# An Investigation into Expanding the Capabilities of Peer-To-Peer Networks: Combining Centralized Network Infrastructure into Decentralized Peer-to-Peer network topology

by

Rui Zhang

BSc Computer Science with Distributed Mobile Systems,
Cardiff University, 2008

THESIS SUBMITTED IN ACCORDANCE WITH
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY

in the
Faculty of Computing, Engineering and Science
University of South Wales

December 2014

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

# Abstract

The Session Initiation Protocol (SIP) has been adopted by the 3rd Generation Partnership Project (3GPP [30]) as the standard signalling for session management over wireless networks. It's the basis of a wide range of IP multimedia services. SIP relies on centralized servers according to its standard specifications and current usage. However, server based approaches are typically non-scalable as they may involve a large amount of capital investment and costly maintenance, also administrative efforts would be required to maintain the network infrastructure. This would be suitable for large commercial firm but not practical for small organizations, especially in the home environment. Therefore research has started on the integration of Peer-to-Peer (P2P) principles to SIP to take advantage of decentralization in last decade.

Peer-to-Peer network is based on decentralized and distributed network architecture; In contrast to the centralized client-server network, each individual connected node (or called peer) on peer-to-peer network, is both resource consumer and supplier in same time. In peer-to-peer networks, tasks like resource publishing, sharing and discovering are shared amongst multiple interconnected peers, who voluntarily make their own resources available to other connected network participants. The Peer-to-Peer network has the ability of self-organizing and great expandability; therefore peer-to-peer network is great platform for small organization and home environment.

This thesis investigates the possibility of integrating SIP into a JXTA-based P2P infrastructure, with the aim to contribute to research in decentralizing the SIP protocol. This proposed theory is to directly integrate SIP into P2P overlay with self-organized proxies and distributed registrars. Unlike other proposed approaches of P2P-SIP architectures to date, this proposed theory does not extend SIP message header and is middleware independent. Also the self-organizing mechanisms are initiated amongst each participant voluntarily, not passively. Therefore the proposed theory in this report would theoretically be able to interoperate with legacy SIP and/or JXTA platforms to ensure the

compatibility and intercommunication ability, and most importantly is to maintain SIP registrar within the network constantly, also actively self-organizing to optimize the network topology from time to time.

# Acknowledgements

I would like to convey my gratitude to Professor Khalid Al-Begain for his valuable guidance, his critical remarks and constructive contribution to this thesis.

# Table of Contents

# Table of Figures

## Chapter 1

# 1. Introduction

## 1.1 Background and Motivation

The unprecedented increase in the number of smart phones has created a new era in mobile and wireless communications. These new generations of smart phones offer a much wider range of services beyond the basic voice and data services. In addition, wireless networks are providing higher and higher bit rates allowing more exciting applications. One area that has attracted significant interest among researchers and professionals is the interoperability and collaboration between devices. This is a new trend that allows devices to share resources and content through efficient and attractive applications. So far, mobile interconnection and collaboration applications are mainly based on a client-server model. Server based approaches are typically non-scalable as they may involve significant amount of capital investment and costly maintenance. The Session Initiation Protocol (SIP [1], [7]), a typical client-server based protocol, has been adopted by the 3rd Generation Partnership Project (3GPP [30]) as the standard signalling for session management over wireless networks. Also SIP is an Internet Engineering Task Force (IETF) protocol for session management in general, which is based on client-server infrastructure. The End user (User Agent) is acting as the client which sits on the edge of the network, while the Registrar service stores client's presence and contact information. The Proxy server is responsible for forwarding the message between the clients. Therefore the traditional client-server infrastructure requires not just capital investment and costly maintenance on a server, but also a set of administrative efforts and has a potential single point of failure to cause the whole network to fall, e.g. the SIP Proxy server or Registrar in this case.

An alternative approach for collaboration and session establishment among devices is called Peer-to-Peer communications (P2P) which has been widely used in wired networks. The characteristics of P2P networks can overcome many issues in traditional client-server networks. Currently, many different approaches for P2P networks exist, such as Gnutella [6], Bittorrent [20], Juxtapose (JXTA [2]) etc. The reason the author considers JXTA to be one of the most appropriate P2P protocols for the purpose of decentralizing SIP, is because the JXTA protocol is a programming language independent and has multiple implementations for different environments, which allows it to be used on various network devices like sensors, mobile phones, PDAs, laptops, workstations, servers and supercomputers, to communicate and collaborate with each other. The self-organizing characteristics of JXTA networks are appropriate solutions for self-organized proxies and distributed registrars for SIP networks. However JXTA doesn't have the ability to establish a session between a JXTA peer and traditional SIP client, and vice versa. The integrating of JXTA and SIP would make intercommunication and collaboration between traditional JXTA and SIP networks to become possible.

## 1.2 Project Summary and Objectives

In this thesis, the concept of a JXTA and SIP protocol integrated communication platform has been theoretically proposed to overcome the current issues of collaboration between mobile devices and computers in a decentralized peer-to-peer wired/wireless network. There is also a review of related work and a comparison to this proposal in theory. Both JXTA and SIP are text based protocols, which have the characteristic of being programming language and platform independent. This proposed network infrastructure adopts JXTA as a P2P network overlay and implements the BitTorrent concept to takes advantage of the efficiency of resource distribution in a P2P network environment. The proposed network infrastructure also has the advantage of being light-weight for session initialization by integrating SIP into the JXTA overlay. Thus the objectives of this project are as follow:

1. To perform a survey on traditional SIP and JXTA P2P networks, and study their implications on modern telecommunication networks

2. Survey the solutions for de-centralization of SIP.

3. To propose a solution to integrate SIP into P2P network environment at a theoretical level.

4. To simulate a P2P (JXTA implement BitTorrent) environment to test the possibility of integrating centralized SIP registrar (which effort is to keep registered SIP peer information) within the P2P network

5. To evaluate the efficiency of distributing the SIP registrar in the simulated decentralized P2P network.

The proposal of this thesis is constructed on a theoretical level. The effort of this study has focused on the method of integrating SIP into the decentralized P2P communication networks, by maintaining the SIP registrar in a decentralized environment. Also, to create a simulation network, in order to test the feasibility of decentralising the function of a centralized SIP registrar server and distributing it amongst a number of peers in a repeatedly changing P2P environment.

## 1.3 The Structure of the Report

This thesis is submitted in fulfilment of the requirements for the degree of MPhil in Computer Science at University of South Wales. In this respect it constitutes the written part of the MPhil thesis by presenting the theoretical work that has been done, also outlining the implementation work that has been undertaken which will examine the workability of the proposed theory.

The structure of this report is organized as follows: Chapter 2 presents a literature review on past related work and compares applied methods. Chapter 3 explains the background, characteristics, advantages and disadvantages of SIP and JXTA respectively; introduces the benefit of combining JXTA and SIP, and also introduces the author’s proposed approach. Chapter 4 talks about the

design of the proposed network infrastructure on a theoretical level. Chapter 5 outlines the implementation of a simulation network aiming at maintaining the registered SIP peer's information within a P2P environment. Chapter 6 contains the conclusions of this research and covers future work to be undertaken.

# Chapter 2

# 2. Literature Review

## 2.1 Basics of P2P and SIP Integration

The initial concept of the Internet is that most of the services and resources are located at some super-power computer called a Server. Every other peer who wishes to use published services and shared resources must connect to the Server and those peers are called Clients. This type of network topology is called the centralized client-server Model. The Session Initiation Protocol (SIP) [3] is one of the typical client-server model protocols for session establishment, where all clients has to register with registrar and all session establishment has to be through SIP proxy.

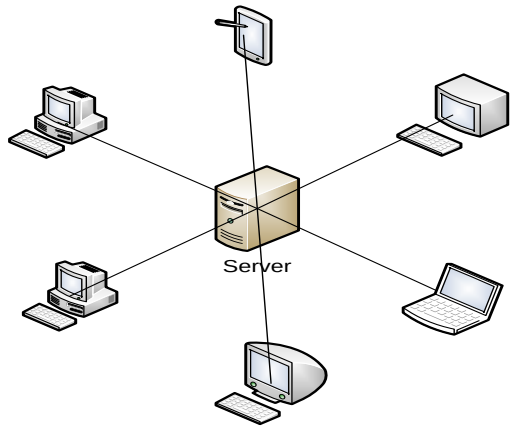



**Figure 1: Client-server network topology**

The main entities of conventional SIP networks are user agent, the Proxy, the Registrar and the redirect server [1] [7]. SIP Proxy, Registrar and Redirect Server are permanently centralized nodes. Therefore the SIP network relies on single point, as the whole network fails if the Server fails. However, to a certain extent, the SIP itself contains the principle of peer-to-peer communication characteristics. An established session will facilitate a direct communication

channel between the two SIP UAs; the connected SIP UAs can manage and terminate the session themselves. Therefore, to decentralise the SIP server into a P2P network which can take advantage of the network's self-organizing and scalability traits, would be a good method for solving a problem in a centralized SIP network, such as a single point failure.

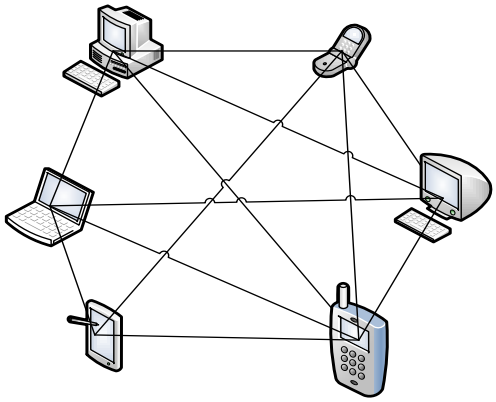


**Figure 2: Peer-to-Peer network topology**

Peer-to-Peer (P2P) [29] network is overlay network, which is built on top of physical network. P2P networks are normally without hierarchical organization or centralized control [8]. Unlike the client-server network topology, peer-to-peer (or "P2P") computer networks use diverse connectivity between participants in a network and the cumulative bandwidth of network participants rather than conventional centralized resources where a relatively low number of servers provide the core value to a service or application. In a peer-to-peer network, each participating peer may have roles as a client and also may be a server. Peers can be simultaneously acting as "Server" and "Client" at the same time.

## 2.2 Related Works

Combining P2P with SIP techniques has gained great interest [9]. The services delivery through the client-server model of networks involves a high cost for building network infrastructure and comes with costly maintenance on the server. However those issues can be solved by adopting a P2P approach. There are has been much work and research carried out on concepts, network topology, infrastructure and other outcome issues for integrating SIP into P2P

approach [10]. The administrative efforts for maintaining networks are needed in the conventional client-server mode of SIP network. Also client-server mode SIP networks have a potential single point of failure in the network causing the whole network to fail [1], e.g. when the server goes down. Skype [14] is a well known commercial company that has gained great success in supplying P2P-based VoIP services. They have adopted the network topology based on the Kazaa [25] architecture and trend, to use P2P techniques in VoIP to overcome these issues [12]. But Skype use their own proprietary protocol to manage the sessions instead of SIP and cannot inter-communicate with other VoIP platforms. There is also some other early work that has been carried out for combining SIP with P2P including the SOSIMPLE project at the College of William and Mary [13]. However, these systems mentioned above have some specific architecture and do not easily inter-communicate with each other.

Currently there are two possible ways of integrating the SIP to the P2P concept [15]. The first approach is P2P-SIP, by extending the SIP message and adding additional headers to the build to maintain the SIP overlay within a P2P environment. A Reserved SIP server is required to maintain the SIP peer in the P2P network. Bryan [17] has proposed a Chord-based P2P-SIP approach that follows the concept of this approach. Chord-based P2P-SIP extends the SIP message to perform the SIP contact information discovery method in the P2P overlay.

In the Chord-based P2P-SIP approach proposed by Bryan, the SIP peer first queries the Distributed Hash Table (DHT) [18] using P2P location mechanisms to select the known adjacent SIP peer, when they want to initiate a session with the destination peer in the network. Then an extended register message is sent to this known adjacent peer from the session initiator peer. The message then would go through one or more SIP peers to reach the destination peer. When the destination peer receives the message, it will respond with 200 OK messages but with an extended head containing the required contact information. The above described network architecture is based on a P2P middleware called Chord [19]. References [35], [36] also adopted this approach for integrating SIP into a P2P environment.

The second way to approach integrating SIP with a P2P network is to use the conventional SIP peer to register and retrieve contact information in the P2P overlay. In this approach P2P location services would responsible for SIP peer contact information registrar and provide discovery mechanisms. SIP peer would no longer require a permanently centralized node acting as SIP registrar, proxy and redirect server, the SIP peer would directly perform P2P publish and lookup instead.

The architecture proposed by Singh and Schulzrinne [21] is based on the use of the DHT interface to publish peer's presence and discovering other peers on the network. This approach extended P2P location services for SIP registration and discovery in P2P overlay. Peers registered with the network are using DHT-based messages to interact with DHT. This would allow the proposed architecture to work with other DHT based middleware. Jennings [22] provided another DHT based P2P-SIP architecture for SIP resources publishing and discovery in P2P overlay, which will have same issues as described above.

Holger Schmidt, Burcinput [23] has provided SIP integration into JXTA [2] overlay. In this approach they have kept a SIP server by defining a services location database, and used this database to store SIP peer's presence and services information. The network architecture is dependent on a services location database to overcome the issues of the SIP server has to be centralized.

Chuan Zhu, Guiran Chang, Wei Ning, have also proposed a JXTA based SIP communication platform for a P2P wireless network environment but with a very complex infrastructure [4]. This architecture uses a SIP domain for centralized SIP registrar, also there is no self-organization mechanism in their proposed network.

P. Houngue, R. Glitho, E. Damiani [24] has proposed a P2P based SIP network architecture by integrating the location services with the proxy. This architecture uses P2P mechanisms for SIP location services, integrating the location

services into the proxy and distributing the registrar. The peers have to predefine the role they are playing in the network (e.g. proxy, registrar) when they join the network. Therefore the proposed network architecture has a passive self-organization character. Also the proxy and registrar function hand over are active, but don't have mechanisms for passive hand over, e.g. if the proxy node or registrar node suddenly become offline, due to power failure.

Z. Wu, J. Zhou, W. Wang implemented a JXTA [2] based SofiaSIP [27] P2P-SIP network architecture [26]. This approach of network topology has skipped SIP proxy for the implementation of publishing and location services. JXP protocol [26] has been designed as a crucial part of this network to convert SIP operations into P2P operations. But in addition, some necessary code changes have to be made in the SIP user agent, in order to turn the standard SIP user agent into a peer client to work with this network approach. This would resolve the issues of inter-communication with conventional SIP networks.

Overall, the first way (P2P-SIP) of integrating SIP to P2P is by extending the SIP message header and adopting the Distributed Hash Table (DHT) for the routing mechanism in P2P based SIP networks; also a SIP server has been reserved in the network. The second way of SIP and P2P integration would be to integrate the conventional SIP registrar and proxy functions into a P2P overlay. Some of the approaches in this category require a reserved SIP proxy to register the SIP peer into the P2P network; others somehow require changes in the location mechanism of a P2P network in order to integrate SIP, or modifications on the standard SIP UA. Also, a limited passive self-organization function has been described in this approach.

## 2.3 The Proposed Network Architecture

In this proposed P2P SIP network architecture, the author has adopted the method of integrating the SIP proxy, registrar and redirect server into P2P layer. In contrast to the approaches mentioned above, the proposed network is different from the first category of integrating P2P mechanisms to SIP by reserving the centralised SIP server and extending the SIP message header;

e.g. similar to reference [17], a scenario which cannot smoothly inter-communicate with a conventional SIP platform, and is not completely decentralized. The proposed approach in this report belongs to the second category which aims at directly integrating SIP mechanisms into the P2P overlay, but unlike the work in references [21], [22], [23], it does not require changes of the P2P location mechanism or a services location database for storing and retrieving peer's contact information The inter-communication with the conventional SIP platform would be another issue to deal with, due to the modifications made on the standard SIP UA and the proprietary protocol being used.

Also, the proposed P2P-SIP network has the ability of self-organizing. The similar work would be in references [24], [26]. The proposed architecture in reference [24] only has passive self-organization function, which all peers have to predefine their role of responsibility while joining the network. This network architecture has a lack of fault tolerance when the peer who is acting as SIP proxy or registrar fails, due to SIP proxy or registrar function hand over mechanisms being voluntary. However the network architecture present in reference [26] seems to have overcome those issues, but still doesn't declare the fault tolerance when the peer who is acting as Locate and Publish rely for SIP peer-client fails.

In this proposal, the author has also adopted JXTA as the P2P platform. The author is proposing to integrate the SIP presence registrar function directly to the JXTA advertisement publishing mechanisms. This approach will not require extending the P2P location service; neither does it require a modification on the standard SIP UA. The SIP registrar information is stored amongst all capable peers existing in the P2P layer. Therefore, the proposed network can smoothly interoperate with conventional SIP and JXTA networks. The standard SIP UA would publish their presence in the network through super-peers, such as a JXTA overlay advertisement. All peers in the network would have a self-organization mechanism; peers can become super-peers based on the peers' status and their knowledge of the local network environment.

# Chapter 3

# 3. Related Technologies and Proposed Network Architecture

## 3.1 Related Technologies

### 3.1.1 Session Initiation Protocol

Session Initiation Protocol (SIP) [1] was originally designed by Henning Schulzrinne and Mark Handley in 1996. It was then chosen as the protocol for IP-based multimedia call control for 3G wireless networks by the Third Generation Partnership Project (3GPP) [30] in 2000. The latest version of SIP specification was published by IETF in June 2002.

SIP protocol provided the ability for users to be able to locate other UA on the network; send session invitation and establish a session in between each other regardless of session type. Also, SIP protocol allows users to manage the session and have the ability to terminate a session at any time. Hence, SIP is commonly used in multimedia communication sessions such as VoIP (voice over IP) or video calls, and provides a similar type of service to H.323 [16].

However, H.323 is designed for a telecommunication approach, and has a set of well defined protocol to take care of entire call setup, call control, and the media used during the call. On the other hand, SIP is a text based protocol, which is designed to work with other Internet protocols, such as HTTP. Therefore, SIP provides similar functions to H.323 but with much lower complexity, richer extensibility, and better scalability [40].

#### 3.1.1.1 SIP Components:

In SIP protocol, four main components are involved to form a session: SIP User Agent, SIP Registrar Servers, SIP Proxy Servers and SIP Redirect Servers [7].

- **SIP User Agent (UA)** can be implemented on any device, e.g. desktop computer, mobile phone, tablet or any other hand hold devices. A SIP UA can initiate a session with any other SIP UA, and manage the initiated session itself.

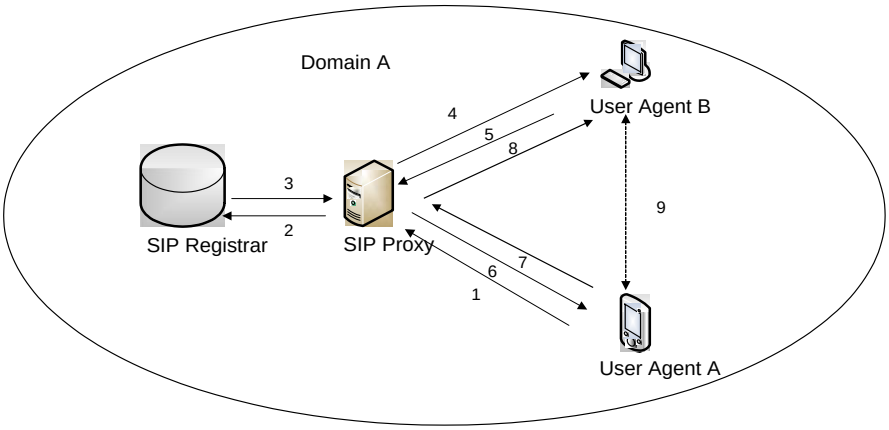
- **SIP Proxy Server** is one of the SIP servers acting as a relay when a SIP UA wants to initiate a session with another SIP UA. It has function of both SIP UA and Server, and has the responsibly, to ensure the session request is sent to the destination UA, or another proxy server who knows the location of the destination UA. The SIP Proxy Server can also be used to enforce a policy during the session initialization process.

- **SIP Registrar Server** registers a user's presence and stores the User Agent location into a location service within a domain. The SIP registrar server usually runs at the same location as the SIP Proxy Server.

- **SIP Redirect Server** redirects the SIP session invitation to domains other than the current hosting one. The SIP redirect server is also usually run at the same location as the SIP proxy server.

### 3.1.1.2 SIP Session Establishment

Establishing A SIP Session within the Same Domain [7]

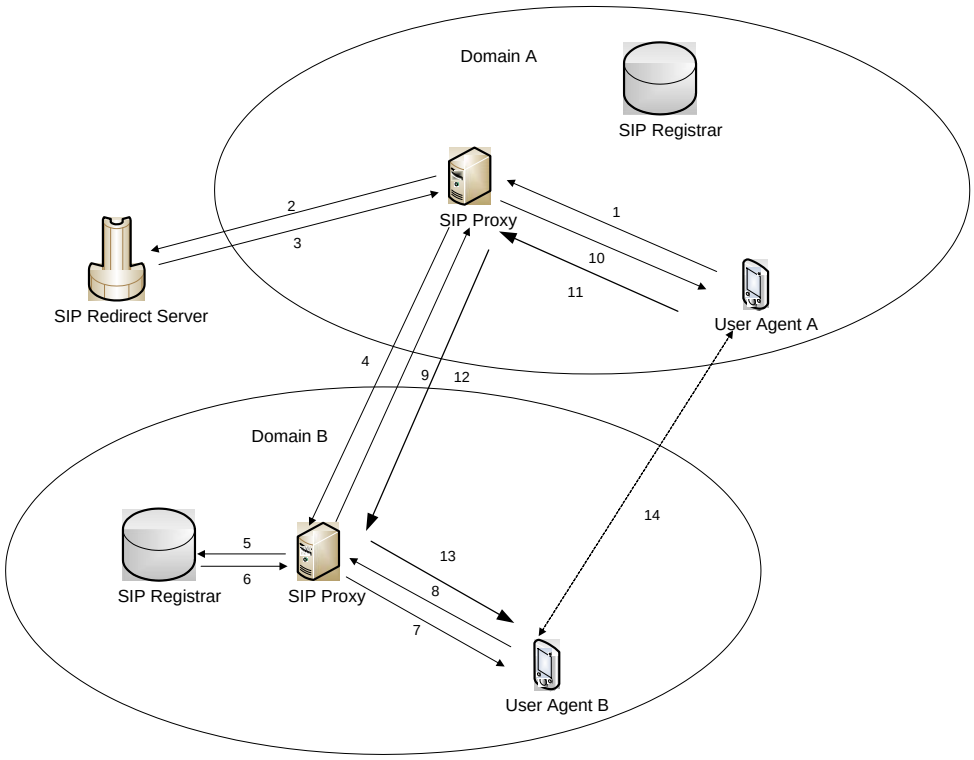



1. Invite B
2. Request address of B
3. Response with SIP address of B
4. Forward invitation to B from A
5. Response from B
6. Forward response to A from B
7. ACK send from A
8. ACK forward to B
9. Session Channel Established

**Figure 3: Establish SIP Session within same domain**

The diagram above illustrates the procedure of how to establish communication between user agent A and user agent B within the same domain. In this scenario, both A and B are registered within a same domain. To establish communication with B, A first will send an invite request to SIP proxy in Domain A; then this SIP proxy server will look up B's address from SIP registrar, and forward this invitation to B. Once B receives the invitation and decides to accept it, B will reply with "200 OK" message to A via SIP proxy. Upon receiving 200 OK from B, A replies with an ACK message sent back to B. Then a direct communication channel can be established between A and B, once A has received this "200 OK" message from B.

Establishing A SIP Session in Multiple Domains [7]




1. Invite B
2. Obtain Domain proxy address of B
3. Response with Domain proxy address of B
4. Forward Invitation to Domain proxy of B from A
5. Request SIP address of B
6. Response with SIP address of B
7. Forward invitation to B from A
8. Response from B
9. Forward response to Domain proxy server of A from B
10. Forward response to A from B
11. A Send ACK message
12. ACK message get forwards to B's Domain
13. B receive ACK message
14. Session Channel Established

**Figure 4: Establish SIP session in dissimilar domains**

In this scenario, user agent A and user agent B are registered to different domains. When A sends an invite request to B via SIP proxy in Domain A, this

SIP proxy recognizes that B is registered with a different domain; it then looks up the SIP proxy address of the domain where B is registered. Then SIP proxy in Domain A will forward the invite to SIP proxy in Domain B, who then will forward to user agent B. The “200 OK” message will reply back to user agent A via the same route if B has accepted the invite. Then A will send an ACK message back to B via the same route. A direct communication channel will be established once A has received the “200 OK” message from B.

## 3.2 JXTA

### 3.2.1 JXTA Architecture

JXTA [2] has defined a set of open XML protocols that allows any connected device on the network to discover and exchange the resources [5]. JXTA enabled devices ranging from smart phones to normal forms of computers. To start a JXTA platform, JXTA peers create a virtual network protocol and overlay on an actual network protocol. This would allow the peers to communicate with each other even when some peers are behind a firewall or Network Address Translation (NAT). The JXTA protocol enables peers to self-organize into peer-groups to communicate and collaborate, or supply services.

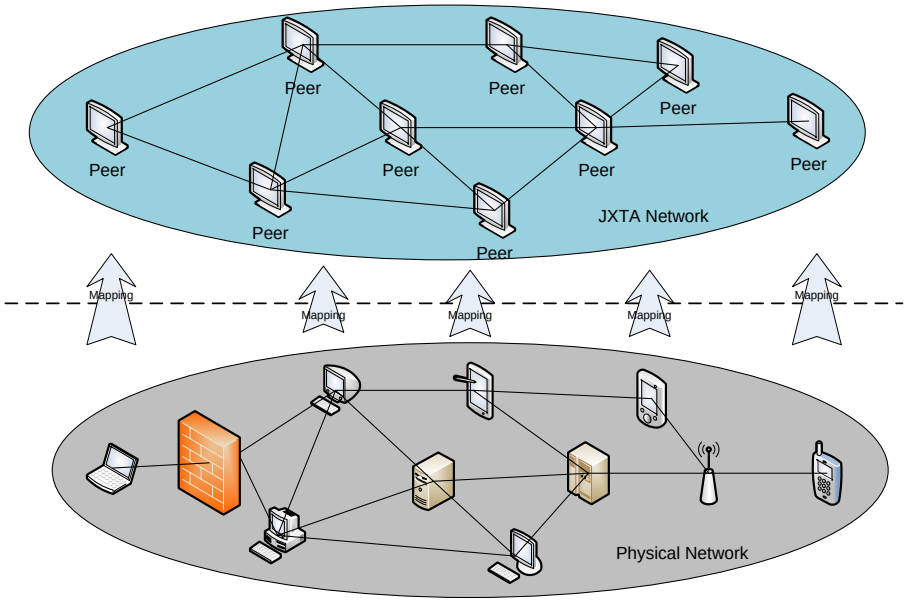



**Figure 5: JXTA Network**

The JXTA protocols standardize the way the peer behaves [2]:

- Discover Peers and resources
- Self-organize into peer groups
- Advertise resources
- Communicate with each other
- Monitor each other

The JXTA P2P platform provides all necessary functions to host all types of network services [2]:

- JXTA is defined by six protocols only, which are easy to implement and integrate into P2P services and applications. Thus, services offered by one vendor can be used transparently by other communities with a different vendor's system.

- The JXTA protocols are defined to be independent of programming languages, so they can be implemented in C/C++, Java, Perl, and many other languages. Devices running completely different operating systems can interoperate with each other using the JXTA protocols.

- The JXTA protocols are designed to be independent of transport protocols. They can be implemented on top of TCP/IP, HTTP, Bluetooth, Home PNA, and many other protocols.

### 3.2.2 The Six JXTA Protocols

There are six JXTA protocols that have been designed to be easily implemented on unidirectional links and asymmetric transports, which will allow the discovery (querying), organization, monitoring and communication between peers [2]:

- **Peer Resolver Protocol (PRP)** is the mechanism which allows peers to send a query to other peers or peer-groups, and receive responses to the query. Each query has a unique ID.

- **Peer Discovery Protocol (PDP)** is the mechanism which allows peers to advertise its own resources, and discover the resources from other peers (peer groups, services, pipes and additional peers). Every peer resource on a JXTA network is described and published using an advertisement which is represented as XML documents.

- **Peer Information Protocol (PIP)** is the mechanism which allows peers to obtain status information about other peers; including state, uptime, traffic load, capabilities, and other information.

- **Pipe Binding Protocol (PBP)** is the mechanism which allows peers to establish a virtual communication channel (JXTA pipe) between other peers. The PBP is used for the peer to bind with one or more pipe endpoints.

- **Endpoint Routing Protocol (ERP)** is the mechanism which allows peers to discover a route (sequence of hops) used to send a message to another peer. If one peer wishes to send a message to another peer and cannot establish a direct link to the recipient, then the sender peer needs to use ERP to find relay peer(s) whom will route the message to the recipient.

- **Rendezvous Protocol (RVP)** [2] is the mechanism which allows peers to subscribe to a propagation service. Peers can listen to rendezvous peers or be a rendezvous peer themselves. The RVP is used by the Peer Resolver Protocol and by the Pipe Binding Protocol in order to propagate messages.

The characteristics of JXTA are focused at the application level. A JXTA based P2P network can be applied when server or resource centralization is not

possible or desired, and the relationships between peers are transient or peers stay at the edge of the network (e.g. ad hoc network). The JXTA application can participate in actions among selected groups or individuals, and collaborate on projects from anywhere using any connected device. JXTA also enable computer services to be shared among all peers regardless of where the systems or the peers are located.

### 3.2.3 JXTA Layers

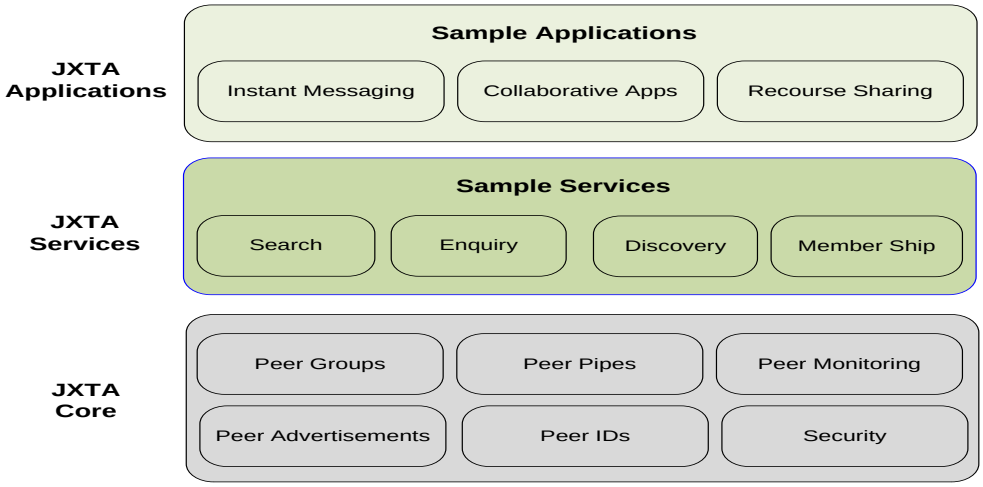



**Figure 6: JXTA Layers**

#### Applications Layer

The applications layer includes implementation for the applications used in a peer-to-peer network, such as resource sharing, entertainment content management and instant messaging systems.

#### Services Layer

The JXTA Services Layer is above the JXTA Core layer. The services layer includes network services that may not be necessary for a peer-to-peer network to be functional, but may be used in a peer-to-peer environment if needed. Those services include searching available resources, file sharing, resource propagation, protocol translation, authentication, and PKI (Public Key Infrastructure) services.

#### JXTA Core

The JXTA Core layer is at the bottom of the hierarchy and the function is for allowing the peer to have basic peer-to-peer networking functionality.

## 3.3 Comparison between JXTA and SIP

SIP services are based on the SIP server. The SIP servers include redirect server, proxy server and registrar. Peers register their presence with the registrar using SIP URL. Every SIP peer has a designated local SIP server which relays SIP requests to the SIP proxy serving the domain given by SIP URL. Redirection may take place if a recipient is roaming to a different domain and has left a forwarding SIP URL. However, to a certain extent, SIP itself has the principle of peer-to-peer communication characteristics. An established session will facilitate a direct communication channel between two SIP UA; the connected SIP UA can manage and terminate the session themselves.

SIP is based on the Server Centric architecture, which relies on a central SIP register server, whereas JXTA is a completely decentralized P2P system, consisting of peers and super peers. In JXTA, a peer can become a super peer based on their credentials and position in the network environment. The super peer can act as a rendezvous peer for delivering queries and advertisement, distributing resource lists, etc; or as relay peers for relaying the traffic between the peers that cannot directly connect to each other. Thus the JXTA peer located in a public network can connect to the devices which are located behind the NAT or firewalls.

In general, the SIP protocol is only for session initiation and management, but JXTA framework does much more. The Six JXTA protocols take care of endpoint routing, peer resolving, rendezvous management, peer discovery, information exchange and virtual communication channel binding (JXTA pipe). JXTA establishes a virtual network on top of an IP or non-IP network to hide the complexity of the underlying network protocols. Like SIP, every JXTA resource (e.g. peer, pipe, advisement, etc) has a unique ID. The JXTA pre-defined protocol has the ability to traverse the firewall and NAT, whereas SIP always relays on a proxy (e.g. STUN, TURN, ICE) placed on the edge of the network

between private (network behind firewall or NAT) and public networks. JXTA is more versatile, but is heavier than SIP. The table below shows the differences between JXTA and SIP.

| Main Feature | SIP | JXTA |
|---|---|---|
| Open Standard | Yes | Yes |
| Inherently P2P | No | Yes |
| Lightweight Protocol | Yes | No |
| Widely deployed | Yes | No |
| Self organizing | No | Yes |
| Relies on a central server | Yes (SIP Server) | No (Super Peer involved) |
| NAT and Firewall traverse | Poor | Good |

**Figure 7: Comparison of SIP, JXTA and JXME [4]**

SIP and JXTA are partly competing with each other in some senses, but they are quite different. Looking at the table above, we can clearly see that SIP is typically a centralized client-server mode network, and has been widely deployed in wireless IP networks. SIP is a lightweight protocol and can be easily implemented into mobile devices that have weak hardware capacities, thus SIP protocols are only responsible for session initiation. However, the conventional SIP platform has high cost for infrastructural investment and maintenance due to the nature of client-server based mode. Also, the conventional SIP networks do not have the ability of self-organizing into optimal topologies through time. I contrast, JXTA is a purely decentralized P2P network which has the ability of self-organizing into an optimal network topology by each individual peers' knowledge of the network environments and resource distribution. The JXTA super-peer can take responsibility of the SIP registrar and proxy services, forwarding SIP messages between different domains. But JXTA has six pre-defined protocols, which aim to take care of all possible scenarios during the communication; all six protocols have to be implemented in order to initiate the JXTA network. Therefore, JXTA is a heavyweight protocol and cannot easily

integrate with mobile devices that have weak hardware capacities. Plus the SIP network has already been widely deployed; therefore we need to consider the compatibility and intercommunication ability with legacy SIP networks when decentralizing SIP to take benefits of P2P networking. Therefore a JXTA based SIP communication network should be created to take advantages of both JXTA and SIP protocols, and be compatible with existing JXTA and SIP networks. A mechanism of converting a SIP operation into JXTA platform operation is need to be designed to be make interoperation between the JXTA overlay and SIP overlay become possible. Through this mechanism, peers can advertise and query their resource (JXTA advertisement mechanism), communicate, interact and collaborate with each other; the peer would also able to store traditional SIP User Agent information.

## 3.4 Difficulties of Integrate JXTA and SIP

To implement a JXTA based SIP communication network, there are issues and difficulties that have to be addressed. The method to look up resources is very different in JXTA and SIP networks. In a JXTA network, every resource is published on the network with a resource advertisement. When a peer looks up a resource [37], it will flood the request to all the adjacent peers or super-peers within the peer group. It's just like one person standing in the middle of the room and yelling to everyone in the room for what he/she wants. Then another peer will start to give a response either with the destination of the resource or a specific address for a peer who may know the answer; whereas in an SIP network, every peer's presence and resource destination are stored and managed by an SIP server. When one peer wishes to talk to another peer, it just needs to connect to the SIP proxy server within the domain, and the SIP proxy server will take care of the rest. Also, due to the nature of P2P networks, continuously maintaining the SIP registry and registered SIP peer's information is a big challenge in the proposed network.

To address the above issues, a prototype JXTA network has been implemented to simulate and examine the possibility of distributing the SIP registry information in a P2P environment. Also, a derivative concept of the BitTorrent

protocol has been designed and implemented within the prototype JXTA network, in order to increase the efficiency of distributing the SIP registry information.

### 3.4.1 BitTorrent

The BitTorrent protocol is an idea that came from Bram Cohen, who designed it in April 2001 and released the first implementation on 2 July 2001. It is now owned by Cohen's company BitTorrent, Inc.

BitTorrent is a peer-to-peer based file sharing protocol, and enhances the efficiency of the file sharing and network communication capability of the peer-to-peer network. The idea of BitTorrent is to distribute large amounts of data widely without the original distributor incurring the entire costs of hardware, hosting, and bandwidth resources. When a BitTorrent network starts data distributing, the original distributor will upload distant small pieces of sharing files to each different file requester. Then the requester who has the different parts of the sharing files can exchange with each other for the data parts they do not have.

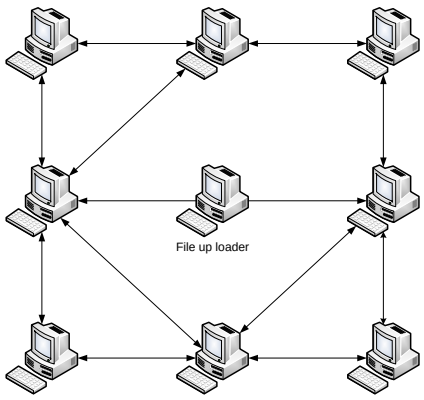



**Figure 8: BitTorrent File sharing Network**

### 3.4.2 The Derivative of BitTorrent

A BitTorrent client is any program that implements the BitTorrent protocol. Each client is capable of preparing, requesting, and transmitting any type of computer file over a network, using the protocol. A peer could be any computerized device which is running an instance of a BitTorrent client.

To share a file, a peer first creates a "*.torrent"* file e.g. "*myFile.torrent*". The torrent file contains metadata of the file which is going to be shared on the network, and a location of a tracker for this shared file. The peer who wants to download this shared file will need to obtain the torrent file first, then use the tracker address contained in the torrent file to connect to the specified tracker. The tracker would tell this peer to download a specific requested file part from a particular peer (or group of peers) on the network.

In a normal client-server mode network, all download requests are made to a single server. The download request for a file is in sequence order. In a BitTorrent network, a peer will only request to download a small part of the file from a peer. Although it could simultaneously send many download requests to different peers on the network. BitTorrent will request to download file chunks in a random order and from random available resources. This would ensure the high availability in the BitTorrent network. Overall, comparing the characteristics of a BitTorrent network and client-server network, the file sharing in a BitTorrent would be more efficient and faster. However, due the unique file sharing characteristic in a BitTorrent network, it may take a while for a peer to increase the number of connections with other peers. Therefore, a typical BitTorrent download speed will gradually increase from a very low speed to a very high speed.

But the question is: how to make a BitTorrent network functional when there is no tracker existing on the network? The design of a BitTorrent peer-to-peer network has inherited the majority of the BitTorrent lineage, except for the tracker. The proposed P2P network is based on JXTA, which means any request or data exchange is achieved by the passing XML messages. Therefore when one peer wishes to find a resource, it will send a broadcast XML message to neighbouring peers on the same network and waits for a reply instead of referring to a tracker. When a reply message has been received, a new Download Module (DM) is created in order to establish a connection to the replier and start downloading data. Therefore one active peer may have many Download Modules (DM) created at the same time depending on the number of replies received from other peers.

As every peer on the network is downloading and uploading at the same time, it is really difficult to track which peer has what part of the file in real time. So a local Download Control Module (DCM) has been implemented to manage Download Modules (DM) to download from which peer on the network and for what specific part of a wanted file. Download Control will first highlight how many chunks are there for a wanted file, and then tell each Download Module to download specific chunk(s) of the wanted file. Consequently, the Download Module may return the downloaded file chunk, or return NULL if there is no required data on its connected peer; and then a new download task will assign to this Download Module. In the case of NULL being returned, the Download Control Module (DCM) will re-assign the same download task to the next available Download Module (DM). The Download Control Module (DCM) will loop the above process until all the chunks of the wanted file have been downloaded.

## 3.5 Summary

In the Simulation platform described above, SIP can be viewed as the service registered with the JXTA network overlay: the function of the SIP registrar and proxy can be implemented into the JXTA-SIP super peer. Also the SIP peer's presence within the network can be published as an advertisement in the JXTA overlay. The JXTA-SIP super-peer (or rendezvous peer) acts as a relay peer between the JXTA peer and the traditional SIP user agent. Each peer in the JXTA network has the function of a SIP UA (User Agent). Those super-peers who act as SIP server also act as SIP registrar and are responsible for publishing the SIP peers' presence onto the JXTA P2P network. A more detailed network design will be described in following chapter.

## Chapter 4

# 4. Design of JXTA and SIP Combined P2P Network

The standard SIP session initialize procedure is implemented by the three way handshake approach: INVITE, 200OK and ACK message [30]. When a caller wishes to initialize a call with someone, the caller would send an INVITE message to the person he/she wish to reach through SIP proxy; The 200OK message will be replied to by the this person through the same SIP proxy; then a direct communication channel will be established after the ACK message. In the above SIP architecture, SIP proxy would always enquire the address of the person who receives the call from SIP domain location services.

In the proposed JXTA-SIP P2P network architecture, the SIP proxy function will be integrated into JXTA-SIP super-peer, who will act as a proxy peer for session initialization between conventional SIP, and act as a rely peer for conventional JXTA peers to communicate with conventional SIP peers. The SIP domain location services will replace P2P resource publish and look up mechanisms, which in the proposed network architecture would be the JXTA resource advertisement mechanism.

## 4.1 Critical Participant in Proposed Network

The prototype of JXTA based SIP communication has been designed to contain four user activities: conventional JXTA peer, conventional SIP peer, JXTA-SIP peer and JXTA –SIP super-peer. The conventional JXTA peer and conventional SIP peer exists in traditional JXTA and SIP networks respectively. Whereas JXTA-SIP peer and JXTA-SIP super exists in JXTA based SIP communication networks. Moreover, the JXTA-SIP super peer acts as the bridge between the

traditional existing JXTA and SIP networks, which are responsible for translating the message between conventional JXTA and SIP platforms.

**JXTA-SIP Super-Peer**

The function of JXTA-SIP super peer can have Network Transport function blocks: The JXTA-SIP super-peer network transport function block is responsible for all JXTA and SIP communications. We can divide this function block into three sub function blocks: SIP transaction block, Message control block and JXTA transaction

- The SIP transaction block would extend the SIP stack which is responsible for SIP operation in lower layers. This sub function block also implements SIP User Agent, SIP Registrar and SIP Proxy.

- The Message control block is the most important function block, which is responsible for translating between SIP message and JXTA message. When the message control block receives a text based SIP message from the SIP transaction block, it will extract the SIP message and convert it into different XML format based on the SIP message type, and vice verse. Then, based on the message type, the message control block will invoke corresponding methods provided by the JXTA function block that is responsible for message transport transaction over JXTA layer. In reverse way, the message control block will invoke the method provided by SIP transaction block that is responsible for SIP message transaction over SIP layer.

- The JXTA function block extends all traditional JXTA characteristics and is responsible for all JXTA layer transactions, publishing advertisements on the JXTA network, discovering the JXTA network resource advertisement, etc.

### JXTA-SIP Peer

JXTA-SIP peer has inherited all functions from JXTA-SIP super-peer. But in the SIP transaction block, the function of SIP User Agent, SIP Registrar and SIP Proxy may not be initiated simultaneously. If JXTA-SIP peer is only acting as the peer on the edge of the network, only the SIP User Agent function will be initiated. Due to the characteristics of the decentralized JXTA P2P network, any ordinary peer can become a super peer at any time based on their credentials and position in the network environment. Therefore the function of SIP Registrar and SIP Proxy can be initiated at any time when needed. The characteristic of initiating different SIP functions in the SIP transaction block would be of benefit in a mobile wireless network due to the device capability and dynamic network environment.

## 4.2 Network Communication Flow

In the proposed JXTA-SIP P2P network architecture, the communication between conventional JXTA peers would be remaining the same. The peer's presence register and session establishment between conventional SIP, conventional JXTA peer and novel JXTA-SIP peer has changed in order to fit in with the proposed network architecture.

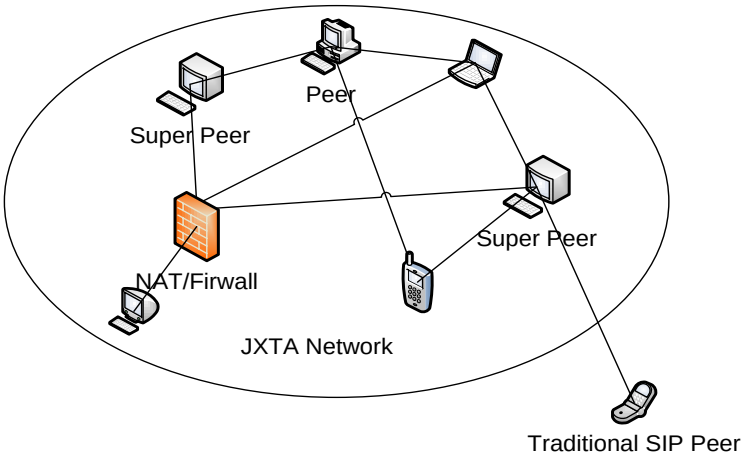



**Figure 9: JXTA-SIP network topology**

Figure 9 above illustrates the topology of a proposed JXTA-SIP network. In the proposed JXTA-SIP approach, conventional SIP peers would register their presence with a JXTA-SIP Super peer. The JXTA-SIP Super peer is

responsible for publishing the other SIP peer's presence onto the JXTA P2P network overlay via the JXTA advertisement. The JXTA overlay will take care of resource publishing and network routing.

When JXTA-SIP or conventional JXTA peers wish to join the network, they will first search advertisements from the JXTA-SIP super-peer to register their presence, before starting other activities. When conventional SIP peers who do not have the ability of the JXTA search function wish to join the network, they will register with a pre-published JXTA-SIP super-peer's domain to register their presence. Then the regular SIP operation will be performed and this JXTA-SIP super-peer will act as relay between this conventional SIP peer and the JXTA based SIP communication platform.

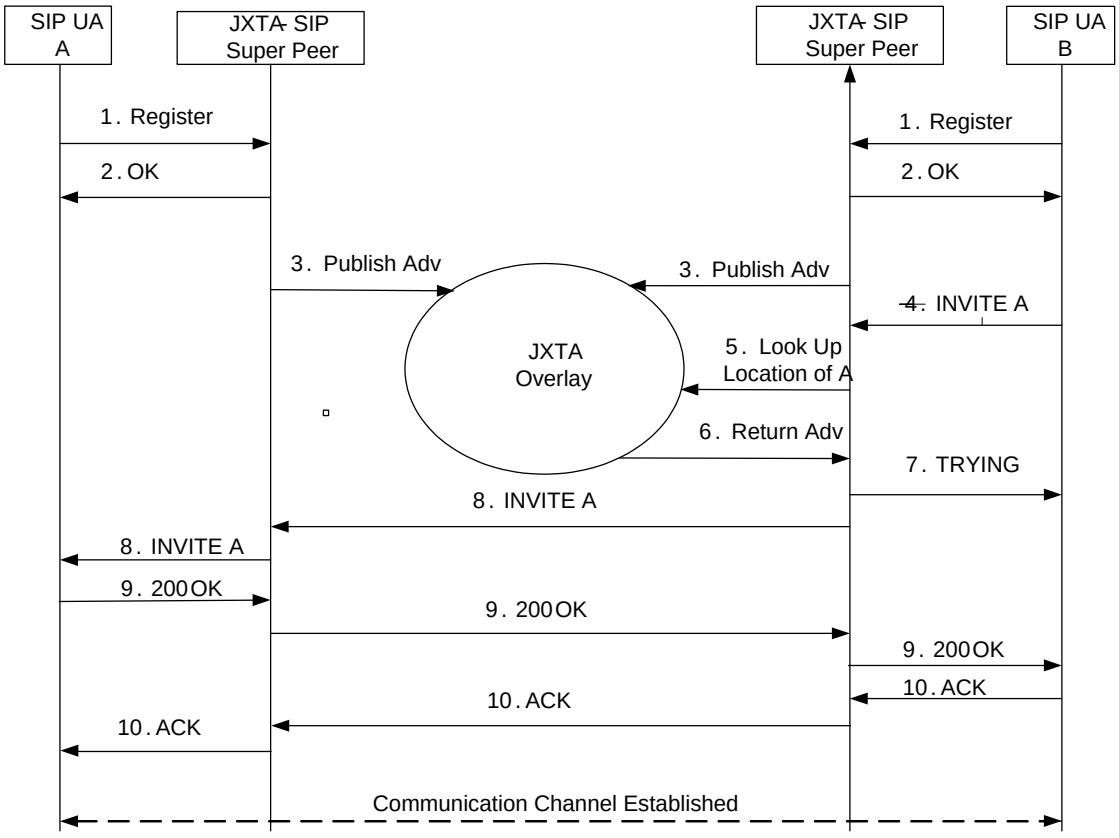



**Figure 10: Communication between SIP peer in the JXTA-SIP network**

The Step 1 and 2 in figure 10 illustrate the registration process for conventional SIP user agent. SIP UA A (A) and SIP UA (B) will send register request to JXTA-SIP super-peer respectively. Then JXTA-SIP super-peer would publish registered SIP peer's presence as a JXTA advertisement onto the JXTA network.

When B wishes to establish a session with A, B would send INVITE to JXTA-SIP super-peer (we call it Super-B) which B has registered with. Then Super-B would look for advertisement of A's presence in the JXTA network. Once Super-B gets the advertisement, Super-B will forward INVITE message to the JXTA-SIP super-peer which A has registered with (we call it Super-A). Then Super-A will again forward message to A. A will then reply the 200ok message back to B though the same path, like the step 8, 9 and 10 in figure 10. The direct communication channel has been establish between A and B after the ACK message.

The communication between a JXTA-SIP peer and a conventional SIP peer are similar to above, due to JXTA-SIP having the ability of looking up functions in the JXTA network as well as all the traditional SIP functions. Therefore JXTA-SIP peer would be able to search the advertisement of the conventional SIP user agent and directly send SIP INVITE message by itself.

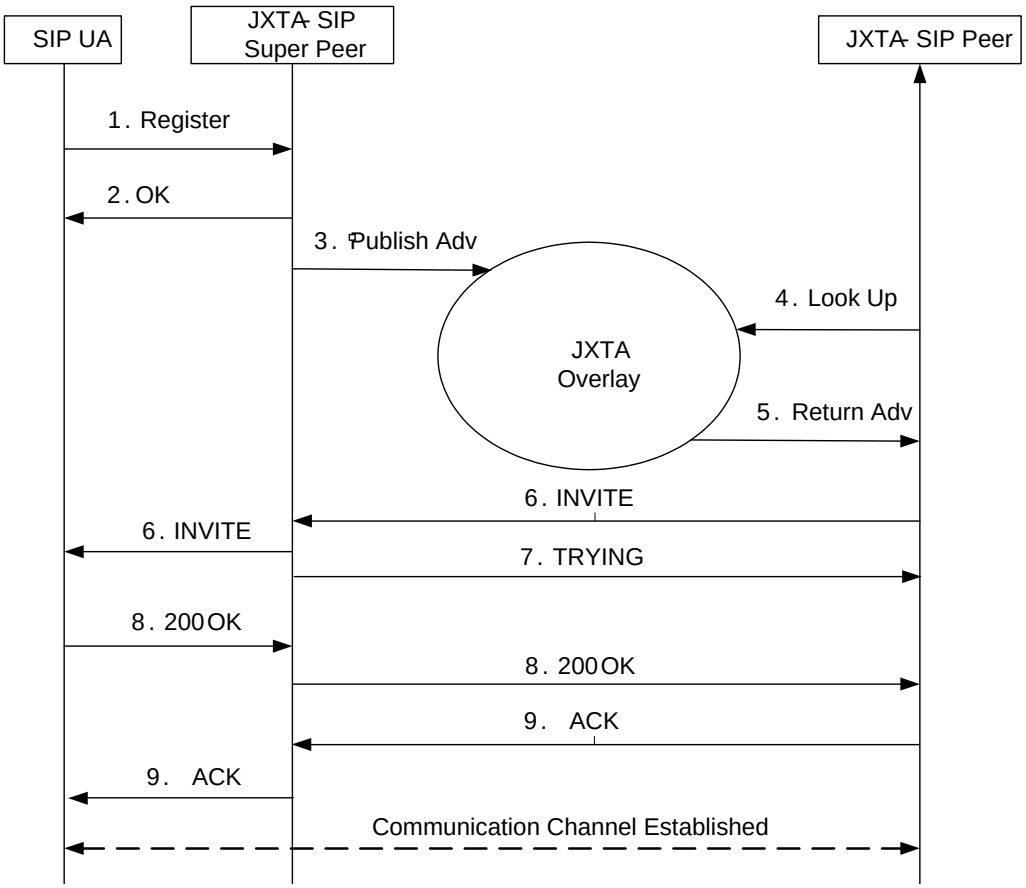



**Figure 11: Communication between SIP peer and JXTA-SIP peer in the JXTA-SIP network**

Figure 11 above illustrates the steps of session establishment between JXTA-SIP peer and conventional SIP user agent (UA). JXTA-SIP peer would first look up the advertisement of SIP UA's presence, and then the session establishment procedure is similar with SIP-SIP UA in figure 10. But instead of going through two JXTA-SIP super-peers, JXTA-SIP peer can directly talk to the JXTA-SIP super-peer that SIP UA has registered with by itself.

The session establishment between conventional SIP and JXTA peer are somehow different from the steps in figure 10 and 11.

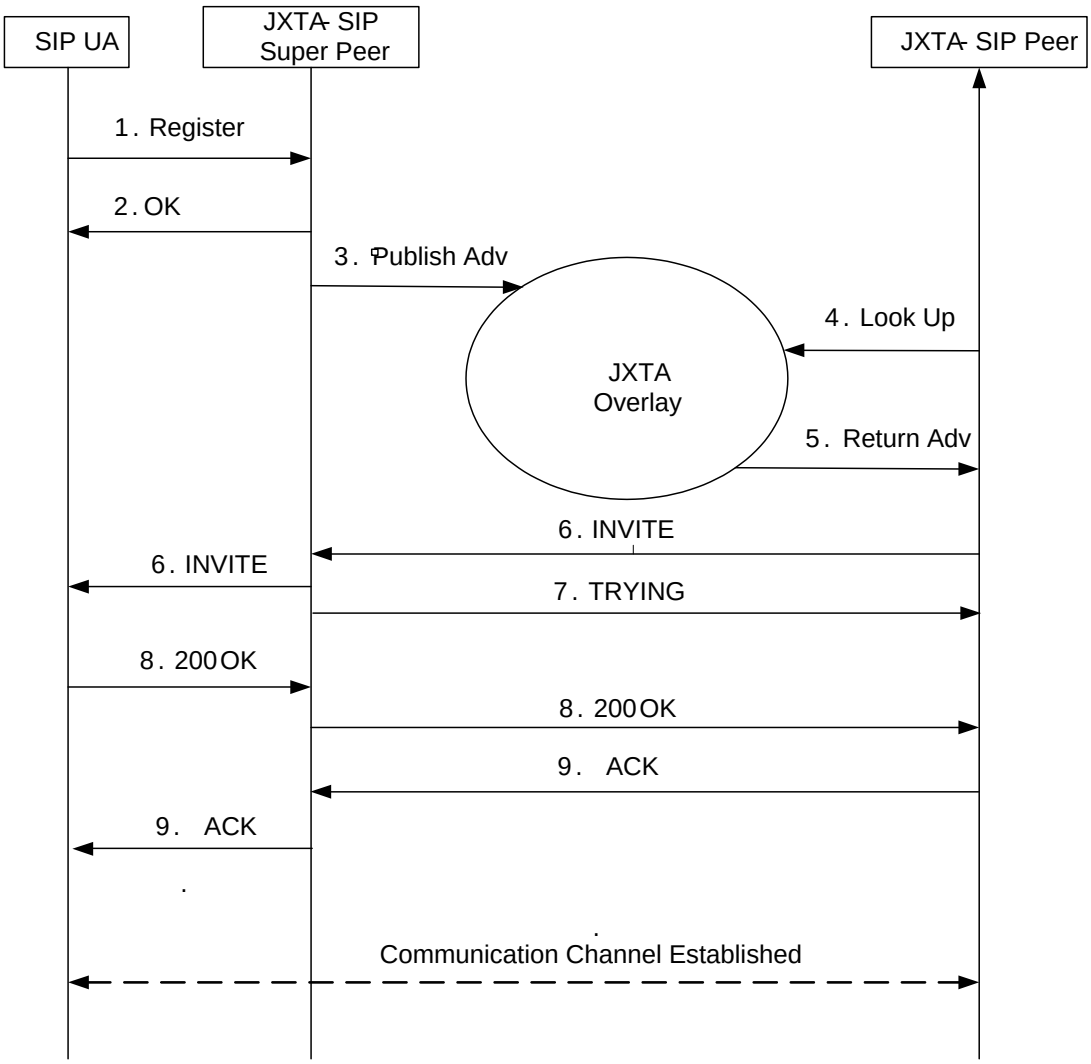



**Figure 12: Communication between conventional JXTA and SIP peer in the JXTA-SIP network**

In figure 12, the session establishment is the same as in figure 11. But JXTA-SIP super-peers are acting as rendezvous/relay peers in between JXTA peers and SIP peers, as JXTA peers cannot communicate to SIP peers directly due to differences in JXTA and SIP protocols.

To sum up, Figures 10, 11 and 12 only illustrate the communication flow amongst the peers within the same domain. The inter-domain communication can be achieved by a rendezvous peer who registers in multiple domains. This rendezvous peer will have the knowledge of all the domains it has registered with, and can reply to the network traffic in between domains. Therefore, the

inter-domain communication flow will be the same except when the traffic crosses the domain boundaries and travels through a rendezvous peer.

## 4.3 Self-Organizing and Fault Tolerance Mechanisms

In the proposed JXTA-SIP approach, self-Organizing [28], [38], [39], and fault tolerance are also important for the network architecture. It's similar to JXTA's self-organizing mechanism.

The self-organization mechanism is done by the participant peer's self experience. After a peer has joined the network, this peer can start querying about the available resources from other peers on the network. After this peer has started receiving replies about the available resources, this peer can also remember which other peers have greater knowledge about this network. Then, this peer will have the ability to reply to queries which have been received from other peers in the network. In time, this peer will gather more and more knowledge; the longer this peer stays on the network and the more positive query results this peer can produce, the higher the rank will be given by other peers on the network. Thus, ordinary peers can become super-peers based on their credentials and position in the network environment. It's like a local shop keeper in real life. The shop has been in the local area for a number of years, and the shop keeper has great experience of the local community and of the environment. Therefore if anyone is new to this area they would go to the local shop keeper seeking answers. The local shop keeper would become a super-peer for this particular local environment in this situation because of his greater knowledge.

Analogous to the example above, we may talk about peer A joining a network of architecture of that of the proposed network. The A would send the request message to all its adjacent peers which is defined by JXTA protocol. Then adjacent peers who received the request either send a reply containing the destination of the requested resource or forward the request to the peer who may know the answers. When A has received a reply and found the requested

resource, A would cache the location of the resource as well as the repliers who replied with the resource location.

After a number of requests have been made through the network, A will have its own list of available resource locations and peers who have great knowledge about the local environment. Therefore when A requests other resources in the future, A will initially send the request to only those peers on the list instead of flooding requests to all adjacent peers (the adjacent peers are defined by JXTA protocol); only broadcasting the request to all adjacent peers if A doesn't get an appropriate reply from those listed peers. Those peers who are on A's list are more likely to be on lists which have been created and cached by other peers in the network. Thus those peers who are well known by every other peer on the network automatically become (and are treated as) super-peers for the local network environment.

In the same way, after A has been on the network for a while, A would also have a list of available resource locations in the network. A could start receiving requests from other peers in the network, and A could reply with the requested resource location information. Then A would start being stored by other peers on the network. The longer A stays on the network, the more positive search results A receives, the more resource locations are on A's list. On other hand, the more positive replies produced by A, the higher the rank A stays on other peers' resource list, the more likely the chance is that A will be ranked as a super-peer by other peers in this network. Therefore, the super-peer on the network is subjective to each individual peer, not objective to all peers. This ranking mechanism will also prevent inappropriate peer becoming a super-peer, e.g. a low powered device will never provide enough positive search result to other peer, and hence it will never have a high enough rank to become a super-peer.

However, when A begins to stop replying to the search requests, this means that either A has lost connection or powered off. Then the rank of A will be lowered by other peers. This is critical when A is already ranked as a super-

peer by other peers on the network, which will lead to the next topic in this proposed network -- Fault Tolerance Mechanisms.

The Fault Tolerance Mechanisms refers to when JXTA-SIP super-peer suddenly goes offline without any premonitions, especially when this JXTA-SIP super-peer is acting as SIP proxy and registrar. This can be caused by a power cut or hardware failure. In order to overcome this issue: when a SIP S who wishes to register with a specific domain within the JXTA-SIP network, there would be more than one JXTA-SIP super-peer or JXTA-SIP peer who can possibly become a super-peer in the future. They will cache the SIP's presences and publish an advertisement of presence for registered SIP peers. When other peers in the network wish to establishment a communication channel with S, they can search for advertisements related to S, and they would get multiple replies. Therefore even when one of the JXTA-SIP super-peers goes off-line for some reason, other peers who have S's registration information would automatically replace the SIP registrar and proxy function for S. Therefore this mechanism would always make sure there is at least one super-peer within a network.

However this would lead to future work on evaluating the trade off between network reliability and network traffic capacity because of the duplicate replies from different peers on the network increasing the network traffic [31]. The fault tolerance mechanisms as described in the previous paragraph will cause duplication of advertisements for one specific SIP peer that exists within the network, due to more than one super-peer will wish to publish this SIP peer's presence within the network. Therefore when one other peer searches for the advertisement of this specific SIP peer, a number of duplicate replies would be received by the request initiator, but only one reply would be enough. But duplicate replies can increase the network traffic and significantly reduce the network communication capacity. The filter function can be implemented to filter the duplicate replies. But filter can only filter the number of replies and does not have the ability to monitor the duplicate advertisements existing on the network which causes the duplicate replies. Therefore some monitoring mechanisms will be needed to monitor the number of duplicate advertisements that exist in the

network. For example only two duplicate advertisements are allowed, which means other JXTA-SIP super-peers only publish the advertisement of one specific SIP peer's presence when there are less than two advertisements for this specific SIP peer exists within the network.

## 4.4 Potential Problems Relating to the Choice of Design

### 4.4.1 Trade-off between Broadcast Scope and Search Quality

In a peer-to-peer network search requests are done by broadcasting XML messages to other available peers on the network. If one peer is broadcasting in a large sized network, this peer may receive a tremendous number of reply messages and the number of reply messages may exceed its message handling capability. Therefore, it may be necessary to limit our broadcasting scope. On the other hand, limiting the broadcasting scope will directly reduce the quality of a peer's search. Hence there is a trade-off between broadcast scope and search quality.

### 4.4.2 Potential heavy traffic on the network

Furthermore, while running a SIP registrar based on peer-to-peer ad-hoc network, there is no guarantee that once a JXTA-SIP peer starts acting as SIP registrar it will continue existing on the network because peers can connect and disconnect to a network at any time. Disconnection may occur for many reasons, for example due to power failure or any other uncontrollable situation. To overcome this issue, all registered SIP peers' information will be stored in a Distributed Hash Table, and will be forward by JXTA-SIP Super peer to any known acting JXAT-SIP who chose to receive this information. Therefore once a known JXTA-SIP Super is down, other JXTA-SIP peers who have the SIP registrar information can become a JXTA-SIP Super peer to continue to maintain the SIP registration information within the network. Thus, the mechanism above will potentially cause a heavy traffic on the network.

### 4.4.3 Potential failure and Security Risk

One critical element of the network is the JXTA-SIP super peer, who has the ability of bridging the conventional JXTA and the SIP platform. Therefore, at least one JXTA-SIP super peer will be required to start and maintain the

functionality of the network. Also, any registered peer can publish resources and generate traffic on the network. Thus, the malicious peer could publish harmful information or generate enormous amount of traffic to jeopardise the network. One possible method to overcome these issues is to place one or more permanent JXTA-SIP peers in the network and register this JXTA-SIP peer in multiple domains, to ensure the communication can travel between different domains. Also, the peer who starts the network will have the less motivation to jeopardise it afterwards. Therefore, the peer who started the network could force a security policy, e.g. a private and public key, to ensure the genuineness of the published resource and network traffic.

## 4.5 Summary

This chapter has discussed the detailed theory of the proposed JXTA and SIP combined P2P network. The following chapter will be introducing the implementation of the simulation platform described in chapter 3. Also, an evaluation will be carried out on the outcome of the test results based on the simulation platform.

## Chapter 5

# 5. Implementation and Evaluation

The core question of the proposed JXTA-SIP network is how to maintain the registered SIP peer information within the network. A characteristic of peer-to-peer ad-hoc networks is that peers are able to join and leave the network at any given time, and that includes the JXTA-SIP Super peer who is acting as SIP registrar. Thus, the most important aspect of the proposed network is sharing the Distributed Hash Table (DHT), which contains the registered SIP peer information, between all JXTA-SIP peers who have the ability to become JXTA-SIP Super peers at a later stage. When an acting JXTA-SIP Super peer has left the network, any existing JXTA-SIP peer can become a JXTA-SIP-Super peer and continue maintaining the SIP registrar. Therefore, the ability and efficiency of distributing the DHT within the P2P layer is the key variable when evaluating the proposed JXTA-SIP network.

To create the test environment, the author has implemented a prototype of a peer-to-peer ad-hoc network based on JXTA. In order to simulate the distribution of a reasonable size of DHT, various sizes of JPEG format images have been distributed amongst all participating test peers. The BitTorrent file swarming method has also been adopted in the test platform in order to increase the efficiency of file distribution.

## 5.1 System State and Implementation of Simulated Network

### 5.1.1 The System State

After starting the network, a registered peer is also ready to publish resource, broadcast searching requests, respond to search requests, and share its resources. The system‘s state has been described; and two UML state

diagrams have been used to help understand the transition between each system state.

**Remote Search**

To start remote search, the following action will be performed:

a. Assembling a broadcast message with the DHT sharing request;
b. Broadcast the message to other peers on the network;
c. Waiting for a specified amount of time for a reply;
d. Start Download service IF reply is received.

**Download Service** (Figure 13)

The Download service will be initiated once a reply message is received from the other peers. The Download service will first establish a connection to replier and initialize the Download Control:

a. Local Download Control Module will assign the download task;
b. Send request to resource peer;
c. Waiting for file chunk sent back;
d. Pass received file chunk to Local Download Control Module to process;

Download Control Module will control the download service to download the Distributed Hash Table (DHT) in the BitTorrent manner. This method will monitor the current DHT file completeness and all the downloading connections for this file. It indicates each connection to download specific parts of the DHT file from each connected peer. When the Download Control Module (DCM) has been initialized, it will obtain the total DHT file size, file chunk size and how many chunks for this particular file from first connected peer.

**Upload Service** (Figure 14)

Hash the file into small chunks and waiting for connection from other peers
Once a new connection has been established
Waiting for requested file part
Upload specific file part to requested peer
(UML represented at next page)

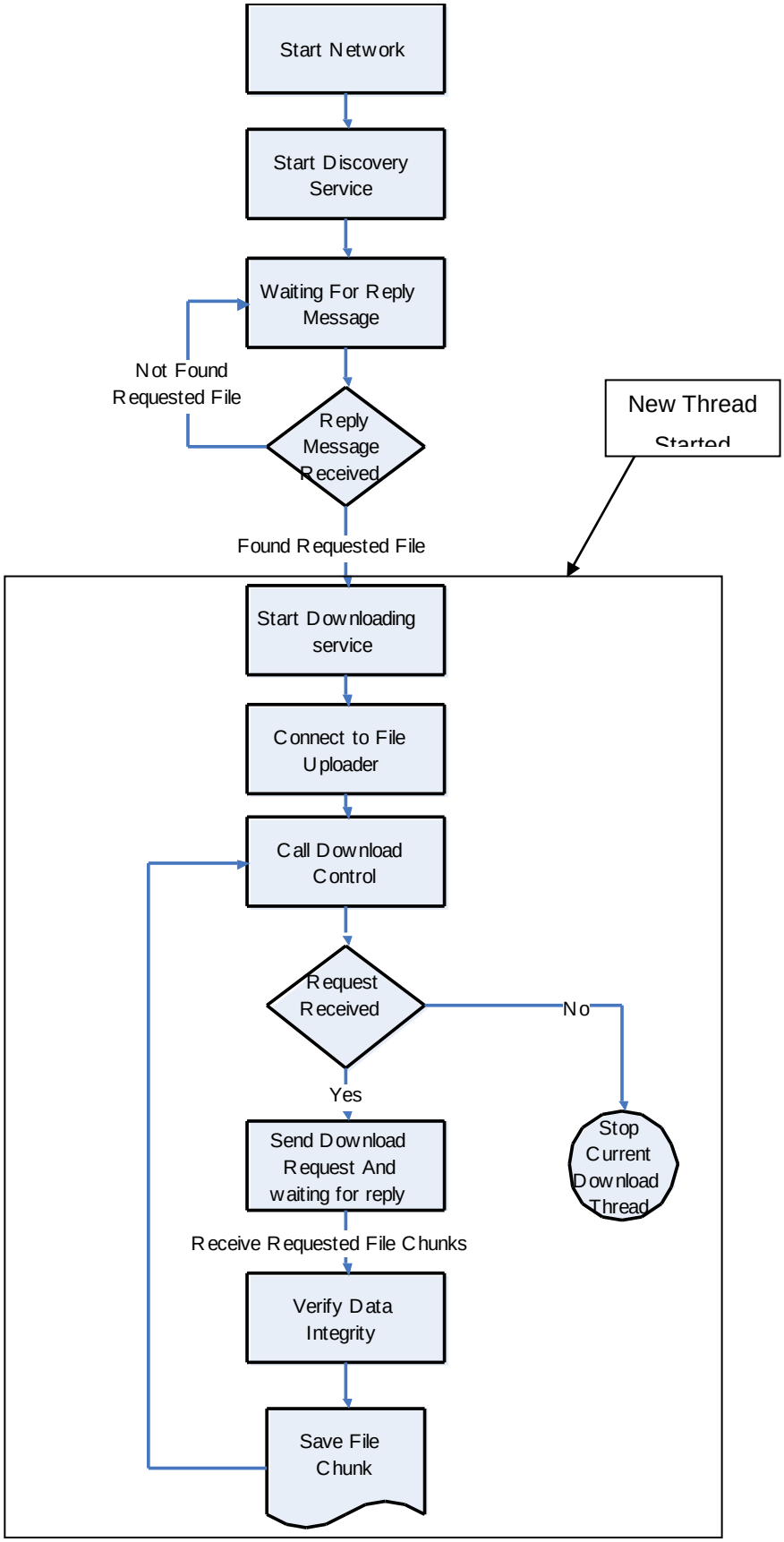



**Figure 13: Download control**

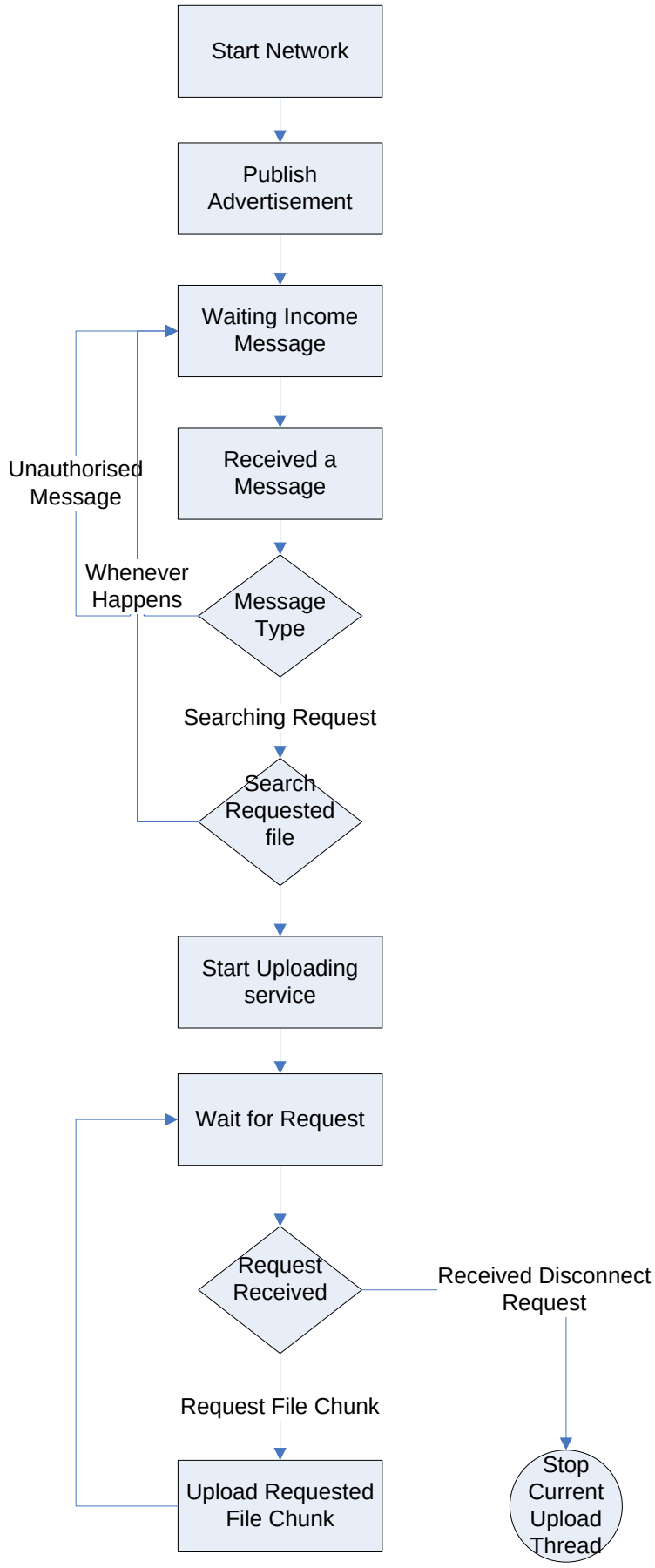



**Figure 14: Upload control**

### 5.1.2 Implementation

**Network Communication Protocol – JXTA**

Ensuring that the JXTA based peer-to-peer SIP network is functional, the author feels it's important to describe details of the JXTA network protocol.

#### 5.1.2.1 Key Aspects on Proposed Communication Network

Referring to Chapter 3 JXTA, there is five essential aspects of the JXTA architecture that have been used in this proposed project have been listed below:

- The open XML protocol to describe network resources
- Abstraction endpoint to endpoints without reply on address resolver such as DNS
- Unique uniform peer addressing scheme (UUID)
- The pipe for propagating the messages
- Shared Distributed Hash Table (DHT) which contains information of registered SIP peer

##### 5.1.2.1.1 Advertisement

Advertisement is the core JXTA objective that is used for JXTA peers to publish pipe services, peer group services and other JXTA service resources. Each JXTA peer may stay on a different network, which may have been implemented by different protocols. Therefore JXTA advertisement is defined by a set of open XML protocol that allows JXTA peers to propagate messages regardless of the network platform.

The *Advertisement.getDocument(MimeMediaType)* method is used to generate the advertisement. Different advertisement representations are available via mime type selection. Typical mime types for JXTA advertisement are "text/xml" or "text/plain" which generate textual representations for the advertisements. Advertisement instances are created via *AdvertisementFactory()*.

#### 5.1.2.1.2 Messages and Message Elements

All the communication in JXTA is done by message passing. JXTA messages contain the data or information sent between JXTA peers. Each data or information is contained in the Message Element. One message can have many message elements or none at all. Each Message Element within a Message is associated with a namespace. Namespaces are used for peers to retrieve relative data from messages (note: that uploading peers and downloading peers must agree on the same namespace).

A JXTA Message Element may be used to store any type of data. The message elements are able to contain Strings, byte arrays, documents, input streams, etc. But the data in elements must be converted to raw bytes before it can be transmitted by a JXTA message.

#### 5.1.2.1.3 Pipes

JXTA peer use pipe to send messages between each other. Pipes are logical connections, which mean pipes connect two peers by a virtual communication channel without having any direct physical link. The pipe is unidirectional, which mean bidirectional communication between two peers can be understood as a pair of pipes, an input pipe and an output pipe. Pipe endpoints are dynamically bound to other endpoints when the pipe is opened. JXTA pipes can have endpoints that are connected to different peers at different times, or may not be connected to any peer at all. All pipe resolution and communication is done within the scope of a peer group, which means the output and input pipes must belong to the same peer group.

#### 5.1.2.1.4 Create PipeID

In the JXTA world, every object has been uniquely identified by JXTA ID e.g. peers, peer groups, pipes and any services. In my project, all the communication is made though JXTA pipes, so each pipe created must have a unique *PipeID*. To create a unique *PipeID* we must:

1. Created initial *seed* for each peer – I use peer's name
2. Give an *expression* to ID creator – I use the name of the pipe

3. Returns a SHA1 hashed string of *seed* + *expression*
4. Use *IDFactory.newPipeID(PeerGroupID, Seed)* to create a unique *PipeID*

See code list below:

```
String expression = PiPeName + PeerName
MessageDigest digest = MessageDigest.getInstance(algorithm);
byte[] expressionBytes = expression.getBytes(charsetName);
byte[] hashResult = digest.digest(expressionBytes);
PipeID pipeID = IDFactory.newPipeID(PeerGroupID, hash(seed);
```

#### 5.1.2.2 Start the Network Platform

Now it is time to start our peer-to-peer network. If a peer wants to share a DHT file with other peers on the network, it first needs to start a JXTA network and needs to join a peer group. This can easily be done in three steps:

1. Create JXTA platform by using *new NetworkManager()* construcor;
2. Start the network by calling *NetworkManager().startNetwork();*
3. Joining the specific peer group (in this project it is the default peer group) *NetworkManager().getNetPeerGroup()*;
4. Start the Discovery service and PipeService (used for creating any type of pipes)

See the code listed below:

```
NetworkManager networkManager = new
NetworkManager(NetworkManager.ConfigMode, instanceName, instanceHome);
networkManager.startNetwork();
netPeerGroup = networkManager.getNetPeerGroup();
discovery = netPeerGroup.getDiscoveryService();
 pipeService = netPeerGroup.getPipeService();
```

After the peer has started the network platform and joined the peer group, it can perform two operations:

- Search and download files from other peers, or

- Share and transfer resources with and to other peers.

From now on we will split the implementation into two parts:

1. Server part – Share and transfer resources with and to other peers
2. Client part – Search and download file(s)

#### 5.1.2.3 Uploading Service – JXTA-SIP Super peer

The JXTA-SIP Super peer will establish an advertisement to share Distributed Hash Table (DTH) which contains the information of registered SIP peers, and then continuously listens for any DHT file share requests. When JXTA-SIP Super peer receives a DHT share request from a JXTA-SIP peer, the JXTA-SIP Super peer will then prepare the uploading service and send a reply message back to the request sender with connection details and public key (The public key is part of a public-private key cryptography system. This key is used to confirm "signatures" on incoming messages, so the receiving peer can verify the incoming message's integrity). The uploading service will start transmitting related file parts when it has received a connection request from the peer who sent the sharing request.

##### 5.1.2.3.1 Starting File Sharing Service

In order to publish a DHT sharing advertisement, a JXTA service is required. JXTA services are services that are published by a *ModuleSpecAdvertisement*. *ModuleSpecAdvertisement* and include a pipe advertisement that can be used by other peers to create output pipes in order to use the service. Each JXTA service is identified by a unique *ModuleSpecID*. There are three separate service-related advertisements:

*ModuleClassAdvertisement* — defines the service class. Its main purpose is to define an advertisement of a module class. It has been identified by *ModuleClassID* uniquely. The following steps need to be taken in order to create *ModuleClassAdvertisement*:

- Create *ModuleClassAdvertisement*

- Set name for *ModuleClassAdvertisement*
- Set Description for *ModuleClassAdvertisement*
- Assist unique ID
- Publish it locally and remotely

See the code list below:

```
ModuleClassAdvertisement moduleClassAdvertisement =
                    (ModuleClassAdvertisement) AdvertisementFactory
                    .newAdvertisement(ModuleClassAdvertisement.getAdvertisem
                    entType());
moduleClassAdvertisement.setName(String JXTAmodName);
 moduleClassAdvertisement.setDescription(String AdertismenttDescription);
 ModuleClassID moduleClassID = IDFactory.newModuleClassID();
  moduleClassAdvertisement.setModuleClassID(moduleClassID);

discovery.publish(moduleClassAdvertisement);
discovery.remotePublish(moduleClassAdvertisement);
```

*PipeAdvertisement* --- defines the service for input pipe. Its main purpose is to enable other peers to create output pipes to connect to this service and to use this service. It has been identified by *PipeID* uniquely. There five steps to create *PipeAdvertisement*:

1. Create *PipeID*
2. Create *PipeAdvertisement*
3. Set *PipeID*
4. Set *PipeType*
5. Set PipeName

See the code list below:

```
PipeID pipeID = createPipeID(DefaultNetPeerGroupID, pipeName);
PipeAdvertisement pipeAdvertisement = (PipeAdvertisement) AdvertisementFactory

                    .newAdvertisement(PipeAdvertisement.getAdvertisementType(
          ))
                        .setPipeID(pipeID)
                        .setType(PipeService.UnicastType)
                        .setName(String PipeName);
```

*ModuleSpecAdvertisement* — defines a service specification. Its main purpose is to provide references to the document which is needed in order to create a confirmation of implementations of that specification; the secondary use is to make running instances usable remotely. It has been identified by *ModuleSpecID* uniquely. The way to create *ModuleSpecAdvertisement* is almost the same as the way to create *ModuleClassAdvertisement*, but we need to add *PipeAdvertisement* to *ModuleSpecAdvertisement* in order for other peers to create an output pipe to connect and use the services.

```
ModuleSpecAdvertisement moduleSpecAdvertisement = (ModuleSpecAdvertisement)
AdvertisementFactory.newAdvertisement(ModuleSpecAdvertisement.getAdvertisement
Type())
                    .setName("JXTASPEC: File sharing")
                    .setDescription("File sharing system at testPeer1")
                    .setModuleSpecID(IDFactory.newModuleSpecID(mcID));

PipeAdvertisement pipeAdvertisement = createPipeAdvertisement(AdvertismentName);
moduleSpecAdvertisement adv.setPipeAdvertisement(pipeAdvertisement);

discovery.publish(moduleSpecAdvertisement);
discovery.remotePublish(moduleSpecAdvertisement);
serviceInputPipe = pipeService.createInputPipe(pipeAdvertisement);
```

#### 5.1.2.3.2 Waiting for File Searching Requests

After DHT sharing advertisement has been published, and an input pipe for other peers to use the service has been created, JXTA-SIP Super peer now needs to listen to the input pipe for any incoming sharing request messages.

```
Message message = serviceInputPipe.waitForMessage();
```

#### 5.1.2.3.3 Hash the File (*parallel running with* 5.1.2.3.4, 5.1.2.3.5)

Once JXTA-SIP Super peer has received a DHT sharing request, it will first hash the DHT file into small chunks in order to distribute. This is the concept of BitTorrent: Instead of downloading an entire file from on peer on network, BitTorrent breaks a file into chunks and distributes them among several file

requesters. Therefore the uploading service should be able to cut file into small chunks and transmit only a small part of a file at a time. To implement the BitTorrent function, the uploading service will first hash the file into many small chunks, and then store those chunks into a hash map with an association key. The association key also indicates the order of chunks and the position of this chunk in the complete/entire file. The key of chunk makes it easy to retrieve the particular file chunks, and indicates in what order these chunks belong and their specific position in the particular file. The file hash function is implemented by following these steps:

1. Read in file

```
DataInputStream dataInputStream = new DataInputStream(new FileInputStream(infile));
```

2. Initialize a matrix byte[][] to contain data for a file. Calculate how may chunks we need for a file, and the number of rows in a matrix byte[][] is equal to number of chunks (each row of matrix contains a chunk of the data). We then put each row of matrix into hash map and associate with a key.

```
byte[][] bfin = new byte[row][buffersize];
int tmp;
for (int i=0; i<row; i++ ) { for (int j=0; j<buffersize; j++) {
        if((tmp = fin.read()) != -1 ) { bfin[i][j] = (byte)tmp; }   }       }
for (int i=0; i<row; i++) { hashmap.put(Integer.toString(i), bfin[i]); }
```

#### 5.1.2.3.4 Start up the Uploading Service (*parallel running with 5.1.2.3.3, 5.1.2.3.5*)

To start the uploading service, a JXTA *BiDiPipe* has been created and a *BiDiPipe* advertisement has been sent back to the requester, and then wait for the requester to connect on *BiDiPipe*.

Create server pipe:

```
serverPipeAdvertisement = testserver.createPipeAdvertisement(fileName);
serverPipe = new JxtaServerPipe(netPeerGroup, serverPipeAdvertisement);
serverPipe.setPipeTimeout(int TimeOut);
```

Create a JXTA BiDiPipe to wait for a connection and to listen for any incoming messages.

```
bidirectionalPipe = serverPipe.accept();
bidirectionalPipe.setMessageListener(MessageListener);
```

#### 5.1.2.3.5 Create Security Key Pair and Reply Requester (parallel running with 5.1.2.3.3, 5.1.2.3.4 but depending on 5.1.2.3.4)

Security is another important issue in a peer-to-peer network. Also JXTA pipes are only virtual communication channels and there are no physical connections between two peers. Therefore we use a digital signature to verify the authenticity of the message.

Create Key pair and signature

```
KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");
keyGen.initialize(1024, new SecureRandom());

KeyPair keyPair = keyGen.generateKeyPair();
PrivateKey privKey = keyPair.getPrivate();
signature = Signature.getInstance("SHA/DSA");
signature.initSign(privKey);

PubicKey pubKey = keyPair.getPublic();
publicKeyArray = pubKey.getEncoded();
```

After the public key have been created, a *BiDiPipe* advertisement needs to be obtained in order to enable the DHT sharing requester to connect to the file uploading service, send a request and download the wanted file chunks.

Create *BiDiPipe* advertisement: *serverPipeAdv = testServer.createPipeAdvertisement(fileName);*

#### 5.1.2.3.6 Transmitting the Requested File Chunk to the Requester

Once a connection has been established between the upload service and DHT sharing requester, the requester will send a message to the upload service on the other end of the *BiDiPipe* to request a particular chunk of a DHT file.

First: the upload service will get the message element, which indicates the requested file chunks:

```
String wantedFilePart = requestMessage.getMessageElement
                                        (testServer.messageNameSpace,
                                        testServer.Request_Part);
```

Second: the upload service will retrieve the requested file chunk from local hash map (note: all the file chunks are stored in byte array):

```
byte[] dataArray = (byte[])hashmap.get(wantedFilePart);
```

Third: to use private key (generated in *5.1.2.3.5*) to sign the *dataArray*

```
signature.update(inputDataArray);
result = signature.sign();
```

Finally: upload service assembles a reply message, which contains the requested file chunk and the digital signature. Send the *response* message back to the DHT sharing requester though JXTA *BiDiPipe:*

```
Message respond = new Message();
respond.addMessageElement(testServer.msgNameSpace,
        new StringMessageElement(testServer.Request_Part, wantedPart, null));

respond.addMessageElement(testServer.msgNameSpace,
        new ByteArrayMessageElement(testServer.Data_Array,
                                    MimeMediaType.AOS, dataArray, null));

respond.addMessageElement(testServer.msgNameSpace,
                        new ByteArrayMessageElement(testServer.Hash_Code,
                                                    MimeMediaType.AOS,
                                                hashCode, null));

bipipe.sendMessage(respond);
```

#### 5.1.2.4 Download Service

The goal of the download service is to search for DHT file sharing advertisement on the network and download all pieces of the DHT file from different available resources locations. The method of downloading is extended from the BitTorrent file swarming system.

If a peer (download peer) wants to find and download DHT files, it first searches for any peers on the network whom have published the “DHT File Sharing Service”, and then sends a sharing request to the peer (upload peer) who has published DHT file sharing service advertisement. If the upload peer returns a message to say: “Hi I have got the DHT file (or part of the file), and here are my connection details”, the download service on the download peer will then use these connection details to create a JXTA *BiDiPipe* to connect to the upload peer and start downloading the DHT file.

#### 5.1.2.4.1 Start Discovery Service

The discovery service is used by the download service, which it uses to discover the “DHT File Sharing Service” advertisement that has been published by other peers on the network.

```
DiscoveryService discovery = PeerGroup.getDiscoveryService();
```

The code above is to show how to create discovery within our peer group.

#### 5.1.2.4.2 Discovery DHT File Sharing Advertisement

Once the discovery service has been started, the download peer can then discover the “DHT File Sharing Service” that has been published on the network. The download peer then adds a listener to the discovery service to listen if any “DHT File Sharing Services” have been found. Once the discovery service has found the “DHT file sharing advertisement”, it will send out a share request to the upload peer to ask for the DHT file.

```
DiscoveryService.addDiscoveryListener(DiscoveryListener);
DiscoveryService.getRemoteAdvertisements (peerID, type, attribute, threshold, DiscoveryListener);
```

#### 5.1.2.4.3 Handle Discovery Result

After the discovery request has been started, many “DHT File Sharing Services” that have been published on the network could be found. Discovery Listener will catch each “DHT File Sharing Service” that has been found on the network, and

get the service publisher's input pipe advertisement. This advertisement will be used later for the download service to create a pipe to connect to the "DHT File Sharing Services" publisher and send a sharing request.

```
DiscoveryResponseMsg discoveryResponseMsg = ev.getResponse();
Enumeration enumeration = discoveryResponseMsg.getAdvertisements();
ModuleSpecAdvertisement moduleSpecAdvertisement =
                                        (ModuleSpecAdvertisement)
                                        enumeration.nextElement();
PipeAdvertisement pipeAdvertisement =
moduleSpecAdvertisement.getPipeAdvertisement();
```

Now create output pipe use *PipeAdvertisement* we have got from discouvery service:

```
OutputPipe outputPipe = PipeService.createOutputPipe(pipeAdvertisement, timeOut);
```

#### 5.1.2.4.4 Sending a Searching Request to the "File Sharing Service" Publisher

After "DHT File Sharing Service" has been discovered along with the service input pipe connection details, the download service can send a message to the "DHT File Sharing Service" publisher to request sharing the DHT file.

```
Message searchMsg = new Message();
outputPipe.send(searchMsg);
```

#### 5.1.2.4.5 Initialize Download Service

When the DHT share request message has been sent out to each of the "DHT File Sharing Service" publishers, a number of replies may already have been received from other peers. Some repliers may say: "I am sorry that I don't have DHT file", and some repliers may say: "Hi I have the DHT file". The positive replier will also send back a JXTA *BiDiPipe* advertisement for the download server on the download peer to connect to the replier and download the DHT file from them.

After creating a JXTA *BiDiPipe* using the pipe advertisement that has been received from the service publisher, the download service then register a listener to listen to any input messages (e.g. reply messages from the “DHT File Sharing Service” publisher which contains the requested DHT file chunks):

```
JxtaBiDiPipe JxtaBidirectionalPipe = new JxtaBiDiPipe(PeerGroup, PipeAdvertisement,
                                                      timeOut, PipeMsgListener, boolean
                                                      reliable);
```

#### 5.1.2.4.6 Initialize the Download Control Module

The Download Control Module (DCM) is the download manager for the DHT file that download peer has requested from the other peers. A DCM need to be initialized in order to setting the total number of chunks for the DHT file and the size of the chunks. The download peer will get this initialization information from the first peer, which has this information, which it connects to.

Author introduced in *5.1.2.3.3 Hash the file* how to hash a file into small chunks and store in a hash map. The DCM performs the reverse process of hash file, in which it will collect each DHT file chunk from different resources available on the network and reassemble the original DHT file. The initialization information includes details of the size of the hash map. The size of the hash map also indicates how many chunks there are for a DHT file. The DCM will then initialize the same size of hash map for storing the file chunks it will receive in the future. By retrieving the hash map, DCM will be able to check the file chunks that it do not has locally, and assign download task to a download service.

```
ConcurrentHashMap hashmap = new ConcurrentHashMap()
```

The code above has initialized a ConcurrentHashMap. Due to the fact that the download service simultaneously downloads file chunks from different peers; it may receive several file chunks at the same time from different peers. DCM then use ConcurrentHashMap as it supports the full concurrency of retrievals and adjustable expected concurrency for updates.

#### 5.1.2.4.7 Download DHT File Chunks

Each download service is connected with a different DHT file holder on the network. After the download service received the download task from Download Control Module, it will send download request to the other end of the communication channel (pipe) and wait for a reply message. The reply message may say: “I don’t have the DHT file chunk you want”; or “Hi, here is the DHT file chunks you requested and here my signature for the data”. Once the download service has received a positive reply message, there are two actions performed:

1. Using the obtained public key and the signature contained in the reply message to verify the data’s integrity and identify the sender.
   *signature.verify(Data of File Chunk);*

2. Pass the DHT file chunk to Download Control Module; DCM will then insert DHT file chunks into an appropriate place in DHT file hash map once the received data has been verified.

Repeat this above process until all the chunks of the DHT file have been downloaded.

## 5.2 Results and Evaluation

The present author has introduced the method of implementation of the proposed project in a previous chapter. In this Chapter, the author will carry out tests on the performance of the proposed project.

### 5.2.1 Testing Environment

The proposed project will request multiple peers to be involved to make the network fully functional. Due to having limited computing equipment, the author will start and configure 4 peers on different computers, and all tests will be carried out in three different network configuration scenarios:

1. All five participating peers are connected within the same local network – connected under the same network router.

2. Two participating peers are connected to a local network (router), then this router is connected to the University's network; three participating peers are directly connected to university's network. In this way, the author hopes to simulate the network environment of each participating peer being hosted in a different network domain.

3. Based on scenario 2, the author has disconnected one peer from the university network (directly connected without router in the middle), and connected this peer directly to the Internet via a 3G dongle. In this way, the author hopes to simulate the network environment with mixed connections.

Operating System: Mac OS 10.8 x 2, Windows 7 x 2 and Windows XP x1

Java Platform:

Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_13-b05-237)

Java HotSpot(TM) Client VM (build 1.5.0_13-119, mixed mode, sharing)

JXTA Platform:

JXTA-JXSE 2.5

### 5.2.2 Start Network Platform

The JXTA network platform is a key part of the proposed project. All the communication between each peer is through this network platform. To test the network platform, the author has consecutively configured 5 peers based on the scenarios described above; each peer has been using the JXTA default peer configuration and is set to join the JXTA default peer group.

The expected result: all peers should successfully start and stand by for any incoming search requests.
The actual result: all peers have been started and are standing by for any incoming search requests.

### 5.2.3 Discover File Sharing Service and Send Search Request

This is another critical part of the project, each peer should be able to find and locate a DHT file on a completely decentralized network. This test section will simultaneously evaluate two functional areas of this proposed project: Publish "DHT File Sharing Service" advertisement and Start Discovery Service to find advertisements.

Carry on testing step in *5.1.2.2 Start Network Platform.* There are 5 peers that have been started and exist on the network. Now, the author has randomly selected a peer and a start to discovery service to search for "DHT file sharing service". A new DHT file share request message will be sent out to "DHT file sharing service" advertisement publisher every time we discover a new "DHT File Sharing Service" advertisement.

#### 5.2.3.1 Locate and Upload DHT File

Extending the test in section in *5.1.2.3 Discover DHT File Sharing Service and Send Search Request*, a peer should be able to locate the DHT file when it has received a DHT share request from another peer on the network, and send the result back to the DHT share requester.

Test Environment: Five peers started, two acting as DHT file share requesters, and the other three acting as DHT file upload peers.
Pre-setup Condition: the author has copied a specific file in the local DHT file sharing folder for three of the upload peers.
The expected result: the download peer should receive three positive replies for file sharing request.

The actual result: Three positive reply messages have been received.

#### 5.2.3.2 Network Resource Discoverability and Search Performance Measure

In order to test network resource discoverability and search performance, the author has consecutively established 100 file sharing requests based on scenarios described in section 5.2.1 (Testing Environment). The resource discoverability and search quality point is measured from the time of file sharing request being established on the network, until the first time each participating peer has been connected with bidirectional pip (and ready for file swarming). The figure below shows the resource discoverability and search quality based on all three scenarios in section 5.2.1.

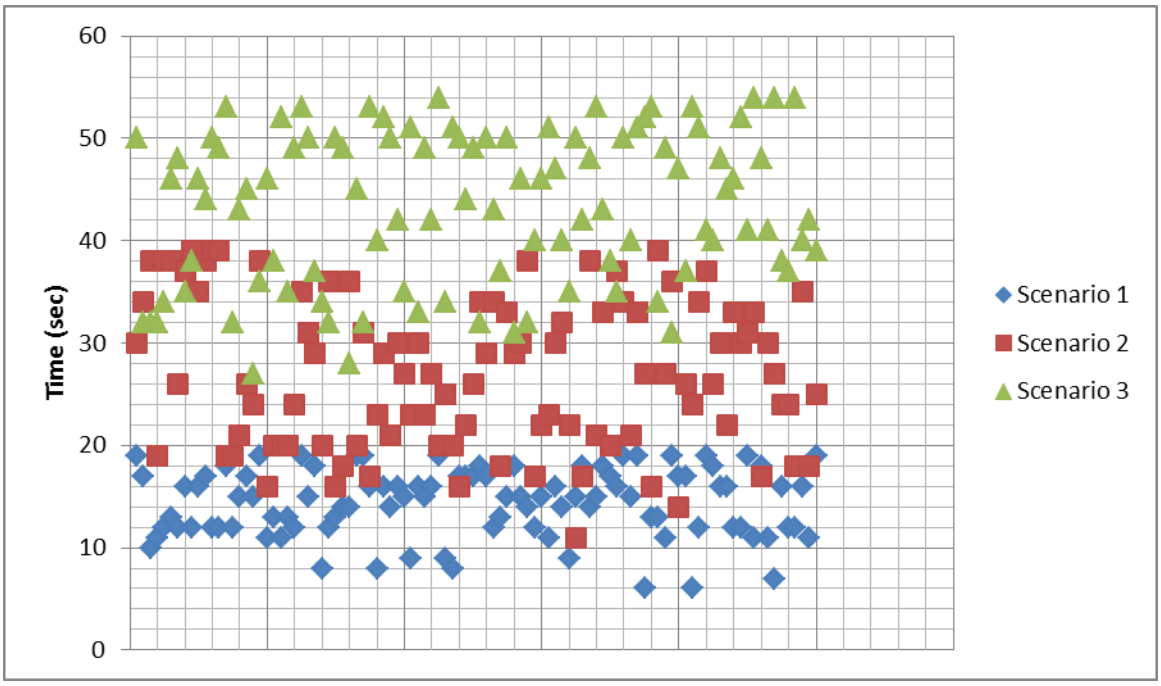



**Figure 15: Finish time when the same file size (100KB) is completely spread amongst all the peers in each scenario**

Figure 15 (one-dimensional figure) illustrates the test results of the following case:

- *The time is calculated from the first generated search request appearing in the network to the last file chunk that has been sent to the peer who sent the last file request*
- Same file size (1000KB) has been repeatedly distributed 100 times in each scenario described in *5.2.1 Testing Environment*

- *Each dot presented in figure 15 is registered as the finish time when the file is completely spread amongst all the peers*
- *The time is measured in milliseconds, then converted into seconds and presented in figure 15*

Refereeing the figure 15: In scenario 1, all participating peers are connected within a local network environment, therefore it's taking much less time to discover the shared resource and establish file swarming around all peers, and also the performances are more predictable. In scenario 2 and 3, with increased network connection complexity, more time will be required to search the shared resources and establish file swarming; also the result will become more dispersed, which means less predictable performance.

### 5.2.4 Establish Bidirectional Pipe and Simultaneously Download

The goal of this test is to test the following aspect of the proposed project:

1. The DHT file requester peer should be able to establish a bidirectional JXTA pipe connection to each discovered DHT file holder

2. The file requester should simultaneously download small parts of the file chunks from each file resource holder

3. As the DHT file holder will upload a small part of the DHT file as requested by the download peer, the DHT file holder should be able to hash the file into lots of small chunks with identical size.

Test method: A series of file swarming requests was started based on all scenarios described in 5.2.1, with gradually increasing file size, to measure the time required to finish the file swarming around all participating peers.

Test Environment: Five peers started, select two random peers acting as DHT file requesters, and other three peers acting as DHT file holders.

Pre-setup Condition: All DHT file holders have the file or part of the file that has been requested by the DHT file requester (download peer).

The expected result: The DHT file requester (download peer) should be able to simultaneously download small parts of the file from each of the DHT file holders upon request.

The actual result: The DHT file requester (download peer) is simultaneously downloading different parts of the DHT file from different DHT file holders.

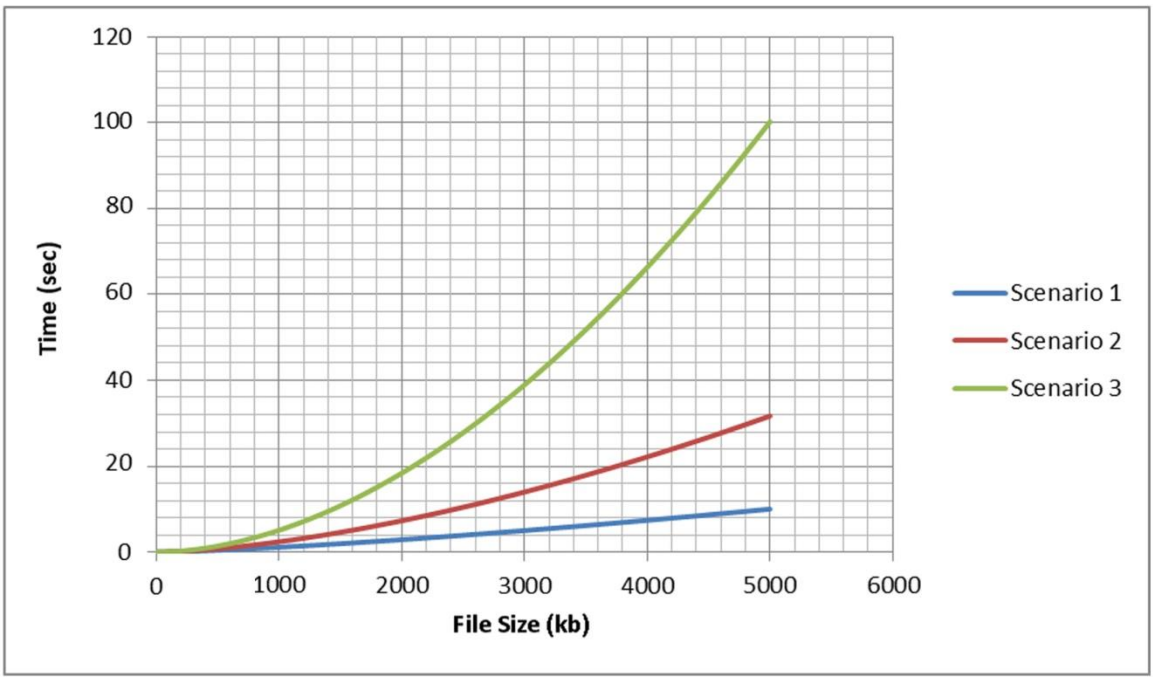



**Figure 16: Incensement of the file destitute completion time when the file size increases in each scenario (file size starts form 50KB, and increases by 50KB each time)**

Figure 16 (two-dimensional figure) illustrates the test results of the following case:

- *The time is calculated from the first generated search request appearing in the network to the last file chunk that has been sent to the peer who sent the last file request*
- *Each line represents the incensement of the file destitute completion time when the file size increases* in each scenario described in *5.2.1 Testing Environment*

- The x-axis represents the size increases in KB (kilobyte); y-axis represents the time increases in seconds
- File distribution has been tested 100 times in each scenario described in *5.2.1 Testing Environment*
- *File size starts form 50KB, and increases by 50KB each time*

The figure above shows that in each scenario, the larger the file is, the more time is required to finish the file swarming. Also the ratio between file size and the time required for file swarming is very different, especially in scenario 3. In scenario 3, time increase dramatically compared to scenario 1 and 2, due to the complexly of network connections.

### 5.2.5 Network Speed Consistency Test

The network speed consistency is also another important factor in this proposed network, which has direct influence on the efficiency of file swarming.

Test method: The speed consistency has been measured by monitoring the time (millisecond) it takes to transmit a fix amount of data (byte), and gradually increasing the size of the data set to find out if the time consistently increases with the size of the data set.

According to the test result, the file transfer rate can be varied in each different scenario. The figure below shows the time (millisecond) required to transfer a fixed amount of data (byte).

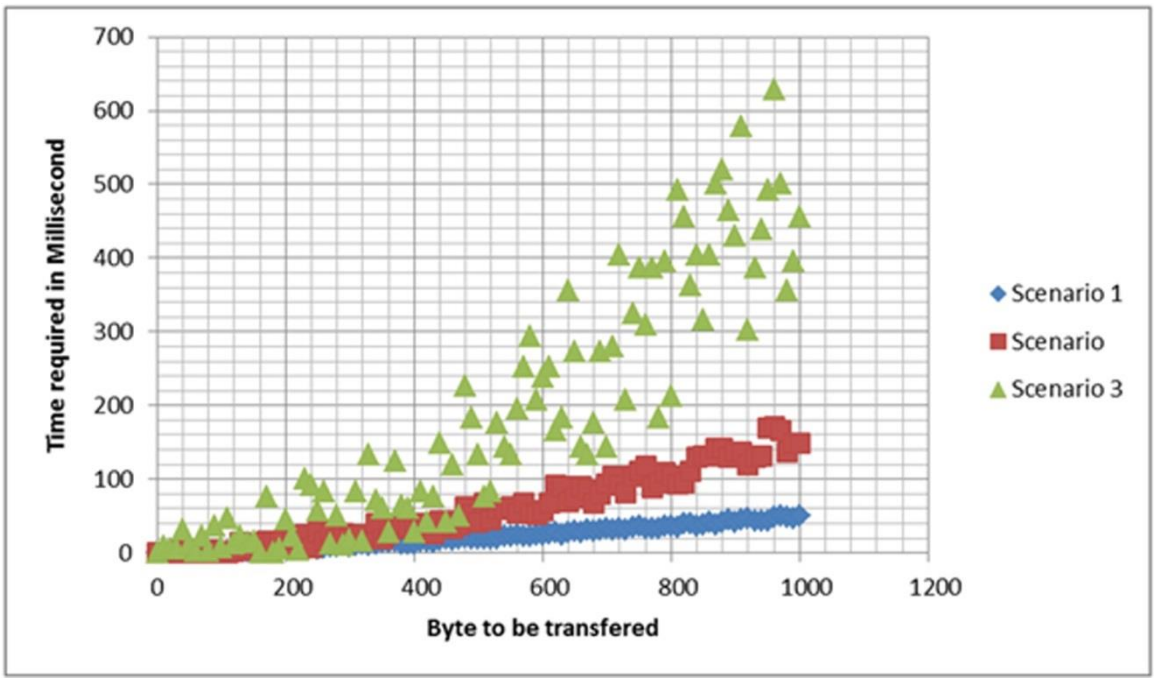



**Figure 17: Finish time when the file is completely spread amongst all the peers in each scenario (data chunk size starts from 10bytes, and increases by 10bytes each time)**

Figure 17 (two-dimensional figure) illustrates the test results of the following case:

- *The time is calculated from the first generated search request appearing in the network to the last file chunk has been sent to the peer who sent the last file request*
- *Each dot presented in Figure 17 is registered as the finish time when the file is completely spread amongst all the peers*
- The x-axis represents the size increases in B (byte); y-axis represents the time increases in milliseconds
- Data chunk distribution has been tested 100 times in each scenario described in *5.2.1 Testing Environment*
- *Data chunk size starts from 10bytes, and increases by 10bytes each time*

In scenario 1, the network speed is highly consistent, the time it takes to transmit a data set (number of byte) increases in the same ratio as the increase in the size of a data set. In scenario 2, the network speed is rather consistent, but there is noticeable variation on the ratio between increasing the size of data

set and required transmitting time. However, in scenario 3, the ratio between the size of the data set and the required transmitting time can be very different from time to time.

### 5.2.6 Test Summary

The implemented network has met the functional design based on testing aims, each participating peer can search and publish resources on the network, also be able to establish bi-directional communication pipe and create file swarming network to distribute the required files. The performance of the proposed network is more stable in local network environment; when the complexity of the connection increases in the network, e.g. in scenario 3, the performance can be varied depending on the route of establishing the bi-directional pipe at the time. According to the test result, the proposed network would be more suitable for a local or more manageable network. Therefore a real life project in a home environment is discussed in the following chapter, where the proposed network can be applied.

## Chapter 6

# 6. Conclusions

This thesis studied the concept of JXTA-SIP P2P networking. It also looked at other related work to the research, and compared it with the proposed network infrastructure. In contrast to other similar works that have been done, the proposed approach takes advantage of powerful resource distribution in P2P and a light weight SIP protocol for session establishment. Without extending the SIP message, the proposed work can smoothly inter-communicate with a conventional SIP network. Also the mechanisms of active self-organizing and fault tolerance have been discussed this approach.

However, there are also some limitations in the author's proposed network. In order to increase the search quality and maintain the high level of fault tolerance, heavy network traffic will be generated as a result. Therefore, when constructing the network, it is imperative to design in detail how to balance the trade off between network reliability and maintenance of network traffic efficiency. Moreover, security mechanisms should be in place to ensure all network traffic is genuine.

Based on the proposed concept in this project, a test network has been implemented to study the feasibility of the proposed network. Although, the test result shows that the performance and efficiency is unstable in a less manageable network environment, the concept of this project has been approved. The test result produced in this thesis is based on the network which only has basic functionality being implemented, due to limited resources and time. However, the performance and efficiently of the implemented test network in a large, less managed environment can be improved by introducing additional mechanisms, which will be discussed in the following paragraph.

## 6.1 Future Work

The prototype system that has designed and implemented has partially achieved the goals of the proposed project. However, a comprehensive system with improved functionality can be achieved if the author has more equipment available and more time.

### 6.1.1 Increase Efficiency on Publish DHT Sharing Advertisement

In this project, the resource discovery is done by asking each individual peer on the network. This is infeasible in a large sized network. A more appropriate solution could be:

1. The DHT file original distributor starts a new peer group and publishes a group advertisement for sharing the DHT file;

2. For each peer who wishes to download this file, it only needs to search the peer group advertisement and join the group;

3. Once the peer joins the group, this peer can start downloading the shared files and exchange file chunks with other peers in the group.

### 6.1.2 Implementing a more reliable Supper Peers

A more reliable JXTA-SIP Super peer could be implemented, whom will stay on the network consistently, have full knowledge of the local network environment and resource distribution. In the implemented network, all DHT share requests are sent by broadcasting to the entire network. This could be improved by referring to current acting JXTA-SIP Super peers first, and then asking every other neighbour peer if a negative share request has been received. If one JXTA-SIP Super peer does not have the knowledge of the required DHT file, then this super peer will forward the request to other super peers who will in turn forward the request wherever necessary, until a desired DHT file is found or the request has exceeded its lifecycle.

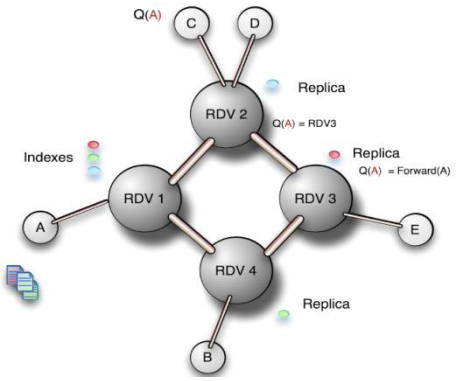



**Figure 18: JXTA-SIP Super Peers involved P2P network [11]**

### 6.1.3 Increased Security Level

As the author has previously discussed, in a peer-to-peer network, any peer can act as a upload peer to upload DHT files or join a peer group to distribute the DHT file. The big issue is how to protect the network from unauthorized/ malicious peers. A JXTA PSE (Personal Security Environment) Membership Service can be implemented in order to prevent malicious peers destroying the network.

PSE membership service is a secure membership service for each peer group; the members use the Public Key Infrastructure certificate to identify themselves and other members within a peer group. This would allow peers to protect themselves from any unauthorized peers or resources.

As well as the work proposed above, there is some future work that has been summarised below as they might become directly relevant during the completion of the MPhil, and can be a direct extension of the MPhil thesis.

## 6.2 Potential Extension on Current Work

The work that has been proposed above is currently focused on a small home environment, with services always provided by a single peer. But as the proposed network topology is a completely decentralized P2P network, the capability of a single peer (device) on the network can often become the bottleneck of service publishing on the network. Therefore collaborative services published among a group of peers (devices) will require creation of a

complex service which involves large amounts of calculations and interactions. This would involve Grid computing concepts, which focus on providing high performance collaboration between a numbers of computers to efficiently accomplish complex tasks. Also, continually optimizing self-organizing mechanisms are also important for future works, which would enable the proposed network infrastructure to be suitable for a large schedulable commercial network.

# 7. References

[1] J. Rosenberg – dynamicsoft, H. Schulzrinne - Columbia U., G. Camarillo – Ericsson, A. Johnston – WorldCom, J. Peterson – Neustar, R. Sparks – dynamicsoft, M. Handley – ICIR, E. Schooler - AT&T. "*SIP: Session Initiation Protocol*". The Internet Society, June 2002

[2] Sun Microsystem Inc, The Internet Society. JXTA V2.0 Protocol Specification: https://jxta.kenai.com/Specifications/JXTAProtocols2_0.pdf - last seen: 05/02/2014

[3] D. Howie, M. Ylianttila, E. Harjula, J. Sauvola. *"State-of-Art: SIP for Mobile Application Super-Networking"*. MediaTeam Oulu Group, Department of Electrical Engineering, Erkki Koiso-Kanttilankatu 3, FIN-90014 University of Oulu, Finland

[4] C. Zhu, G. Chang, W. Ning. "*A JXTA-Based SIP Communication System for P2P Wireless Network"*. IEEE 4th International Conference: Wireless Communications, Networking and Mobile Computing. 2008.

[5] Janice J. Heiss. "*JXTA Technology Brings the Internet Back to Its Origin*". Oracle Technology Network Articles. August 11, 2005.

[6] Marshall Brain, " *How Gnutella Works*", http://computer.howstuffworks.com/file-sharing.htm, last seem: 05/02/2014

[7] Ubiquity Software Corporation, "Understanding SIP". Ubiquity Software Corporation, 2003

[8] C. Plaxton, R. Rajaraman, A. Richa. *"Survey and Comparison of Peer-to-Peer Overlay Network Schemes".* IEEE Communications Surveys & Tutorials. Second Quarter 2005

[9] D. Bryan, B. Lowekamp. *"FEATURE: Q Focus: Session Initiation Protocol (pages 34 – 41): Decentralizing SIP"*. ACM Queue. 2007.

[10] D. Willis, D. Bryan, P. Matthews, E. Shim. *"Concepts and Terminology for Peer to Peer SIP"*. IETF draft-willis-p2psip-concepts-04. 2007. http://tools.ietf.org/html/draft-willis-p2psip-concepts-04

[11] "JXTA Java Standard Edition v2.5: Programmers Guide" Sun Microsystems, September 10, 2007

[12] Salman A. Baset and Henning G. Schulzrinne. *"An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol*". INFOCOM 2006. 25th IEEE International Conference on Computer Communications. 5 Dec 2004

[13] D. Bryan, B. Lowekamp, C. Jennings. *"SOSIMPLE: Towards a Serverless, Standards-based, P2P Communication System"*. IEEE First International Workshop: Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA). 15 June 2005.

[14] D. Yang, L. Niu, X. Wang. *"A P2P-SIP Architecture for Real Time Stream Media Communication"*. IEEE 9th International Conference: Hybrid Intelligent Systems. 2009.

[15] S. AHSON, M. Ilyas. *"Sip Handbook Services, Technologies, and Security of Session Initiation Protocol"*. CRC Press. 2008.

[16] ITU-T -- Telecommunication Standardization Sector of ITU. *"SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS -- Infrastructure of audiovisual services – Systems and terminal equipment for audiovisual services"*. H.323 (12/2009).

[17] D. Bryan, B. Lowekamp, C. Jennings. *"A P2P Approach to SIP Registration and Resource Location"*. IETF draft-bryan-sipping-p2p-02. 2006. http://tools.ietf.org/html/draft-bryan-sipping-p2p-02

[18] Z. Xu, R. Min, Y. Hu. *"HIERAS: a DHT Based Hierarchical P2P Routing Algorithm"*. IEEE International Conference: Parallel Processing. 2003.

[19] I. Stoica, R. Morrisz, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, H. Balakrishnan. *"Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications" (*pp. 149-160*)*. ACM SIGCOMM 2001, San Deigo, CA. August 2001.

[20] Ryan Toole, Vinod Vokkarane "BitTorrent Architecture and Protocol", 2006

[21] K. Singh, H. Schulzrinne. *"Using an External DHT as a SIP Location Service"*. Tech. Report CUCS-007-06, Columbia University. 2006.

[22] C. Jennings, B. Lowekamp, E. Rescorla, J. Rosenberg, S. Baset, H. Schulzrinne. *"Resource Location and Discovery"*. IETF draft-bryan-p2psip-reload-03. 2008. http://tools.ietf.org/html/draft-bryan-p2psip-reload-03

[23] H. Schmidt, B. Aksoy, F. Hauck, A. Kassler, *"How Well Does JXTA Fit Peer-to-Peer SIP?"*. IEEE International Conference: Communications, ICC '08. 2008.

[24] P. Houngue, R. Glitho, E. Damiani. *"A Novel Architecture for a Peer-to-Peer Session Initiation Protocol"*. IEEE Symposium: Computers and Communications (ISCC). 2010.

[25] J. Liang, R. Kumar, K. W. Ross. *"The KaZaA Overlay: A Measurement Study"*. Manuscript, Polytechnic University. 2004.

[26] Z. Wu, J. Zhou, W. Wang. *"The Protocol Design and Implementation of a JXTA Based P2P-SIP System"*. IEEE International Conference: Internet Technology and Applications. 2010.

[27] Sofia-SIP Library. http://sofia-sip.sourceforge.net/. Last Seen: 05/02/2014

[28] M. Junginger, Y. Lee. *"A Self-Organizing Publish/Subscribe Middleware for Dynamic Peer-to-Peer Networks"*. IEEE Communications Conference: Network. 2004.

[29] G. Fox. *"Peer-to-Peer Networks"*. IEEE Conference in Computing in Science & Engineering. 2002.

[30] 3GPP Specification Series – TS 24.229. *"IP Multimedia Call Control Protocol Based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)" Stage 3.* 2006.

[31] E. Halepovic, R. Deters, and B. Traversat. *"JXTA Messaging: Analysis of Feature-Performance Tradeoffs and Implications for System Design"*. In DOA'05, volume 3761 of LNCS. 2005.

[32] R. Fielding, UC. Irvine, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. *"RFC2068: Hypertext Transfer Protocol -- HTTP/1.1"*. 1997. http://tools.ietf.org/html/rfc2068

[33] D. Atkins, R. Austein. *"Threat Analysis of the Domain Name System"*. RFC 3833, Internet Engineering Task Force, 2004. http://tools.ietf.org/html/rfc3833

[34] M. Handley, V. Jacobson, C. Perkins. *"SDP: Session Description Protocol"*. IETF RFC4566. July 2006.

[35] B. Meyer, M. Portmann. *"Practical Performance Evaluation of Peer-to-Peer Internet Telephony Using SIP"*. IEEE 8th International Conference: Computer and Information Technology Workshops. 2008.

[36] C. Cheng, S. Tsao, J. Chou. *"Unstructured Peer-to-Peer Session Initiation Protocol for Mobile Environment"*. IEEE 18th International Symposium: Personal, Indoor and Mobile Radio Communications. 2007.

[37] S. Waterhouse. *"JXTA Search: Distributed Search for Distributed Networks"*. Sun Microsystems, Inc. 2001.

[38] E. Daly, A. Gray, M. Haahr. *"On Using Peer Profiles to Create Self-organizing P2P Networks"*. IEEE 6th International Symposium: A World of Wireless Mobile and Multimedia Networks. 2005.

[39] H. Kobayashi, H. Takizawa, T. Inaba, Y. Takizawa. *"A Self-organizing Overlay Network to Exploit the Locality of Interests for Effective Resource Discovery in P2P Systems"*. IEEE Conference: The 2005 Symposium on Applications and the Internet. 2005

[40] H. Schulzrinne, J. Rosenberg. "*A Comparison of SIP and H.323 for Internet Telephony*"