

## **Abstract**

This project aims to develop a fully functioning peer-to-peer file sharing application. The advantage of this type of network system as opposed to the traditional client-server networks is that the information stored across the peer-to-peer network is uniquely decentralized. The design strategy begins with a literature review of computer network systems followed by the discussion of JXTA technology – the framework utilized for the application development. To establish the design flowchart for the final file sharing application, it is imperative to identify the application requirements including secure login, chatting, content sharing, uploading resources and many more. Moreover, a user-friendly graphical interface was created to ease the use of this application. The main frame window is the parent to all the components encapsulating the functionality of application components. This mainframe is then categorized into tabs, each being responsible for a specific task. Having explained the graphical outcome of this application, this report then develops on to the application implementation giving examples of various sections of source codes were appropriate for better understanding. Clearly, application testing is the final and one of the most essential stages in the application development procedure. This application has shown successful results in the Windows XP Professional, Solaris 10, Linux Ubuntu 7.4 and Linux Slackware 10 operating systems. Once the fully-functioning application has been established, there is always room for improvement. This could be included in the future work such as the download of multiple files at the same time and improving the speed of downloads.

## **Table of Contents**

### **Chapter 1: Project Introduction**

<b><u>4.....</u></b>	<b><u>1.1INTRODUCTION</u></b>
<b><u>5.....</u></b>	<b><u>1.2AIMS AND OBJECTIVES</u></b>
<b><u>5.....</u></b>	<b><u>1.3REPORT STRUCTURE</u></b>
<b><u>6.....</u></b>	<b><u>2.1INTRODUCTION</u></b>
<b><u>7.....</u></b>	<b><u>2.2COMPUTER NETWORKS</u></b>
<b><u>7.....</u></b>	<b><u>2.3.1Client-Server architect</u></b>
<b><u>7.....</u></b>	<b><u>2.3.2Peer-to-Peer architect</u></b>
<b><u>9.....</u></b>	<b><u>2.3HISTORY OF PEER-TO-PEER</u></b>
<b><u>9.....</u></b>	<b><u>2.3.1Napster</u></b>
<b><u>9.....</u></b>	<b><u>2.3.2LimeWire</u></b>
<b><u>9.....</u></b>	<b><u>2.4P2P APPLICATION DEVELOPMENT</u></b>
<b><u>10.....</u></b>	<b><u>2.4.1What is JXTA Technology?</u></b>
<b><u>11.....</u></b>	<b><u>2.4.2JXTA Protocols</u></b>
<b><u>12.....</u></b>	<b><u>2.5JXTA ARCHITECT</u></b>
<b><u>13.....</u></b>	<b><u>2.6WHAT CAN BE DONE WITH JXTA TECHNOLOGY?</u></b>
<b><u>14.....</u></b>	<b><u>2.7CONCLUSION</u></b>
<b><u>15.....</u></b>	<b><u>3.1INTRODUCTION</u></b>
<b><u>15.....</u></b>	<b><u>3.2FUNCTIONAL REQUIREMENTS</u></b>
<b><u>16.....</u></b>	<b><u>3.3P2P NETWORK DESIGN</u></b>
<b><u>17.....</u></b>	<b><u>3.4APPLICATION DESIGN</u></b>
<b><u>19.....</u></b>	<b><u>3.5GRAPHICAL USER INTERFACE (GUI)</u></b>

<u>26.....</u>	<u>3.6CONCLUSION</u>
<u>27.....</u>	<u>4.1INTRODUCTION</u>
<u>27.....</u>	<u>4.2JXTA ENTITIES</u>
<u>29.....</u>	<u>4.3STARTJXTA CLASS</u>
<u>31.....</u>	<u>4.4CHATTING CLASSES</u>
<u>32.....</u>	<u>4.5PEER LISTING CLASS</u>
<u>33.....</u>	<u>4.6SAEEDSHARING CLASS</u>
<u>34.....</u>	<u>4.7SEARCHFILE CLASS</u>
<u>36.....</u>	<u>4.8DOWNLOADFILE CLASS</u>
<u>36.....</u>	<u>4.9FRMMAIN CLASS</u>
<u>39.....</u>	<u>4.10TESTING</u>
<u>41.....</u>	<u>4.11SYSTEM REQUIREMENTS</u>
<u>41.....</u>	<u>4.12CONCLUSION</u>
<u>42.....</u>	<u>5.1INTRODUCTION</u>
<u>42.....</u>	<u>5.2APPLICATION OBJECTIVES</u>
<u>42.....</u>	<u>5.3TEST RESULTS</u>
<u>43.....</u>	<u>5.4CONCLUSION</u>
<u>44.....</u>	<u>6.1INTRODUCTION</u>
<u>44.....</u>	<u>6.2PROJECT CONCLUSION</u>
<u>44.....</u>	<u>6.3FUTURE WORK</u>
<u>45.....</u>	<u>7.1APPENDIX A – TERMS OF REFERENCE</u>
<u>48.....</u>	<u>7.2APPENDIX B: PROJECT PLAN</u>

<a href="#">49.....</a>	<a href="#">7.3APPENDIX C: ETHICS LIST</a>
<a href="#">50.....</a>	<a href="#">ETHICS CHECK FORM</a>
<a href="#">51.....</a>	<a href="#">8.1REFERENCES</a>
<a href="#">51.....</a>	<a href="#">8.2BIBLIOGRAPHY</a>

## **1.1 Introduction**

The internet comprises of thousands of computer networks over the globe that are interconnected with one another. Its usage is increasing with the population of mankind, encouraging the use of technology so as to maximise the usage of limited available resources. This is where Peer-to-Peer (also know as P2P) computing comes in - to help overcome problems such as lack of bandwidth and cost of software/hardware over the internet.

Peer-to-Peer computing is known to some as a type of networking and to others as a technology; however the most recognised aspect of P2P is its file sharing applications: Napster, eDonkey, eMule (based on eDonkey network) and LimeWire are some examples. P2P is distributive computing that can be used for a variety of purposes such as file sharing, hence reducing the cost and strain on systems that are sharing

resources among users. This project is based in and around the environment of this technology.

The knowledge gained throughout the research phase of this project helped to design, implement and develop a unique and fully functioning P2P file sharing application. The application developed in this project is a decentralised (i.e. does not need a main server) P2P file sharing application. This is the project deliverable and will be fully documented and tested throughout this report. The performance of this application is to aim for P2P manner: including Peers discovery, Peers connectivity and then the sharing and downloading of the resources once the peers are successfully connected to each other. To achieve this goal, this project will use the JXTA technology, developed by Sun Microsystems. This technology is suitable for use in P2P concept because it enables the devices that are connected to JXTA to share resources without the knowledge of their network architecture.

## **1.2 Aims and Objectives**

The aims and objectives of this project are:

- An initial study of the following topics:
  - ⇒ Computer Networks
  - ⇒ P2P technology
  - ⇒ History of P2P computing
  - ⇒ Description of some P2P file sharing applications
  - ⇒ JXTA technology
  - ⇒ History of JXTA technology
  - ⇒ Usage of JXTA technology in this project
  - ⇒ Future of JXTA technology
- The design and development of the final file sharing application using JXTA technology
- Fully testing the application with different computers in network so as to produce the final results
- Future of developed file sharing application.

## **1.3 Report Structure**

This report will cover the following areas:

- Project Introduction – A brief introduction of the project at hand as well as the final P2P file sharing application.
- Literature Review / Analysis – Context of the work and any technical details necessary for understating this project. Moreover, a description of JXTA technology is provided and an investigation into P2P networks is conducted.
- Design – Identify project requirements, show the design diagram and describe the File sharing application.
- Implementation and Testing – Describe the work undertaken and the results obtained with explanations of some parts of codes that have been used in the main application.

- Evaluation – Examine the completed work and the result achieved.
- Conclusion – Briefly restate the work undertaken, JXTA technology future, as well as the application future.
- Appendices A – Terms of References ( TOR)
- Appendices B – Project plan
- Appendices C – Ethics From
- References & bibliography – Resources studied and used for this project

## **2.1 Introduction**

This chapter presents the literature study and research that was conducted in order to establish the final working application. This includes a description of JXTA technology as well as the understanding of computer's network topology. This chapter will briefly introduce computer network's architect, P2P network's architect followed by a history of P2P networks. In addition, a brief history of Napster; the most popular multimedia file sharing application, will be presented. After a obtaining a basic understanding of computer Networks, the JXTA technology and its usage in the final output will be examined in depth.

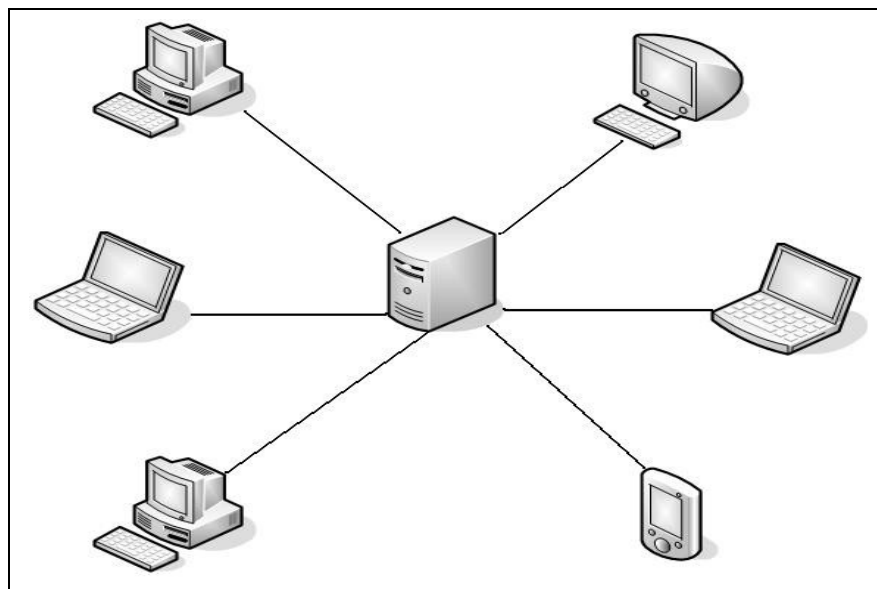
## **2.2 Computer Networks**

A computer network system involves the interconnection of two or more computers. This section includes the basic discussions of Client-Server and Peer-to-Peer networking architects.

### *2.3.1 Client-Server architect*

Client-Server architecture is a well-known network architecture in which the application processing is divided between client workstations and servers. It is essential that all clients' computers are connected to the server independently, so as to provide service for each individual client. Figure 1 shows the traditional client-server architect. Although this architect holds many advantages including location flexibility and maintainability, however as it can also be seen from Figure 1, this system involves numerous disadvantages including:

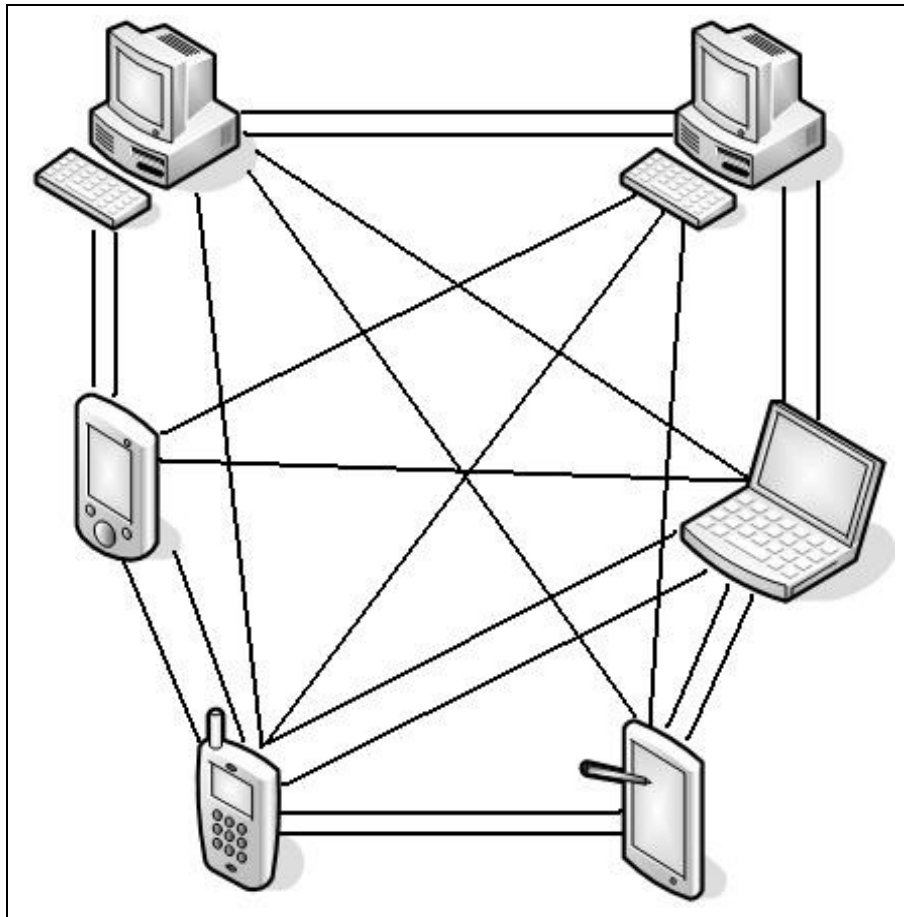
- Leak of bandwidth.
- Network communication failure if the service's provider server goes down.



**Figure 1: Client/Server Architect**

### *2.3.2 Peer-to-Peer architect*

Another type of network architecture, in wide usage today, is P2P – the subject of this project. In this architect, computers or any other devices that are capable of network communication are known as Peer. The main differentiating advantage of P2P networks is that each peer in the network can simultaneously act as server and client. An illustration of this architect given in Figure 2.



**Figure 2: Peer-to-Peer Architect**

Perhaps the most distinguishing characteristic of P2P is that each computer can have both a server module and a client module. This enables computers not only to have access to the software modules of other computers but also provide services for each other. The aim of this architect is to allow all clients to provide resources such as bandwidth, storage space, and computing power. As the number of nodes increase, the total capacity of the system increases, whereas in the client-server architect, adding more clients would result in slower data transfer [*Peer to Peer architecture*, 2004]. Another advantage of P2P networks is the increased robustness in the case of failures through the replication of data over numerous peers.

Having described the computing architectures, along with the advantages and disadvantages of each type, it is then possible to explore the history of P2P Networks as well as the widely known P2P application including Napster.

### **2.3 History of Peer-to-Peer**

Peer to Peer, previously known as the host-to-host concept, was at first used to establish equal communication between two peers. In this concept each peer can act as server and client simultaneously. The basic P2P technology has been around from when USENET was created in 1979 by two graduate students "Tom Truscott and Jim Ellis" in Duke University. USENET was established for the purpose of exchanging



information between UNIX machines. The next section will present the typical examples of P2P file sharing applications include Napster and LimeWire.

### *2.3.1 Napster*

Napster is one of the best known P2P file sharing applications; written by "Shawn Fanning" in 1999. Napster was first created for those who had difficulty in finding and downloading MP3 music files. Basically, it is a combination of a search engine and a file-sharing application.

Napster is a file sharing service that facilitates the location and exchange of files, which usually include images, audio or video, via access to the internet. Being one of the first sharing applications, it then paved the way for other programs including Kazaa, Morpheus, LimeWire and Bearshare.

However, in 1999, Napster was penalized by the Industry Association of America (RIAA) for copyright reasons and was forced to pay a fee of \$26 million to resolve its legal battle with songwriters and music publishers. Today, it includes a subscription service and receives percentages from sale.

### *2.3.2 LimeWire*

Another popular file sharing application is LimeWire. This application makes use of a decentralised P2P network and enables any type of file to be shared between the clients.

## **2.4 P2P Application Development**

One of the well known frameworks for developing P2P application is JXTA (**Juxtapose**) which is an open source P2P protocol developed by Sun Microsystems in 2001. In this protocol, all devices that are connected to the network can exchange messages in XML format and work together, independent of the underlying network topology. JXTA is based on XML format and it can be used in any modern languages such as: Java, C# and C/C++. The goal of JXTA protocol is that peers can share resources without worrying about network topology. For instance, peers can share resources that are behind NAT or Firewall same as direct peers to the internet.

### *2.4.1 What is JXTA Technology?*

JXTA is an open source peer-to-peer framework developed in Apache open source model by Sun Microsystems. According to Sun Microsystems: "*JXTA technology is a set of open, generalized peer-to-peer protocols that allows any connected device*

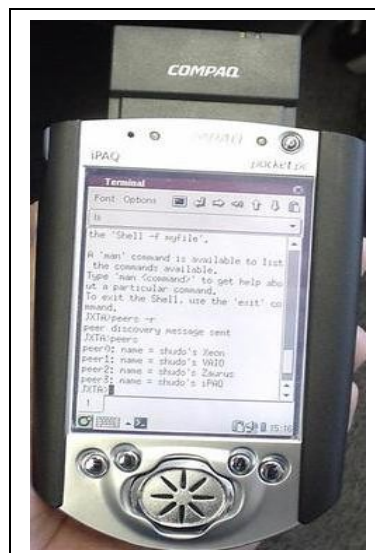
*(cell phone to PDA, PC to server) on the network to communicate and collaborate"*  
[Sun Microsystems, 2001]

The basic goals that the JXTA platform achieves include:

- Peers discover each other without any centralised server
- Peers have to login into JXTA default group
- Peers can then search, create and join to their specified groups. In the other hand, Peers should be self-organised into peer groups.
- Peers then advertise and try to discover available network resources
- Peers should be able to communicate with each other
- Peers are able to monitor each other
- Peers in JXTA network are not required to know network topology to be able to communicate with each other

As mentioned earlier, the JXTA platform does not require the use of any particular programming language or any operating system. Furthermore, the JXTA protocols are designed to be independent and do not require the use of any authentication, security or encryption model. JXTA is very simple and generic platform and it is designed with the basic functionality to host all types of network services. Its protocols can also be implemented in many other network protocols such as: TCP/IP, Bluetooth and HTTP.

Development and implementation of JXTA can be done in numerous programming languages such as: Java, C/C++, Perl and Python. Because its independency, the JXTA protocols can be easily implemented in various devices each with completely different software stacks. Figure 3: JXTA on PDA is an illustration of JXTA on handheld devices.



**Figure 3: JXTA on PDA**

(Available from: [http://jxta.free.fr/JXTA-SHELL/JXTA\\_Shell.html](http://jxta.free.fr/JXTA-SHELL/JXTA_Shell.html), 06/04/2008)

## 2.4.2 JXTA Protocols

There are only six protocols in the current JXTA system (the current version of JXTA is 2.5), with each protocol designed for handling JXTA services.

These six protocols that are creating the JXTA core layer are listed below:

1. Peer Resolver Protocol (PRP)
2. Peer Discovery Protocol (PDP)
3. Peer Information Protocol (PIP)
4. Pipe Binding Protocol (PBP)
5. Peer Endpoint Protocol (PEP)
6. Rendezvous Protocol (RVP)

Each protocol is responsible for handling specific services in the JXTA platform and the collection of these protocols will create the core layer of the JXTA platform.

### ***Peer Resolver Protocol (PRP)***

This protocol is used to send a query to one or more peers and receive their response to the query. The queries can be sent to all the peers or to a specific peer within the group.

### ***Peer Discovery Protocol (PDP)***

In this protocol peers can advertise their resources as well as discover other peers' resources, for example: Peer group advertisement, services and pipes. Within this protocol every peer publishes their resource's advertisement through JXTA network. Advertisements here are represented as XML document and they have metadata structure that describes the peer's network resources.

### ***Peer Information Protocol (PIP)***

This protocol will obtain the peers' status information such as: uptime, traffic load, capabilities and other information.

### ***Pipe Binding Protocol (PBP)***

Through this protocol, peers can establish communication or pipe with one or more peers.

### ***Peer Endpoint Protocol (PEP)***

Endpoint protocol is responsible for peer's route discovery, so peers can discover a route or sequences of hops used to send messages between peers. For example if peer "Alice" wants to send message to "David" and there is no direct route between "Alice" and "David" then peer "Alice" needs to find intermediary peer(s) who can redirect the message to "David". Endpoint protocol is also known as Endpoint Routing Protocol (ERP) and it is used to detect the route information between peers. Therefore, if network topology changes or it becomes unavailable, ERP can be used to determine an alternative route. Figure 4: Endpoint Routing Protocol [JXTA Programming Guide, 10/09/2007] is an illustration of Endpoint routing protocol – it can be seen from this figure that when there is no direct route between peer "A" and "B", then an alternative route such as peer "C" will be used to send messages through the peers.

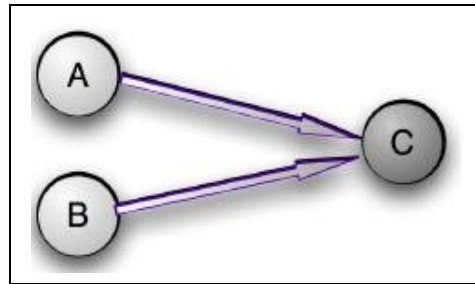


Figure 4: Endpoint Routing Protocol [JXTA Programming Guide, 10/09/2007]

### ***Rendezvous Protocol (RVP)***

Rendezvous protocol is used to propagate services through the entire peer group. In the peer group, peers can be either Rendezvous peer or peers that are listening to rendezvous peer. This protocol allows peers to send messages to all listening instance of the services. Rendezvous peer can be used by Peer Resolver and Pipe Binding protocols in order to propagate messages through Peers group. However, there is no guarantee that the messages propagated by the rendezvous peer will always reach all the peers in the peer group. Figure 5 is an illustration of Rendezvous peer and message propagation through the peer group.

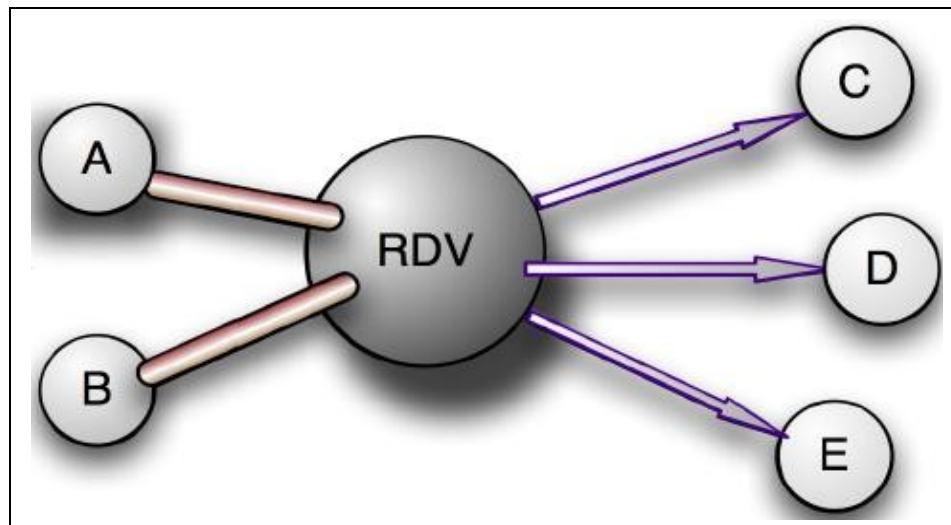


Figure 5: Rendezvous Protocol [JXTA Programming Guide, 10/09/2007]

## **2.5 JXTA Architect**

The JXTA software is made up of three layers, as shown in Figure 6:

- Core Layer
- Services Layer
- Applications Layer

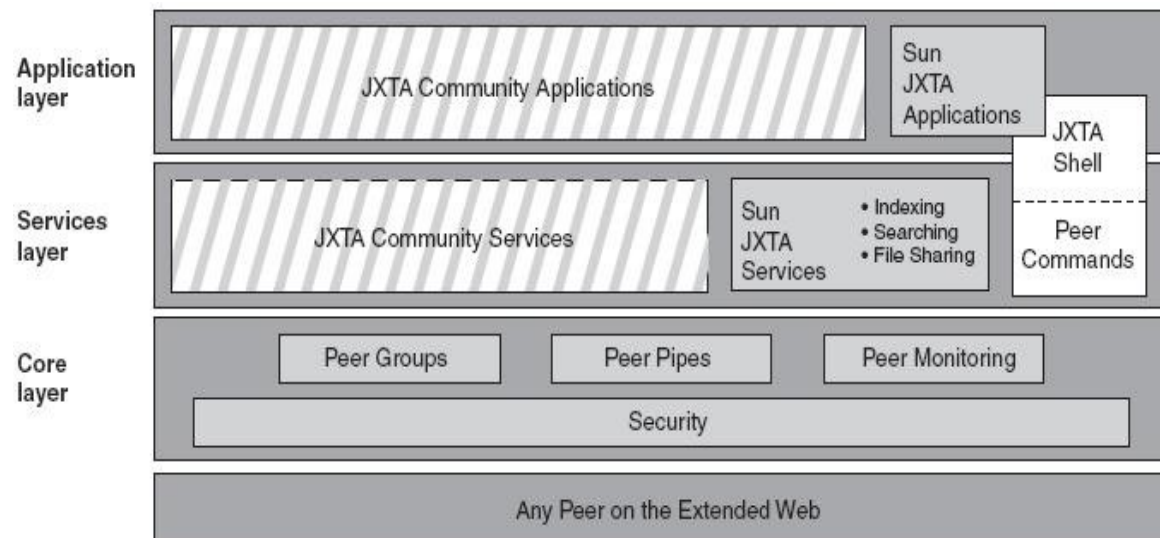


Figure 6: JXTA Layers [Joseph D. Gradecki, 13/09/2002]

### ***Core Layer***

The Core layer is made of all the essential and common factors of P2P networking, such factors are: Peers, Peer Discovery, Peer monitoring, Peer communication and associated security primitives. This layer is shared between all Peer-to-Peer devices so as to enable interoperability.

### ***Services Layer***

This layer is above Core layer and it is a common layer for P2P networking however it is not an absolute necessity for P2P networking. Examples of network services include: File Sharing, Message transferring services (i.e. AOL messenger), Storage, distributive file systems and protocol translations.

### ***Application Layer***

In this layer an application view of software can be viewed as a service by other peers. Consequently, this layer can be developed by developers who are creating JXTA applications and it can be used to create service layer for other peers.

## **2.6 What can be done with JXTA Technology?**

The JXTA technology helps developers to deploy a P2P application and services all using the same P2P standard which is independent of any computer programming language or any operating systems. It can also be implemented in numerous devices with completely different software stacks as well as the ability to work on any network topology without knowledge of the network architect.

## **2.7 Conclusion**

This chapter described different types computing architectures, including client-server and P2P. Although each method has advantages, they are also accompanied by several disadvantages. The client-server architect has a maintainability advantages, however it would have slow data transfer when the number of clients are increased, while this is not the case for P2P architects. Having had a brief understanding of these concepts, it is then possible to move to the next phase of this project - analysis and design so as to produce the final working application.

### **3.1 Introduction**

The JXTA platform, which this project is based on, was discussed in the previous chapter. This chapter will analyse the steps that have been taken to create a fully functioning file sharing application. Moreover, it will state the requirements for the final product as well as showing design diagrams with an attempt to fully explain the produced P2P file sharing application.

### **3.2 Functional Requirements**

These requirements are based primarily on the areas that are critical for this P2P application to compete with other applications. The most popular applications around have similar features which users have come to expect from P2P applications. These features could be shortened to the ability to log on as an individual to a network of peers. Once logged on to the network the user should be able to search for resources available from other peers, make connections to these peers and download the resources to be stored locally. Moreover, the users should be provided a means for sharing their own resources on the network and being able to upload these resources to the waiting peers whenever shared files are requested.

Peers in P2P networks should also be able to find one another and communicate with each other in order to share their contents. Hence, peer discovery is another functional requirement for the establishment of a fully functioning P2P file sharing application. The import goal, to have successful communication between peers is to use technology that is independent of the network topology and is usable in different networks such as: NAT networks and Firewalled networks. Hence, the only thing that the peers need to do is to log into global network and then advertise their resources to other peers in the group.

Logging into peer's global network might have different approaches such as: Secure login or Anonymous login. If peers log securely into the network, then they can be identified by a unique username and password. This will make it easier to find the owner's resources that have been shared through the network as well as the verification of user peer if the network is designed to be secure and only trusted users have to be able to login.

Once peers are logged into the global network then they can start to listen to other peers to propagate their request or they can start to search for requested contents when demanded.

Requested contents are often a file name that are supplied by the user to be shared within other peers in current group. Peers start to search for given name in the entire network and if the appropriate results are found, peer returns file name as well as file name's owner details such as file JXTA ID, so peer can download the file by knowing the file ID.

### 3.3 P2P Network Design

In P2P networks there are only two types or responds to requested messages:

- Return Results if the peer has the requested message
- If not, pass it to other peers

These methods can be illustrated in Figure 7, therefore "Peer1" sends its request to the nearest which is "Peer2", if "Peer2" has the contents of requested message then it will respond to it, otherwise the request will be propagated to the other peers.

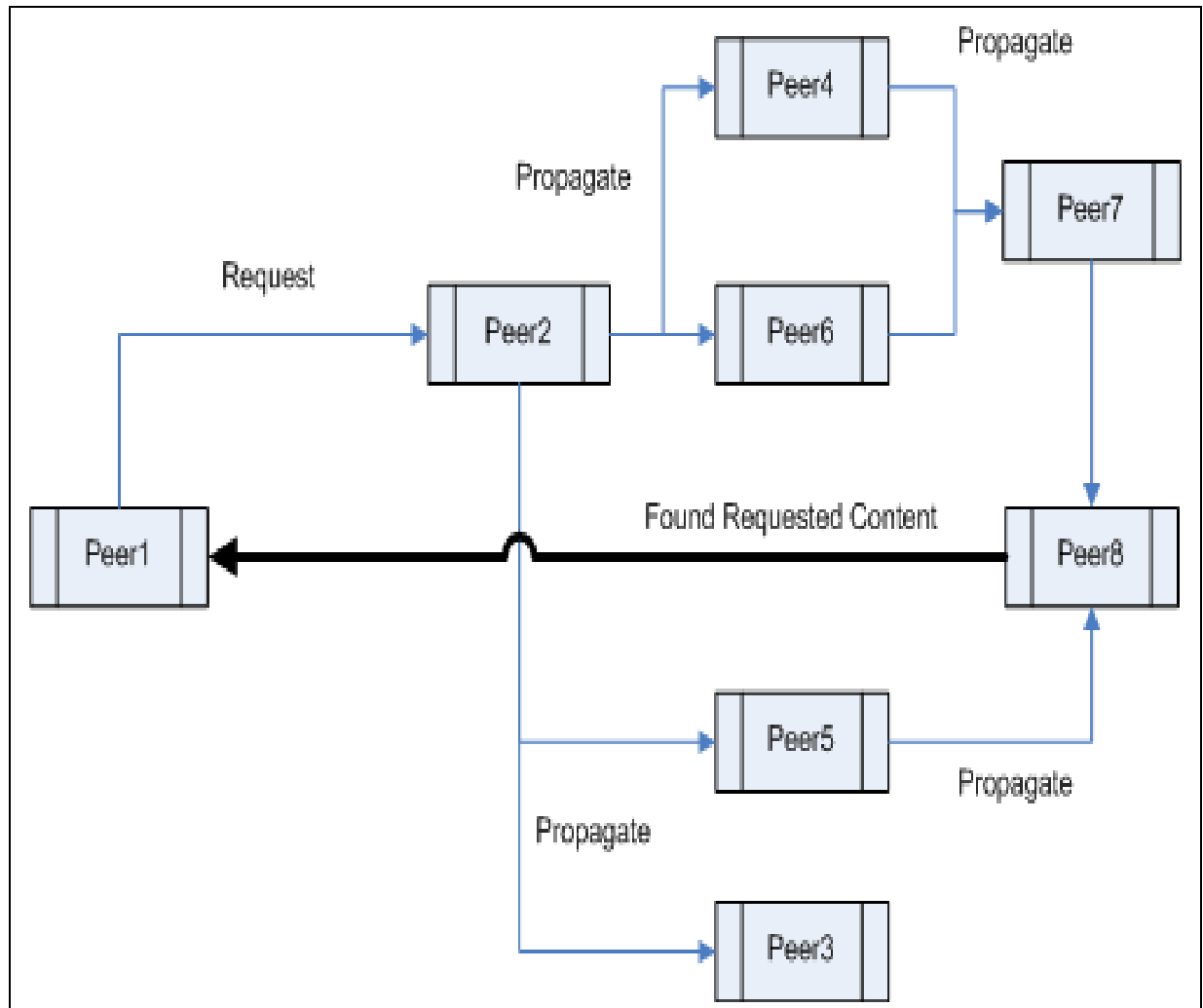


Figure 7: Peer-to-Peer Networks

Knowing these methods will help in the design phase so as to create a better and more user friendly application.



### **3.4 Application Design**

The system requirements and network design have been investigated and the areas that have to be incorporated into the system have been identified for the final file sharing application. The following aims need to be accomplished for creating a fully functioning application:

- A secure way for peers to log into JXTA network
- Users should be able to and have privilege to create configuration file for further usage.
- Users should be able to choose the directory with its contents to share it over peer group.
- Peers should be able to login into JXTA default network (also known as NetPeerGroup)
- Peers should be able to search for predefined group (SaEeDGroup)
- If SaEeDGroup is not found in the searching process, peers should be able to create their group with predefined group UID.
- Peers should be able to login into SaEeDGroup
- Peers should be able to discover each other within the group
- Peers should be able to share their contents with other peers in the same group
- Peers should be able to start listening to received queries and be able to respond to received queries from other peers
- Peers should be able to communicate with each other and search for specified contents
- If peers could not find the requested queries in their contents they should propagate the request to other peers in group
- Peers should be able to chat with each other
- Peers should be able to list other peers in same group
- Peers should be able to upload their resources
- A graphical user interface for application must be provided for ease of use
- Users must accept the application's terms and conditions. Otherwise, they should be able to continue the use of application
- Users should be able to list and view their own shared resources
- Peers should ensure that the download was not corrupted by verifying the downloaded file, once it has been completely downloaded.

Figure 8: Main Application Flowchart presents a well designed flowchart of the final file sharing application. This will facilitate the better understanding of application's process hierarchy.

The next phase of this chapter will investigate the Graphical user interface (GUI) of the final output, attempting to discuss each item of the final application in more depth.

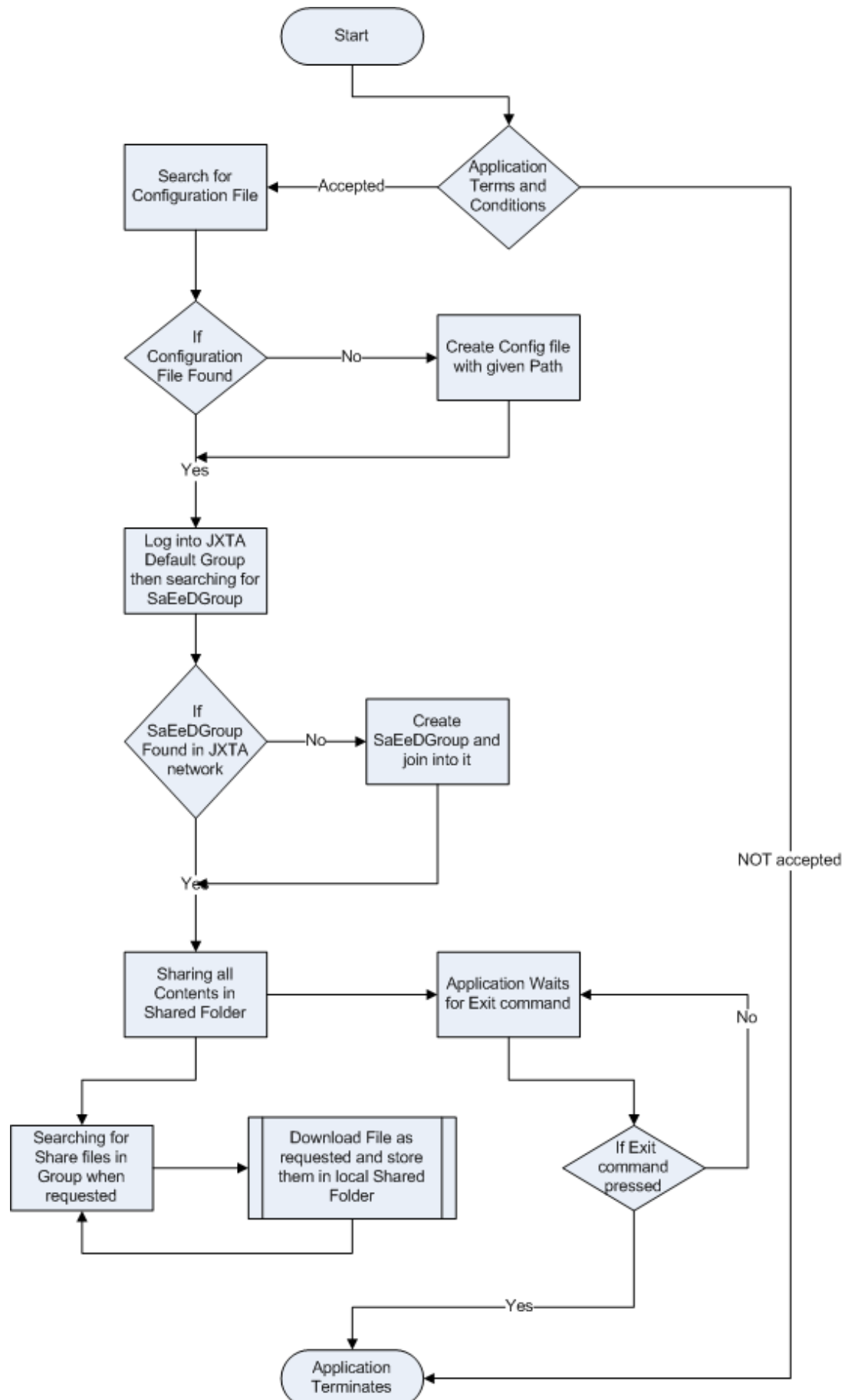


Figure 8: Main Application Flowchart

### 3.5 Graphical User Interface (GUI)

The Graphical Interface of an application is basically made to ease its use. This will allow users to increase their performance in using the application's components.

The final output of this project has a main Frame window which encapsulates the functionality of all the components used in this application. It is also the Parent of other used components. Figure 9 shows the main frame that has been used to create final application. The following part will describe the functionality of each item in the mainframe.

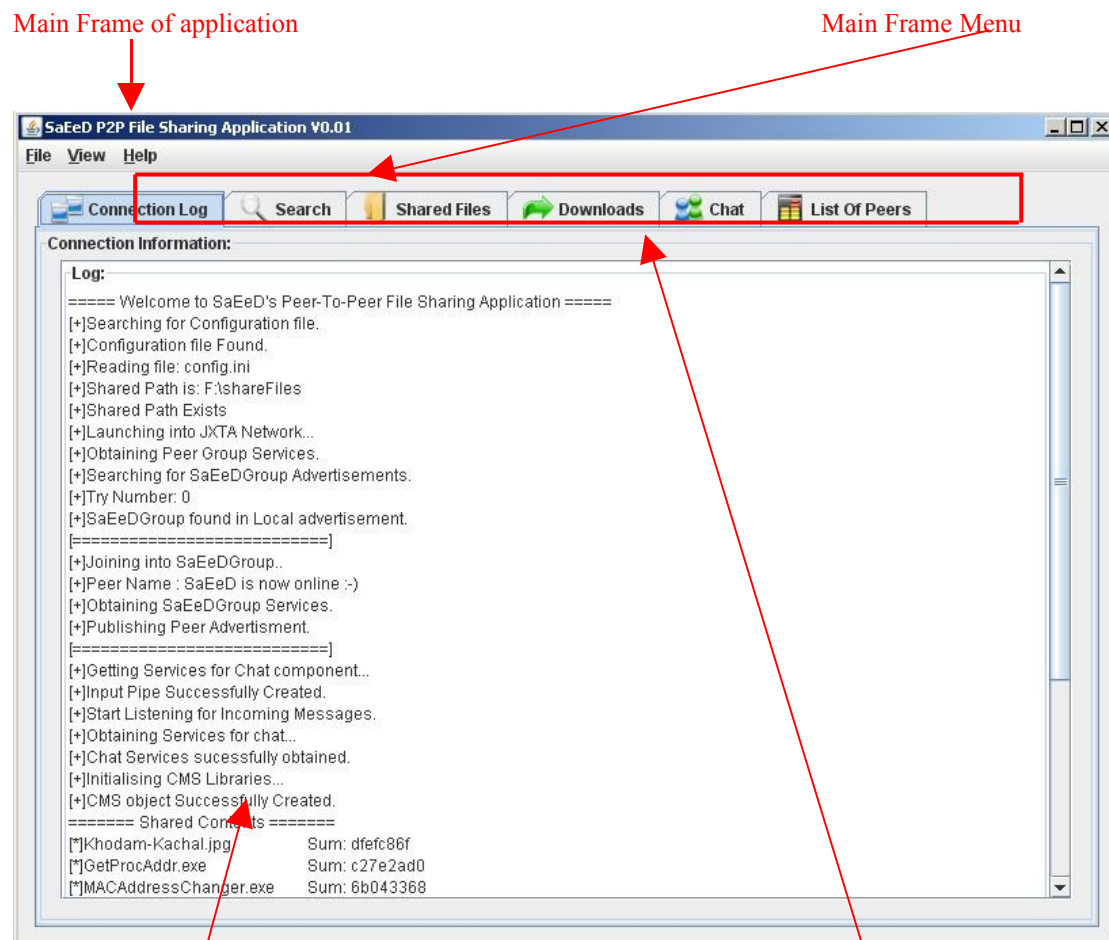


Figure 9: Main Frame of Application

Logging the information  
Obtains from classes that  
Been used in application

Tabs in application, each  
tab represents different  
Functionality.

## Main Frame Menu

This item is made of three elements with each representing a different functionality:

- 1) File
- 2) View
- 3) Help

With the **"File"** item, users can disconnect from sharing group and reconnect to sharing group. In addition, user can easily exit from main program. An overview of this item is presented in Figure 10



Figure 10: File Menu Item

**"View"** item is designed so that the user can have different appearance styles for the main frame, such as: Window looking style, Swing looking style and other styles. It is up to the user to choose the frame style they prefer. Figure 11 shows the different designs in main frame.

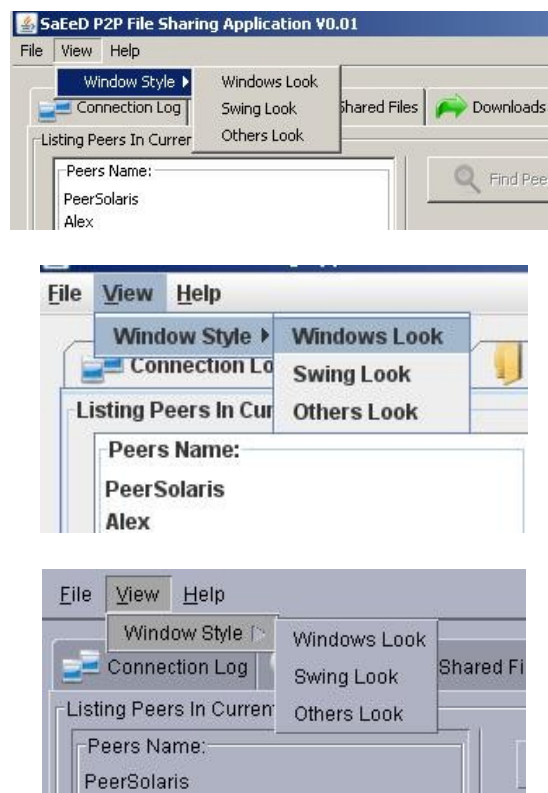


Figure 11: Frame Styles

“**Help**” item is used to show a brief description of this application as well as showing the Terms and Conditions of using this application. This made of dialog box which is also pops up at the beginning of application to make sure that the terms and conditions are accepted by the user. Figure 12 gives an preview of this dialog box:

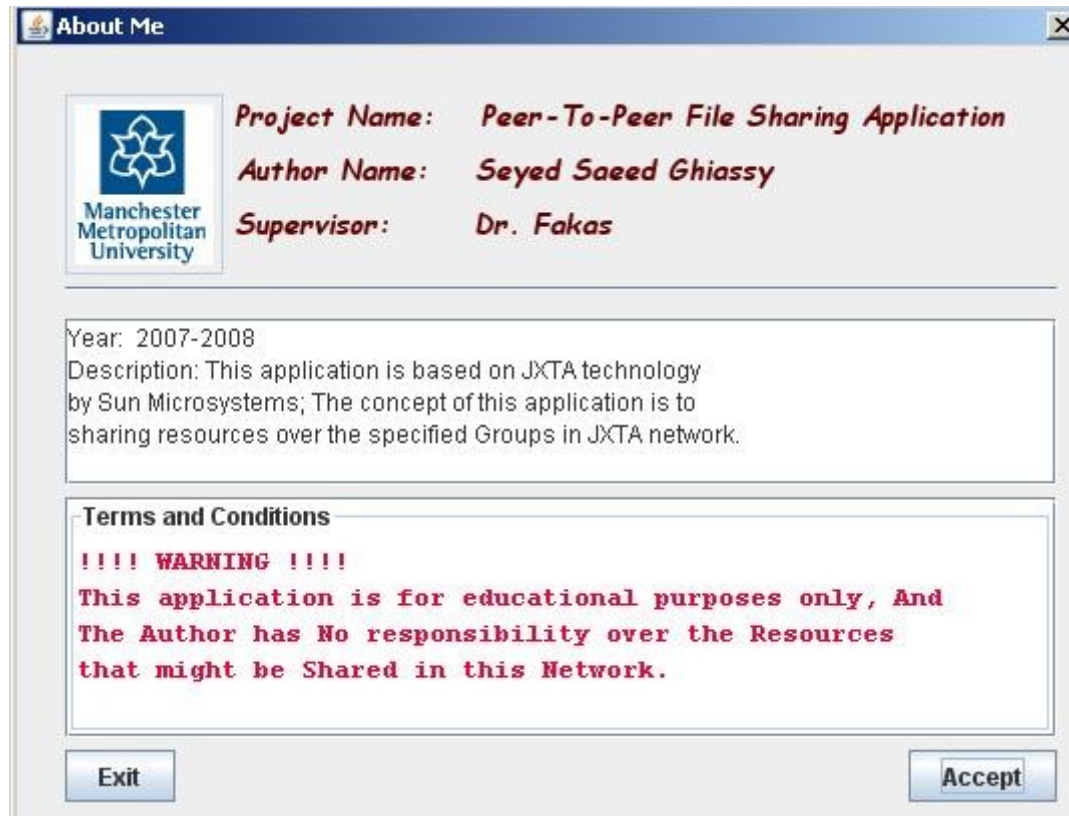


Figure 12: AboutMe, Terms and Conditions

### Tabs Panels:

The main Tab in main application is made of different panels, and each panel is responsible for a specific functionality of this application. They have been used in Tab component for better accessibility, so that the frame can be well categorised. In here, each Tab panel and its functionality in the overall frame will be investigated independently.

#### *‘Connection Log’ Tab*

The Connection Tab panel is responsible for logging the connection status and collecting all the information of the classes that have been used in this application. Consequently, user can keep track of what is really happening in the application. Consequently, it can be utilised for debugging purposes. Some examples of information collected by this tab are: Name of peer, status of peer in JXTA global network, shared path, shared contents and also status of all classes that have been used in this application.

### 'Search' Tab

This tab has been designed so that peers can search for a specified file name in the group. The content request will then be sent to all available peers in group. If any peer has requested content then they will reply to the peer with details of the content, so they can be downloaded later. Figure 13 is an illustration of the search tab with details of each of its buttons.

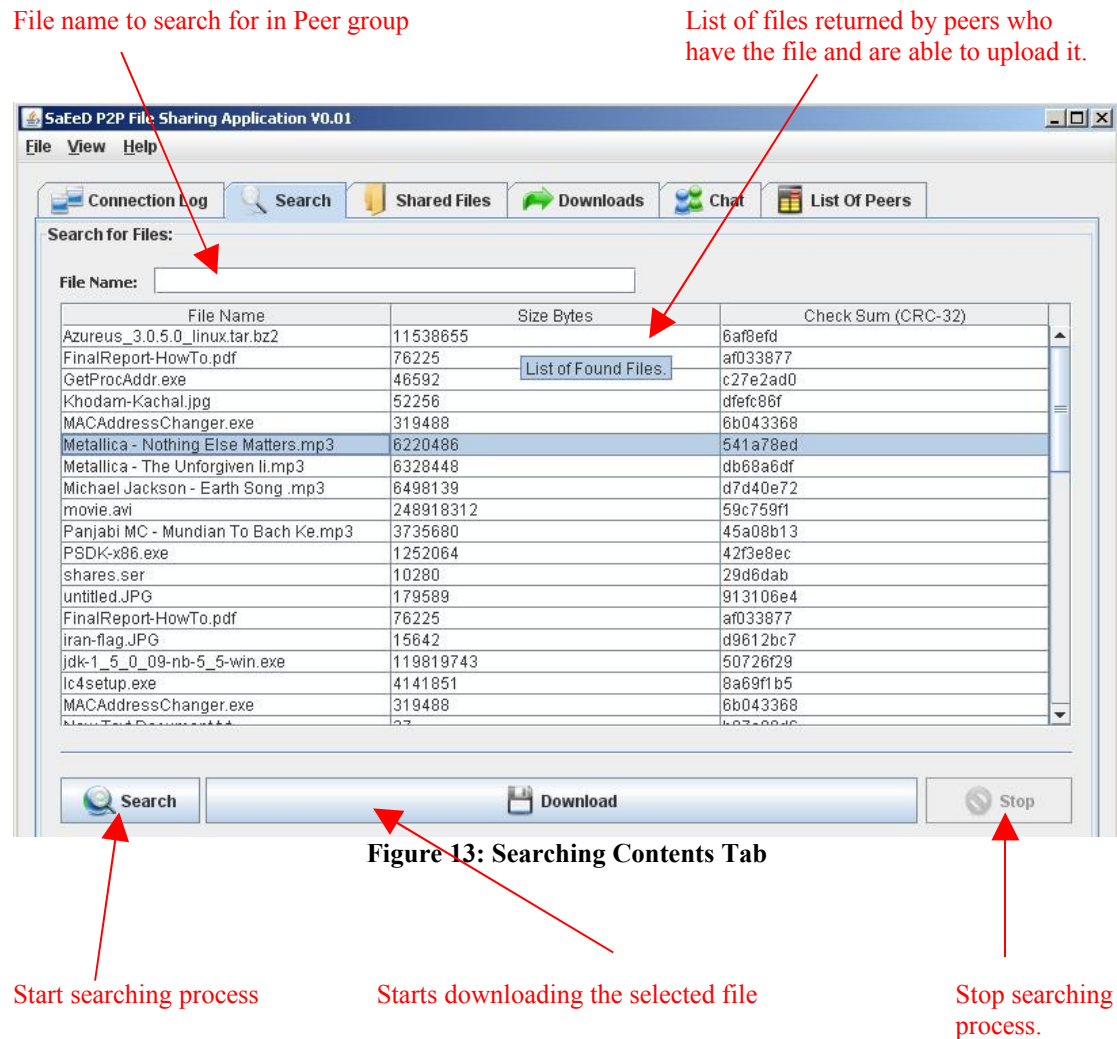


Figure 13: Searching Contents Tab

As it can be seen from this figure, there is a column in files lists which contains the file check sum - this check sum has been received from the peer who is providing the file. So that once the file has been downloaded by user, then they can regenerate check sum of file so as to make sure download was successful and file was not corrupted.

When user enters a file name, this will start to search for file within the group, and the searching progress will be continued until user stop this progress. Figure 14 is a well demonstrated flowchart of 'searching for file' progress.

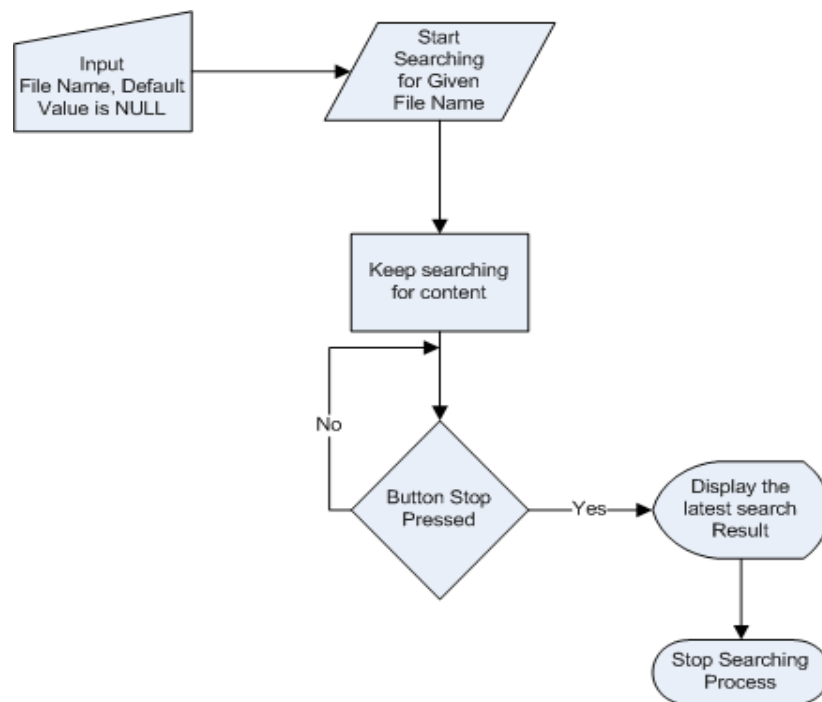


Figure 14: Searching for File flowchart

### 'Shared Files' Tab

This Tab panel has been designed so that users can view their shared and downloaded files. This is also includes information about files in shared folder such as: File Type, File Name, File size in byte and file checksums. This Tab is shown in Figure 15.

List of Shared Files:			
Type	File Name	Size Bytes	Check Sum (CRC-32)
	Khodam-Kachal.jpg	52256	dfefc86f
	GetProcAddr.exe	46592	c27e2ad0
	MACAddressChanger.exe	319488	6b043368
	FinalReport-HowTo.pdf	76225	af033877
	movie.avi	248918312	59c759f1
	untitled.JPG	179589	913106e4
	Michael Jackson - Earth Song ....	6498139	d7d40e72
	Punjabi MC - Mundian To Bach ...	3735680	45a08b13
	Metallica - Nothing Else Matters...	6220486	541a78ed
	Metallica - The Unforgiven II.mp3	6328448	db68a6df
	shares.ser	14486	f179f17f
	PSDK-x86.exe	1252064	42f3e8ec
	Azureus_3.0.5.0_linux.tar.bz2	11538655	6af8efd

Figure 15: Shared Files Tab

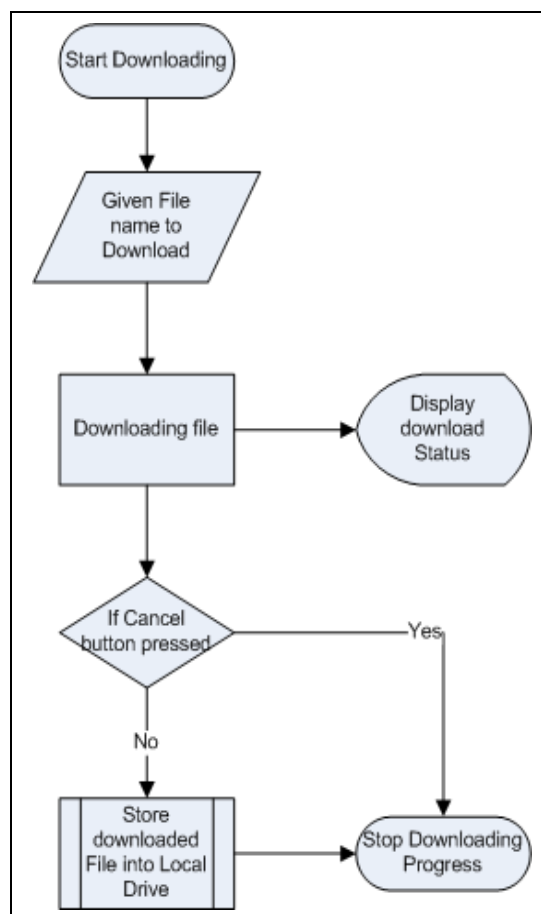


### ***'Downloads' Tab***

This tab presents all the information regarding the current downloading process. It includes information such as: Downloading File Name, Status of downloading component, percentage of completed download. In addition users can interrupt their download process by the cancel button. Verification button is also designed to check the file once download has been completed to ensure that the download was not corrupted. This verification and downloading progress will be discussed in more detail in the following chapters. Figure 16 gives a preview of the Downloading Tab. The downloading process can be demonstrated by the flowchart model given in Figure 17:



**Figure 16: Downloading Tab Panel**



**Figure 17: Downloading Flowchart**

### ***'Chat Tab' Panel***



To have a complete application, it is necessary that in addition to sharing their contents, users are also able to chat with one another. Thus, this tab panel is designed for this purpose so that users can transfer messages between one another – as shown in Figure 18.



Figure 18: Chat Tab Panel

### ***'List of Peers' Tab***

Peers monitoring is one of the most important aspects in JXTA P2P file sharing, enabling the peers to save information about one another. This Tab panel attempts to find the peers who are located within the same group. Figure 19 give a preview of this tab panel. This is followed by Figure 20 – demonstrating the flow chart for the peer listing procedure.

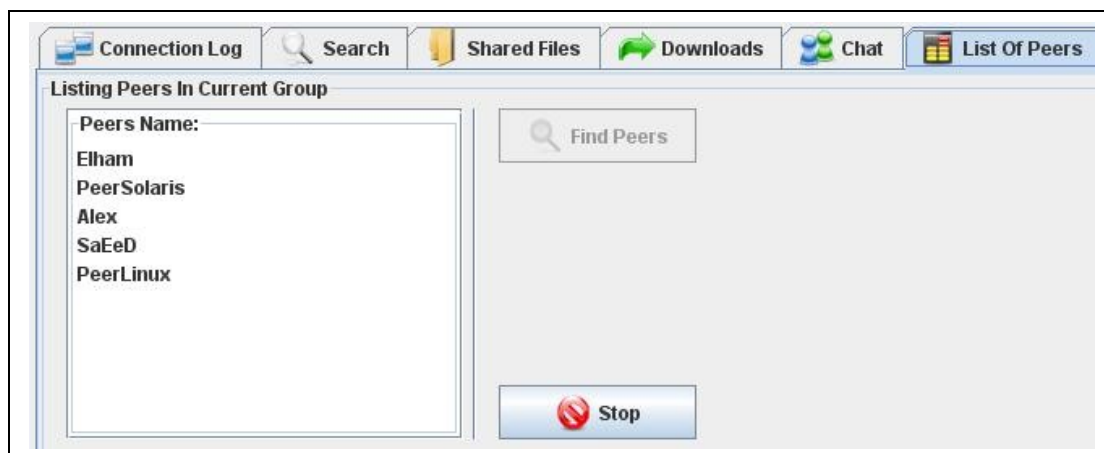


Figure 19: Peers Listing Tab Panel

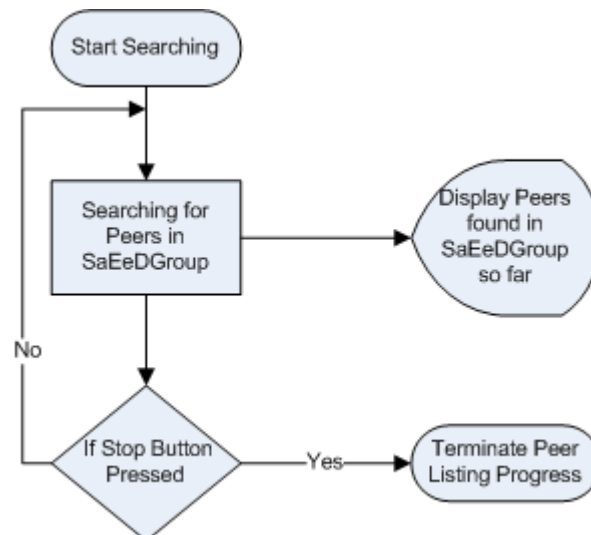


Figure 20: Peers Listing Flowchart

### 3.6 Conclusion

The analysis and design phase discussed in this section is the most important phase because it will determine the outcome of final file sharing application. This chapter will also help to give a better understanding of the implementation phase. Moreover, the basic outcome of application has been explained in this chapter. The next chapter will investigate around implementation environment of the application and discuss how the application actually works.

## **4.1 Introduction**

Having fully explained the graphical outcome of the application in pervious section, this chapter will investigate into the implementation of the application and the components used to create final output. In addition, the classes used in this application and their aims will be also considered in this chapter. The aim of this part is to explain the components used in this application, as well as the usage of JXTA in the application functionality.

## **4.2 JXTA Entities**

There are several entities that are common in JXTA applications to act in P2P manner.

Entities include:

- Peer
- Identifiers
- Peer Group
- Advertisements
- Messages
- Pipes

### ***Peers***

The most common component in all P2P systems is the peers. The Peer is software that works on the computer device with the ability to communicate with other peers. To create a P2P system, it is fundamental that peers have ability to communicate with one another.

The essential peer's elements in JXTA network is listed below:

- Peer identifier (ID)
- Peer membership
- Peer transport – communication within other peers in group

### ***Identifiers***

Every resource in the P2P networks need to be referenced somehow. In JXTA network, entities can be referenced by unique ID; JXTA uses 128-bit Universal Unique Identifier (UUID) to identify and reference resources in the P2P network. These UUIDs are generated by the JXTA platform and are guaranteed not to be the same by adding the network details and name to them. The simple example of group UUID is: "jxta:uuid-4E0742B0E54F4D0ABAC6809BB82A311E02". These UUIDs can be used to identify different entities in the P2P network such as: Peer, Module Advertisement, Services, Peer Group...etc.

## Peer Groups

Peer Group entity in JXTA network is made of peers cooperating to one another to share their resources and services with each other. By default, all peers are a member of NetPeerGroup, which is JXTA default group. When peers use JXTA protocols, they first join to the NetPeerGroup which is a root group of JXTA and then they can search for specified group and join or create a new group.

## Advertisements

Advertisements are sets of XML messages and metadatas that are used by peers to advertise their services to other peers within same group. When a peer publishes its advertisement over the P2P network, other peers come to know its available resources and services which are meant to be shared over network. For example: Files advertisement and Pipe Advertisement. Hence, when peers discover these advertisements, they will know the provider source and will establish communication to the owner of the advertisement so as to use its services as well as its resources. An example of XML advertisement file is represented in Figure 21 -showing the pipe advertisement. Peers can use pipe advertisement to transfer messages with one another - also known as chatting services.

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
  <Id>
    urn:jxta:uuid-59616261646162614E5047205032503337165B7AB98642E6A6918591D053591704
  </Id>
  <Type>
    JxtaUnicast
  </Type>
  <Name>
    SaEeDPipe
  </Name>
</jxta:PipeAdvertisement>
```

Figure 21: Pipe XML advertisement

## Messages

Messages are XML objects that can be exchanged between peers. Peers can decide on the format of the message being sent: it can be either Binary or XML.

## Pipes

In JXTA network, pipes are used to send message between peers. According to JXTA developer's guidance article: "*Pipes are an asynchronous, unidirectional and non-reliable (with the exception of unicast secure pipes) message transfer mechanism used for communication and data transfer*" [JXTA Programming Guide, 10/09/2007]. Figure 22 gives an illustration of the virtual communication channel or pipe in JXTA network:

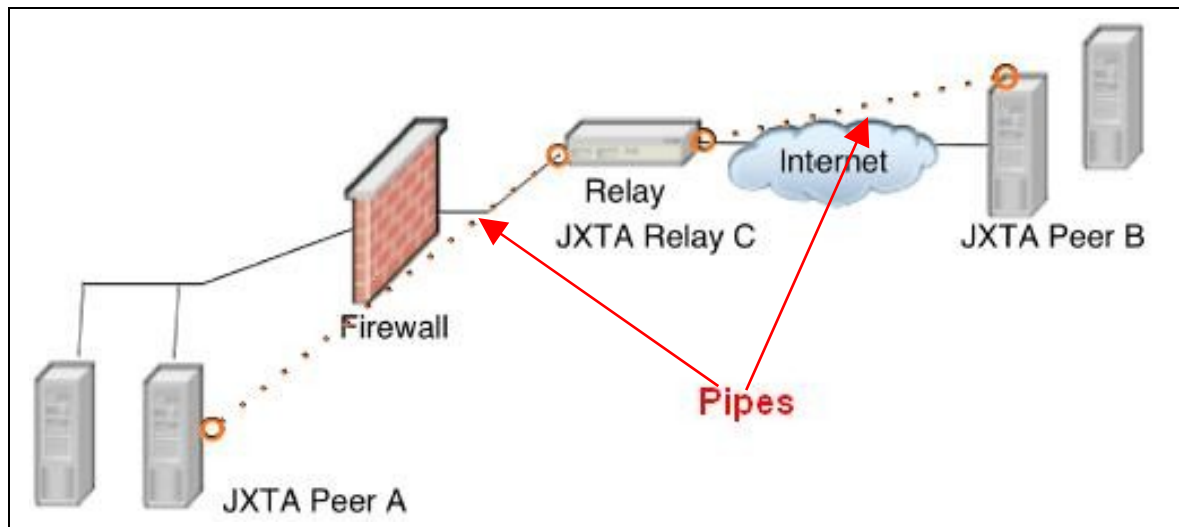


Figure 22: Pipes in JXTA network [Jxta Programming Guide, 10/09/2007]

### Content Management Service (CMS)

The Content Management Service is an extension of JXTA libraries and it is used to provide basic functionality for file sharing purposes. For example this library contains components for: sharing contents within other peers in the group, responding to requested content queries from other peers, uploading contents to other peers, downloading contents from other peers, information about the downloading progress and many other components that are beyond the concept of this report.

The next section of this chapter will investigate the components that have been used to create the final file sharing application as well as showing the applications class diagram. Each class and the code used for its creation will be explained individually with its diagram. Then, the final output will be tested and a screen shot of the test will be presented to help users for future usage of this application.

### 4.3 StartJXTA Class

The StartJXTA class is part of the whole application which provides the needed communication to JXTA network in order to join into Peer's group so that peers can share their contents. This class will first launch itself into JXTA global network also known as NetPeerGroup. When it is logged into NetPeerGroup then available services such as Discovery Services will be obtained from NetPeerGroup. The following code is a demonstration of this progress:

```
try{
// Log into JXTA network
netPeerGroup = PeerGroupFactory.newNetPeerGroup();
}catch(PeerGroupException e){
//if could not login then Exit
System.out.println("[-]Fatal Error:" + e.getMessage());
e.printStackTrace();
System.exit(-1);
}
//Obtaining the Services
myDiscoveryService = netPeerGroup.getDiscoveryService();
```

Once, peer logged into JXTA network it will start trying to find "SaEeDGroup" in NetPeerGroup. After couple of tries and searching the NetPeerGroup, if "SaEeDGroup" not found, peer will assume that it the owner of group and will attempt to create SaEeDGroup then the advertisement of group existence will be sent to other peers, so later when peers searching for "SaEeDGroup" they will be able to find in Global group. Afterward, either SaEeDGroup found in group discovery progress or creation of group, peer will attempt to log into SaEeDGroup. So, then peer can advertise itself and its contents that can be shared with other peers in SaEeDGroup. The following code is essential part of this progress:

```
//This is Group UUID so it can be used as Group identifier when
application attempts to crate group
private final String stringID = "jxta:uuid-
4E0742B0E54F4D0ABAC6809BB82A311E02";
//Searching local cache for any information about SaEeDGroup
adv =
myDiscoveryService.getLocalAdvertisements(DiscoveryService.GROUP, "Name", "SaEeDGroup");
//Searching remote peers for any existence of SaEeDGroup
myDiscoveryService.getRemoteAdvertisements(null, DiscoveryService.GROUP, "Name", "SaEeDGroup", 1);
//calling functions for group creation
SaEeDGroup = createGroup();
//joining group either way
joinToGroup(SaEeDGroup);

//Essential part of code needed to Create SaEeDGroup as subgroup in
NetPeerGroup
ModuleImplAdvertisement myMIA =
netPeerGroup.getAllPurposePeerGroupImplAdvertisement();
myNewGroup = netPeerGroup.newGroup(getGID(),
myMIA,
"SaEeDGroup",
"SaEeD P2P File Sharing Application");
```

The class diagram of this component is illustrated in Figure 23:



Figure 23: StartJXTA class

After StartJXTA class executed successfully, there is method in class that returns information about SaEeDGroup for other classes that might go to use it.

## 4.4 Chatting Classes

Once the peer has successfully joined into SaEeDGroup it will then start executing chatting component. Consequently, peers are able to transfer messages with one another. There are two classes used for this purpose:

- Chat Input class – responsible for incoming messages from other peers
- Chat Output class – responsible for sending messages to other peers

Both classes use XML file called: "saeedPipe.adv". The existence of this file in the current directory, from which the application is executing, is absolutely critical. This file contains information such as: Pipe UUID, Pipe Type and Pipe Name. By having this information, peers are able to find one another's pipe and transfer messages through it. Without having this file application, peers will not be able to execute chatting component successfully and the program will be terminated.

### *Chat Input class*

Once, this class called by peer (who joined into SaEeDGroup) it will try to obtain Pipe Service from SaEeDGroup in order so as to create input pipe. Then this class will use information available in "saeedPipe.adv" file to create a new input pipe and advertise itself to other peers available in the current group. Once, input pipe has been successfully created, the interface called "Pipe Event Handler" will be implemented to start listening for incoming messages and fire the event handler as soon as new messages are received from other peers. Transferred messages between input and output pipes are in XML format and contain elements such as: Sender's Peer Name, Sender's Peer UUID, Chat Content and Time of message creation. The important section of the source code of this class is given below:

```
//Obtaining Pipe Services from SaEeDGroup
myPeerID = SaEeDGroup.getPeerID().toString();
//Reading "saeedPipe.adv" file
FileInputStream is = new FileInputStream("saeedPipe.adv");
//Creating new Pipe using information from advertisement file
pipeAdv = (PipeAdvertisement)AdvertisementFactory.newAdvertisement(
    MimeMediaType.XMLUTF8,is);

//Starting the Pipe Event Handler interface
public void pipeMsgEvent(PipeMsgEvent ev)
{
    log.append("[+]Message Received...\n");
    Message myMessage = null;
    try{
        myMessage = ev.getMessage();
        if(myMessage == null){
            return;
        }
    }catch(Exception e){
        System.out.println("[-]Exception!");
        e.printStackTrace();
    }
    //XML file that received, and its elements
    MessageElement me = myMessage.getMessageElement("peerName");
    MessageElement me2 = myMessage.getMessageElement("peerID");
    MessageElement me3 = myMessage.getMessageElement("chatMessage");
    MessageElement me4 = myMessage.getMessageElement("Time");
}
```

In addition, class view of the chat input class is shown in Figure 24:

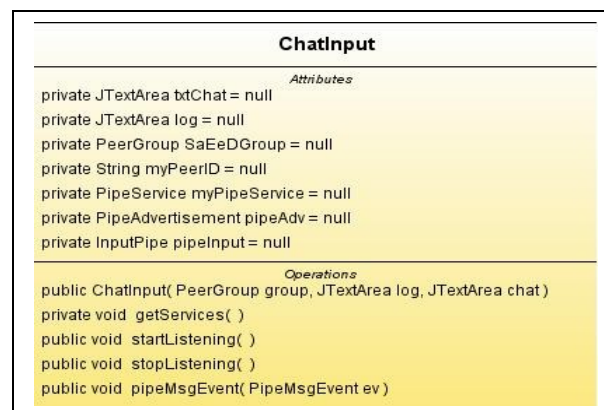


Figure 24: ChatInput Class

### Chat Output Class

This class uses the same procedure as chat input class, except it creates an output pipe rather than an input pipe. After it has created the output pipe, it will then add a XML message to it with elements such as: Peer ID, Peer Name, Message Content and Time of message created. Finally, it will send this XML message through output pipe to other peers in the current JXTA group. The overview of class structure can be viewed in Figure 25.

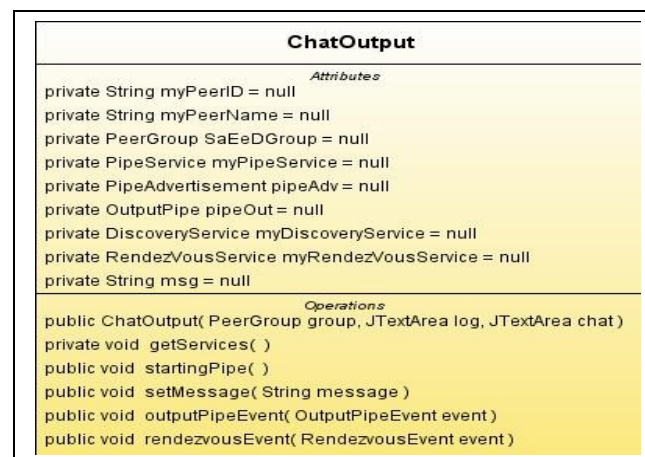


Figure 25: ChatOutput Class

## 4.5 Peer Listing Class

The purpose of the Peer Listing class is to list the available peers' name in SaEeDGroup. In order to do so, this class first obtains Discovery Services from current group, then it will create object from interface in JXTA library called "Discovery Event Handler". This object is responsible for updating the peers list as soon as a peer's advertisement is caught from peers in SaEeDGroup. The following code is presents how this is done. This is followed by the class diagram in Figure 26.



```
//obtaining Discovery Services from SaEeDGroup
myDiscoveryService = SaEeDGroup.getDiscoveryService();
//Creating listener object from JXTA library
public void discoveryEvent(DiscoveryEvent event)
{
    DiscoveryResponseMsg res = event.getResponse();
    String name = "unknown";
    boolean isInList = false;
    PeerAdvertisement peerAdv = res.getPeerAdvertisement();
    if(peerAdv != null){
        name = peerAdv.getName();
    }
    PeerAdvertisement myAdv = null;
    Enumeration en = res.getAdvertisements();
    Vector peerList = new Vector();
    while(en.hasMoreElements()){
        myAdv = (PeerAdvertisement) en.nextElement();
        peerList.addElement(myAdv.getName());
    }
    //updating the List of peers name
    updatePeerList(peerList);
}
```

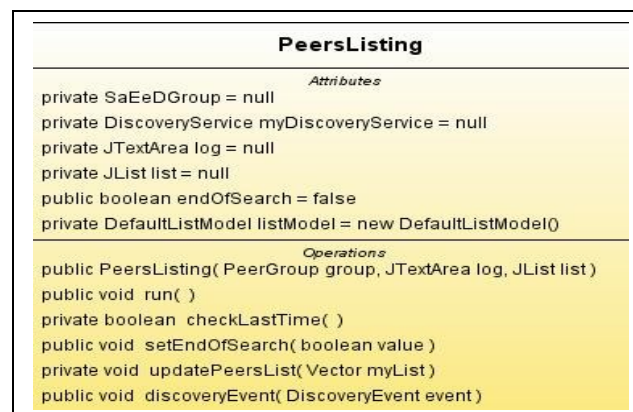


Figure 26: PeersListing Class

## 4.6 SaEeDSharing Class

This class will share all contents (i.e. Contents are referred to Files in here) of each peer with the other peers in the SaEeDGroup. In order to do so, Content Management Service library also known as CMS has been used. CMS is an extension library of JXTA which is made for file sharing purposes. This class will first create an object from CMS library, then CMS object needs to be initialised so, it can bind its services to the current group (SaEeDGroup in this case). Then this class will start to search for files in the shared directory – once the files are collected in the share directory then CMS object will create a unique advertisement for each file and make them available for other peers. Moreover, if any of the peers in the current group requests to download the shared file, then CMS will handle that request and will send the file to the other endpoint peer. There is also another class used here called "ChecksumCalc" - responsible to generate checksum for files that being shared within other peers in the

group. Therefore when other peers are downloading files, they regenerate checksum of file using the same algorithm (CRC-32 algorithm used to generate 32-bit hexadecimal value of file) to double check that download was successful and file was not corrupted. The following code shows the usage of CMS in order to share files:

```
//Creating CMS object
cms = new CMS();
//Initialising CMS object
cms.init(SaEeDGroup,null,null);//binding CMS object to SaEeD Group
if(cms.startApp(myPath) == -1){
    System.exit(-1);
}

//using Content Manager in CMS object for file sharing
ContentManager contentManager = null;
contentManager = cms.getContentManager();
//Sharing files in current JXTA network
if(list[i].isFile())
{
    //Sharing Files and check sums in network
    contentManager.share(list[i],checkSum.getFileSum(list[i]));
}
```

The structure class view of this class is presented in Figure 27:

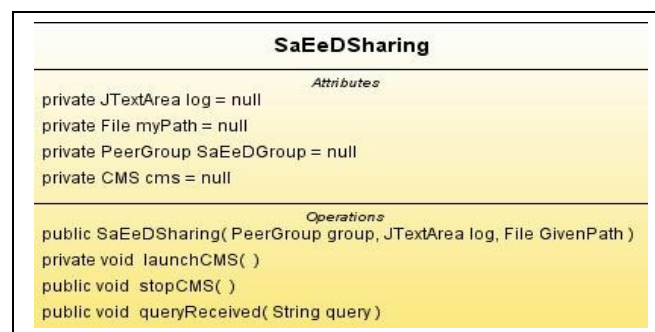


Figure 27: SaEeDSharing Class

## 4.7 SearchFile Class

The goal of this class is to search for shared contents from other peers in the current JXTA group (SaEeDGroup). Once this class has been executed by the user, it will start searching for specific a filename advertisement in SaEeDGroup until user interrupts the searching progress. When advertisements of file are found in group, basic information needed for downloading progress will be obtained form advertisement. Consequently, user can download shared files with basic knowledge obtained from advertisement of files in SaEeDGroup. The file advertisement contains information such as: Owner Peer, MD5 id of file, File Size and JXTA address of file. By gaining this information, then the users can download the file with other class - explained later in this chapter. The following code shows a small part of the codes that have been used to create this class:

```
//This class uses CachedListContentRequest as super class
class ListRequestor extends CachedListContentRequest
{
//initialising super class with SaEeDGroup and Name of file
//we are looking for.
public ListRequestor(PeerGroup SaEeDGroup , String SubStr){
    super(SaEeDGroup,SubStr);
}

//notify the application if file advertisements found
//in SaEeDGroup
public void notifyMoreResults()
{
    log.append("[+]Searching for More Contents.\n");
    searchResult = getResults();

for(int i=0; i < searchResult.length;i++){
    log.append("[*]Found: " + searchResult[i].getName()+"\n" +
        "Size: " + searchResult[i].getLength() + " Bytes\n");
    }
}
}
```

The class view of this class as well as its dependencies is demonstrated in Figure 28.



Figure 28: SearchFile Class

After this information is obtained from search class, the application will be ready to download the file from its provider.

## 4.8 DownloadFile Class

When file advertisements are returned from searching process, this class will handle the downloading request of file. This class is a subclass of "GetContentRequest" in CMS library. So, when this class is executed by the user, it will try to download the specified file with information of that file in current peer group. After file downloading has been completed, then user can regenerate checksum of downloaded file and compare it to the checksum of original file which has been obtained from file advertisement to make sure download was successful and file was not corrupted. The following code is a demonstration of downloading progress:

```
//this class is a subclass of GetContentRequest
class GetRemoteFile extends GetContentRequest
{
public GetRemoteFile(PeerGroup group, ContentAdvertisement contentAdv, File
destination)
{
//Calling the super class and initialise it
super(group, contentAdv, destination);
this.progressBar = progress;}
//inform the user about current download progress
public void notifyUpdate(int percentage)
{progressBar.setValue(percentage); }
//inform user when download has been finished
public void notifyDone()
{
log.append("[+]Donwloading Process is sucessfully finished.\n");}
```

In addition, class view of class structure and its dependencies is given in Figure 29.

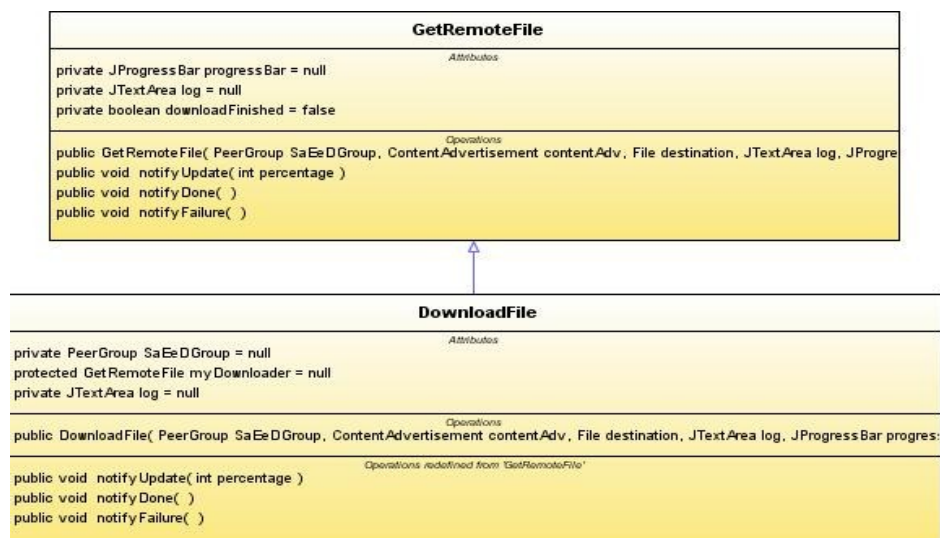


Figure 29: DownloadFile Class

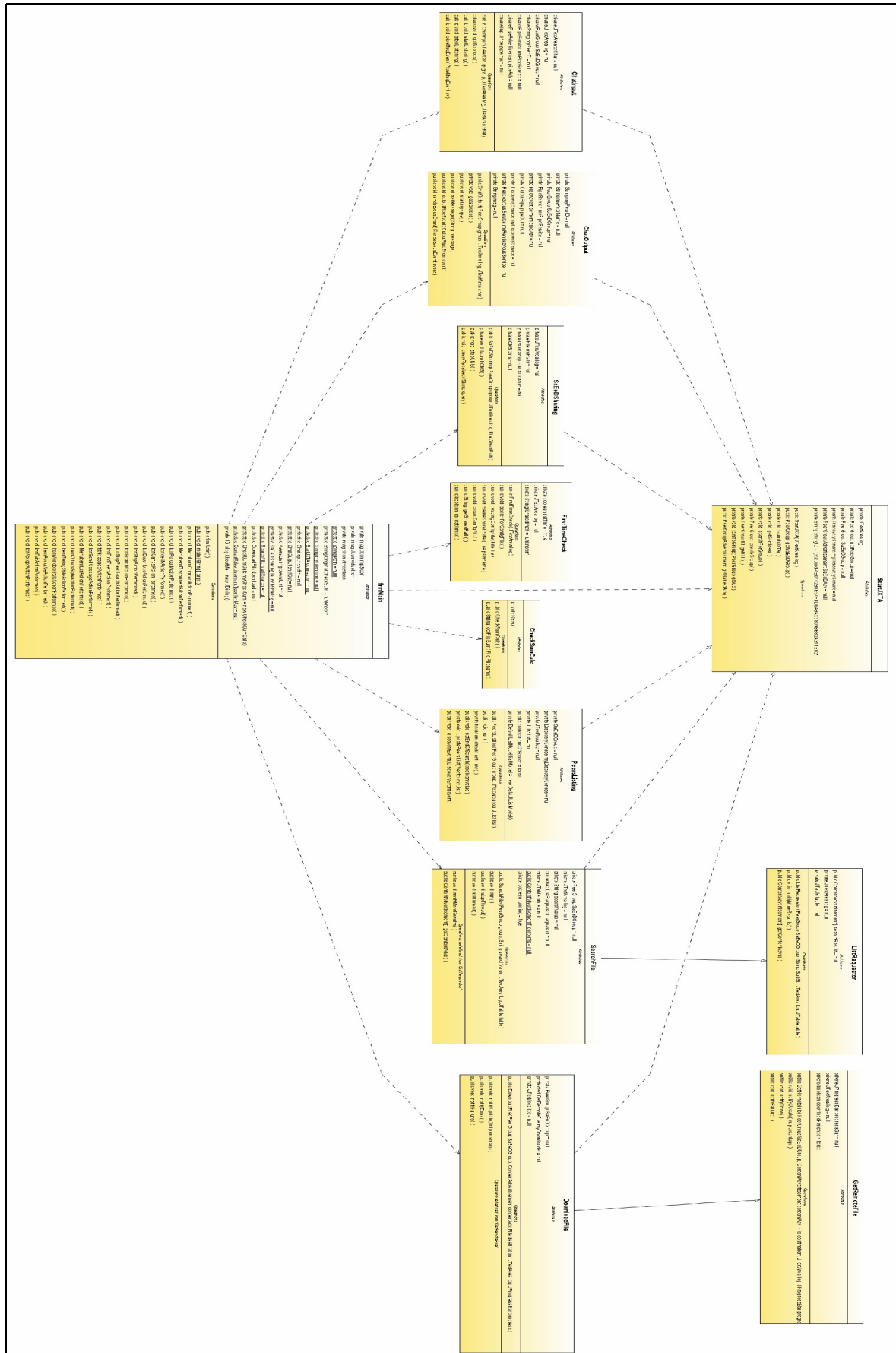
## 4.9 FrmMain Class

The FrmMain is the main class of the whole application responsible for controlling the execution of all the other classes included for file sharing application. This class is made of a variety of standard Java components such as: JButton, JTable, JList and many more. In addition, implementation of this application is highly dependant on this class and it acts as the parent to all other classes. The class view of structure of this class is illustrated in Figure 30.

frmMain
<p><i>Attributes</i></p> <pre> private ImageIcon mp3Icon private ImageIcon videoIcon private ImageIcon othersIcon protected String Path = null protected String OriginalChecksum = "unknown" protected String myFilename = null protected StartXTA connection = null protected ChatInput chatIn = null protected ChatOutput chatOut = null protected PeersListing peersList = null protected SaEeDSharing launchSharing = null protected SearchFile startSearch = null protected DownloadFile download = null protected CheckSumCalc myChecksum = new CheckSumCalc() protected ContentAdvertisement[] contentAdv = null private JDialog AboutMe = new JDialog() </pre>
<p><i>Operations</i></p> <pre> public frmMain( ) public void main( String[] args ) public void MenuItemConnectActionPerformed( ) public void MenuItemDisconnectActionPerformed( ) public void btnReloadActionPerformed( ) public void btnVerifyActionPerformed( ) public void btnCancelActionPerformed( ) public void btnDownloadActionPerformed( ) public void btnStopActionPerformed( ) public void btnSearchActionPerformed( ) public void btnStopPeerSearchActionPerformed( ) public void btnFindPeersActionPerformed( ) public void btnSendMessageActionPerformed( ) public void MenuItemExitActionPerformed( ) public void ItemOthersStyleActionPerformed( ) public void ItemSwingStyleActionPerformed( ) public void ItemWindowStyleActionPerformed( ) public void ItemAboutMeActionPerformed( ) public void btnExitActionPerformed( ) public void btnAcceptActionPerformed( ) </pre>

Figure 30: frmMain Class

Figure 31 presents the class view of entire application as well as its dependencies. This is followed by the next section that will investigate the usage of this application.



## 4.10 Testing

This section will go through execution of "SaEeD-P2P" file sharing application and briefly explain each step, serving as the user manual of this application.

### Starting of SaEeD-P2P

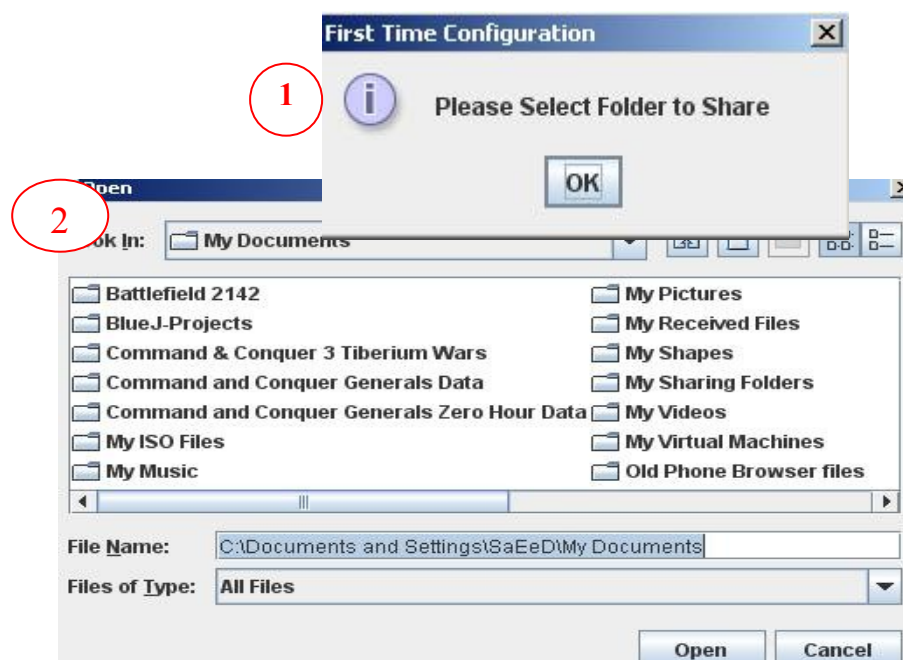
This application is written and compiled in java and there are two ways to execute it: either by double clicking on it or running the following command in Windows Command Prompt or Linux Console:

```
>java -jar SaEeD-P2P_File_Sharing.jar
```

**In order to run this application, JXTA libraries must be present in folder called "lib" in the same directory that application is executing from. As mentioned before the pipe advertisement file needs to be in current directory as well.**

After the application executed by user, the terms and conditions dialog box will pop-up initially, in order to make sure that user has accepted the application's conditions. Otherwise, the application will be terminated.

Then application will look for the configuration file called "config.ini" so as to determine whether it is the first time that it is being used by user. If it is first time that the application is being used or the configuration file doesn't exist in same directory, then the application will ask the user to choose the folder that they want to share their contents from it over the JXTA peer group. This configuration dialog demonstrated below:





If it is first time that the user is using the application, then the user needs to configure its peer as well so that it can operate within the JXTA network. The window which pops-up for JXTA configuration is presented in Figure 32.

This is Advance configuration for Peer. By this peer can be set to operate As Rely peer or Rendezvous Peer.

This is configuration used for peers who are behind NAT/Firewalled network. So they can communicate with outside world, and share their Contents with other peers outside the Network.

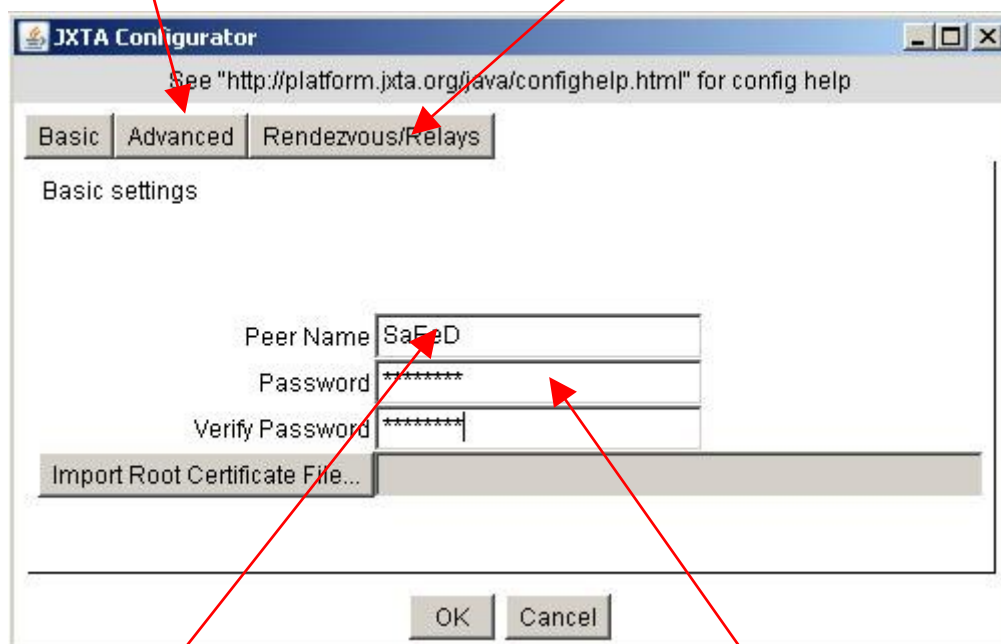


Figure 32: Jxta Configuration Window

In this field user can choose its peer name - this name will be visible to other peers in JXTA Network.

This field is for peers to choose password for themselves. The password Must be at least 8 characters.

Once the peer has successfully configured these steps, then the application will be launched into JXTA network and files from chosen shared directory will be shared with other peers in SaEeDGroup.



#### **4.11 System Requirements**

In order for the users to run SaEeD-P2P file sharing in their computers, they need to have some basic requirements - as listed below:

- A Computer which is capable of sharing and storing Data – PC, Laptop, PDA
- An Operating system which supports Java virtual machine such as: Windows, Solaris, Linux, Macintosh, Palm OS, Windows CE.
- Hardware for network communication
- Network connection – that can be internet or intranet connection and if neither is available, "LoopBack" connections can be used for testing purposes.
- Java Runtime Environment (JRE) version 1.6 or higher
- Files to share with other peers – Optional

#### **4.12 Conclusion**

This chapter described part of codes used in final P2P application as well as class diagrams of each class to represent functionality of this application. In addition a class diagram was established for the whole application with their dependencies. This chapter looked into the testing of its application in the real world. Furthermore, the basic system requirement of this application was also mentioned. The entire source code of this application and diagrams are available within the CD included in this project.

## 5.1 Introduction

Having examined the design and implementation of the final output in the previous chapter, this section aims to investigate and critically examine the completed work. The results achieved in the testing procedure of this application are presented. Relative achievement or lack of them to original objective will be explained as well.

## 5.2 Application Objectives

The aim of this project is to create a P2P file sharing application using JXTA platform. In order to do so, the following goals have been achieved throughout the development of this project:

- Study of the JXTA Platform – understanding how the JXTA platform can be used for developing file sharing applications and P2P computing.
- Creating friendly and easy to use graphical user interface for final output – similar to those of Napster or LimeWire.
- Testing application on different computers over the network such as Local Area Network. Because it is network-based application, better results would be obtained if it will be tested on different machines.
- Evaluating the final application – testing the application and its weakness to recreate more stable application
- Final output

### *Problems Encountered*

The main problem encountered throughout this project involved a lot of preliminary study and research to understand JXTA topology before carrying out the actual project. In general JXTA is a very broad and complicated subject and this project only touches the surface. Other problems include lack of literature and related examples available for study.

## 5.3 Test Results

This section deals with collecting results that have been taken from the testing procedure of the final file sharing application in different operating systems. These results are given in Table 1.

Operating System	JXTA Version	JRE version	Results
Windows XP Professional	2.4.1	1.6	Successful
Solaris 10	2.4.1	1.6	Successful
Linux Ubuntu 7.4	2.4.1	1.6	Successful
Linux Slackware 10	2.4.1	1.6	Successful
Windows XP Professional	2.4.1	1.5	Failed

Table 1: Testing results in different Operating Systems

Table 2 presents the testing results that have been taken from testing the speed of file transfer between peers. This test is based on 2Mbit ADSL internet connection and the average download speed is: 228.7 KB/Sec.

Size (Bytes)	Transfer Speed (KiloBytes/Second)	Duration (Seconds)
1,024	256	0.004
4,987	250.6	0.019
156,871	248.9	1.15
351,002,413	200.5	608.49
704,132,015	10240	240.24

**Table 2: Transfer Speed Results**

## **5.4 Conclusion**

This chapter has pointed out the achieved objectives of this application. The test results for the final application are also presented. The next chapter will look into the future of this application and improvements that can be carried out to the final file sharing application.

## **6.1 Introduction**

This chapter will attempt to conclude this report by giving an overview of final output including the positive aspects of the project at hand. It will also explore the future of this application and improvements that can be carried out to final output.

## **6.2 Project Conclusion**

This project attempts to investigate P2P computing, in relation to File Sharing applications, to provide a detailed insight into P2P technology. The main aspect of this project has been to create a decentralised P2P file sharing application using available technologies such as: JXTA, JAVA and XML. SaEeD-P2P was achieved in this task and more. SaEeD-P2P is a completely decentralised P2P application and used no indexing servers or network routes. Thus means that if there are more users, then the application will be further expanded to suit the users' needs. SaEeD-P2P is a user friendly application designed for users who do not have any related knowledge in P2P fields. All users can only set their directory to share and the application will do the job for them, and files will be shared in the network with the other peers in a decartelised manner. Furthermore, as long as one peer remains in the peer group, the group will be visible to others. This application is also used to overcome networking problems that may have been encountered by P2P user. Network problems such as: NAT connections, Firewalls and different networks topologies.

The only problem was encountered during development of this project was lack of documentation for JXTA technology. JXTA technology is an excellent platform for developing P2P applications especially for creating file sharing applications – however the lack of documentation was problematic in the development of this application.

## **6.3 Future Work**

This section will examine into the future work that could be used to improve the application.

### ***GUI***

In this area, the file monitoring of the application can be improved, in order to monitor uploading files, moreover improve downloading monitoring, so that the user can download and monitor multiple files simultaneously.

### ***Transfer Speed***

To improve the speed of downloads; users should be able to download the same file from different sources. This can be done by using additional libraries from JXTA platform and hopefully will be considered in the next release of the SaEeD-P2P file sharing application.

## 7.1 Appendix A – Terms of Reference

### Project Title

Peer-to-Peer File sharing application

### Project Background

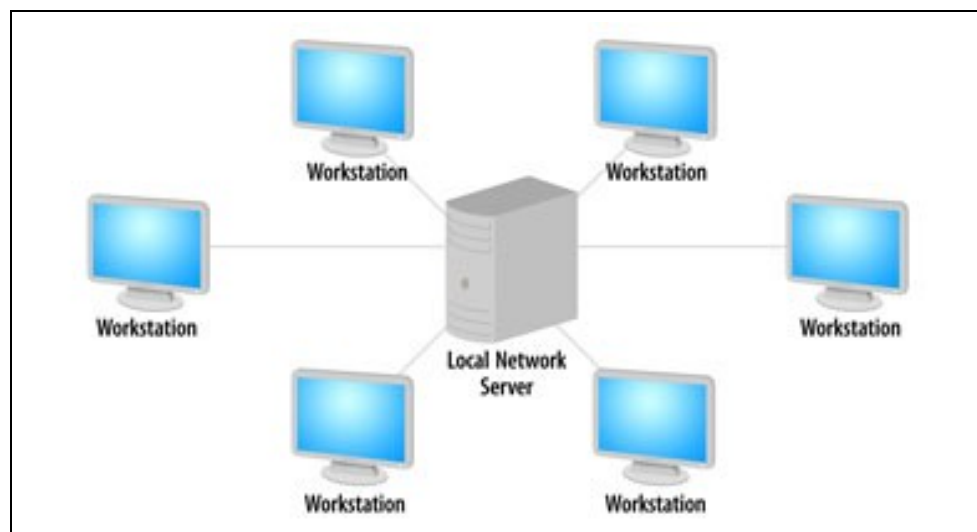
I am currently a final year student at Manchester Metropolitan University studying a BSc Honours degree in Applied Computing. For my final year project I have decided to develop a peer to peer application for file sharing

### Computer Networks

#### *Client-Server Architect*

Client-Server architecture is a well-known network architecture in which the application processing is divided between client workstations and servers. In order to provide service for the clients, it is required that all clients' computers connect to the server independently. Figure 33 gives a demonstration of the traditional client-server architect and as it can be seen - there are some disadvantages:

- Lack of bandwidth
- The whole network goes down if the server or service provider goes down.

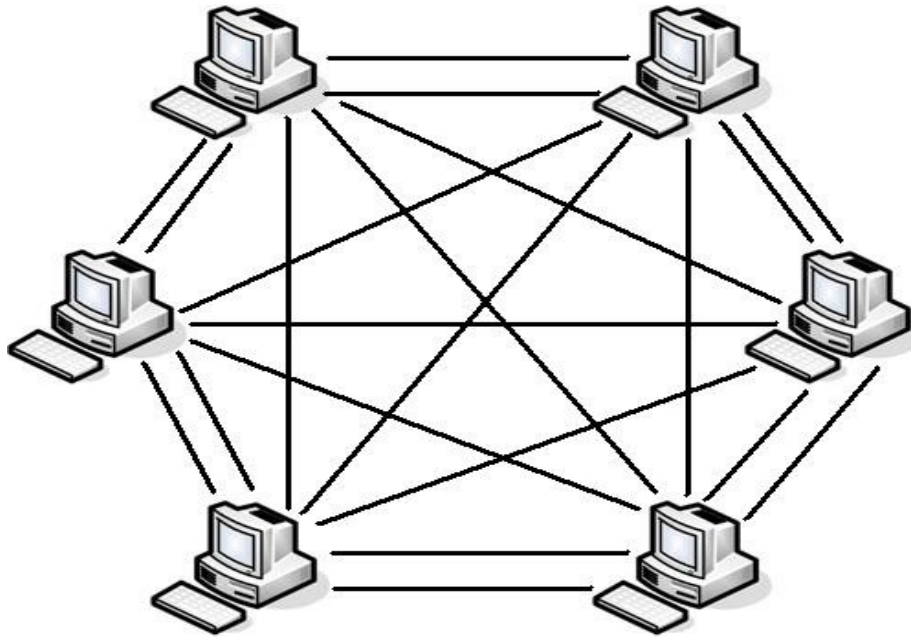


**Figure 33: Client Server**

Available from: <http://www.marketwebsolutions.com/images/client-server.jpg> (01/10/07)

#### *Peer-to-Peer Architect*

Another type of network architecture is known as peer-to-peer which is in wide usage today. In this architect, each computer in the network is known as a peer and can simultaneously act as both a client and a server. Figure 34 illustrates a Peer-to-Peer architect.



**Figure 34:P2P Architect**

I have chosen to research on this topic for the following reasons:

- Considerable knowledge in Java programming
- Interest in network based applications
- Interest in Peer-to-Peer architect
- Interest in JXA Protocol

### **Aims**

This project includes the development of a peer-to-peer file sharing application based on JXTA protocol. It aims to give users ability to share their files over their own P2P network; however it is not required to write a peer-to-peer protocol from scratch as they are available to use in the JXTA libraries.

### **Objectives**

In order to achieve the project goals it is required to complete a series of tasks:

1. Research and learn about JXTA protocol. Resources such Mastering JXTA book can be used.
2. Create suitable and user friendly interface for an application; similar to Napster or KaZaa.
3. Create an application in GUI.
4. Test the application on different computers over the network such as my home Local Area Network. Since it is a network based application and so it has to be tested in different machines on the network.
5. Evaluate: both centralize and decentralize methods should be tested.
6. Prepare and complete final report

### **Project Deliverables**

1. Project report
2. Interim Report
3. P2P file sharing application with source code

### **Required Resources**

Hardware:

- Two or more laptops or computers since it is network based application

Software:

- JXTA Libraries (2.4.1 >=)
- JRE 1.6
- NetBeans 5.5
- JXTA shell (optional) – can be used for testing purposes
- File to share with other peers over network

7.2 Appendix B: Project Plan

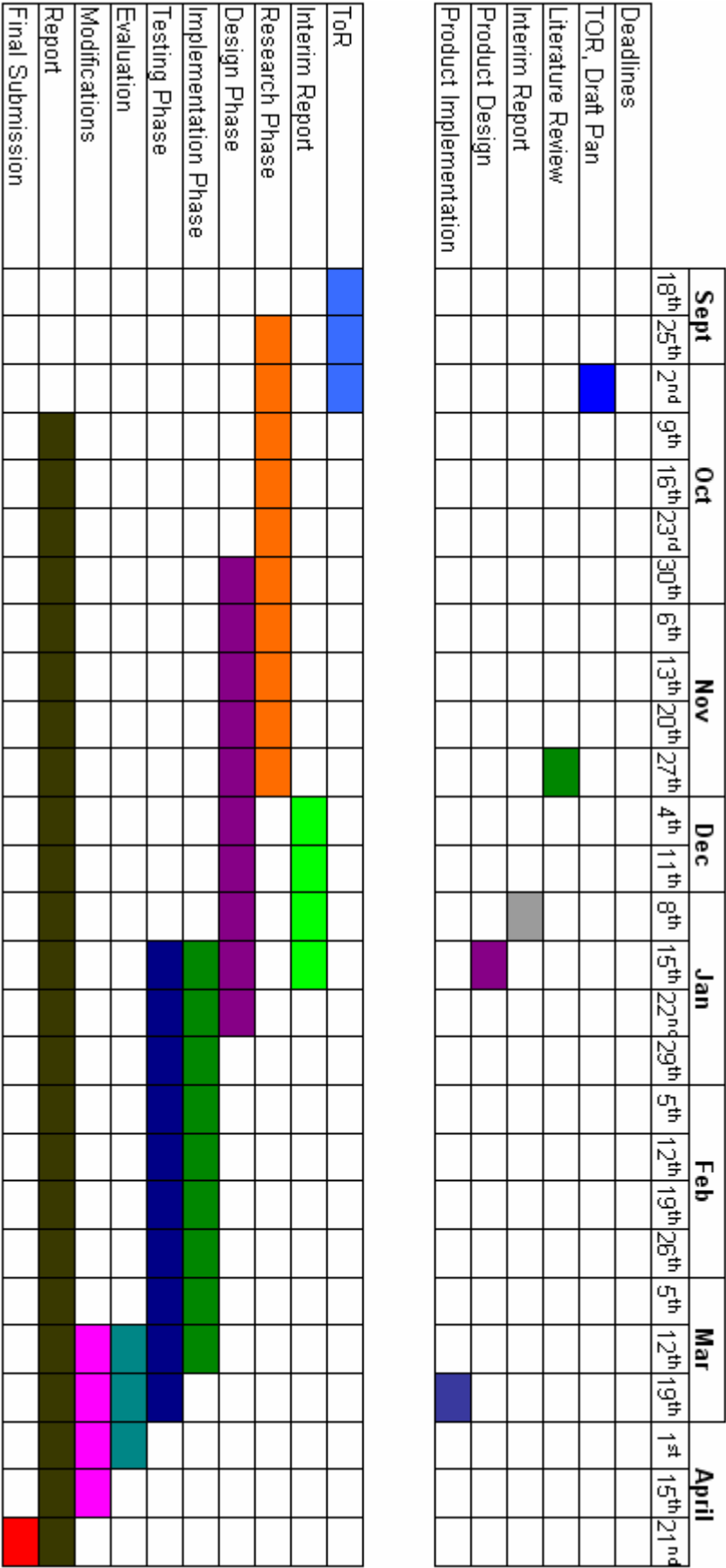


Figure 35: Project Plan



### 7.3 Appendix C: Ethics List

#### Ethics Check Form: Notes for Guidance

Before completing the Ethics Check Form the person undertaking the activity should consider the following questions:

		YES	NO	N/A
1.	Is the size of sample proposed for any group enquiry larger than justifiably necessary?			
2	Will any lines of enquiry cause undue distress or be impertinent?			
3	Has any relationship between the researcher(s) and the participant(s), other than that required by the academic activity, been declared?			
4	Have the participants been made fully aware of the true nature and purpose of the study?			
	If NO is there satisfactory justification (such as the likelihood of the end results being affected) for withholding such information? (Details to be provided to the person approving the proposal).			
5	Have the participants given their explicit consent?			
	If NO is there satisfactory justification for not obtaining consent? (Details to be provided to the person approving the proposal).			
6	Have the participants been informed at the outset that they can withdraw themselves and their data from the academic activity at any time?			
7	Are due processes in place to ensure that the rights of those participants who may be unable to assess the implications of the proposed work are safeguarded?			
8	Have any risks to the researcher(s), the participant(s) or the University been assessed?			
	If YES to any of the above is the risk outweighed by the value of the academic activity?			
9	If any academic activity is concerned with studies on activities which themselves raise questions of legality is there a persuasive rationale which demonstrates to the satisfaction of the University that:			
	i the risk to the University in terms of external (and internal) perceptions of the worthiness of the work has been assessed and is deemed acceptable;			
	ii arrangements are in place which safeguard the interests of the researcher(s) being supervised in pursuit of the academic activity objectives;			
	iii special arrangements have been made for the security of related documentation and artefacts.			
	<i>Appropriate expert advice should be sought as appropriate</i>			
10	Have the ethical principles and guidelines of any external bodies associated with the academic activity been considered?			



## Ethics Check Form

1. Names(s) of applicant: **Sayed Saeed Ghiassy**
2. Department: **Computing and Mathematics**
3. Name of Supervisor: **Dr.Georgios Fakas**
4. Title of Project: **GF.3 Peer-to-Peer File Sharing Application**
5. Resume of ethical issue: **N/A**
6. Does the project require the approval of any external Agency? **NO**

If YES has approval been granted by the external agency? **N/A**

### 7. Statement by applicant

I confirm that to the best of my knowledge I have made known all relevant information and I undertake to inform my supervisor of any such information which subsequently becomes available whether before or after the research has begun

Signature of Applicant: \_\_\_\_\_ Date: \_\_\_\_\_

### 8. Statement by Supervisor/Line Manager (*please sign the relevant statement*)

#### **Approval for the above named proposal is granted**

I confirm that there are no ethical issues requiring further consideration.  
(Any subsequent changes to the nature of the project will require a review of the ethical considerations):

Signature of Supervisor: \_\_\_\_\_ Date: \_\_\_\_\_

#### **Approval for the above named proposal is not granted**

I confirm that there are ethical issues requiring further consideration and will refer the project proposal to the appropriate Committee\*\*

Signature of Supervisor: \_\_\_\_\_ Date: \_\_\_\_\_

- \*\* For work forming part of an MMU taught programme– refer to Faculty Academic Standards Committee
- \*\* For work forming part of an MMU research programme – refer to Faculty Research Degree Committee
- \*\* For PhD by published work – refer to Research Degree Committee
- \*\* For any other work – refer to appropriate Faculty/Department Committee or line manager.

## 8.1 References

1. [Gradecki, Joseph D. Gradecki, Mastering JXTA, 13<sup>th</sup> /Septembr/2002]
2. [JXTA Java Standard Edition: Programming Guide, 10<sup>th</sup>/September/2007], Available: <https://jxta-guide.dev.java.net/> , Accessed: 17<sup>th</sup>/October/2007
3. [JXTA on PDA, 3<sup>rd</sup>/March/2008], Available: [http://jxta.free.fr/JXTA-SHELL/JXTA\\_Shell.html](http://jxta.free.fr/JXTA-SHELL/JXTA_Shell.html), Accessed: 4<sup>th</sup>/April/2008
4. [Sun Microsystems, What is JXTA, 2002] , Available: <http://www.sun.com/software/jxta/> , Accessed: 4<sup>th</sup>/April/2008
5. [Media Wiley, Peer to Peer architecture, 2004], Available: [http://media.wiley.com/product\\_data/excerpt/98/04714636/0471463698.pdf](http://media.wiley.com/product_data/excerpt/98/04714636/0471463698.pdf), Accessed: 20<sup>th</sup>/March/2008

## 8.2 Bibliography

1. JXTA, Available: <https://jxta.dev.java.net>
2. JXTA platform, Available: <http://www.jxta.org>
3. JXTA Protocols, Available: <https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html#id283731>
4. JXTA, Available: <http://www.sun.com/software/jxta/>
5. JXTA, Available: <http://www.onjava.com/pub/a/onjava/2001/04/25/jxta.html>
6. JXTA technology, Available: <http://en.wikipedia.org/wiki/JXTA>
7. Scott Oaks , Bernard Traversat, Li Gong. JXTA in a Nutshell
8. J. Wilson, Brendon. JXTA
9. Daniel Brookshier, Darren Govoni , Navaneeth Krishnan, Juan Carlos Soto. JXTA Java P2P Programming.