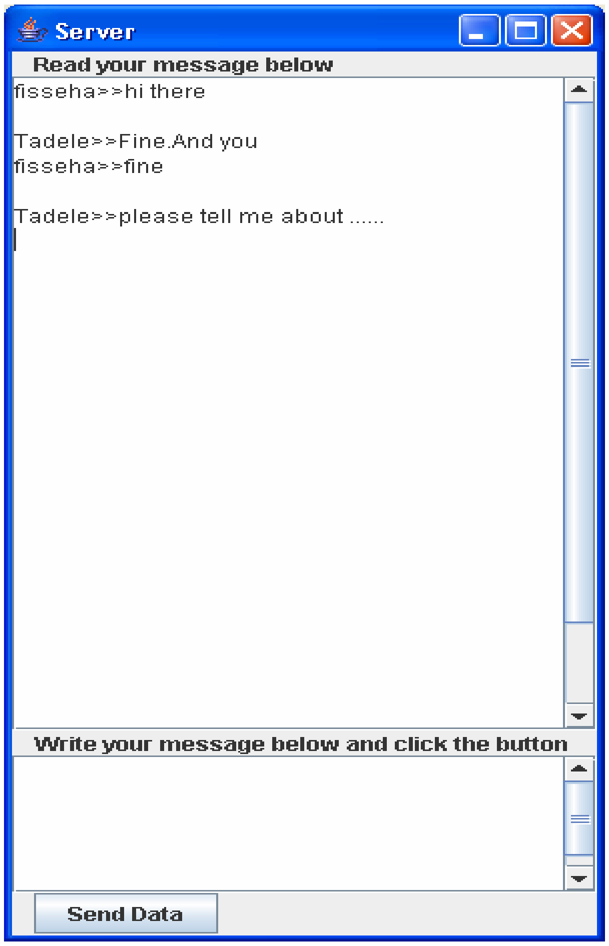
We use pipes to send data between peers in a peer group. JXTA provide three types of pipes; Unicast, UnicastSecure, and propagate pipes. Once the application creates a pipe it publishes advertisement for the pipe. The purpose of publishing a pipe advertisement is to let all the peers in a group know of the pipe's existence. The discovery service offers two publish methods: publish() and remotePublish(). The publish() method lets peers that are directly accessible to the advertising peer know about the new pipe. The remotePublish() takes the publishing one step further by trying to use all available transports, such as TCP/IP and HTTP, in order to get the widest distribution. When a remote peer discovers a pipe advertisement, it can bind to it in order to exchange data. The JBPTPCS application handles three types of communication.

6.2.2.1 Unicast communication

The most basic type of pipe, Unicast allows communication from one peer to another in a single direction. If information has to be passed between peers in both directions, both of the peers need to publish pipe advertisements, or one of the peers can pass pipe information through the one Unicast pipe. When the other peer receives the pipe advertisement, it will create another pipe. This function of the JBPTPCS application provides a service in such a way that two JXTA peers can communicate using unicast pipe in one way. One peer sends a message and the other receives the message.

6.2.2.2 Bidirectional communication

Bidirectional communication is implemented using the BidirectionalPipeService class. This class uses the core JXTA unicast pipes (Input Pipe and Output Pipe) to simulate bidirectional pipes. This function of the JBPTPCS application provides a feature in such a way that two JXTA peers can communicate. Here the communication is in two direction; a peer sends as well receives a message. Fig 6-6 shows a screen shot of this service.

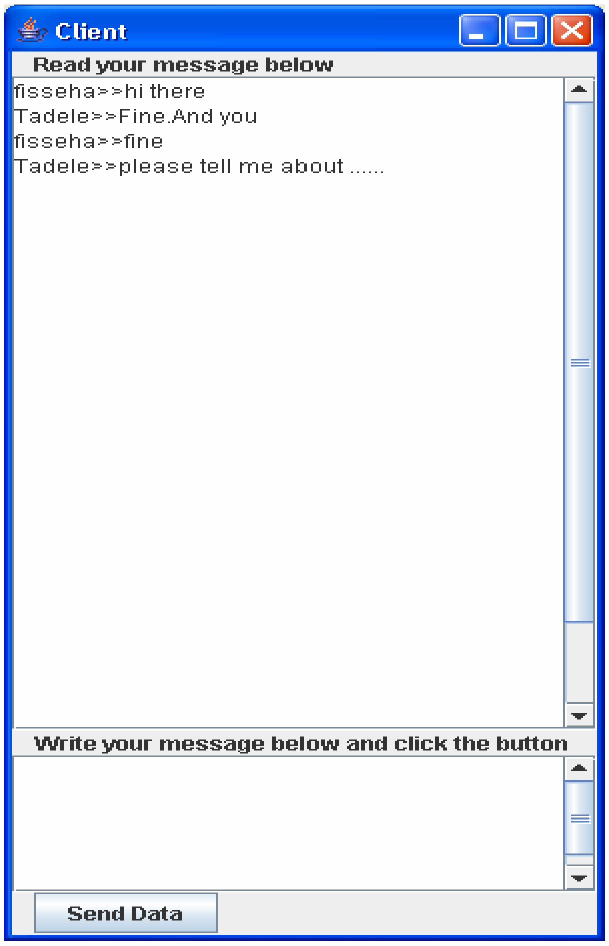


```
fisseha>>hi there

Tadele>>Fine.And you
fisseha>>fine

Tadele>>please tell me about ......
```

(a)



```
fisseha>>hi there
Tadele>>Fine.And you
fisseha>>fine
Tadele>>please tell me about ......
```

(b)

Figure 6-6 the communicating peers in the bidirectional communication

In the figure above, Figure 6-6(a) is the peer that initiates communication and Figure 6-6(b) is a peer that searches advertisement to bind to the server.

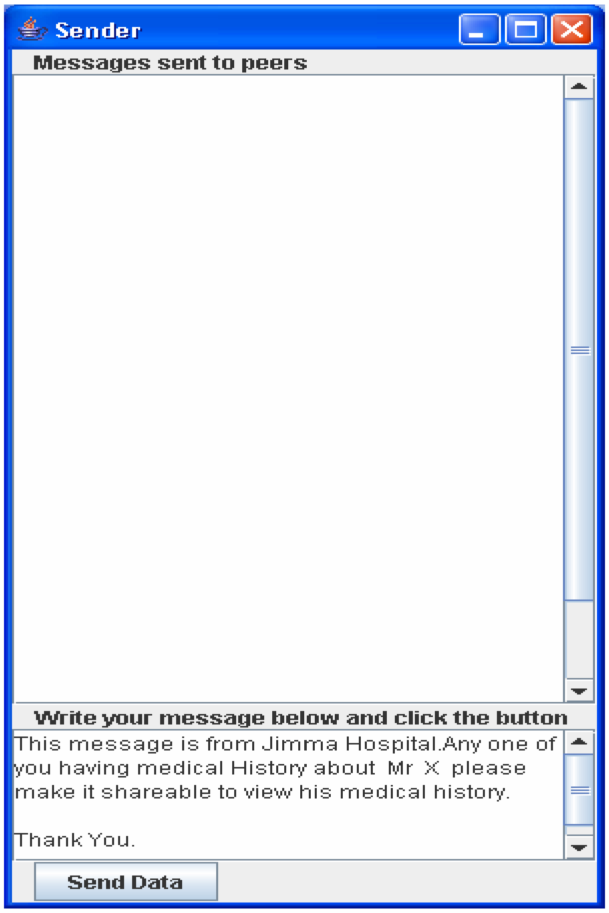
6.2.2.3 Propagate message

In some application designs, a single peer will need the ability to send information to a number of different peers at the same time. A propagate pipe connects one output pipe to multiple input pipes. Messages flow from the output pipe (the propagation source) into the input pipes. All propagation is done within the scope of a peer group. That is, the output pipe and all input pipes must belong to the same peer group.

The JBPTPCS application implements multicast data exchange in two varieties

- The first alternative is that one peer sends a multicast message and the other peers receive the message. Here the receiving peers can not send response to the sending peer.
- The second alternative is, one peer sends a message and other peers receive the message and respond to the sending peer turn by turn.

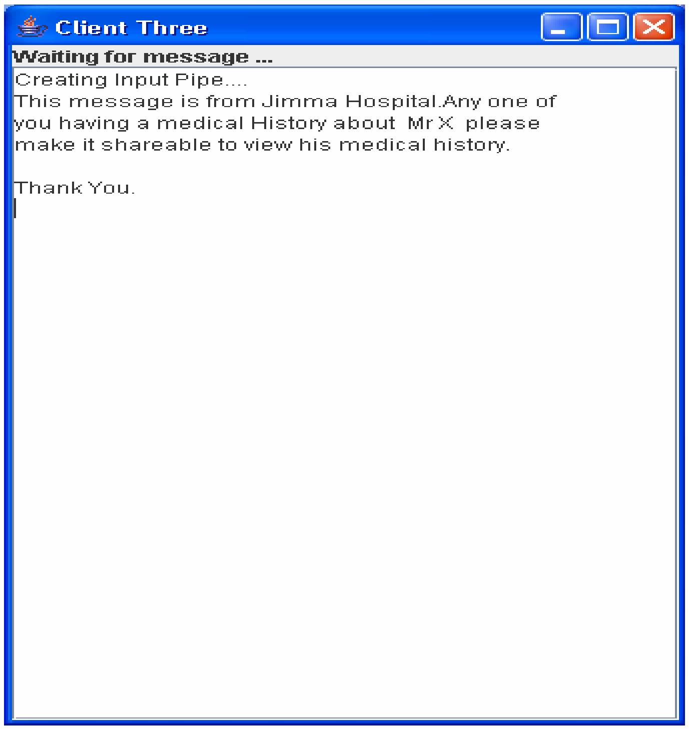
A sample output for multicast communication of the first option is shown in Figure 6-7 and 6-8.



This message is from Jimma Hospital.Any one of you having medical History about Mr X please make it shareable to view his medical history.

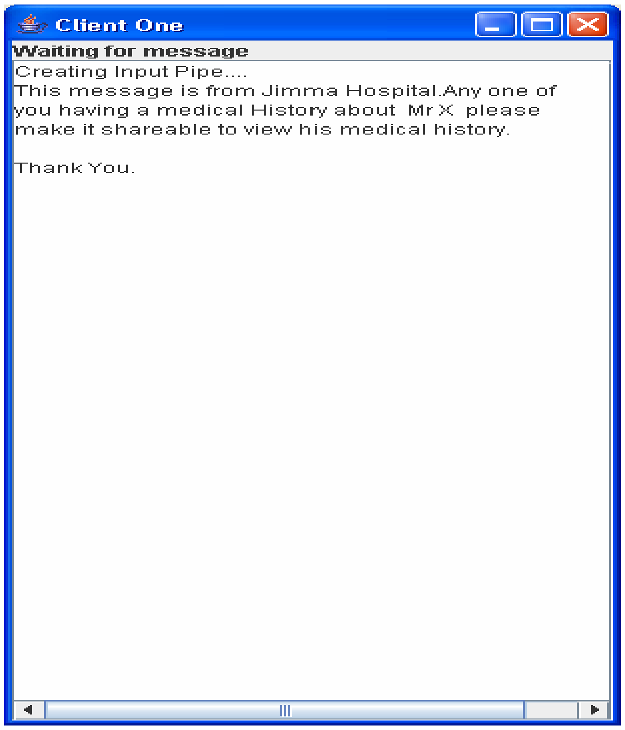
Thank You.

Figure 6-7The sending Peer (a peer that multicasts)



```
Waiting for message ...
Creating Input Pipe....
This message is from Jimma Hospital.Any one of
you having a medical History about  Mr X  please
make it shareable to view his medical history.

Thank You.
```



```
Waiting for message
Creating Input Pipe....
This message is from Jimma Hospital.Any one of
you having a medical History about  Mr X  please
make it shareable to view his medical history.

Thank You.
```

(a) **(b)**

Figure 6-8 two receiving peers

Figure 6-8 (a) and Figure 6-8(b) are receiving peers that receive a multicast message.

6.2.3 Creating and joining a peer group

All JXTA peers want to be a part of a peer group, so one of the first things an application will do is create a default peer group with the name NetPeerGroup. As a general rule, all JXTA applications will belong to this peer group. The group itself provides a number of services for finding and creating additional groups. Peer groups provide a way of segmenting the P2P network space into distinct communities of peers organized for a specific purpose.

This function of the JBPTPCS application provides a service by which a new peer group is created and joined by interested peers.

Chapter 7

Conclusion and Future work

7.1 Conclusion

As described in chapter 3, JXTA is a platform and programming language independent technology. Using this technology the following achievements are attained.

- Active JXTA peers can be discovered and listed.

  This function provides currently live peers that we are able to communicate with.

- Active JXTA peer groups can be discovered and listed.

  This function provides currently existing peer groups that we can join.

- Shared contents are easily discovered among JXTA peers.

  This application gives the opportunity to know in advance what contents are currently available to be shared among JXTA peers.

- Shared contents can be downloaded.

  A peer can download and use any shared content.

- Messages communication among peers is possible in different ways

  - ➢ Unicast communication: - Message exchange between two peers in one direction only.
  - ➢ Bidirectional communication:- Message exchange between two peers in both directions
  - ➢ Multicast communication: - A peer sends a multicast message and two or more peers can receive the message. The implementation has two scenarios. One scenario is, a peer sends a multicast message and others receive the sent message. The other scenario is the receiving peers are given a privilege to respond to the sending peer turn by turn.

7.2 Future Work

As stated above, besides a platform and programming language independent technology, JXTA is also available for any digital device. Though it has such a tremendous application area, JBPTPCS application was developed on a local area network. Hence we recommend the following as a future work.

- To incorporate security issue on the application.

In this application, we tried to show that a peer joins a peer group with out any security consideration. A peer joining a peer group should be authenticated in order to minimize destruction from malicious actions of some peers. For the application to be reliable security issue is non neglected .Hence, it is left as a future work to be incorporated in the application.

- To expand and incorporate pervasive devices.

Pervasive devices are resource hungry devices. Because of their size they encounter shortage of resources. Using JXTA technology, computations can be performed in other devices and the output can be returned to this resource hungry devices.

- To expand the application in order to incorporate peers found in different networks as well as peers that are found beyond a NAT or a Proxy.

Consider a big organization which has branch offices distributed and covered a wide area network where each branch has peers behind a NAT or proxy device. In such situation incorporating super peers (i.e. rendezvous and relay peers) can help to utilize content sharing and message exchange among peers in different branches.

Appendix A

JBPTPCS classes

This appendix consists of a comprehensive summary of the JBPTPCS classes and important methods in each class. This material is used to provide a quick reference when needed to explore the class. A short description of the classes and methods and also a code example is given for a chosen method.

Class DiscoverPeers

The DiscoverPeers class provides the basis for discovering all peers currently alive in a Net Peer Group, which is the default peer group.

Constructor Summary

public DiscoverPeers() :-creates user interface

Method Summary

private void startJxta() :- launches JXTA

public void discoveryEvent(DiscoveryEvent ev) :- Notifies when response comes

Example

```
// by implementing DiscoveryListener we must define this method
// to deal with discoveryService responses

public void discoveryEvent(DiscoveryEvent ev) {
        DiscoveryResponseMsg res = ev.getResponse();
        String name = "unknown";

        // Get the responding peer's advertisement
        PeerAdvertisement peerAdv = res.getPeerAdvertisement();

        // some peers may not respond with their peerAdv
```

```
            if (peerAdv != null) {
                    name = peerAdv.getName();
            }

            //System.out.println ("Got a Discovery Response [" +  res.getResponseCount()+ "
            elements] from peer : " + name);
            displayArea.append("Got a Discovery Response [" +  res.getResponseCount()+ "
            elements] from peer : " + name + "\n");

            // printout each discovered peer
            PeerAdvertisement adv = null;
            Enumeration enumerator = res.getAdvertisements();

            if (enumerator != null ) {
                    while (enumerator.hasMoreElements()) {
                            adv = (PeerAdvertisement) enumerator.nextElement();
                            //System.out.println (" Peer name = " + adv.getName());
                            displayArea.append(" Peer name = " + adv.getName() +      "\n");
                    }
            }
    }
```

Class CreateAndJoinPeerGroup

The CreateAndJoinPeerGroup class provides a service to create and join a peer group.

Constructor Summary

public CreateAndJoinPeerGroup() :- creates user interface.

Method Summary

private String peerGroupName():- takes group name from a user.

private PeerGroup createGroup():-creates a peer group by the name it takes from a user.

private void joinGroup(PeerGroup grp):-Joins a peer group.

Example

```
private PeerGroup createGroup() {
        peerGroupName=peerGroupName();
        peerGroupDescription=peerGroupDescription();
        PeerGroup pg; // new peer group
        PeerGroupAdvertisement adv; // advertisement for the new group
        System.out.println("Creating a new group advertisement");
        displayArea.append("Creating a new group advertisement \n");
        try {
            // create a new all purpose peergroup.
            ModuleImplAdvertisement implAdv =
            myGroup.getAllPurposePeerGroupImplAdvertisement();
            pg = myGroup.newGroup(null, // Assign new group ID
            implAdv, // The implem. adv
            //"HealthGroup", // The name
            peerGroupName,


            //"Health Centers Group"); // descr.
            peerGroupDescription);
            // print the name of the group and the peer group ID
            adv = pg.getPeerGroupAdvertisement();
            PeerGroupID GID = adv.getPeerGroupID();
            System.out.println(" Group = " +adv.getName() + "\n Group ID = " +
            GID.toString());
            displayArea.append(" Group = " +adv.getName() + "\n Group ID = " +
            GID.toString() + "\n");
        } catch (Exception eee) {
            System.out.println("Group creation failed with " + eee.toString());
            displayArea.append("Group creation failed with " + eee.toString() + "\n");
            return (null);
        }
```

```
            try {
                    // publish this advertisement
                    // (send out to other peers and rendezvous peer)
                    discoveryService.remotePublish(adv);
                    System.out.println("Group published successfully.\n");
                    displayArea.append("Group published successfully.\n");
            }catch (Exception e) {
                    System.out.println("Error publishing group advertisement");
                    e.printStackTrace();
                    return (null);
            }
            return(pg);
    }

private void joinGroup(PeerGroup grp) {
            System.out.println("Joining peer group...");
            displayArea.append("Joining peer group...\n");
            StructuredDocument creds = null;
            try {
                    // Generate the credentials for the Peer Group
                    AuthenticationCredential authCred = new AuthenticationCredential( grp,
                    null, creds );
                    // Get the MembershipService from the peer group
                    MembershipService membership = grp.getMembershipService();
                    // Get the Authenticator from the Authentication creds
                    Authenticator auth = membership.apply( authCred );
                    // Check if everything is okay to join the group
                    if (auth.isReadyForJoin()){
                            Credential myCred = membership.join(auth);
                            System.out.println("Successfully joined group " +
                            grp.getPeerGroupName());
                            displayArea.append("Successfully joined group " +
                            grp.getPeerGroupName() + "\n");
                            // display the credential as a plain text document.
                            System.out.println("\nCredential: ");
                            StructuredTextDocument doc =
                            (StructuredTextDocument)myCred.getDocument(new
                            MimeMediaType("text/plain"));
                            StringWriter out = new StringWriter();
                            doc.sendToWriter(out);
                            System.out.println(out.toString());
                            out.close();
                    } else
                            //System.out.println("Failure: unable to join group");
                            displayArea.append("Failure: unable to join group\n ");
            } catch (Exception e){
                    System.out.println("Failure in authentication.");
                    displayArea.append("Failure in authentication.\n");
                    e.printStackTrace();
            }
        }
```

Class ShareFile

This class provides a service to discover and share content.

Constructor Summary

public ShareFile()

Method Summary

public void notifyMoreResults():-Notify when response about content arrives.

private String fileName():-takes name of a file to down load.

Example

```
public void notifyMoreResults() {
        System.out.println("Search Done");


        ContentAdvertisement[] result = listRequestor.getResults();
        displayArea.append("The following Files are from Peer " +
        netPeerGroup.getPeerName() +"\n");
        if ( result != null ) {


                displayArea.append("Total Files from this peer are = " + result.length +
                "\n");
                for (int i=0;i<result.length;i++) {
                        ContentAdvertisement myAdv = result[i];


                        displayArea.append(myAdv.getName() + "\n");


                    //} //changed me
                        //if (!gotOne) {
                        if (gotOne) {
                                displayArea.append("Starting Download\n");
                                File tmpFile = new File( "DownLoadedFileNameIs_" +
                                myAdv.getName());
                                ContentListener myListener = new ContentListener() {
                                        public void finishedRetrieve(String url) {

                        displayArea.append("File Download Finished\n");
                                        //key=FileName();
                                        }
                                };
                        try {


                                GetSharedFileFounder request = new
                                GetSharedFileFounder(
                                netPeerGroup, result[i], tmpFile, myListener );
                        } catch ( InvocationTargetException e ) {
                                e.printStackTrace();
                        }
                        //gotOne = true;
                        }
                }

                //gotOne = true;
        }else {
                System.out.println("No results");
        }
}
}

public void run() {
        displayArea.append("Click on Button to send a remote search  request...\n");
        displayArea.append("On the message window please enter yes if you know the
        file name to download ...\n");
        displayArea.append("else no to see all shared files among the peers ...\n");
        displayArea.append("On the next window Either write the exat file name as
        image.jpg   \n");
        displayArea.append("Or  *.jpg to see only shared JPEG files or *.* to see all types
        of shared files ...\n");
        displayArea.append(" After wards you can do similarly to down load different
        files ...\n");
        try {
                myCms.getContentManager().share(new File("image.jpg"));
        } catch (IOException ex) {
                System.out.println("Share command failed.");
        }
}
```

Class MulticastMessageSender

This class provides a service in which a single peer is able to send a multicast message.

Constructor Summary

Public MulticastMessageSender():- Creates a user interface.

Method Summary

private void buildAndPublishOutputPipe()

private void createOutputPipe(PipeAdvertisement myPipeAdvertisement)

private void sendData(String data)

Example

```
private void buildAndPublishOutputPipe() {
        PipeAdvertisement aPipeAdv = null;
        try {
                FileInputStream is = new FileInputStream("PropPipeAdv.txt");
                aPipeAdv =(PipeAdvertisement)AdvertisementFactory.newAdvertisement(
                new MimeMediaType("text/xml"), is);
        } catch (Exception e) {
                System.out.println("failed to read/parse pipe advertisement");
                e.printStackTrace();
                System.exit(-1);
        }
        try {
                //myDiscoveryService.publish(aPipeAdv,DiscoveryService.ADV);
                myDiscoveryService.publish(aPipeAdv);

                myDiscoveryService.remotePublish(aPipeAdv,DiscoveryService.ADV);
        } catch (Exception e) {
                e.printStackTrace();
                System.exit(-1);
        }
        createOutputPipe(aPipeAdv);
}
```

Class MulticastMessageReceiver

This class provides a service in which one of the multiple peers able to receive a multicast message.

Constructor Summary

Public MulticastMessageReceiver():- Creates a user interface.

Method Summary

private void createInputPipe(PipeAdvertisement pipeAdv)
private void findAdvertisement(String searchKey, String searchValue)

Example

```
private void findAdvertisement(String searchKey, String searchValue) {
        Enumeration myLocalEnum = null;
        displayArea.append("Trying to find advertisement..."+ "\n");
        try {
                myLocalEnum = myDiscoveryService.getLocalAdvertisements(
                DiscoveryService.ADV, searchKey, searchValue);
                if ((myLocalEnum != null) && myLocalEnum.hasMoreElements()) {
                        displayArea.setText("Found Local Advertisement..." + "\n");
                        myPipeAdvertisement =
                        (PipeAdvertisement)myLocalEnum.nextElement();
                        createInputPipe(myPipeAdvertisement);
                }
                else {
                        DiscoveryListener myDiscoveryListener = new
                        DiscoveryListener() {
                        public void discoveryEvent(DiscoveryEvent e) {
                        Enumeration enumer;
                        displayArea.setText("Found Remote Advertisement..." + "\n");
                        DiscoveryResponseMsg myMessage = e.getResponse();
                        enumer = myMessage.getResponses();
```

```
                    while (enumer.hasMoreElements()) {
                              try {
                              String str = (String)enumer.nextElement();
                              myPipeAdvertisement =
                              (PipeAdvertisement)AdvertisementFactory.newAdvertisem
                              ent(XMLMIMETYPE, new
                              ByteArrayInputStream(str.getBytes()));
                              displayArea.setText("Trying to build pipe" +"\n");
                              createInputPipe(myPipeAdvertisement);
                              } catch(Exception ee) {
                                        ee.printStackTrace();
                              }
                    }
                    }
                    };
                    displayArea.setText("Launching Remote Discovery Service..." +
                    "\n");
                    myDiscoveryService.getRemoteAdvertisements(null,
                    DiscoveryService.ADV, searchKey, searchValue,
                    1,myDiscoveryListener);


          }
          } catch (Exception e) {
                    //displayArea.setText("Error during advertisement search");
                    System.exit(-1);
          }
}
```

Class BiDiMessageServer

This class provides a service that initiates a pipe to exchange message in both direction.

Constructor Summary

public BiDiMessageServer():- Creates a user interface.

Method Summary

```
public void run()
private String getUserName()
private void sendData()
```

Example

```
public void run() {
        try {

                BidirectionalPipeService.AcceptPipe incomingAcceptPipe =
                myBiPipeService.bind("bipipe");
                displayArea.append("Pipe Bind...\n");
                while (true) {
                        try {
                                BidirectionalPipeService.MessageListener
                                myListenerService =
                                new BidirectionalPipeService.MessageListener () {
                                public void messageReceived (Message msg, OutputPipe
                                responsePipe) {

                                String myMessageContent;
                                myMessageContent = msg.getString("DataTag");
                                Message sendMsg = null;
                                if (myMessageContent != null) {
                                        displayArea.append(myMessageContent + "\n");
                                        //displayArea.append("Sending Response...");
                                       outputPipe=responsePipe;
                                        //sendData(responsePipe);
                                        //try {
                                         //  String data=writeArea.getText();
                                                                      //
                                        sendMsg = myPipeService.createMessage();
                                        //        sendMsg.setString("DataTag", data);
                                        //        responsePipe.send(sendMsg);
                                        //} catch(Exception e) {}
                                        //displayArea.append("Waiting for message...\n");
                                        return;
                                } else {
                                        displayArea.append("Invalid tag\n");
                                        return;
                                }
                        }
                        };
                                //BidirectionalPipeService.Pipe newPipe =
                                incomingAcceptPipe.accept(30000, myListenerService);
                                newPipe = incomingAcceptPipe.accept(30000,
                                myListenerService);
                                displayArea.append("Accepted a pipe connection\n");
                                userName = getUserName();
                                displayArea.setText("");
                                } catch(Exception e) {}
                        }
                } catch(Exception e) {}
        }
```

Class BiDiMessageClient

This class provides a service that searches a pipe to exchange message in both direction.

Constructor Summary

Public BiDiMessageClient()

Method Summary

private String getUserName()

private void sendData()

private void findAdvertisement(String searchKey, String searchValue)

private void createPipe(PipeAdvertisement myPipeAdvertisement)

Example

```
	private void sendData() {
		//String data = "Hello my friend!";
		String data=userName + ">>" + writeArea.getText() + "\n";
		 displayArea.append(data);
		Message msg = myPipeService.createMessage();
		msg.setString("DataTag", data);

		try {
			 //displayArea.append(data + "\n");
			//displayArea.append(data);
			myPipe.getOutputPipe().send(msg);

			msg = myPipe.getInputPipe().poll(300000);
			//displayArea.append("From Server: " + msg.getString("DataTag"));
			displayArea.append(msg.getString("DataTag"));
			writeArea.setText("");
			} catch (Exception e) {
				System.out.println("Unable to create input/output pipe");
				e.printStackTrace();
				System.exit(-1);
			}
}
```

REFERENCES

[1]. Brendon J. Wilson, 2002, JXTA, Indiana 46290.201 West 103rd Street.

[2]. CLARKE, I., *SANDBERG, O., WILEY, B., AND HONG, T. W.* Freenet: 2000, A Distributed Anonymous Information Storage and Retrieval System. In Proceedings of the ICSI Workshop on Design Issues in Anonymity and Un observables Berkeley, California.http://freenet.sourceforge.net. Visited on March 9, 2008

[3]. Dejan S. Milojicic, *Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja1, Jim Pruyne, Bruno Richard, Sami Rollins 2 , Zhichen Xu* , 2003,Peer-to-Peer Computing, HP Laboratories Palo Alto HPL-2002-57 (R.1) July 3rd ,

[4]. GNUTELLA DEVELOPMENT FORUM. The Gnutella v0.6 Protocol, 2001. http://groups.yahoo.com/group/the_gdf/files/. Visited on March 10, 2008

[5] Grady Booch , 1994,Object Oriented Analysis and Design, Benjamin Cummings.

[6]. Nico Maibaum, Thomas Mundt.JXTA: A Technology Facilitating Mobile Peer-To-Peer Networks. http://doi.ieeecomputersociety.org/10.1109/MOBWAC.2002.1166946 Visited on March 9, 2008

[7] NIIT, 2005, System analysis and design for software engineers, Prentice-Hall.

[8]. http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/ Visited on March 10, 2008

[9]. http://whatis.techtarget.com/definition/0,289893,sid9_gci343142,00.html Visited on March 9, 2008

[10]. http://www.napster.com. Visited on March 9, 2008

[11]. http://gnutella.wego.com/. Visited on March 9, 2008

[12] http://www.baltimoresun.com/news/health/bal-bz.health21may21,0,2114464.story. Visited on April 12, 2008

[13] http://www.healthnet.org/hnethiopia.php aed satellite. Visited on May 12, 2008

[14] http://www.mnlab.cs.depaul.edu/seminar/spr2004/JXTAOverview.pdf. Visited on May 13, 2008

[15] http://users.cs.cf.ac.uk/Ian.J.Taylor/DistributedSystems/PPTSlides/JXTA.ppt Visited on June3, 2008

[16] http://www.csie.ntu.edu.tw/~b89014/JXTA_Intro.ppt. Visited on May 5, 2008

[17] http://www.thinkuknow.co.uk/parents/chat/good.aspx. Visited on May 5, 2008

[18] http://www.cragsystems.co.uk/why_use_use_cases.htm Visited on May 5, 2008

[19] http://www.agilemodeling.com/style/sequenceDiagram.htm Visited on May 5, 2008

[20] http://www.ibm.com/developerworks/rational/library/3101.html Visited on May 5, 2008

[21] http://www.agilemodeling.com/style/classDiagram.htm Visited on May 5, 2008

[22] http://www.objectmentor.com/resources/articles/umlClassDiagrams.pdf Visited on May 5, 2008

[23] http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/activity.htm Visited on May 5, 2008

[24] http://www.developer.com/design/article.php/2247041 Visited on May 5, 2008

Declaration

I, the undersigned, declare that this project is my original work and has not been presented for a degree in any other university, and that all source of materials used for the project have been duly acknowledged.

Declared by:

Name: ______________________________

Signature: ______________________________

Date: ______________________________

Confirmed by advisor:

Name: ______________________________

Signature: ______________________________

Date: ______________________________

Place and date of submission: Addis Ababa, June 2008.