P2P SIP: Current Situation

Submitted in partial fulfilment
for the requirements of the degree of

Bachelor of Science (Honours) in Computer Science

at Rhodes University

Erasmus Tyapa

Grahamstown, South Africa
November 4, 2010

Abstract

Session Initiation Protocol (SIP) is one of the popular protocols used for the exchange of text, voice and video on Internet Protocol (IP) or next generation networks which happens in a client-server environment. The servers are used to provide centralised control of the entire network environment. While there are advantages for client-server environment, the servers create a single point of failure. This it not ideal for resource limited settings, such as in environments with limited Internet connectivity and infrastructure.

To avoid the use of centralised servers, the SIP community via the Internet Engineering Task Force (IETF), has been working on decentralising SIP by creating a Peer-to-Peer version of SIP called P2PSIP. In this thesis, we investigated the progress of this work. We also tested some of the implemented P2PSIP systems with the view of comparing how these systems have addressed various issues that need to be resolved before P2PSIP is declared a standard for decentralised SIP communication. We then went further to compare tested implementations using some of the designs decisions made in the P2PSIP working group. These comparisons helped us to choose two implementations, 39 Peers and SIP2P that can be used for research purpose, in particular within the Rhodes University Convergence research group.

ACM Computing Classification System Classification

ACM Computing Classification Thesis classification under the ACM Computing Classification System (1998 version, valid through 2010):

C.2.1 [Network Architecture and Design]: Packet-switching networks

C.2.2 [Network Protocols]: Applications (SIP)

C.2.0 [General]: Data communications

C.2.4 [Distributed Systems]: Distributed applications

C.3 [SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEM]: Real-time and embedded system

General Terms: DHT, SIP, P2P, Standardisation and P2PSIP

Acknowledgements

I take this opportunity to express my profound gratitude and respect to all those who assisted me throughout the work of this project. Working on my honours project was a daunting task. I am glad to have had encouragement, support, and advice of a great set of mentors, colleagues, and friends.

I want to start by thanking my supervisors, Prof. Alfredo (Alf) Terzoli and Mr. Mosiuoa (Mos) Tsietsi, for their help in guiding me with my research. Alf helped me understand how to do research while Mos helped me understand the technical parts of the research. I ascribe my little success to them. You gave me this opportunity to undertake the project and providing crucial feedback that influenced me and provided opportunity to undertake the project work in esteemed concern. I am also deeply thankful to Dr. Kundan Singhn (Columbia University) and Li.lichun for useful suggestions, gentle and soothing attitude and pointing me in the right direction this helped me a lot to learn and understand various concepts. I am also extremely grateful to my convergence research group colleagues (Ray, Walter, Grace, Shange, Kooper, Moses, Mathe and Zelalemss) at Rhodes University for advising and willing to help when ever I needed them. I must specifically thank Mathe for her fair criticism which brought the best out of me, for sacrificing her time to attend to my problems, thanks Mathe you are a superwoman.

I must also take this opportunity to thank SANTED for sponsoring my studies. This work would have been significantly harder to do without the funding from Telkom SA, Comverse SA, Stortech, Tellabs, Easttel, Bright Ideas Projects 39 and THRIP through the Telkom Centre of Excellence at Rhodes University. Thank you so much.

MeKauna, tangi unene (Thanks very much fo making it possible for me to study at Rhodes university and for your support).

Finally, I would like to thank my family. Your consistency in keeping me focused on my studies has been like a soccer referee refereeing a soccer game, my mother and cousins seemed to have a perfected the art of making me work on the most important tasks, either by sharing their wisdom, showing their love or by just reminding me that working hard always pays off.

Thank you guys!

Erasmus Tyapa

Table of Contents

List of Figures

List of Tables

Acronyms

AOR	Address of Record
CAN	Content Addressable Network
DHT	Distributed Hash Table
ICE	Interactive Connectivity Establishment
IETF	Internet Engineering Task Force
IM	Instant Messaging
IMS	IP Multimedia Subsystem
IPTV	Internet Protocol Television
JSAP	Jain SIP Applet Phone
NAT	Network Address Translation
PSTN	Public Switched Telephone Network
P2P	Peer-to-Peer
P2PSIP	Peer-to-Peer Session Initiation Protocol.
RELOAD	REsources LOcation And Discovery
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
SS7	Signalling Solution No.7
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extension
SIP	Session Initiation Protocol
STUN	Session Traversal Utilities for NAT
TURN	Traversal Using Relay NAT
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
VoIP	Voice over IP
XMPP	Extensible Messaging and Presence Protocol
3G	Third Generation
3GPP	Third Generation Partnership Project

Chapter 1

Introduction

The Session Initiation Protocol (SIP) was created to facilitate the set-up, management and tear down of multimedia sessions [53]. It is regarded as one of the most popular protocols for providing real-time multimedia services. SIP communication typically happens in a client-server environment, where centralised servers are used to control connected endpoints.

The centralised servers however create single points of failure. This makes SIP less suited for some environments. Consider for example wireless ad-hoc networks in extreme emergency scenarios (e.g. The Haiti earthquake), where infrastructure is limited but communication is required. Another example of P2PSIP usage is global P2P VoIP network such as open standard Skype like system that may connect with Public Switched Telephone Network (PSTN) and work with reduced number of infrastructures such as servers and maintaining reliability and security with minimal authorities. Such scenarios and the benefits of a decentralised architecture have motivated proposals to move the SIP architecture from client-server to Peer-to-Peer (P2P) architecture, where there are no centralised servers and single points of failure.

The Internet Engineering Task Force (IETF) [30] is working on a new protocol that combines SIP with P2P. This protocol is called P2PSIP [43] and its objective is to use a collection of intelligent endpoints to establish and manage sessions, rather than centralised servers as currently deployed in SIP. In this project, we explored the work done by IETF (through the P2PSIP IETF working group [43]) to standardise P2PSIP as a P2P alternative for the client-based SIP. We also scanned intensively the Internet searching for implementations that combine SIP and P2P.

1.1 Project Objectives

This project is an extension of a project which was carried out at Rhodes University during the early stages of the P2SIP IETF working group discussion on how to standardise P2PSIP. In this project a framework called OverCord [67] was developed to give P2P capability to a SIP user agent, the JAIN SIP Applet Phone (JSAP). The task of this project is to continue the investigation on the evolution of the standard. The specific objectives of the project are to:

- Analyse and compare the designs proposed in the drafts published by IETF.
- Identify open issues about P2PSIP that are still being discussed by P2PSIP IETF working group.
- Test, analyse, and compare some of the available P2PSIP systems.

1.2 Deliverables

There are two main deliverables from the work done in this project. The first deliverable is to provide an overview of the standardisation progress of P2PSIP in the IETF. The second deliverable is to provide a comparison of different P2PSIP implementations in order to recommend one that may be suitable for use in future research activities, in particular within the Convergence research group at Rhodes University. That is, recommend an implementation that seems to follow the same path being taken by the P2PSIP working group.

1.3 Document Overview

The rest of the thesis is organised as follows.

Chapter 2: In this chapter we will look at SIP. This chapter will help us understand how SIP is able to support distribution of proxies and registrar roles. To this end, our discussion will only focus on the SIP protocol design, SIP network structure, how SIP works and some of its limitations.

Chapter 3: In this chapter we will look at P2P networking. This chapter will help us understand what is going to be discussed in Chapter 4. The two classes of P2P, structured and unstructured, will be discussed to help us understand and identify which type of class is good to be used in SIP.

Chapter 4: In this chapter we will investigate the work done within the P2PSIP IETF working group. Essentially, this chapter will explore different proposals brought forward as part of P2PSIP standardisation.

Chapter 5: In this chapter we will have a look at P2PSIP implementations that we have tested during the course of this project.

Chapter 6: In this chapter we will analyse, compare and make recommendations on the P2PSIP systems that we have tested.

Chapter 7: This chapter will provide conclusions and discuss future work.

Chapter 2

Session Initiation Protocol

This chapter describes the basicSIP protocol as in RFC 3261. SIP was also adopted by the Third Generation Partnership Project (3GPP) as the IP-based multimedia call control protocol for Third Generation (3G) wireless networks, called IP Multimedia Subsystem (IMS). The adoption essentially made SIP as a common standard for session management, context exchange, and Instant Messaging (IM) in the new generation of mobile networks. In the following sections, the SIP protocol design and network structure will be explained briefly.

2.1 Basic SIP Protocol

The basic SIP protocol is text based and similar to Signalling System No. 7 (SS7) [38]. SS7 is a set of protocols used to set up PSTN telephone calls. A major difference between SIP and SS7 is that SS7 has dumb end points (standard telephone handsets) whereas in SIP, the end points are intelligent.

SIP was originally developed to manage negotiation and provide a rendezvous for session establishment on the Internet. It supports the following fundamental features: registration, session initiation, modifying, termination, and call redirection. SIP does not define the type of session that is being established. It only concerned about how it should be managed. Since its inception, many features have been added to SIP by different groups.

The Voice over IP (VoIP) community for example, added presence exchange and IM to SIP so that it can support IM and presence. Other applications and services that can be supported by SIP include collaborative gaming and Video on Demand (VoD) as well as voice, video and web conferencing.

2.1.1 Media Handling

Even though SIP is used to identify, locate and join parties who want to communicate using any media type, it does not transport the media itself. The media is transported differently by Real-time Transport Protocol (RTP) [55]. For the purpose of describing the media, SIP uses the Session Description Protocol (SDP) [23], which carries within it information about the session (namely: the type of media, the codec to use, and the protocol for actually transporting the media). SDP is usually used to describe multimedia communication sessions for the purposes of session announcement, session invitation and parameter negotiation.

2.1.2 SIP Addressing

SIP uses the Simple Mail Transfer Protocol (SMTP) [46] scheme for addressing purpose. SIP addresses have similar structure to an email address and comprises of two parts: the username and the domain separated by the symbol @ as Resource Identifier (URI). Example of SIP address is erasmus@sip.ru.ac.za.

The SIP address can be obtained from an online service provider in a similar way to creating an email account with Google or AOL. Some SIP services also allow one to create SIP addresses within the users own domain as well. An example of such as SIP services provider is OnSIP. OnSIP allow customers to use registered domain name of their companies to create SIP addresses free of charge.

SIP address is very important because it is the means for SIP based communication over the Internet. Just like an email address, SIP addresses follows the person it belong to, this is in contrast to a telephone number which is registered to a specific device. SIP user can also be reached by other users calling from the old telephone by having a regular telephone number that forwards to the SIP address. SIP service providers also provide a means to make a call to regular telephone numbers.

2.2 SIP Messages

The SIP design is similar to Hypertext Transfer Protocol (HTTP) [19] in that it uses the request/response transaction model. The transaction involve the client request that triggers the SIP method or function on the server and at least one response. Just like HTTP, SIP is a human readable text-based format and reuses most of the header fields, encoding rules and status codes of HTTP as shown in Figure 2.1.

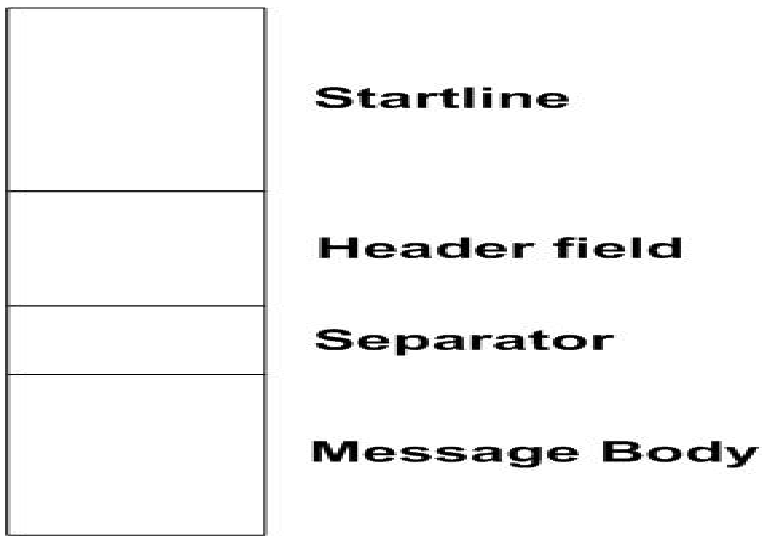


Figure 2.1: SIP Message Structure.

As shown in Figure 2.1, a SIP message consists of the startline, message header, and optional message body. The startline denote the beginning of a SIP message. A startline can be either a request or a response, and it contains a method name, the SIP URI to which the message is sent, and the SIP version number. The header field provides additional information about a message such as *To*, *From*, *Subject*, and *Via*. These fields are needed by clients and servers for routing the messages. The message body contains the information that must be passed to the receiver and is independent of the SIP protocol. The message body usually contains information either plain text message or multimedia message.

2.2.1 SIP Requests

The original SIP specification soecified in [24] included six types of request methods. These request methods are listed below. The response to these methods are listed in Table 2.1.

1. REGISTER: Used to register a SIP User Agent (UA) or client.

2. INVITE: Is used to invite a user or a service to a new session or modify parameters of the established session.

3. ACK: Used to confirm that the client has received a final response to an INVITE request.

4. CANCEL: Cancels any pending searches but does not terminate a call that has already been accepted.

5. OPTION: Queries the capabilities of servers.

6. BYE: Terminates a session.

SIP requests mentioned above and other additional methods defined by SIP protocol determine the different SIP functionalities. These functionalities are categorised into session-related functions and non-session-related functions [60]. The SIP methods or requests are mostly related to session-related functions, whereas additional methods such as *SUBSCRIBE*, *NOTIFY*, and *MESSAGE* are related to non-session-related functions.

2.2.2 SIP Responses

As mentioned in the above subsection for each request there is a corresponding response, Table 2.1 capture these responses.

Response class	Responses
Informational (1XX)	100 Trying, 180 Ringing, 181 Call is being forwarded, 182 Queued, 183 Session Progress.
Successful (2XX)	200 Ok, 202 Accepted
Redirection (3XX)	300 Multiple Choices, 301 Moved Permanently, 302 Moved Temporarily, 305 Use Proxy, 380 Alternative Service.
Client Failure (4XX)	400 Bad Request, 401 Unauthorised Registrars, 402 Payment required, 403 Forbidden, 404 Not Found, 405 Method Not Allowed, 406 Not Acceptable, 407 Proxy Authentication Required, 408 Request Timeout, 410 Gone (exists, but not available), 413 Request Entity Too Large,414 Request-URI too long, 415 Unsupported Media Type, 416 Unsupported URI Scheme, 420 Bad Extension, 421 Extension Required, 423 Interval Too Brief, 480 Temporarily Unavailable, 481 Call/transaction Does Not Exist, 482 Loop Detected, 483 Too Many Hops, 484 Address Incomplete, 485 Ambiguous, 486 Busy Here, 587 Request Terminated, 488 Not Acceptable Here, 491 Request Pending, 493 Undecipherable, 494 Security Agreement Required.
Server Failure (5XX)	500 Server Internet Error, 501 Not Implemented, 502 Bad Gateway, 503 Service Unavailable, 504 Server Time-out, 505 Version Not Supported, 513 Message Too Large.
Global Failure (6XX)	600 Busy Everywhere, 603 Decline, 604 Does Not Exist Anywhere, 606 Not Acceptable.

Table 2.1: SIP Responses. Adapted from [35].

2.3 SIP Functional Entities

Basically, SIP network consists of the following entities: SIP user agents [64], SIP registrar, SIP proxy, SIP redirect, SIP location, and SIP presence servers. These elements are have different roles.

End-user devices, such as IP phone, a wireless Personal Digital Assistant (PDA) or a soft phone are examples of SIP user agents. These devices, also called user agents, are used to establish and tear down sessions with other SIP user agents. SIP user agents are divided into two parts: the User Agent Client (UAC) and User Agent Server (UAS). The

UAC and UAS are not different in architecture but different in such a way that the UAC sends SIP requests messages and receive SIP responses messages while UAS receives SIP requests messages and sends SIP responses messages.

The proxy servers are the core of SIP networks. They make the SIP network to work in a client-server model. The proxy servers perform lookup, call routing, and forwarding of SIP messages and their responses. They also have other functionalities such as authentication, authorisation and accounting (AAA). There are two types of proxy servers: stateful and stateless. Stateful proxy servers remember current state of the SIP transaction and they are able to fork and allow retransmission of the message if an error occurs. Stateless proxy servers on the other hand do not keep any state regarding SIP messaging. The main benefit of stateful proxy servers is that it provides services that stateless cannot provide. For example, stateful proxy server can provide call forwarding busy but stateless cannot. However, stateless proxy is performance is better than stateful proxy server. This due to the functionality that stateful proxy perfomance such as re-transmission and generating billing reports.

The server entity in the SIP network that receives registration is called registrar server. This can be a database that contains the location of all UAs within a domain. UAs register their current location on a network with this server. Sometimes there is another server that is called SIP location server which might be running on the same server as the registrar. The registrar server in this case does not real store the UA location but just UA registration. Most of the time location server and registrar server are exist together. As a result, just refered them as registrar even though we might be talking about location server, registrar or both. The main purpose of location database is to map the domain to the Internet Protocol (IP) address of the device being used. The location database usually contains a list of IP addresses, usernames, and port numbers.

Another special server is the SIP redirect server which allows SIP proxy servers to direct SIP session invitation to external domains. SIP redirect servers may reside in the same hardware as SIP registrar server and SIP proxy servers, or exist separately.

SIP client's use Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) as a transport protocol and run the stack on default port number 5060 to connect to SIP servers and other SIP endpoints. Port number 5060 is for unencrypted traffic signalling while one can use port number 5061 for traffic encrypted with Transport Layer Security (TLS) [1].

SIP presence server is the SIP entity that accepts, stores, and distributes presence information to allow users to see the availability of other people they want to contact. The presence server has two distinct sets of clients:

1. Presentities (producers of information) provide presence information about themselves to the server to be stored and distributed.

2. Watchers (consumers of information) receive presence information from the server. Watchers can subscribe to certain users, much like IM users choose which “buddies” to add to their list.

2.4 SIP Services

SIP is the IETF protocol that helps initiate an interactive session. This interactive session usually involves multimedia elements such as video, voice, gaming, and virtually reality. In this section we will look at some of the service that SIP can provide.

2.4.1 Instant Messaging

IM refers to the transfer of messages between users in real-time. SIP for Instant Messaging and Presence Leveraging Extension (SIMPLE) working group [20] is an IETF working group that has extended SIP to include presence exchange and instant messaging information. The Instant messaging extensions are defined in RFC 3428 [13]. Just like with voice or video call, the SIP session in IM is established by SIP INVITE method, then the session participants can send instant message to each other using the MESSAGE method. When a participant receives the instant message, it sends back a confirmation message 200 Ok. Figure 2.2 illustrate a process of instant messaging between two users in the same domain and using the same proxy.

User1 forwards MESSAGE 1 to the server for User 2. The proxy receives this request, and recognises that it is destined for the same domain. It then looks up User 2 in its database (built up through registrations), and gets the binding information for the user. It then forwards MESSAGE 2 to User 2. The message is received by User 2, displayed, and a response is generated which is MESSAGE 3, and send to the proxy. The proxy strips off some header fields and send it to user as MESSAGE 4.

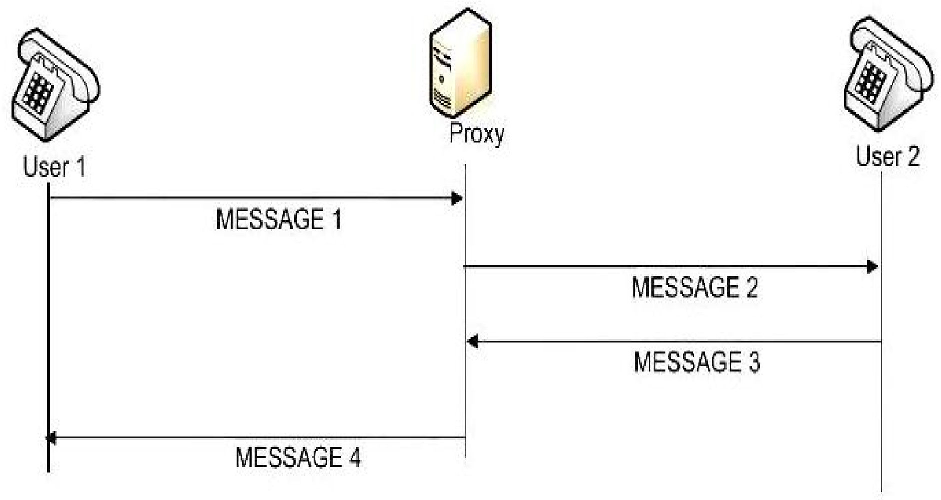


Figure 2.2: Example of Message Flow. Adapted from [60].

2.4.2 VoIP

The delivery of voice communications over IP network has grown tremendously over the past few years. Its growth is visible in many websites, with websites having a click-to-call button. VoIP [6] employ session control protocol to control the set-up and tear down calls. As mentioned above the establishment of SIP has brought the innovation of integrating voice with other Internet services. In VoIP, an audio codecs is used to encode speech to allow it to transmit over an IP network as a digital audio. Basically, the steps involved in VoIP telephone call are signalling and media channel setup, digitisation of the analog voice signal, optionally compression, packetisation, and transmission as IP packets over a packet-switched network. On the receiving side the reverse is done to reproduce the original voice stream.

2.5 SIP Operations

In this section we will consider few examples of SIP operations namely SIP registration, SIP session establishment, and a basic SIP call flows.

2.5.1 SIP Registration

As it was mentioned in Section 2.3, the registrar server is to collect information from the UA in order for other SIP components (e.g., proxy) to be able to manage sessions. Since initial IP information is only known by the client (UA), there is a need of a specific communication between the UA and the registrar in order to inform its address. This specific exchange is performed during UA start-up and on a regular basis in order for the registrar to have up to date information. By default the exchange refresh is done every hour, but this period can be adjusted. The registration phase between each UA and the registrar server is using a SIP specific method called REGISTER. Figure 2.3 shows an illustration of a registration exchange trace between IP phone at 146.231.124.30 and an asterisk server acting as a registrar at 146.231.124.96. The user (146.231.124.30) provides credentials which is checked against database by Asterisk using the configuration file *sip.conf*, in which each user has a matching entry. After the user is authenticated, the user is registered and a 200 Ok messages is sent back to the user.

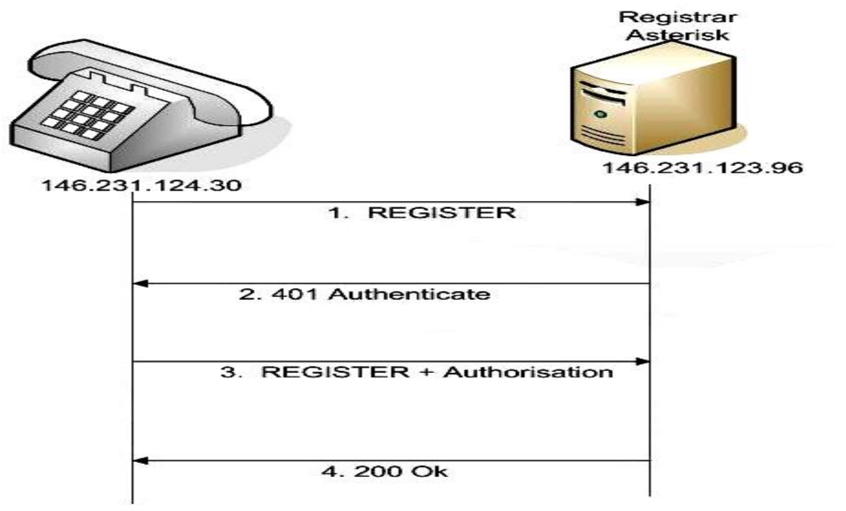


Figure 2.3: Simple SIP Registration Flow. Adapted from [60].

2.5.2 Basic SIP Session Establishment

The process to establish a session starts with an INVITE message, which is sent from a calling user (caller) to a called user (callee), inviting the callee to participate in a session. The caller might receive a number of interim responses (listed in Table 2.1) before the callee accepts the call. For example, the caller also informed that the callee is being alerted (the phone is ringing) by the proxy. When the callee answers the call, an OK response is generated and sent to the callee. The callee sends an ACK message to finalise the session establishment and after which media, such as voice, video, or text, start to be exchanged. When one of the users hangs up, a BYE message is generated and sent to the other client, which will confirm that the session is over and end the session. Figure 2.4 illustrate the whole process of session establishment.

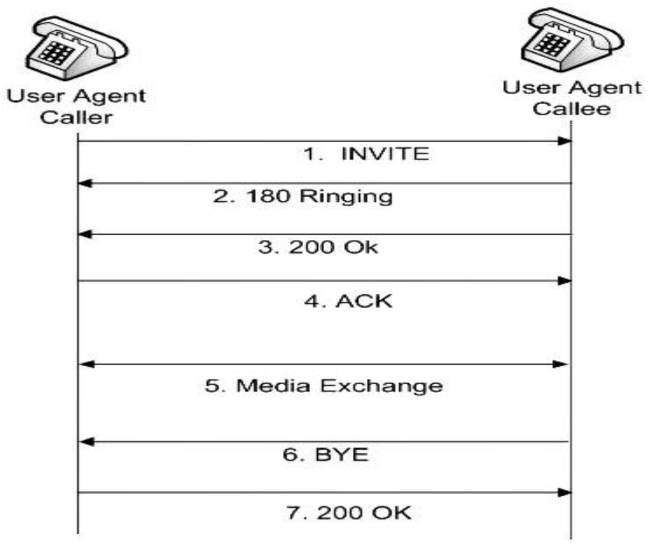


Figure 2.4: Typical SIP Session Establishment. Adapted from [53].

2.5.3 SIP Calls Flows

In a basic SIP call a caller sends a SIP INVITE request to invite the callee to establish a voice session. As can be seen from the Figure 2.4, the callee responds with a 180 status code message to indicate the phone is ringing. A response with 200 status code message is sent to the caller as soon as the phone is picked up as an indication to accept the invitation. An ACK is sent by the caller to confirm that the media exchange, which in this case is a voice session is established. Once the session is established, the actual digitised voice conversation typically transmits via a transport using Real-time Transmission Protocol (RTP) [55]. As as the conversation ends, a SIP BYE is sent followed by a response with a 200 status code to confirm the voice session termination.

Figure 2.5 shows a detail call flow through proxy for SIP based calls. UA 1 sends an INVITE message to the proxy to ask eras@sip.ict.ru.ac.za to participate in the session. The

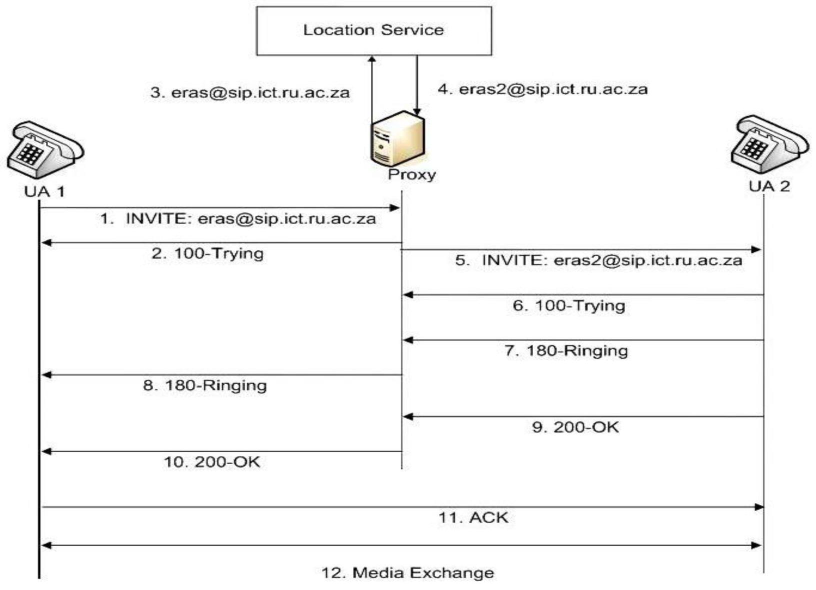


Figure 2.5: SIP Call Flow through Proxy. Adapted from [53].

proxy responds to UA 1 that it is trying and lookup for eras@sip.ict.ru.ac.za at location service. Since Eras has at least two SIP addresses and eras@sip.ict.ru.ac.za was unavailable, the proxy instead fetched and invited eras2@sip.ict.ru.ac.za, another SIP address for Eras. Using this address, response with SIP message 100-Trying and 180-Ringing are sent to the proxy. The proxy notifies UA 1 that UA (in this case eras2@sip.ict.ru.ac.za) is ringing, the UA 2 forwards a 200-OK message to the proxy as an indication that it wants to participate in a proceed sessions with UA 1. An acknowledgment message is then sent straight from UA 1 to UA 2 and the voice conversation starts between the UAs without the proxy being involved.

2.6 Summary

In this chapter we have seen SIP as a session initiation protcol for multimedia. SIP is based around entities called user agents, proxies, and registrars. Proxies and registrars are the entities used to find SIP URIs. The proxies only route messages without exercising any control. Basically, user agents use direct symmetric communication and have more intelligence than SIP proxies. However, user agents lack ability to perform lookup and route functionality in order to build a P2P network. Hence we need to a find a way to give the user agents ability to lookup and route SIP messages, so as to remove proxies and registrars from the SIP network.

Centralised call server in SIP network is subject to all shortcomings of client-server model of communications. There are also additional costs involved in the deployment of applications in smaller and ad-hoc environments. In actual practise, it also requires deployment, maintenance, and configuration redundancy. All this shortcomings are calling for SIP network without a centralised server.

Chapter 3

Introduction to Peer-to-Peer Networking

In P2P systems, information processing and storage is done by the nodes in the system, as opposed to one centralised server system. A P2P system is an optimal solution in deployment scenarios where the network needs to be made of ad-hoc devices [47]. The P2P systems are less costly and require fewer configurations. For example, there is no need to purchase servers, as data is stored among the peers. In this chapter will look at P2P network model.

3.1 Background

P2P is not a new idea. The Internet original design was very friendly to a P2P network [37]. The deployment of firewalls and the growth of asymmetric network links such as Asymmetric Digital Subscriber Line (ADSL) and cable modems as well as the spread of NAT are some of those issues that impact on our ability to create P2P applications in today's Internet.

Even though there are a lot of issues in today's Internet, there has been tremendous growth in P2P application in recent years, originally used for file sharing. Recently, the technology has greatly evolved and is being used for VoIP, IP Television (IPTV) [69] and distributed data storage. Even though the applications and services of P2P have increased in numbers, most of these applications are proprietary applications. On top of that, there are still security and latency problems in locating resources. P2P systems are generally more complex to implement than client-server based systems.

3.2 Overview of P2P Systems

The best way to understand a P2P network is to compare it to the client-server model as shown in Figure 3.1. A P2P network consists of peers that help each other with request processing unlike in a client-server model where a centralised device processes requests from clients. The fundamental distinction between client-server networking and P2P networks is an entity called servent, used in P2P networks. Servent is a term derived from the combination of letters from the term server ("serv-") and the term client ("-ent") [47]. The term refers to a node that performs the function of both server and client.

In a P2P network a peer will not necessarily provide service or data: some peers will provide service or store data but some just use services without contributing anything. We will look more on this in Section 3.5.

Many definitions seem to describe P2P based on the application aspect. One definition of P2P that is well suited to this thesis is the following:

"P2P systems are distributed systems consisting of interconnected nodes able to self-organise into network topologies with the purpose of sharing resources such as content, Central Processing Unit (CPU) cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralised server or authority" [60].

As we can see from Figure 3.1, nodes are connected to each other in P2P model while in client-server model nodes are connected to central devices. Before looking into P2P network in more details, we will look at some of the few concepts in P2P networks.

3.3 Concepts of P2P Overlay Network

Overlay networks refer to networks that are constructed on top of another network. P2P overlay network is defined as any overlay network that is constructed by Internet peers in

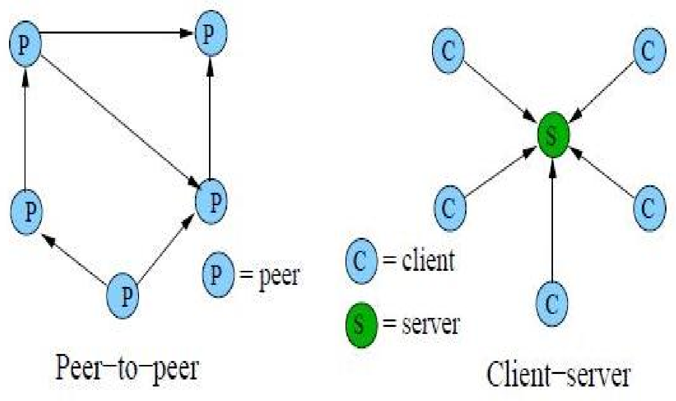


Figure 3.1: Client-server model vs P2P model. Source: [59].

the application layer on top of the Internet Protocol (IP) network [60]. The overlay networks are self-organising and nowadays often use Distributed Hash Table (DHT) protocol. There are three fundamental types of peers.

- **Simple Peers** are designed to serve a single end user. The objective is to allow a user to provide services from the device and consume services provided by other peers on the network. In most cases, a simple peer on a network is located behind a firewall. Hence, peers outside the firewall may in all likehood fail to directly communicate with the simple peer located inside the firewall. Simple peers have the least amount of responsibility in any P2P network because of their limited network accessibility. Unlike other peer types, they are not responsible for handling communication on behalf of other peers or serving third-party information for consumption by other peers. (see Section 3.5 for an example of simple peer).

- **Rendezvous Peers** provide peers with a network location to use to discover other peers and peer resources. Peers issue discovery queries to a rendezvous peer, and the rendezvous peer provides information about the peers it is aware of on the network. A rendezvous peer can augment its capabilities by caching information on peers for future use or by forwarding discovery requests to other rendezvous peers. These schemes have the potential to improve responsiveness, reduce network traffic, and provide better service to simple peers. A rendezvous peer will usually exist outside a private internal network's firewall. However, rendezvous peer could also exist behind the firewall, but such a case it would need to be capable of traversing the firewall using either a protocol authorized by the firewall or through a router peer outside the firewall.

- **Router (Relay) Peers** provide a mechanism for peers to communicate with other peers separated from the network by firewall or Network Address Translation (NAT) equipment. A router peer provides a connection for peers outside the firewall to be able to communicate with peers behind the firewall, and vice versa. To send a message to a peer via a router, the peer sending the message must first determine which router peer to use.

In P2P network, when a new node is joining the network it must get basic information to start up and publish its information about resources it holds. The node can then issue a query for the resources it wants. The destination node to which the query is issued is located using the P2P location protocol.

There are two classes of P2P networks, called unstructured P2P overlay network and structured P2P overlay network. For the purposes of selecting the class to use for SIP applications, it is important to understand how these classes differ from each other.

3.3.1 Unstructured P2P Overlay Network

In unstructured, or random, P2P overlay network, there is neither centralised control nor any control over the network topology or resource placement. Peers join unstructured networks by arbitraly selecting a peer as neighbor. Furthermore, for a new peer to publish its resources, the peer must store the resources or place them on a randomly chosen peer. As a result there are no guarantees for resource discovery: even though a resource may exist in the network, it may not be found.

In unstructured P2P network, the routing is mainly based on broadcasting and the search is based on keywords. The two fundamental routing operations in unstructured P2P are flooding and random walks. Flooding involves asking all nearby nodes, and having them asks their neighbors, until a result is either found or not found. With flooding, a search packet is used with a limited Time-To-Live (TTL) field. In contrast to flooding, random walks forwardes query to one randomly chosen neighboring peer. Even though, random walk has an advantage over flooding in terms of the number of messages sent per query, flooding is more robust and has better response times. Some examples of unstructured P2P overlay network are many traditional protocols like, DC++ [18], Gnutella [22], Napster [27] and KaZaA [36],

3.3.2 Structured P2P Overlay Network

In structured P2P overlays, the network topology is well controlled and resources are placed with peers in a deterministic way. This means that resources are placed at specified locations unlike the unstructured P2P overlay where resources are randomly placed among peers. In structured P2P overlay network, a DHT is used to solve the problem of storage, lookup and to arrange the peers in network. DHT manages the network in which resources location is placed deterministically. DHT is based on consistently assigning each peer a unique node identification (NodeID) value and the data objects (resources) are assigned unique identifiers called key. The P2P overlay network retrieves data objects or searches for them using a $<Key, Value>$ pair as shown in Figure 3.2. $Put<key, Value>$ is a store operation to store a key. $Value=Get<Key>$ is used to retrieve data objects corresponding to the key, which involves routing requests to the peer corresponding to the key. The P2P overlay network is built on top of routing algorithms such as Bamboo [25], Chord [26], Content Addressable Network (CAN) [50] or Pastry [54], all of which use DHT. However, there are also other algorithms in P2P structured overlay which are not based on DHT. An example of such algorithm is Mercury [5].

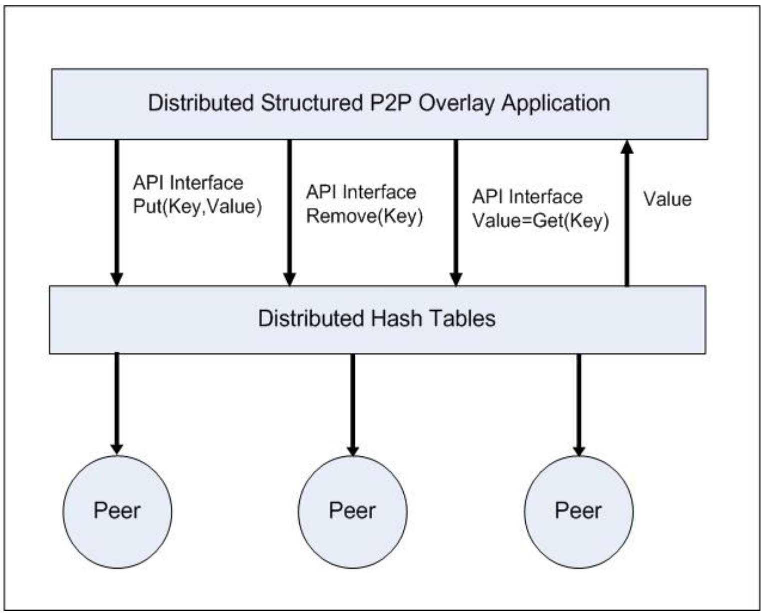


Figure 3.2: Application Interface for Structured DHT-based P2P Overlay. Adapted from [39].

3.3.3 Candidate P2P Overlay Network for SIP Applications

The exhaustive and non deterministic searches make unstructured P2P overlay not suitable for SIP applications. However, unstructured overlay present a low cost to build and maintain. Structured P2P overlay has an advantage over unstructured P2P overlays. Primarily because searching is efficient since broadcasting is not used. This makes it scalable. Overall, even though unstructured P2P has some advantage, Structured is most suitable for SIP applications.

3.4 DHT Implementations

DHT is a P2P algorithm offering *put/get* interfaces and developed to optimise the control of distributed network reliability and efficiency. It is based on structured Key Based Routing (KBR) and uses this to attain decentralisation and efficiency and reliability of nodes within structured P2P overlay network.

DHT provides lookup functionality to locate nodes in a similar fashion local hash tables are used to find memory locations in a computer. A keyspace (collection of all of keys in a space e.g. $[0,2^{160}]$) is used for allowing connected nodes to find the owner of any given key. The main advantage of DHT algorithms is that they are good at distributing resources with known names. Furthermore, they are scalable in a sense that they automatically distribute load when new nodes added. The fact that no central server is needed in DHT algorithms make them robust against node failure as data automatically migrated away from failed nodes except for bootstrap nodes. Bootstrap nodes are nodes that enable the initial discovery of other nodes by joining nodes. If bootstrap node fail, the data will be lost. DHT provide the three basic functions described in Subsection 3.3.2. These functions are summarised in Table 3.1. Table 3.2 shows different DHT algorithms and the complexity comparison between them. In table 'n' represent the number of nodes in the P2P network while 'd' a constant representing a dimensional Cartesian coordinate space.

Following subsections describe the two DHT implementations namely: Bamboo and Open Chord.

Function	Role	Description
Put(Value)	Join	Computes the value's key by hashing its contents, and sends it to the key's successor server for storage
Put(Value,Key)	Store	Stores or updates a signed value, used for root value.The Value must be signed with the given public key. The value's key will be hash of the key
Get(Key)	Search	Fetches and returns the block associated with specified key.

Table 3.1: Basic Functions of DHT.

Protocol	Routing Table Size	Searching Path Length
CAN	O (d)	O (d $n^{1/d}$)
kademlia	O (log n)	O (log n)
Pastry	O (log n)	O (log n)
Tapestry	O (log n)	O (log n)

Table 3.2: Comparison of DHT Algorithms.

3.4.1 Bamboo

Bamboo is an open source DHT implementation based on Pastry or a re-engineered pastry protocol. Bamboo is written in Java and was developed at the University of California, Berkeley [68]. Even though Bamboo is based on Pastry and uses the Pastry geometry. It does not use the same joining or neighbor management algorithms as Pastry. Geometry is a term used to refer to the pattern in which the neighbors are connected in the overlay network, independent of the routing algorithms or neighbor management. Compared to Pastry, the algorithm is more incremental which makes Bamboo better in withstanding membership changes in the DHT, especially in bandwidth-limited environments.

3.4.2 Open Chord

The Open Chord [15] is an open source DHT implementation based Chord DHT as described [62, 63]. It is a Java implemented DHT developed by the distributed and mobile systems group of Bamberg University [42]. Just like other DHTs, Open Chord solves the problem of efficiently locating the node that store particular data item. Furthermore, it solves the problem of load balancing because it uses consistent hashing each node essentially, receives roughly the same number of keys [62, 63]. Chord is full distributed, that means that no node is more important than any other. This improves robustness and makes Open Chord appropriate for loosely organised P2P applications. It has a ring overlay, in which each node is assigned a nodeID and values are assigned keys.

3.5 Classification of P2P Decentralisation

P2P network is calssified with distribution and decentralisation of resources storage, processing, information sharing and information control. There is a need to understand that P2P network does not necessarily mean that every peer in a network will provide service or store data. P2P network is classified according to the degree of decentralisation and how different peers in the network participate in providing services and store data. Classified into three group of decentralisation, namely: purely decentralised P2P, hybrid decentralised P2P, and partially decentralised P2P.

3.5.1 Purely Decentralised P2P

In purely decentralised P2P network, all nodes are equal in terms of performing certain task. There is no central coordination and every peer acts as a client and as a server. These peers are referred to as "servent" . This type of decentralisation purely decentralise P2P networks has disadvantages of data inconsistency, manageability and security. Examples of this include Gnutella and Freenet [16, 21].

3.5.2 Hybrid Decentralised P2P

In this type of network there is a central server that facilitate the interaction between peers and performs the lookups and identifies the nodes of the network, after that the interaction is between peers.

3.5.3 Partially Decentralised P2P

Partial decentralised P2P or super nodes in a P2P network distribute functionalities of central servers depending on which kind of the nodes participate in the distribution. The model in Figure 3.3 is the P2P overlay of nodes but not all nodes have equal capability (bandwidth, CPU, memory) and availability (uptime, public IP address) [58] than the other nodes. A super-node perform the duties of a server, it maintains the information for the joining nodes, and locates other nodes by communicating with other super-nodes.

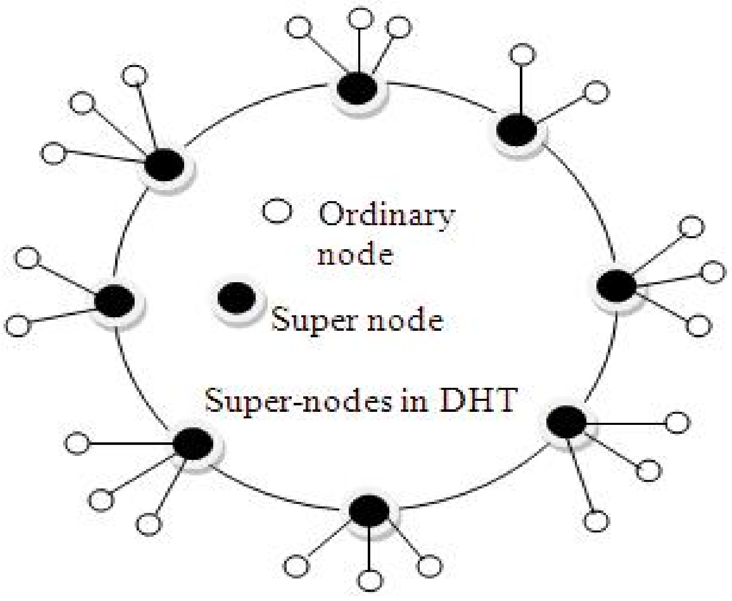


Figure 3.3: Super-nodes in DHT Distribution Model. Adapted from [56].

3.6 Examples of P2P Applications

P2P network is well known for file sharing but over the years it has evolved and include VoIP and IPTV. This happen because of the research and effort done to improve the problems that existed in file sharing applications such as security. Early example of file applications where BitTorrent [7], KaZaA and Napster. New applications like Damaka SIP P2P VoIP, SIPshare and Skype has proven that P2P applications are now more than file sharing. In the following section we prsent the newly dveloped P2P applications.

3.6.1 SOSIMPLE

Around the time Skype [29] was launched, a research project also early work in P2PSIP was started by by College of William and Mary. This work led to the SOSIMPLE application aimed at removing the central severs from SIP architecture. SOSIMPLE is a P2P system that provides VoIP and IM services. In this projects, a Chord based DHT is implemented and the traffic required is added as additional information in the SIP packets. Very few details exist regarding the implementation; the latest publication regarding this project is from 2005 and makes no statement about compatibility with standard clients even though as stated by Bryan and Lowerkamp, SOSIMPE is fully decentralised, standards-based P2P communication [9]. More information about SOSIMPLE cab found in [9, 10].

3.6.2 Skype

The first P2P system to implement VoIP in a decentralised distribution was Skype [29]. Skype stores user information in a decentralised fashion and is an overlay of P2P network with two types of nodes, ordinary nodes and super-node (see Figure 3.3). An ordinary node must connect to a super-node and must register itself with the Skype login server for a successful login operation. The login server is an important entity in the Skype network. Usernames and passwords are stored at the login server where authentication is done. Apart from the login server, there is no other central server in the Skype network. For the purpose of communicating with a PSTN there are SkypeOut and SkypeIn servers which provide PC-to-PSTN and PSTN-to-PC calls respectively. According to Baset and Schulzrinne [2], SkypeOut and SkypeIn servers are not part of P2P network as they do not play a role in PC-to-PC call establishment. Skype online and offline information is stored and propagated in a decentralised fashion and so are the user search queries. Skype is able to work across firewalls and NATs with least amount of infrastructure as it uses the Skype peers to route calls [48]. Currently, Skype is the most popularly used system, even though it uses proprietary protocol. Sometimes it may appear that the central servers are used for to locate resources (but this is very difficult to prove due to its proprietary nature).

3.6.3 IPTV

The popularity of P2P networking has grown tremendously to the level that it is now offering services such as IPTV. P2P-based IPTV systems are relatively new technology but their popularity have grown in the last few years. Nowadays, there are P2P video live streaming applications that have been successfully deployed on the Internet such as youTube [72]. IPTV has a special functionality for viewing TV with features such as recording, fast-forwarding and pausing a live. In addition, this IPTV is a two-way communication for interactive television. The most common and successfully deployed P2P IPTV applications are PPLive, PPStream, SOPCast, and P2P IPTV. We will look at more P2P applications that are related to VoIP in Chapter 5.

3.7 Summary

P2P networks and currently applications have gained popularity in the last decade. As result it is obvious that the majority of Internet traffic today is caused by P2P applications. As discussed in this chapter, these applications have evolved from simple file sharing to complex services like IPTV and VoIP. Part of this evolution can be attributed to research efforts in developing DHTs.

Chapter 4

P2PSIP Standardisation Progress

P2PSIP has been a topic of discussion, debate, and development for sometime. It collapses some of the more complex server functions into the UAs themselves and relies on the SIP philosophy that the intelligence in communications solutions should reside in the endpoint. P2PSIP origins are in academia. Some of the initial papers were published by Bryan and Lowekamp [10] and Singh and Schulzinne [59] in 2005. P2PSIP has been discussed and debated for the past years inside and outside IETF, where a P2PSIP working group was formed in March 2005 as a follow-up on the Columbia University projects SIPpeer [58] and the SOSIMPLE project at William and Mary College. The initiators of P2PSIP claim higher robustness against failure as well as easier configuration and maintenance as the main motivation for P2PSIP [49]. P2PSIP distributes registration, location and lookup steps of SIP. It handles three functions:

1. Registering a user with the P2P overlay network.
2. Looking up a user in the P2P overlay network (when a call to a user is made).
3. Dynamically sharing information when peers join and leave, so that the load is balanced across peers, and so that the sudden loss of one or more peers does not cause the P2P network to lose track of its current registrars.

The primary aim of this chapter is to discuss the progress of P2PSIP standardisation in the IETF.

4.1 P2PSIP IETF Working Group

The P2PSIP working group is the group in charge of the standardisation of the decentralised SIP architecture. The group main task is to define a P2P based VoIP communication that uses SIP. Moreover, it addresses issues such as security and privacy in a P2P communications network. In short, the mandate of the group is to incorporate P2P into the client-server based SIP and form P2PSIP that is not tied to only one organisation like Skype. (Essentially, P2PSIP is an open standard's answer to Skype.)

Any person with interest in P2PSIP can join the group and follow up what is going on. Group members propose different ideas, which, once accepted and mature, are formulated in drafts. Group members can read the drafts at the working group website [43] and the discussion on the issues about the proposed designs are carried via a mailing list. .

The charter of the IETF P2PSIP working group [65] outlines the following as primary tasks of the group:

- Producing an overview document explaining concepts, terminology, rationale, and providing use cases for the remaining work.
- Writing a proposed standard for the P2PSIP peer protocol.
- Writing a proposed standard for the P2PSP client Protocol, the protocol used between a P2PSIP peer and a P2PSIP client. (The terms peer and client will be explained later).
- Writing statements that will address how the previously defined protocols, along with existing IETF protocols can be used to produce systems to locate a user, identify appropriate resources to facilitate communication (such as media relays), and establish communications between the users, without relying on centralised servers.

4.2 Main P2PSIP Concepts

The main concepts used in P2PSIP are defined in the concepts draft [11]. This section describes the most important concepts to the work done in this research.

- **P2PSIP:** It is the set of protocols that extends SIP for P2P. It only includes two protocols: the P2PSIP Peer Protocol used between P2PSIP Peers, and the P2PSIP Client Protocol used between a P2PSIP Client and a P2PSIP Peer.
- **The P2PSIP Overlay:** This refers to a network of nodes that participates in data distribution and provides SIP registration, SIP request routing, and other services.
- **A P2PSIP Peer:** It is a node participating in a P2PSIP overlay that provides storage and routing services to other nodes in the same P2PSIP overlay. A P2PSIP peer can be located behind NATs and still be fully functional. It can perform several operations like joining and leaving the overlay, routing requests within the overlay, storing information, inserting information into the overlay and retrieving information from the overlay.
- **A P2PSIP Client:** It is a node participating in a P2PSIP overlay that does not store resources, run the distributed database algorithm, and is not involved in routing messages to other peers or clients. A P2PSIP client is like a simpler peer. A client insert, modify, examine, and remove records by interacting with a peer of that same overlay.
- **P2PSIP Peer Enrollment:** This refers to initial one-time process a P2PSIP peer follows to get an identifier and credentials for a given P2PSIP overlay. The process is done outside the P2PSIP overlay and is only needed at regular intervals or when the P2PSIP peer looses its identifier or its credentials.

4.3 Distribution Model in P2PSIP

The ultimate aim of P2PSIP is to get rid of centralised proxies and registrars in SIP and distribute their functionalities among the participating nodes. There are different ways of distributing these functionalities depending on the kind of nodes participating in the distribution. The Partial [8] decentralised P2P model described in Subsection 3.5.3 has been chosen by the P2PSIP working group as a distribution model for P2PSIP. In this case a super-node is a P2PSP peer and it performs duties of SIP registrars and proxies, it maintains location information for the joining node, and locates other users by communicating with other P2PSIP peer. The main reason to use a super-node (P2PSIP peer) distribution model is to enable nodes behind NAT to connect to a P2PSIP overlay. Singh [56] states that a node with low bandwidth connection to the Internet or those behind a firewall or NAT may not be able to fully function in a DHT because it may need in-bound connections, significant bandwidth for forwarding P2P messages or significant memory or CPU to process DHT functions. The P2PSIP client and P2PSIP peer distinction takes place only in when the node has joined the DHT. Basically a P2PSIP node enrol in the P2PSIP overlay, and then acts as either a client or a peer depending on the node capability and availability. This essentially addresses the issue that the group was discussing, on wether there is a need of separate credentials for peers and clients, and as it appears, there will be no separate credentials.

The P2PSIP overlay is required to internetwork with conventional SIP networks. The super-node in the DHT model solves the problem of inter-domain connectivity by letting each domain have at least one P2PSIP peer. These P2PSIP peer connect with each other to form upper layer overlay, which provide help when communication is needed between peers in different P2PSIP domain.

The difference in capabilities between super-nodes and ordinary nodes is very similar to the way in which the P2PSIP charter [65] introduces a P2PSIP peer and P2PSIP client. The idea of this model is to select weaker and unstable peers for the lower layer, to make the system more scalable and guarantee peer or resources lookup in the higher overlay. Another model that has been accepted by the working group is the pure P2P model. This model has many issues that need to be solved such as security and NAT drawbacks.

P2PSIP message flow in the overlay network should comply with a few routing styles. Each associated with disadvantages and advantages. We will look at some of the possible routing styles in P2PSIP overlay in the following section.

4.4 Routing Methods

Choosing the routing algorithm is not just a question of the number of hops taken by the message but also the efficient functioning of the algorithm in the overlay. In addition, security threats such as Denial of Service (DoS) attacks from compromised peers must be considered. Zheng and Vladimir [71] elaborated on the way messages can flow in the P2PSIP overlay: iterative, recursive and semi-recursive.

In iterative routing (see Figure 4.1) the source peer (S) is redirected by each intermediate peer to the destination peer (D). Basically, in iterative routing the peer receiving the request from the source peer, replies by suggesting address of a nearer peer rather than forwarding the message. The source peer then sends a new request to the suggested peer, repeating until the target reached. The source peer is able to check the validity and correctness of each response. It might be implemented in security sensitive environment. However, this solution does not provide guarantee for NAT traversal when the destination peer is behind NAT protection.

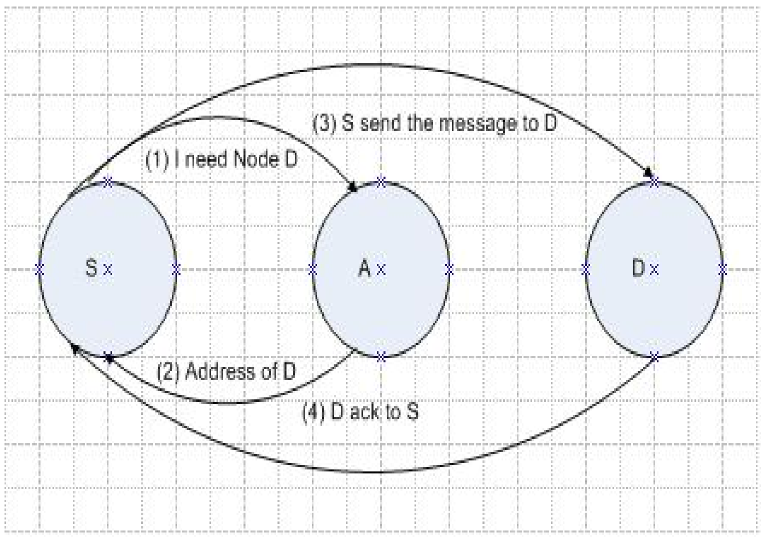


Figure 4.1: Example of Iterative Routing. Adapted from [71].

In recursive routing (see Figure 4.2) the source peer sends a message to the nearest peer in the path to the destination, if the peer is not the target, the message is forwarded to the next nearest peer and the process repeats until the target is reached. Recursive routing has little trouble with NAT traversal; however, it might cause long delay due to excessive message flows. This happens as a result of intermediate peer sending the message to the peer that it think is near the destination peer, causing some delay as the message is being exchanged as shown in Figure 4.2. Therefore, this approach is only recommended for high capability overlays (e.g. more CPU processing power, high bandwidth, etc.).

In semi-recursive or symmetric recursive routing (see Figure 4.3), the request message is forwarded by intermediate peers hop by hop to the destination, while the response is directly returned. Symmetric recursive routing has no problem with NAT traversal and system delay. Also, it provides better security than recursive routing since the response is directly forwarded to the source peer. This approach can be implemented in environment that needs NAT traversal, lower latency, and better security.

Message delay and message delivery rates are not the only aspects to be considered when choosing between routing algorithms but also the presence of NAT which is not friendly for connection establishment between two end-points [71]. When we look at REsource

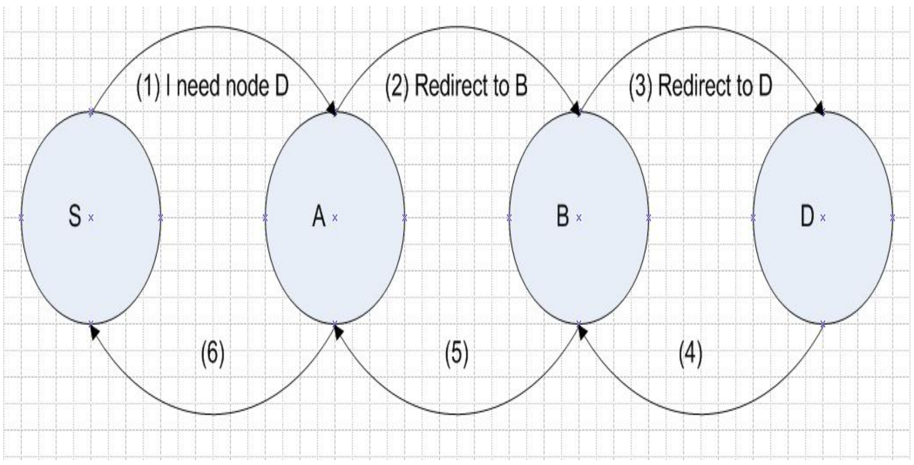


Figure 4.2: Example of Pure Recursive Routing. Adapted from [71].

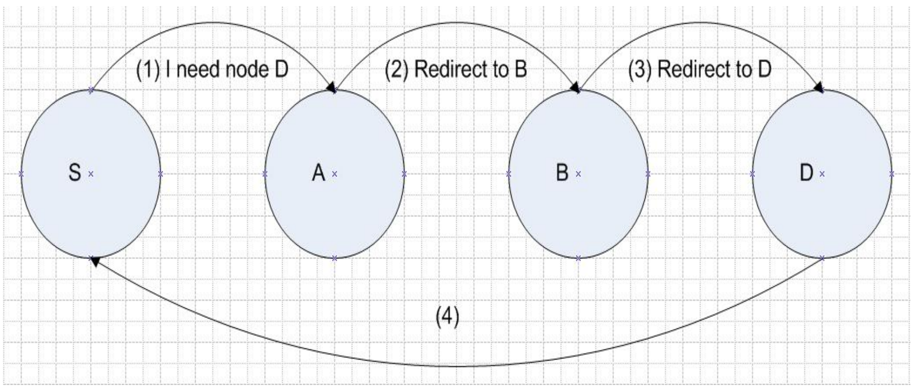


Figure 4.3: Example of Symmetric Recursive Routing. Adapted from [71].

LOcation And Discovery (RELOAD) network in Subsection 4.8, we will see which of the three types of different routing is more appropriate for P2PSIP.

4.5 DHT Choice

Each DHT define some specific message routing type. Hence the choice of the DHT must be selected looking at some of the advantages and disadvantages mentioned in the previous section.

The P2PSIP working group has been debating which DHT is more appropriate for P2PSIP. Some people in the working group were rather interested in choosing one DHT while others were for multiple DHTs. The P2PSIP working group agreed that multiple DHTs seemed a better option, but they have not indicated which DHT algorithms are part of that multiple DHTs. In terms of writing drafts, they have decided that the best solution is to specify the DHT algorithms for testing but this does exclude other DHT from being used. This means that the selection of DHT algorithms is left to the developer. That said, not all the DHT algorithms will applicable to be used in P2PSIP. The working group will decide on which DHT algorithms will be used in P2PSIP. So far, only Chord is recommended for use. This choice of multiple DHTs algorithm allow P2PSIP peer protocol to be extensible to accommodate different overlay technologies such as Bamboo, Pastry, Kademlia, Chord and others, including some future algorithms that may appear.

4.6 Protocol Layering for P2PSIP

The debate on which model for protocol layering is the most appropriate for P2PSIP has been debated in the P2PSIP working group and different proposals were brought forward to support the notion. The P2PSIP working group has chosen one of the two architecture of combining SIP and P2P, proposed by the Singh [56] for P2PSIP telephony. The two architectures are: P2P-over-SIP, SIP-using-P2P. The two architectures will be described next.

4.6.1 P2P-over-SIP

In P2P-over-SIP architecture, SIP messages are not used only for registering users, resource lookup, and establishing session, but also for maintaining a P2P network. Although SIP was not designed with P2P in mind, the design is extensible. To cater for P2P traffic SIPPEER [58] and dSIP [31] have proposed extensions of SIP. In this proposal, SIP REGISTER requests are used to join, build, and pass information between peers. However, tunnelling all P2P messages over SIP REGISTER causes high overhead. P2P-over-SIP architecture is not flexible as it is not interoperable with any other P2P applications without requiring them to implement a SIP stack. Because of the many drawbacks mentioned, the P2P-over-SIP has lost its popularity and was not chosen.

4.6.2 SIP-using-P2P

The problem of overloading SIP with further functionality it was not designed to do e.g., maintaining a P2P/DHT network, can be solved by SIP-using-P2P. SIP-using-P2P architecture uses two separate stacks: a SIP layer for registering users, resource lookup, establishing session and a P2P layer for maintaining a distributed network. Therefore, it diminishes the application overhead and complexity. The SIP stack and P2P stack in P2PSIP applications can be implemented on the same or different nodes. In other words, P2PSIP application can implement DHT on its own, or deploys an external DHT service. Singh and Schulzrinne [59] propose using OpenDHT [28] as a SIP location service. They use a partial P2P architecture where OpenDHT node acts as a server offering a storage service to other clients who does not support P2P functions. As shown in Figure 4.4 (a), DHT layer and SIP layer are clearly separated in SIP-using-P2P. In SIP-using-P2P the P2P wire protocol is independent of SIP, and the SIP entities just use the services provided by P2P layer, e.g., storage, lookup and routing. This made SIP-using-P2P as the appropriate choice by the working group. There are many proposed designs that combine SIP and P2P using SIP-using-P2P, such as RELOAD which will be in Section 4.8.

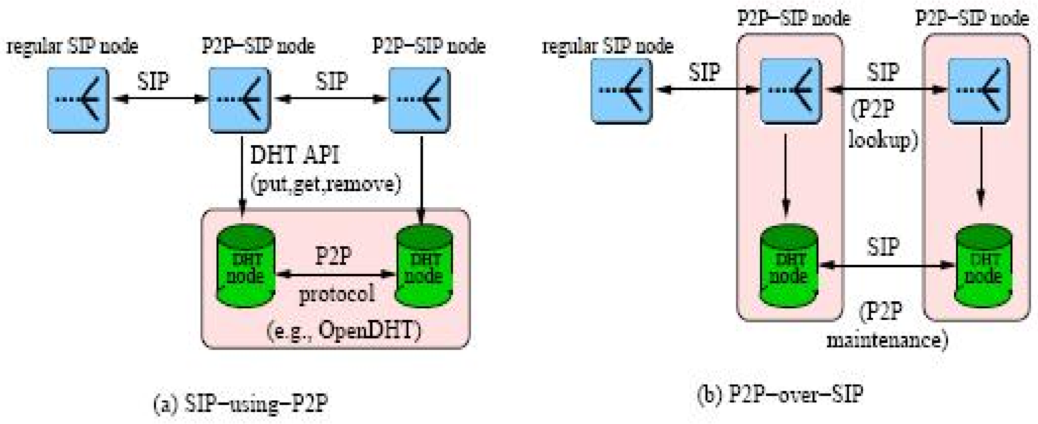


Figure 4.4: SIP-using-P2P vs. P2P-over-SIP Architectures. Source: [56].

In the next section, we will now clarify the structured overlay networks using Chord as it is the most used protocol in the most proposals for P2PSIP.

4.7 Chord-based P2PSIP Overlay

The problem with P2P applications is locating node that stores the desired data item. In P2P network, a peer might use flooding to forward a request for certain data to a destination peer. The problem with flooding is that traffic generated may inhibit the network from retrieving the requested data. Chord [62, 63] offers a solution to this problem, Chord is a P2P lookup algorithm used to map a key to a specific node.

It has been demonstrated [62] that Chord can solve any given lookup by sending information to a maximum of *O(log N)* nodes, where *N* represent the number of nodes in the system. This means that even when *N* is a large number, the number of messages sent by a node using Chord will still be relatively small.

Chord nodes are arranged on a ring topology. On the Chord ring, each node has a successor and a predecessor [62, 63]. Since P2P nodes join or leave the network freely, nodes maintain multiple successor pointers to improve robustness. In Figure 4.5, 4.6 and 4.7 we can see examples of a Chord ring.

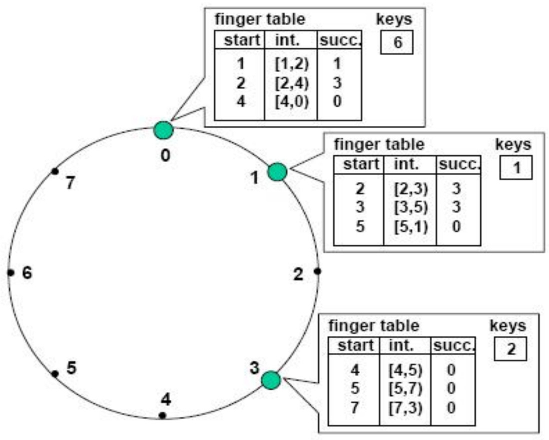


Figure 4.5: Finger Tables and Key Locations for a Network with Nodes 0, 1, and 3, and Keys 1, 2, and 6. Source: [62].

In Figure 4.5, Nodes 0, 1 and 3 are connected while Nodes 2, 4, 5, 6 and 7 are not. The size of the network is 8 and the currently available files/keys map to Nodes 1, 2 and 6. Since node 6 does not exist, the file (key=6) is mapped to the first available node, in this case Node 0. The file (key=2) is mapped onto Node 3.

Chord continually maps the files along the Chord ring as peers join and leave the network. For instance, in Figure 4.6, when Node 6 joined the overlay, the file (key=6) is stored in that node, and also the successors of the nodes vary according to this new topology [63]. In Figure 4.6, when Node 1 leaves the overlay, the finger tables are adjusted, and the file (key=1) is taken by Node 3 as shown in Figure 4.7. Another feature of Chord is that it uses two routing modes, iterative and recursive [63].

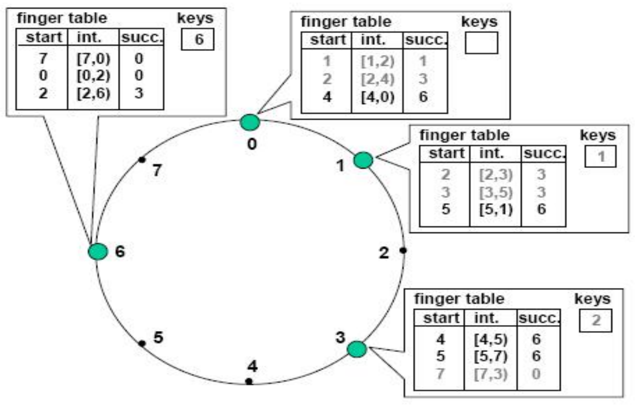


Figure 4.6: Finger Tables and Key Locations after Node 6 Joins the Network. Source: [62].

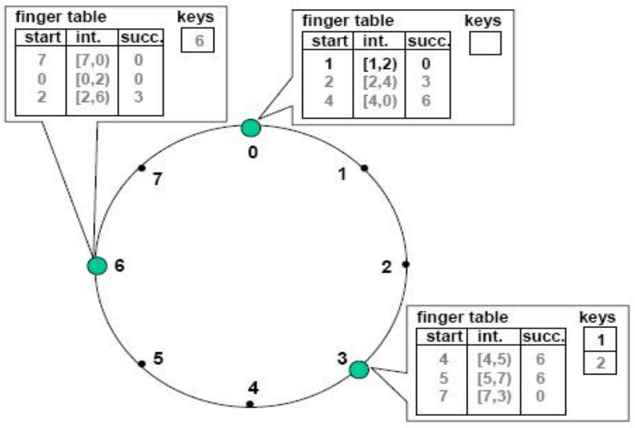


Figure 4.7: Finger Tables and Key Locations after Node 3 Leaves the Network. Source: [62].

4.8 REsources LOcation And Discovery

In 2007, there were many competing proposals for the P2PSIP peer protocol; RELOAD, Address Settlement by peer-to-peer (ASP) [33], Service Extensible P2P Peer Protocol (SEP) [34], Extensible Peer Protocol (XPP) [41] and Host Identity Protocol (HIPHOP) [17]. In February 2008, Peer-to-Peer Protocol (P2PP) [3] was merged to the combined RELOAD/ASP protocol.

In 2008, RELOAD [32] was adopted by the P2PSIP working group as its starting point for the primary P2PSIP protocol. RELOAD can be used for other P2P applications since it has two separate stacks, a SIP stack and P2P stack. RELOAD has been designed with an abstract interface to the overlay layer to simplify implementing a variety of structured

(DHT) and unstructured overlay algorithms. This promotes interoperability and selection of overlay algorithms optimised for a particular application.

4.8.1 RELOAD Architecture

Figure 4.8 shows the basic architecture of RELOAD. Each application that wishes to use RELOAD defines a RELOAD usage; examples include a SIP Usage, Extensible Messaging and Presence Protocol (XMPP) Usage or any other. These usages all talk to RELOAD through a common Message Transport API. The applications can use RELOAD to store and retrieve data, as a service discovery tool or to form direct connections in P2P environments. Currently defined usages are the SIP usage, the certificate store usage, the Traversal Using Relays around NAT (TURN) [40] server usage and the diagnostics usage.

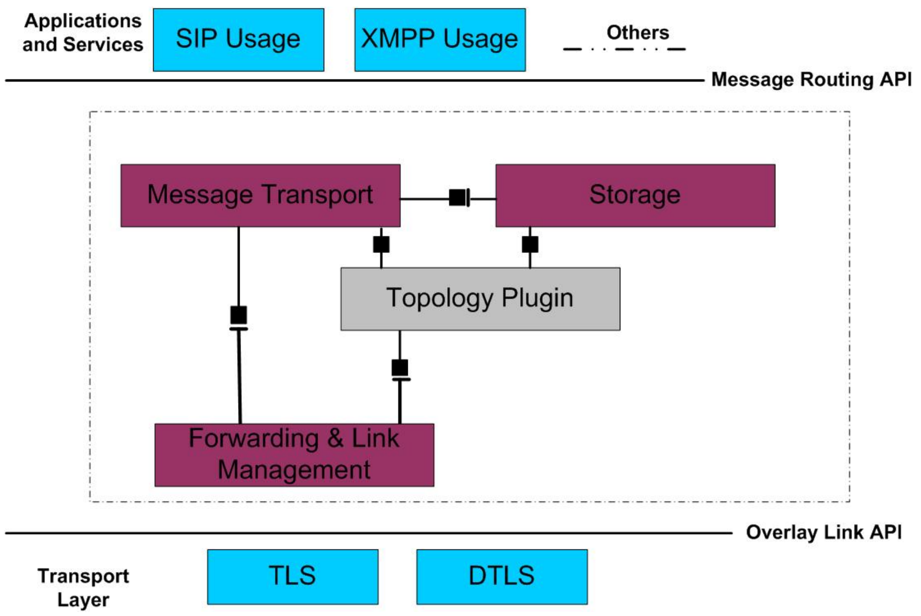


Figure 4.8: Major Components of RELOAD. Adapted from [32, 51].

The *transport layer* provides a generic message routing service for the overlay that is sending and receiving messages from peers. The *storage* component is responsible for processing messages relating to storage and retrieval of data.

RELOAD is specifically designed to work with a variety of overlay algorithms. However, for interoperability reasons, RELOAD defines one algorithm, Chord, that is mandatory to implement. The *topology plugin* defines the content of the messages that will be used in RELOAD, the various procedures to join and leave an overlay, the hash algorithm used (the default algorithm used is Secure Hash Algorithm 1 (SHA1), the replication strategy and the routing procedures. *Forwarding and link management layer* provide packet forwarding services and help setting up connections across NATs using Interactive Connectivity Establishment (ICE) [52]. Currently TLS over TCP and Datagram Transport Layer Security (DTLS) over UDP are the link layer protocols supported by RELOAD for hop-by-hop communication.

4.8.2 RELOAD Network

The RELOAD Network is very similar to any other P2P network. One of the differences is that not all members of the network are peers, some are clients. A node might act as a client simply because it does not have the resources defined in [32] or does not implement the topology plug in required to act as a peer in the overlay. Still, a client uses the same RELOAD protocol as the peers, knows how to calculate Resource-IDs and signs its requests in the same manner as peers. RELOAD is going to have a credential and an enrollment server too. The routing mode in RELOAD is symmetric recursive, which is similar to recursive routing. Iterative routing is not possible since a message may need to be sent to a peer that is not present in the routing table, which requires a new direct connection to be established, making latency too high for the communication to be efficient. The pure recursive routing cannot be applied either for similar reasons: if a node behind a NAT receives a message response that has requested a long time before, the NAT will drop the message, making the communication impossible.

4.8.3 Enrollment and Bootstrapping

In RELOAD, a new user may join the P2PSIP overlay by using a central enrollment sever, from which the new peer may learn about the overlay network including the list of bootstrap peers. The bootstrap peer is not necessarily a super node. The parameters about the overlay networks are encapsulated into an XML file and sent from the enrollment server to the new peer. Based now on the overlay parameters, the new peer chooses the corresponding handling mechanism. The enrollment server also informs the new peer of policy of being a peer in the overlay. The bootstrap peers are just a collection of static peers collected by the enrollment. Their objective is to assist the joining peer to find its neighbors (successors) in the overlay. The bootstrap peer is the first contact point for a joining peer.

4.9 Closed and Open issues

Table 4.1 shows closed and open issues in the P2PSIP working group.

	Closed	Open
P2PSIP peer and client protocol	✓	✗
Distribution Model	✓	✗
Support of multiple or single DHT	✓	✗
Security	✗	✓
RELOAD support for P2P live streaming and Vod Services	✗	✓
NAT Traversals	✗	✓

✓: Represent yes is closed or open. ✗ :represent no is not closed or open.

Table 4.1: Table showing open and closed issues in the P2PSIP working group.

In [67] the work in the IETF in terms of P2PSIP was summarised. In this section we will look at break-throughs in terms in the IETF working group since the conclusion of the OverCord project. The group was debating on the necessity of the P2PSIP client protocol, some section of the working group felt that P2PSIP client is not necessary at all and was pushing for clients and peers to use the same protocol. The working group decided that the P2PSIP client protocol rather be optional. Client protocol is important in terms of small devices like cell phone which have low bandwidth and storage capacity.

The layering of the two main protocols (SIP and P2P) has been debated extensively. This debate was addressed in Section 4.2. The P2PSIP working group has agreed that the distribution models to be used in the P2PSIP is the super-node in DHT model and pure P2P model. The P2PSIP protocol must support multiple DHTs. The P2PSIP peer protocol should be extensible to accommodate different overlay technologies, including future algorithms that may appear. Even though the working group had shown its support for multiple DHTs, not all DHT algorithms will be supported. So far, Chord is the only DHT algorithm recommended by the working group and Chord is not good as other DHT algorithms in some ways. Therefore, the P2PSIP working group needs to recommend more DHT algorithms.

Many issues in P2PSIP seem to be addressed, making the working group to adopt the primary P2PSIP protocol called RELOAD. RELOAD protocol has benefited from the other proposals that was brought forward before it. Some issues that were addressed in those proposals were taken and adopted into RELOAD. The working group is now focusing on improving RELOAD for better. RELOAD needed to be improved to be suitable to be leveraged for both P2P live streaming and VoD services. As for now the evaluation has proved that DHT or RELOAD is not suitable for the chunk discovery in P2P streaming especially the live media streaming. Another issues being debated, is the issues of security and NAT traversals, where NAT solution is proposed in a draft [12], this draft is under Host Identity Protocol (HIP) working group. The only question is whether the P2PSIP working group need to or will it have time to adopt HIP in future as HIP has not been deployed widely. At the moment the issues of NAT is being addressed the same way it is addressed in conventional SIP.

On the security issues, RELOAD security model is based on each node having one or more Public Key Certificates (PKC). It based on levels, namely: Connection level. Message level and object level. Draft [14] propose a way to address the security at mentioned levels.

4.10 Summary

The advanatges of a P2P architecture attraction has led to its adoption for SIP, which led to the formation of the P2PSIP working group in the IETF. In this chapter we have discussed on going work in the P2PSIP IETF as well as looking at routing methods. The work in working group has led to the primary adoption of RELOAD as P2P signalling protocol and currently lot of effort is being put in improving RELOAD. The work in the P2PSIP working group is still going on, efforts has been put into security and defining different usages of RELOAD.

Chapter 5

Available Implementations

As already suggest, P2PSIP is a relatively new technology that emerged few years back. Hence there are just a few implemented P2PSIP systems so far. The majority of implemented P2PSIP systems are proprietary and this makes them difficult to use for academic research purpose example of such systems is Damaka SIP P2P VoIP. In this chapter we are going to discuss these systems and how we tested them.

5.1 OverCord

OverCord [67] is a product of such research, it is a P2P framework that was developed as a part of a Master's thesis at Rhodes University. OverCord is a Java implemented framework based on a SIP-using-P2P (see Subsection 4.7.2). The OverCord framework separates the SIP and P2P like as shown in Figure 5.1.

At the top most layer of the OverCord framework is SIP applications and other distributed services. The lower layer is the transport layer. It consists of UDP, TCP and other optional transport protocols such as DTLS and TLS. The middle layer is the P2P layer, which consists of a resource database, discovery, plugin management, overlay plugin and overlay repository layers. The OverCord framework again demonstrates the importance of combining P2P and SIP using the SIP-using-P2P model which the P2PSIP working group has adapted.

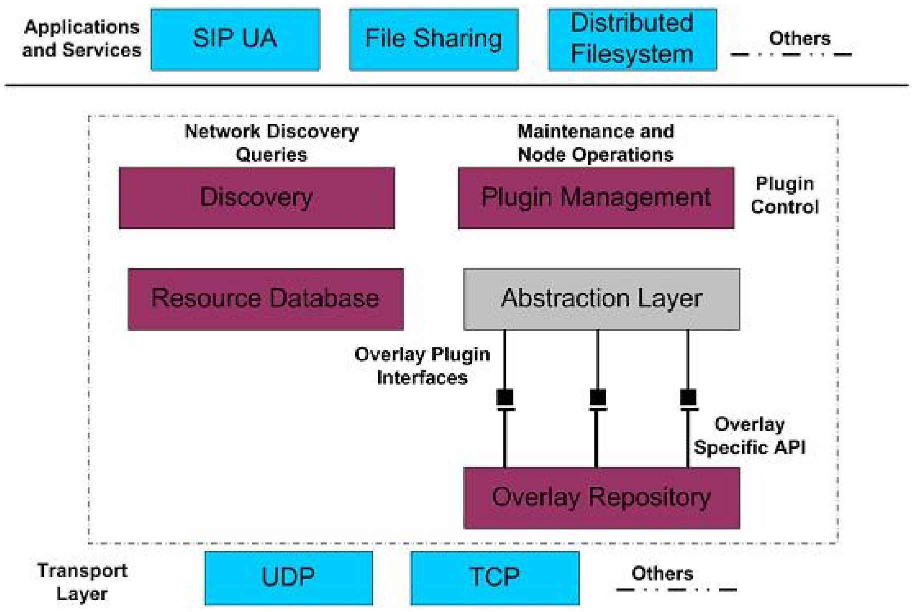


Figure 5.1: The Layered Architecture of the OverCord Framework. Source: [67].

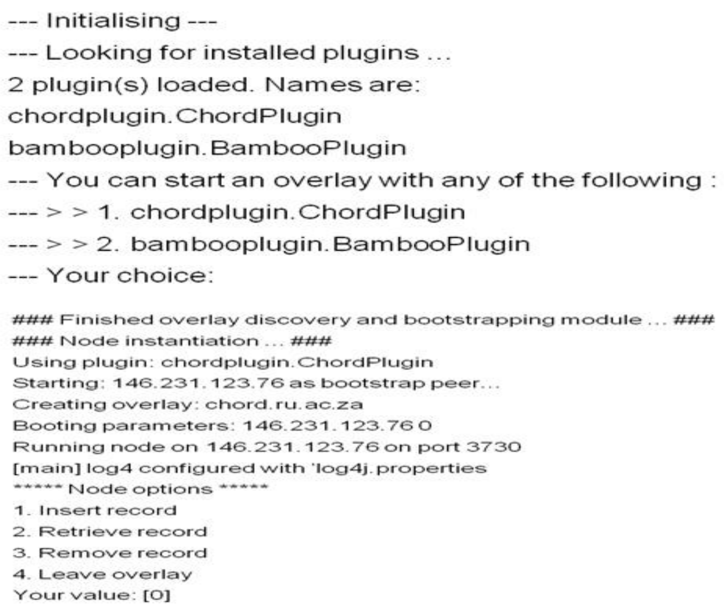
5.1.1 Plugin Management

The plugin management component perform administrate tasks for the OverCord. Basically it detects plugins, verifies and controls plugins. Plugins are the DHT algorithms that are being used by the OverCord. DHT supported by OverCord are Bamboo and Open Chord but can be extended to include other DHT algorithms. The plugin management has a Java class called *PluginManager* that has the main method to run OverCord.

5.1.2 Testing OverCord

There was a need of a Java framework to run the code. The NetBeans framework for Java desktop applications was used (During the process of testing we used both the framework and window command line to test OverCord). OverCord design is good but the implementations is complex.

When OverCord was developed, it was incorporated into the Jain SIP Applet Phone (JSAP) giving JSAP P2P capability. A stand alone OverCord was requested from the developer because the one incorporated into a user agent was quite different. There is no defined mechanism that makes easy to incorporate such a framework into an UA. Basically, this means that OverCord is not designed as an adaptor like SIPpeer [58]. Making it impossible to incorporate it into user agents without its source code.



```
--- Initialising ---
--- Looking for installed plugins ...
2 plugin(s) loaded. Names are:
chordplugin.ChordPlugin
bambooplugin.BambooPlugin
--- You can start an overlay with any of the following :
--- > > 1. chordplugin.ChordPlugin
--- > > 2. bambooplugin.BambooPlugin
--- Your choice:

### Finished overlay discovery and bootstrapping module ... ###
### Node instantiation ... ###
Using plugin: chordplugin.ChordPlugin
Starting: 146.231.123.76 as bootstrap peer...
Creating overlay: chord.ru.ac.za
Booting parameters: 146.231.123.76 0
Running node on 146.231.123.76 on port 3730
[main] log4 configured with 'log4j.properties
***** Node options *****
1. Insert record
2. Retrieve record
3. Remove record
4. Leave overlay
Your value: [0]
```

Figure 5.2: OverCord Output from NetBeans framework.

We then experimented with Open Chord and Bamboo. Understanding these two DHT algorithms was crucial for removing some errors in OverCord.

The *PluginManager* Java class in the plugin management component is used to run OverCord. There is another Java class called *SimulationManager* which is used for creating overlays. Other important classes in plugin management are *PluginCreator* and *PluginDetector*. *PluginCreator* used for storing information about the loaded class whereas *PluginDetector* as its name suggest detect the presence of plugins in a specified folder. Figure 5.2 show a sample output of OverCord.

5.2 39 Peers

The 39 Peers [57] is an open-source P2P Internet telephony software using the P2PSIP and implemented in Python. The project is developed by Singh [57] and it is still incomplete. In 39 Peers, a P2PSIP adaptor component was developed to act as a SIP-to-SIP gateway. Theoretically, this allows one to use any compatible SIP UA to be connected via an adaptor running local. The SIP UA is then able to use a P2P module to perform lookup and store content information, along with other information such as cryptographic credentials. Essentially, an incoming REGISTER request from the UA is translated into a P2P *put* operation that stores information. Any other SIP request from the UA is translated into P2P *get* (lookup) operation for the destination and the request is routed to that destination.

Some of the high level modules that have been completed are *voip*, *dht*, *opendht* and *dhtgui*. These different modules have different roles and use modules that implement different protocols, such as SIP and SDP. The *voip* module handles SIP UA registration, call, IM and conferencing. The DHT implementation supported by this project is Bamboo. The *opendht* module allow the client side to connect to an existing OpenDHT service whereas *dhtgui* module implements a test launch of GUI displaying nodes in a circle like as shown in Figure 5.3.

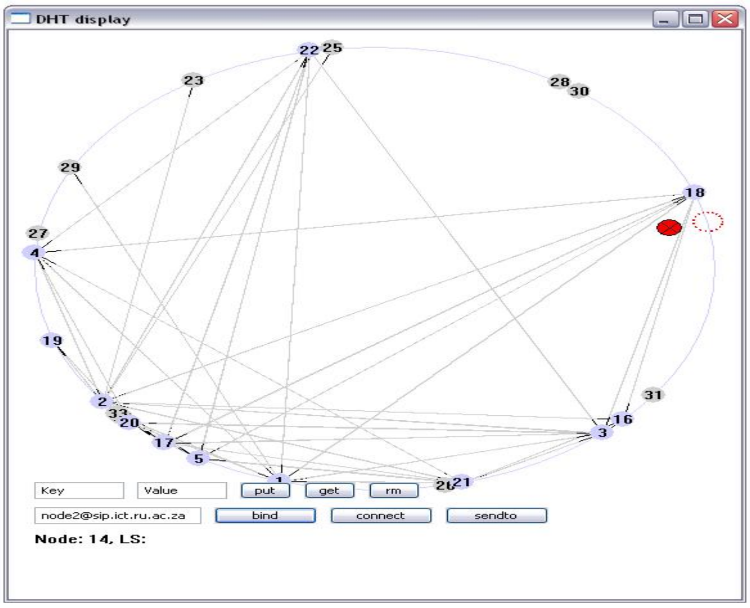


Figure 5.3: Screen Capture of 39 Peers GUI Displaying Nodes in a Ring Topology.

Some modules are still incomplete such as the module that implements the P2PSIP application, SIP registration and modules to implement abstraction between two peers using a DHT. The *p2psip.py* module is still to be finished. In particular the UA class needs to be implemented to enable the UA to do call routing.

5.2.1 Testing 39 Peers

The 39 Peers was tested on a Window32 machine on which Python 2.6 was installed. Because there are many modules that are incomplete, the *dhtgui.py* module was the only module which was tested. The other completed modules do not run by themselves as they just implement logic needed by other modules. Running the *dhtgui.py* module requires the *wxPython* library [70].

The project provides a nice graphical interface for displaying nodes in a circle as shown in Figure 5.3. The 39 Peers project first needs to be downloaded, and inside the unzipped folder simply execute the *dhtgui.py* to launch the application as shown in Figure 5.3. Use the command *python app/dhtgui.py* to run the *dhtgui.py* module. The graphical interface is shown in Figure 5.3.

5.3 OpenVoIP: An Open Peer-to-Peer VoIP and IM System

OpenVoIP [66] is an open source P2P VoIP and IM system of 1000 nodes running on 300 PlanetLab [45] machines, developed at Columbia University. It runs P2PP, an early candidate proposed for P2PSIP and was designed to allow nodes to relay SIP messages and media between nodes possibly behind NAT and firewalls. The P2PP does not define specific structured and unstructured protocols to allow it to implement well-known DHTs or unstructured protocols. Currently, Kademlia, Bamboo, and Chord are the only supported DHTs in P2PP version 1. The security part of it is handled by the hash algorithms. The following algorithms are supported: SHA1, SHA256, MD4 and MD 5. Message routing in OpenVoIP can be recursive or iterative routing.

The P2PP protocol is based on P2PP draft [3] which is now obsolete. The P2PP protocol support both peers and clients. The clients and peers in OpenVoIP follow the same definitions as in Section 4.3. However, OpenVoIP clients and peers are defined in terms of their position with respect to NATs: A node behind NATs join as client, the others are peers. OpenVoIP uses STURN, TURN and ICE for NAT traversals.

5.3.1 Testing OpenVoIP

OpenVoIP can run on both Linux and WindowsXP, but we have tested this project only on WindowsXP. This project can be run in two ways, by connecting to PlanetLab overlay using P2PP or using the Open-Wengo-P2PP SIP phone [66]. The OpenWengo P2PP SIP phone is an open source phone that has been given P2P capability by P2PP. This phone connects to OpenVoIP system running on PlanetLab. If the OpenWengo P2P phone is behind NAT then it operates as P2PSIP client, otherwise it will always operate as a P2PSIP peer. The PJNATH (v0.8.0) library [44] is used to determine whether a node is behind NAT before joining the network.

OpenVoIP has two layers: the P2P layer for searching usernames and SIP layer for establishing sessions. The system has a bootstrap server running on a remote location (128.59.19.185). This server keeps and updates the status of nodes and peers running P2P executable or OpenWengo on Google maps.

To avoid dependency on a remote server, we decided to create our own overlay by creating our own bootstrap server. Once we created our own overlay, the OpenWengo P2P phone was used by users that joined the overlay to call each other.

5.4 SIP2P

SIP2P [61] is an open-source project published at Sourceforge.net. It provides an experimental P2PSIP implementation that makes use of the Kademlia library and allows placing calls between SIP clients. It is written in C++ and currently supports the REGISTER and INVITE methods, which are directly translated into a *put/get* operation on the Kademlia network. The receipt of REGISTER message is used to publish user data binding in the P2P network. When making the calls, the callee is searched in the P2P network, the INVITE message will be modified accordingly and forwarded to the destination. The REGISTER message is processed as follows: when a user receives the REGISTER message, their data is published in a Kademlia network and a SIP Ok response is sent to the sender.

The *<Key, Value>* pair comprises of the username as the key and a combination of a username, IP address and port number of machine on which the SIP client is running on are used as the value. For example, lets say username Erasmus and his SIP client runs on IP address 146.231.123.70 on port 5061. Then the published value will be Erasmus@146.231.123.70: 5061. The port number and preceding colon are omitted if the SIP client is running on a default port 5060. During the processing of an INVITE message, the SIP2P acts as SIP proxy server , see Figure 2.5 and replace the proxy server in that figure with SIP2P to understand how the processing of INVITE message work.

The SIP clients login using the SIP2P server. When a client logs in, it stores its data in Kademlia (which is essentially resembles a P2P database) and when a call comes in, it looks for the data in the Kademlia network, and then forwards the call. Currently the project is not active anymore.

5.4.1 Testing SIP2P

The required library and software to run this project can be found in *Appendix A*. The process of testing started by setting up a virtual Linux machine. Installing the socketcc library, gave us tough time. This was a result of some of the classes being written in both C and C++. We overcame this problem by including both headers in classes. Another issue we experienced during the testing is not get a permission to execute the tinyconfig. (We changed it to be executable using *chmod 777 tinyconfig* and then configured it using *./tinyconfig*). There was a need to update the makefile for building the project, and settings file for running the program. The project was tested on a Linux machine. The SIP2P was running on the virtual Linux machine but the SIP clients can run on any operating systems. In normal SIP client (e.g., X-lite), the address and the port of the machine where SIP2P is running is entered as a SIP proxy.

5.5 Summary

In this chapter we identified and discussed five of the available P2PSIP implementations. We went further to tested. There are still more implementations available such as SIPDHT2 based on XPP draft, Kademlia dSIP implementation based on dSIP draft, Huawei implementation of SEP peer and client protocols and P2PNS, a secure distributed Name service for P2PSIP. Testing these systems was helpful in understanding P2PSIP concepts and in putting in perspective the findings mentioned in Chapter 4.

Chapter 6

Discussion and Recommendations

In the previous chapter, we discussed how we tested some P2PSIP implementations discussed in the previous chapter. In this chapter, we will provide a comparison between the tested systems in order to provide a recommendation on those that may be suitable for supporting research activities, particularly within the Rhodes University Convergence research group.

6.1 Comparison of Tested Systems

We used different criteria to compare tested P2PSIP implementations. The criteria can be categorised into qualitative and non-qualitative criteria. Qualitative criteria is subjective in nature while non-qualitative is objective.

In this section, we will provide our comparisons along these two dimensions. We do not include RELOAD in our comparison because it is not implemented yet.

6.1.1 Comparison Using the Non-qualitative Criteria

Table 6.1 shows the comparison of the tested P2PSIP systems. The columns represent the tested systems while rows represent the criteria used for comparison. Below are the explanations of included criteria.

Architecture: Since we only tested systems that employ SIP-using-P2P architecture approach, this criterion does not really influence the decision about which systems to chose.

Client and peer support: As it was stated in Section 4.2, the adopted P2P version of SIP supports clients and peers. We have identified this criterion so that it would not be difficult to re-implement such project toward a standardised P2PSIP. This criterion does not really mean that clients and peers are defined as the P2PSIP working group has defined.

Complete: Completeness is this case is intended to provide indication wether the project is ready for incorporation into the a user agent. We are looking for a project for that is viable for research in both theoretical and practical learns.

Hash Algorithms: This criterion indicates how strong security of is. Systems without any hash algorithms are more vulnerable then those include hash algorithm .

Language: Some of the programming languages are more complex, popular and compact than others. For example, language like Python is easy to understand and uses less lines of code than corresponding code in Java or C/C++. This gives Python advantage over other languages because it means that one can write software much faster and easily (see [57] for further details).

NAT support: P2PSIP systems need to support NAT since some user agents may reside behind NAT.

Routing Methods: Security and NAT are the main issues that need to be addressed in P2PSIP systems. As a result, we must be aware if a routing method can support NAT traversal without compromising security.

Supported DHT algorithms: The choice of DHT algorithm is very important and this criterion indicates which type of DHT algorithms supported. The best option is to have multiple algorithms supported but that said, we can always re-implement the systems to support different DHTs.

User Agent: Some of the tested P2PSIP systems have already been incorporated into the user agent but some are not. At the same time we are also interested on how easy it is to incorporate such a system into a general user agent. Some of these systems require the UA source codes for it to be incorporated in such a UA.

	39 Peers	OpenVoIP	SIP2P	OverCord
Architecture	SIP-using-P2P	SIP-using-P2P	SIP-using-P2P	SIP-using-P2P
Client and peer support	Yes	Yes	No	No
Complete	No	Yes	Yes	Yes
HASH ALGORITHMS	SHA1	MD4, MD5, SHA1 and SHA256	SHA1	None
Language	Python	C++	C++	Java
Nat support	Yes	Yes	No	No
Routing methods	Recursive	Iterative and Recursive	Recursive	Iterative and Recursive
Supported dht algorithms	Kademlia only	Bamboo, Chord and Kademlia	Kademlia only	Bamboo and Chord
USER AGENT	None	OpenWengo	None	JSAP

Table 6.1: Non-qualitative Comparison of Tested P2PSIP Systems.

As we can see from Table 6.1, half of these systems have do not support security and NAT. Hash algroithm indicate if the systems support security or not. These two issues are major drawbacks in the progress of P2PSIP standardisation. Below we will provide a brief discussion pertaining to each system as part of interpretation of the table.

- OverCord has been influenced by some of the early proposals in the P2PSIP working group. As a result, it does not support clients and peers. It makes the hypothesis that it can be used for other P2P applications besides SIP. This is due to the structure of tthe protocols layering of the project which uses separate P2P stack and SIP stack. OverCord does not employ any security mechanisms. This is one of the issues that need to be addressed as messages are passed from one node to another through the overlay. Hence, messages can be easily compromised as there are no security mechanism to avoid this. OverCord implementation is very difficult to incorporate into a UA for which the source code does not exist. It is also hard to incorporate OverCord implementation into a non-Java implemented UA.

- 39 Peers is based on SIP-using-P2P architecture. The project is incomplete but the design specification shows the potential of the project. The *p2p.py* module implements abstraction that allows establishing a P2P pipe between two peers. It is based on a partial P2P decentralised model with a the super-node that keeps a list of all attached ordinary nodes, and forwards incoming connect and datagram requests to the attached ordinary nodes. A super-node is similar to P2PSIP peer while ordinary node is similar to P2PSIP client. (Refer to Subsection 3.5.3 for partial decentralised P2P model and Section 4.2 for P2PSIP peer and client). 39 Peers has implemented a security using a hash algorithm. NAT is supporrted using Session Traversal Utilities for NAT (STUN). The overall system is fairly easy to understand as it's developed in Python. The major advantage of 39 Peers over other projects is that it runs on a local machine and can be easily incorporated into a UA using IP and port number.

- OpenVoIP is based on the SIP-using-P2P architecture. It has its own platform with a simple registrar and redirect server functionality. In theory, this allows any complient standard UA to use application layer network created by P2PP for SIP registration and call setup. During our testing process, we created our own overlay instead of using the PlanetLab platform (which is proposed by developers). This somehow does not allow a standard UA to use the overlay to make calls. OpenVoIP supports NAT Traversals using STUN, TURN and ICE. The PJNATH library is used for NAT traversal. For nodes that cannot establish calls directly, they use a relay peer for relaying signalling and media traffic through a relay peer. Just like 39 Peers, OpenVoIP employs the partial P2P model with peers and clients defined in the same way as defined by the working group. In OpenVoIP, clients are not part of the P2P overlay and they are attached to a peer who will acts as a super-peer for the attached clients. Clients are simpler peers that do not store resources or t take part in lookups. However, clients are able to insert resources, lookup for peers and resources in the same way as peers through their super-node (super-peer).

- SIP2P is also based on SIP-using-P2P architecture just like the other three systems. SIP2P acts like SIP proxy and it is incorporated into a UA by entering the IP address and its port in a normal SIP client. Basically with SIP2P one does not need the UA source code. The major issue that arises from this is that single point of failure remains a problem. As it was stated in Section 5.5, SIP2P runs on a Linux machine and can be tested by entering the IP address and port number into a user agent as SIP proxy. Once this machine goes down, the users in Kademlia network will not be able to find each other. SIP2P is good that there is very little modification that needs to be done in order to incorporate it into the UA. Another major problem with SIP2P, it does not support NAT and cannot support communication across different domains.

The perfect way is to implements a SIP client having SIP2P functionality within. This will give the SIP client the ability to do what the SIP proxy server and other server would have done but would retrieve the addresses of other users over Kademlia and SIP messages would be sent directly to the called party.

The best scenario for OpenVoIP project, is that we can adapt the P2PP source code and incorporate into some UA but this is likely to be difficult to incorporate into a UA that has no available source code. However, which source code are not available. On top of that we can use the mechanism used in this project to support NAT. The mechanism we are referring to is the PJNATH library, which can be used in other projects to allow them support NAT.

6.1.2 Comparison Using Qualitative Criteria

The ideal system must be easy to change and must support some of the scenarios proposed in RELOAD. At the same time, the system must be able to give practical understanding of P2PSIP to researchers. We identified the following criteria: documentation, technical support, usability and testing as being important qualitative criteria in choosing systems for further research.

Documentation: Is there enough documentation ? what type of documentation exist ? how easy it is to follow? and are the source code available ?. These are the questions that this criterion will answer.

Technical Support: Another key aspect to consider is the assistance, if one is stuck when setting up testing scenarios or there are bugs within code, can one get assistance from forums.

Testing: How the testing scenarios created. These basically refer to the process of configuring and setting up the platform for testing.

Usability: This criterion has to do with how easy the system is to learn, use and the neatness of its source code. Can code be altered and ported to and from other system?.

Table 6.2 shows the comparison of tested systems using qualitative criteria.

	39 Peers	**OpenVoIP**	**SIP2P**	**OverCord**
Documentation	Little documentation, through website and documentation of code.	Well documented, lots of publications about P2PP that is used as P2P protocol in this project.	Little documentation by means of *README* file in the source code, no other means.	Well documented, the design and implementation are explained in a Master Thesis [67].
Technical Support	Technical support via developer email and forum.	Technical support is available via developer email and documentation (website).	Little technical support available, through developer email.	Developer provide technical support via consultation.
Testing	The process of testing was easy. To figure how to set up the project was a big challenge.	The process of testing was easy. This is because all the setup procedure is provided on the website.	The process of testing was hard. Few documentation on how to setup the project.	The process of testing was hard. The Complex implementations made it hard.
Usability	Source code are neat, can be altered and ported. Ready to be incorporated into a UA, project incomplete.	Source code are neat, can be altered and ported. P2PP codes can be used to make a P2P overlay or incorporated into a UA.	The source code are not neat making them difficult to alter or port them to another system. Ready to be incorporated into a UA.	Neat source code that can be altered and ported.

Table 6.2: Qualitative Comparison of Tested P2PSIP Systems.

- 39 Peers provide documentation of the source code through their website. The source code is well commented and in a Portable Document Format (PDF) format. The document explains the methods and variables used in the code. The developer also gives assistance about the project by email. The protocols used in the project are well expalined in the document. However, the process of testing was very difficult initially. This was due to the fact that no information was given on how to test the project in the *README* file. One can attribute this to the fact that the project is incomplete. Although the project is incomplete, major modules were be tested.
- OpenVoIP like 39 Peers has a website. Further there is a lot of documentation about where P2PP. As it was stated before, P2PP is an early proposal candidate for P2PSIP proposed by a team from Columbia University. P2PP source code is available on the OpenVoIP website. OpenVoIP uses a lot of code from different projects but some of its codes but not all this code is made available, some available as a binary executable. The testing process was quite fast thanks to a video tutorial on how to test the project. At the same time, there was clear explanation of the P2PSIP concepts. However, the is too much abstraction. In terms of setting up P2PP overlay, it is very difficult to understand at high level as setting up process is only done using binary executable.
- SIP2P has little documentation available, the only means of documentation we are aware of is the read me file that exist in the source code. The read me file is well written, with most of the required libraries and an explanation on how to run the project. The source code is available at source forge. The process of testing took time. The project itself is ready for use and can be incorporated into a UA.
- OverCord publication is by means of Master thesis, well documented and it can be clearly understood. The source code is available but very complex to understand. This project took us time to test as there was a need to understand how the DHT implementations used in this project work. OverCord is already incorporated into a UA called JSAP but it can be used in another UA.

6.2 System Recommendations

The reason for analysing the tested systems is to help us to choose an ideal system project for research purpose. 39 Peers seems to be the best one to use out of the four implementations. That said, is still unfinished. It is closely aligned with the work by

P2PSIP working group. Hence, it will not be difficult to modify in order to implement all decisions made in the P2PSIP working group. 39 Peers runs as an adaptor that can runs on a local machine. This makes it very easy to incorporate into the UA with little or no implementation modification.

SIP2P can be considered as the second best out of the tested systems. SIP2P has implemented the logic needed to support P2P. Hence, very little needs to be done in terms of P2P network. Furthermore one can borrow PJNATH and some code from OpenVoIP to implement NAT traversal and security mechanisms respectively. These two projects can clearly enhance the future research in P2PSIP, especial within the Convergence research group. This could be trhough implementation of VoIP services which use a P2P overlay instead of a server. Alternatively through implementation of IPTV/video streaming that uses P2P technology to organise TV/video streaming servers. OpenVoIP can be seen as first stepping to understand P2PSIP. The project is complete, make it not applicable in terms of practical implementation understanding of P2PSIP. However, that said the project can be used to create a P2PSIP systems for practical understanding purpose. OverCord implementation is too complex to be used, need a modification to simplify it.

6.3 Summary

The tested P2PSIP systems seem to predict a bright future for P2PSIP. Most of these systems are either incomplete or based or early proposals of the work in the P2PSIP IETF working group. This can be seen from the fact that a system like SIP2P does not support the notion of P2PSIP client and peer. Most of the tested systems seem to ignore interoperability with conventional SIP.

Having tested four P2PSIP systems, we identified some criteria to assist us in choosing the systems for further research. In the end, we recommended two of the four systems: 39 Peers and SIP2P. As the most appropriate systems for further research. It's very surprisingly enough that there is a lack of RELOAD prototype implementation. The work we are aware of, is the analysis [51] of RELOAD based on the first draft of RELOAD, and at the time of writing RELOAD was in draft 11. There is also a project just launched at Google code implementing storage and message encoder decoder for RELOAD.

Chapter 7

Conclusion and Future Work

P2PSIP is a relatively new proposal that aims at capitalising on the benefits that are provided by P2P networks and SIP in order to provide scalable and reliable communication services. There are still some issues to address before P2PSIP can become a standardised.

In this project, we made a determination that to attain standardisation flexibility is key. We saw this through the choices that have been made by the P2PSIP IETF working group on three key requirements issues. Firstly, the group has shown a preference for a partial P2P distribution model instead of a pure model. This means that a variety of endpoints with different capacities can participate in communication as either peers that help in sharing functions that would have been done by servers or as clients that merely make use of services. Secondly, in terms of protocol layering, the group has adopted SIP-using-P2P instead of P2P-over-SIP. As indicated before, SIP-using-P2P uses two separate stacks and this provides the flexibility to use the architecture for other P2P applications without making any modifications. Thirdly, with regard to DHT algorithms to use, the group prefer using multiple algorithms instead of one. However, so far Chord is a popular choice.

Despite making choices or decisions on key issues, a number of issues remain open. With P2PSIP, user information including location address, presence status and buddy lists are routed and stored in the untrustworthy overlay network. Security mechanisms are needed to prevent unauthorized access to data. Unfortunately, mechanisms which rely on trusted entities to provide hop-by-hop security along the routing path are not applicable in the distributed architecture like P2PSIP, where nodes are untrustworthy. Hence, security remains an important issue to be addressed by the working group.

Other issues include support for NAT, dealing with high delays, expiration of resources and other failures on the Internet that must be handled in order to sustain a stable and correctly operating overlay network. All these issues need to be addressed if P2PSIP is to be standardised. This means more research needs to be carried out to address the issues.

P2PSIP may potentiate new market for all the involved actors. Device manufacturers have proven in commercial products that a P2P PBX for example can be implemented such that all the PBX functions reside in the end devices. Companies like Siemens and Avaya have already targeted Small and Medium Business (SMB) VoIP market.

7.1 Revisitation of Project Objectives

Even though there were some challenges, the core objectives of the project were met. We were able to investigate and draw conclusions on the direction taken by the P2PSIP IETF working group on some requirements for P2PSIP standardisation. In this section we revisit the objectives in Section 1.2 by stating our achievements on each objective.

Analyse and compare the designs proposed in the drafts.

We did an analysis on the proposed protocols. These protocols were XPP, P2PP and RELOAD. Our study later shifted more toward RELOAD as it was the primary P2PSIP protocol.

Identify open issues about P2PSIP that are still being discussed by P2PSIP IETF working group.

We identified major design decisons that there were made and summarised them. We also gave a summary of open issues in the IETF working group. Such isuses are security and NAT.

Test, analyse, and compare some of the available P2PSIP systems.

We identified four systems, tested these systems and evaluated them by comparing them using specific requirements criteria. Moreover, we were able to recommend two P2PSIP implementations that could be used to further the work being done on the standardisation of P2PSIP or in the development of decentralised services. In particular, we recommend these systems to be used for further research within Convergence research group.

7.2 Future Work

The work presented in this thesis can be developed and augmented in a variety of ways, be it from a theoretical or a practical point of view. Theoretically, there is a need to continue with the investigation on the work being done by the P2PSIP working group.

Practically, more work could be done to improve OverCord such that it evolves towards RELOAD. Security could be enhanced and a graphical interface for displaying and inserting nodes could be implemented. Further, an adaptor interface could be built to enable easy incorporation of OverCord into UAs without changing implementations; perhaps a graphical user interface could also be provided to allow the user to change the IP address and port number. Following the enhancements, the hypothesis that OverCord is able to support other P2P applications could be tested.

Other tested systems could also be enhanced to evolve towards RELOAD implementation. For 39 Peers, there is a need to finish the unfinished modules first and to ensure that the user agent can perform call routing. A user agent could also be identified to which SIP2P can be incorporated and add some feature to SIP2P to be able to support SIMPLE/IM. On, top of that a security mechanism can be another feature to be added on. The support of NAT can be explored, either using PJNATH library or any other means. Very little has been done in testing implemented P2PSIP systems with small devices such as mobile devices. This could also be explored in the future.

References

[1] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999.

[2] S. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *INFOCOM 2006. The 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, 2006.

[3] S. Baset and H. Schulzrinne. Peer-to-Peer Protocol (P2PP). Internet-Draft (work in progress), draft-baset-p2psip-p2pp-01, Available online, URL: http://tools.ietf.org/html/draft-baset-p2psip-p2pp-01, November 2007. Accessed on: 14 June 2010.

[4] BestSoftphone.com. http://www.etn.nl/voip/softphon.htm. Accessed on: 01 October 2010.

[5] R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 353–366, New York, NY, USA, 2004. ACM.

[6] R. Birke, M. Mellia, M. Petracca, and D. Rossi. Understanding VoIP from Backbone Measurements. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2027–2035, May 2007.

[7] BitTorrent. Available online; URL: http://www.bittorrent.com/. Accessed on: 21 March 2010.

[8] A. Bryan and B. Lowekamp. Decentralizing SIP. *Queue*, 5(2):34–41, 2007.

[9] D. Bryan and B. Lowekamp. SOSIMPLE: A SIP/SIMPLE Based VoIP and IM System, Novemebr 2004.

[10] D. Bryan and B. Lowekamp. SOSIMPLE: A Serverless, Standards-based. In *P2P SIP Communication System, Proceedings. of the International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications*, pages 42–49. IEEE Press, 2005.

[11] D. Bryan, P. Matthews, E. Shim, D. Willis, and S. Dawkins. Concepts and Terminology for Peer to Peer SIP. Internet-Draft (work in progress), draft-ietf-p2psip-concepts-02, Available online, URL: http://tools.ietf.org/html/draft-ietf-p2psip-concepts-02, July 2008. Accessed on: 04 April 2010.

[12] G. Camarillo, P. Nikander, J. Hautakorpi, A. Keranen, and A. Johnston. HIP BONE: Host Identity Protocol (HIP) Based Overlay Networking Environment. Internet-Draft (work in progress), draft-ietf-hip-bone-06.txt. Available online, URL: http://tools.ietf.org/html/draft-ietf-hip-bone-06, April 2010. Accessed on: 03 June 2010.

[13] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428 (Proposed Standard), December 2002.

[14] S. Chen, F. Gao, C. Sun, X. Qiu, and C. Zhang. Public Security Channel(PSC): An Alternative Key Management Mode in RELOAD. Internet-Draft (work in progress), draft-chen-p2psip-psc-00. Available online, URL: http://www.ietf.org/id/draft-chen-p2psip-psc-00.txt, October 2010. Accessed on: 15 October 2010.

[15] Open Chord. Available online; URL: http://open-chord.sourceforge.net/. Accessed on: 21 October 2010.

[16] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *INTERNATIONAL WORKSHOP ON DESIGNING PRIVACY ENHANCING TECHNOLOGIES: DESIGN ISSUES IN ANONYMITY AND UNOBSERVABILITY*, pages 46–66. Springer-Verlag New York, Inc., 2001.

[17] E. Cooper, A. Johnston, and P. Matthew. A distributed transport function in p2psip using hip for multi-hop overlay routing. Internet-Draft (work in progress), draft-matthews-p2psip-hip-hop-00. Available Online, URL: http://tools.ietf.org/search/draft-matthews-p2psip-hip-hop-00, June 2007. Accessed on: 03 June 2010.

[18] DC++. Aailable online; URL: http://dcplusplus.sourceforge.net/index.html. Accessed on: 14 March 2010.

[19] R. Fielding, UC. Irvine, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616, Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Proposed Standard), June 1999.

[20] Simple Status Pages SIP for Instant Messaging and Presence Leveraging Extensions (Active WG). Available online, URL: http://tools.ietf.org/wg/simple/. Accessed on: 21 October 2010.

[21] Freenet. Available online; URL: http://freenetproject.org/download.html. Accessed on: 14 March 2010.

[22] Gnutella. Available online; URL: http://sourceforge.net/projects/phex/, Accessed On: 21 March 2010.

[23] M. Handley, V. Jacobson, and C. Perkins. Sdp: Session description protocol. RFC 4566 (Proposed Standard), July 2006.

[24] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. RFC 2543 (Proposed Standard), March 1999.

[25] Bamboo homepage. The bamboo distributed hash table: A robust, open-source dht. Available online; URL: http://bamboo-dht.org/. Accessed on: 23 April 2010.

[26] Chord homepage. Available online, URL: http://pdos.csail.mit.edu/chord/. Accessed on: 23 April 2010.

[27] Naspter homepage. Available online, URL: http://free.napster.com/napsterhomemain.htm;jsessi
Accessed on: 21 April 2010.

[28] OpenDHT homepage. Available online, URL: http://opendht.org/. Accessed on: 14 March 2010.

[29] Skype homepage. Available online, URL: http://www.skype.com/intl/en/home. Accessed on: 14 March 2010.

[30] The Internet Engineering Task Force (IETF) homepage. Available online, URL: http://www.ietf.org/. Accessed on: 21 October 2010.

[31] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne. A P2P Approach to SIP Registration and Resource Location. Internet-Draft (work in progress), draft-bryan-p2psip-dsip-00. Available online, URL: http://tools.ietf.org/html/draft-bryan-p2psip-dsip-00, February 2007. Accessed on: 03 June 2010.

[32] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne. REsource LOcation and Discovery (RELOAD) base protocol. Internet-Draft (work in progress), draft-ietf-p2psip-base-08. Available online, URL: http://tools.ietf.org/html/draft-ietf-p2psip-base-08, March 2010. Accessed on: 14 June 2010.

[33] C. Jennings, J. Rosenberg, and E. Rescorla. Address Settlement by Peer to Peer. Internet Draft (work in progress), draft-jennings-p2psip-asp-0. Available online, URL: http://www.p2psip.org/drafts/draft-jennings-p2psip-asp-00.txt, July 2007. Accessed on: 14 June 2010.

[34] X. Jiang, H. Zhenga, C. Macian, and V. Pascual. Service Extensible P2P Peer Protocol. Internet-Draft (work in progress), draft-jiang-p2psip-sep-01. Available online, URL: http://tools.ietf.org/html/draft-jiang-p2psip-sep-01, February 2008. Accessed on: 14 June 2010.

[35] A. Johnston. *SIP: Understanding the Session Initiation Protocol*. Artech House, INC, Second edition, 2004. ISBN 1-58053-655-7.

[36] KazaA. Search download and share. Available online, URL: http://www.kazaa.com/. Accessed on: 21 March 2010.

[37] L. Lessig. *Peer-to-Peer Harnessing the Power of Disruptive Technologies*. O'Reilly Media, O'Reilly & Associates, Inc . 101 Morris Street Sedastopol, CA 95472, First edition, March 2001. ISBN: 0-596-001100-X.

[38] G. Lorenz, T. Moore, J. Hale, and S. Shenoi. Securing SS7 Telecommunications Networks. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, June 2001.

[39] E. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, 7:72–93, 2005.

[40] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766 (Proposed Standard), April 2010.

[41] E. Marocco and E. Ivov. Extensible Peer Protocol (XPP). Internet-Draft (work in progress), draft-marocco-p2psip-xpp-01. Available Online, URL: http://tools.ietf.org/html/draft-marocco-p2psip-xpp-01, November 2007. Accessed on: 03 June 2010.

[42] University of Bamberg. Available online, URL: http://www.sewanee.edu/german/Bamberg/bamberg.html. Accessed on: 21 October 2010.

[43] P2PSIP Status Pages. Peer-to-Peer Session Initiation Protocol (Active WG). Available online, URL: http://tools.ietf.org/wg/p2psip/. Accessed on: 21 October 2010.

[44] STUN PJNATH-Open Source ICE and TURN Library. Available online, URL: http://www.pjsip.org/pjnath/docs/html/. Accessed on: 14 October 2010.

[45] PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services. Available online, URL: http://www.planet-lab.org/. Accessed on: 14 October 2010.

[46] J. Postel. RFC821 - Simple Mail Transfer Protocol. RFC 821 (Proposed Standard), August 1982.

[47] B. Pourebrahimi, K. Bertels, and S. Vassiliadis. A survey of peer-to-peer networks. In *Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Proessing*, 2005.

[48] S. Pundkar. Peer to peer communication over the internet: An open approach. Technical report, Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, INDIA, Developed at Telecom Italia, Turin, Italy, May-July 2007. Available online, URL: http://sipdht.sourceforge.net/sipdht2/swapnil-report.pdf, Accessed on: 14 June 2010.

[49] A. Rajini. VOIP in PEER-TO-PEER using Session Initiation Protocol. In *International Journal of Computer Applications (0975 - 8887) Volume 1, No. 28*, 2010.

[50] S. Ratnasamy, P. Francis, M. Handley, Richard Karp, and S. Shenker. A Scalable Content-Addressable Network. In *PROC. ACM SIGCOMM 2001*, pages 161–172, 2001.

[51] A. Roly. Analysis and prototyping of the IETF RELOAD protocol onto a Java application server. Master's thesis, Universitaires NotreDame de la Paix de Namu, 2008-2009.

[52] J. Rosenberg. Indicating Support for Interactive Connectivity Establishment (ICE)in the Session Initiation Protocol (SIP). RFC 5768 (Proposed Standard), April 2010.

[53] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621.

[54] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *IN: MIDDLEWARE*, 3:329–350, 2001.

[55] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Proposed Standard), July 2003.

[56] K. Sighn. *Reliable, scalable and Interoperable Internet Telephony.* PHD project, Columbia University, Department of Computer Science, New york, 2006.

[57] K. Singh. Welcome to 39 peers! Available online, URL: http://www.39peers.net/. Accessed on: 15 September 2010.

[58] K. Singh and H. Schulzrinne. SIPPEER: A Session Initiation Protocol(SIP)-based peer-to-peer Internet telephony client adaptor. Implementation Report, Columbia University, Department of Computer Science, Columbia University, April 2004. This is a full TECHREPORT entry, Available online, URL: http://kundansingh.com/papers/sip-p2p-design.pdf, Accessed on: 17 September 2010.

[59] K. Singh and H. Schulzrinne. Peer-to-Peer internet telephony using SIP. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 63–68, New York, NY, USA, 2005. ACM.

[60] H. Sinnreich and B. Johnston. *Internet communications using SIP: Delivering VoIP and multimedia services with Session Initiation Protocol.* Wiley Publishing, Inc, New York, NY, USA, second edition, 2001. ISBN-13: 978-0-0471-77657-4.

[61] a P2P SIP Proxy sip2p. Available online, URL: http://sourceforge.net/projects/sip2p/. Accessed on: 14 October 2010.

[62] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and*

protocols for computer communications, volume 31, pages 149–160, New York, NY, USA, October 2001. ACM.

[63] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[64] P. Thomas, A. Zmolek, J. Kanclirz, and A. Rosela. *Practical VoIP Security*. Syngress, New York, NY, USA, 2006.

[65] Peer to Peer Session Initiation Protocol (p2psip). Available online, URL: http://datatracker.ietf.org/wg/p2psip/charter/. Accessed on: 21 March 2010.

[66] OpenVoIP: An Open Peer to Peer VoIP and IM System. Available online, URL: http://www1.cs.columbia.edu/ salman/peer/. Accessed on: 14 October 2010.

[67] M. Tsietsi. Prototyping a Peer-to-Peer Session Initiation Protocol User Agent. Master's project, Rhodes University, Department of Computer Science, Grahamstown, South Africa, March 2008.

[68] Berkeley University of California. Available online, URL: http://berkeley.edu/. Accessed on: 14 March 2010.

[69] J. Weber and T. Newberry. *IPTV Crash Course*. Mcgraw-Hill, 2006. ISBN-13: 978-0-07-226392-3.

[70] wxPython. Available online, URL: http://www.wxpython.org/. Accessed on: 14 September 2010.

[71] Z. Xianghan and O. Vladimir. *A survey on peer-to-peer SIP based communication systems*, volume 3 of *Peer-to-Peer Networking and Applications*, chapter 10, pages 257–264. Springer New York, Reading, Massachusetts, Fourth edition, December 12 2010.

[72] youtube homepage. Available online, URL: http://www.youtube.com/. Accessed on: 14 March 2010.

Appendix A

Requirements to run SIP2P

The SIP2P source code can be downloaded at (http://sourceforge.net/projects/sip2p/)

Software for installation :

- A PC running Linux Operating System
- Any soft phone (but tested only with X-Lite)
- C++ compiler.

Libraries:

1. KadC library - for storing and retrieving records in Kademlia can be downloaded at (http://kadc.sourceforge.net)
2. Pthreadcc - library to provide a series of multi-thread programming tools. Can be downloaded at (http://www.ctie.monash.edu.au/SocketCC/PThreadCC.htm)
3. Socketcc - library for socket communication. Can be downloaded at (http://sourceforge.net/pro

Steps to install the system:

Install the above libraries in the order as listed above. Using make command to build and make install to install. Install should be done with root previledge.

Configuration the layout of the config file is pretty straight forward. It just uses key-value pairs for the most important settings.

Below is a simple example of configuration layout, and then follows the explanation.

```
#———SAMPLE BEGIN————-
###################################################
# INI FILE USED FOR STARTING THE SIP2P SERVER
########################################################
# IP and port #######################
S2P_SERVER_IP=<ip address> S2P_SERVER_PORT=5060;
####################
# KadC conf file location
#######################
KADC_INI_FILE=kadc.ini
####################
# Debug TRUE or FALSE
#######################
DEBUG=FALSE
######################
# Publish dummy users
#######################
DUMMY_PUBLISH=TRUE
######################
# Fast search for dummy users
#######################
DUMMY_FAST_SEARCH=TRUE
```

######################

List of dummy users

#######################

[DUMMY_P2P_USERS] localx=konj@<ip address>:5062 local=local@<ip address>:5063 #local=local@<ip address> [DUMMY_P2P_USERS_END]

#————SAMPLE END————————

Comments start with a '#' sign. All lines starting with a hash key will be ignored. Actually if a key name cannot be interpreted, the line containing that key will also be ignored.

The most important settings which must be set for proper use are S2P_SERVER_IP and S2P_SERVER_PORT. They contain the IP address, and the port under which the UDP Server will be started. The other settings are used for debugging purposes and can be omitted.

The file named KADC_INI_FILE specify the location of the configuration file for Kad-Capi, the library used in this project. It should be always kadc.ini, but for debugging purposes, when running more then one process of SIP2P, then different ports can be specified for KadC to use in two different configuration files.

If DEBUG option is set to TRUE (it is case sensitive), then the program may behave different by normal mode. For details, please look into the source code.

Between the lines [DUMMY_P2P_USERS] and [DUMMY_P2P_USERS_END] you may put names of some dummy users and their SIP URLs. Actually these may be real SIP accounts, which are just not available in the kademlia. If proceded by Hash key #, then theses dummy users will also be ignored.

If the flag DUMMY_PUBLISH is TRUE, then the dummy users will be published in the kademlia network. IF the DUMMY_FAST_SEARCH is set to TRUE, then the names from the configuration file will be searched, and not Kademlia. That way you may speed up the development process, without having to wait for kademlia searches.

CLIENTS

————————————————-

v.81 has been tested wtih X-lite v2 which available at [4].

The following attributes need to be set for the client:

System Settings -> SIP Proxy -> add a default proxy with IP being the SIP2P address

Direct Dial IP=yes

Advanced System Settings - SIP Settings

Timeout SIP Messages=60000

Resend SIP Messages=60000

Appendix B

How to setup a P2PP overlay?

The instructions below can be used for building a peer-to-peer overlay in a public or a private network using Windows executable

1. Download the Windows executable file from http://www1.cs.columbia.edu/~salman/peer/overlay.h

The same executable is used by the bootstrap server, peer and clients.

Setting up the Bootstrap Server

1. Download the configuration file for the bootstrap server from http://www1.cs.columbia.edu/~salma

2. Open it any any text editor to edit it accordingly.

3. There are several fields in this file. See below for an explaination.

TCP_PORT: This is the listening port for incoming requests over TCP. The current implementation does NOT support TCP.

UDP_PORT: This is the listening port for incoming requests over UDP.

P2P: This is the DHT to be used in the P2P network. Kademlia and Bamboo are supported by the current implementation.

HASH_ALG: This is the hash algorithm to be used in the DHT. SHA1, SHA256, MD4, MD5 are supported in the current implementation.

HASH_SIZE: Size of the IDs generated by the Hash Algorithm (in bytes). This depends on the value in the HASH_ALG field. Use the pairs of the following

HASH_ALG-HASH_SIZE

SHA1 - 20

SHA256 - 32

MD4 - 16

MD5 - 16

REC: '0' for iterative, '1' for recursive. Specifies whether iterative or recursive routing is to be used. Current implementation supports both recursive and iterative routing.

PARALLEL: '1' for parallel, '0' for sequential. Specifies wether parallel requests are to be sent.

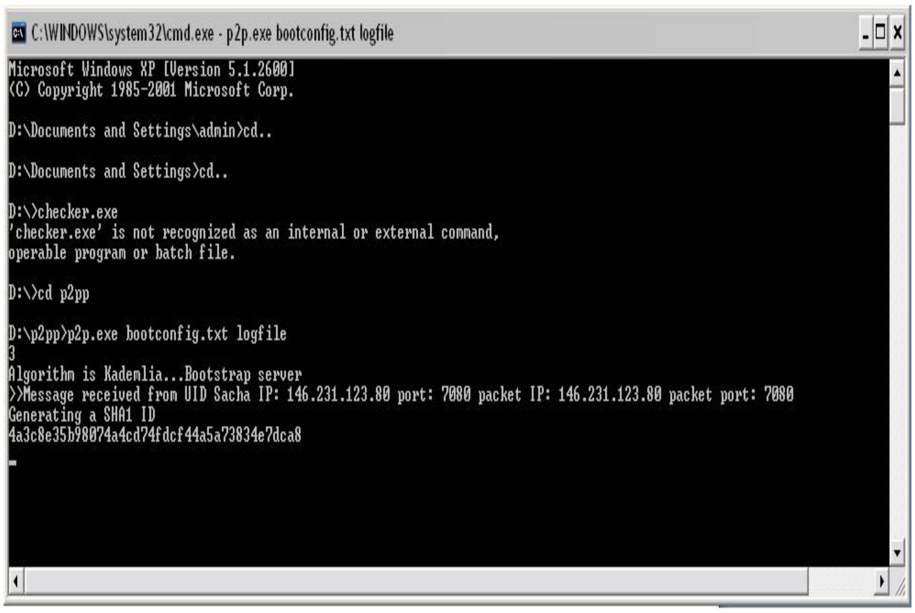
CMD: '1' for command-line usage, '0' for no-command line option. Specifies if command line usage is desired.

BASE: Indicates the base to be used by the DHT. Should be a power of 2.

MODE: Set this to BOOTSTRAP (default).

4. After editing the file, open a terminal and browse to the directory containing the executable and configuration file.

5. Type ">p2p.exe bootconfig.txt logfile" to start the bootstrap server.


C:\WINDOWS\system32\cmd.exe - p2p.exe bootconfig.txt logfile

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

D:\Documents and Settings\admin>cd..

D:\Documents and Settings>cd..

D:\>checker.exe
'checker.exe' is not recognized as an internal or external command,
operable program or batch file.

D:\>cd p2pp

D:\p2pp>p2p.exe bootconfig.txt logfile
3
Algorithm is Kademlia...Bootstrap server
>>Message received from UID Sacha IP: 146.231.123.80 port: 7080 packet IP: 146.231.123.80 packet port: 7080
Generating a SHA1 ID
4a3c8e35b98074a4cd74fdcf44a5a73834e7dca8
_
```

6. Then the Bootstrap server is running

Setting up Peers

1. Download the configuration file for the peer from http://www1.cs.columbia.edu/~salman/peer/ove

2. Open it with any text editor.

3. There are several fields in this file. See below for an explaination.

TCP_PORT: This is the listening port for incoming requests over TCP. The current implementation does NOT support TCP.

UDP_PORT: This is the listening port for incoming requests over UDP.

P2P: SHOULD be the same as that of the bootstrap server.

HASH_ALG: SHOULD be the same as that of the bootstrap server.

HASH_SIZE: SHOULD be the same as that of the bootstrap server.

REC: Set to '0' for iterative routing and '1' for recursive routing.

PARALLEL: Same as that of bootstrap. Not used by current implementation.

CMD: Specifies wether command line usage is desired. See this for using the command line interface

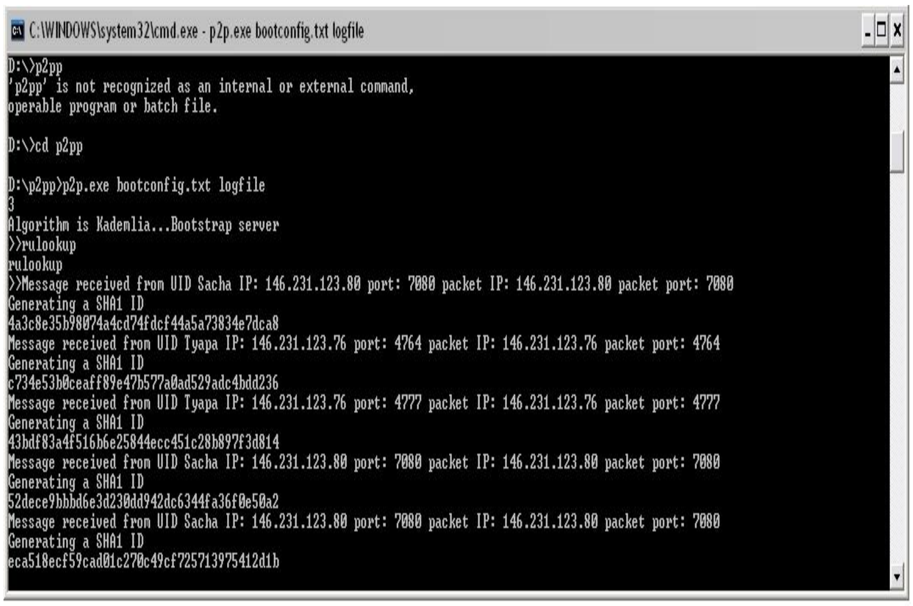
MODE: Set this to PEER (default). BOOTIP: IP address of the bootstrap server.

BOOTPORT: UDP port of the bootstrap server.

USERID: Name of the peer. This should be unique for each peer.

4. After editing the file, open a terminal and browse to the directory containing the executable and configuration file.

5. Type ">p2p.exe configp.txt logfile"



C:\WINDOWS\system32\cmd.exe - p2p.exe bootconfig.txt logfile

```
D:\>p2pp
'p2pp' is not recognized as an internal or external command,
operable program or batch file.

D:\>cd p2pp

D:\p2pp>p2p.exe bootconfig.txt logfile
3
Algorithm is Kademlia...Bootstrap server
>>rulookup
rulookup
>>Message received from UID Sacha IP: 146.231.123.80 port: 7080 packet IP: 146.231.123.80 packet port: 7080
Generating a SHA1 ID
4a3c8e35b98074a4cd74fdcf44a5a73834e7dca8
Message received from UID Tyapa IP: 146.231.123.76 port: 4764 packet IP: 146.231.123.76 packet port: 4764
Generating a SHA1 ID
c734e53b0ceaff89e47b577a0ad529adc4bdd236
Message received from UID Tyapa IP: 146.231.123.76 port: 4777 packet IP: 146.231.123.76 packet port: 4777
Generating a SHA1 ID
43bdf83a4f516b6e25844ecc451c28b897f3d814
Message received from UID Sacha IP: 146.231.123.80 port: 7080 packet IP: 146.231.123.80 packet port: 7080
Generating a SHA1 ID
52dece9bbbd6e3d230dd942dc6344fa36f0e50a2
Message received from UID Sacha IP: 146.231.123.80 port: 7080 packet IP: 146.231.123.80 packet port: 7080
Generating a SHA1 ID
eca518ecf59cad01c270c49cf725713975412d1b
```

6. Peer is running.

Setting up Clients

1. Download the configuration file for the client from http://www1.cs.columbia.edu/~salman/peer/ov

2. Open it with any text editor.

3. Fields are the same as Peer. Except

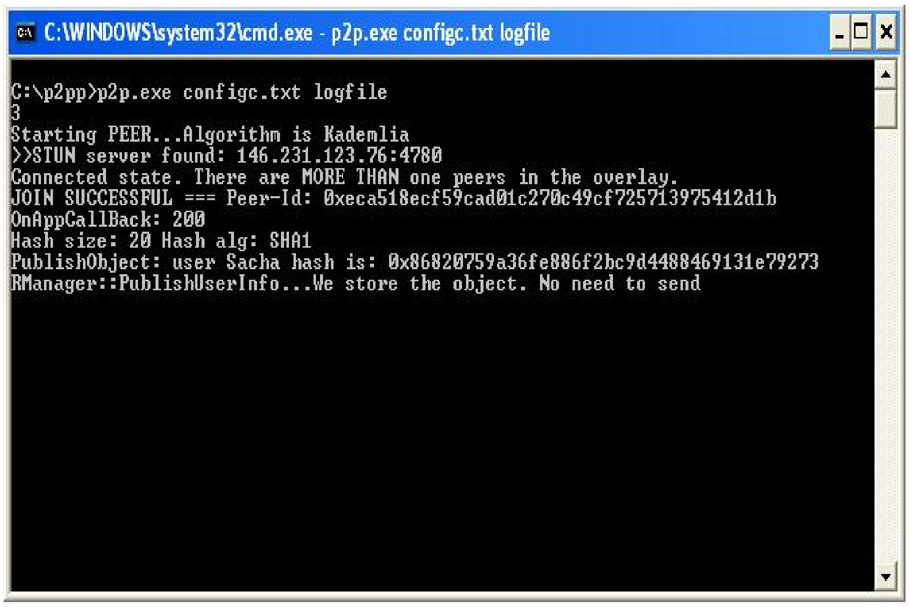
MODE: Set this to CLIENT (default).

BOOTIP: Set this to that of bootstrap server.

BOOTPORT:Set this to the UDP port of the bootstrap server.

4. After editing the file, open a terminal and browse to the directory containing the executable and configuration file.

5. Type ">p2p.exe configc.txt logfile"



```
C:\p2pp>p2p.exe configc.txt logfile
3
Starting PEER...Algorithm is Kademlia
>>STUN server found: 146.231.123.76:4780
Connected state. There are MORE THAN one peers in the overlay.
JOIN SUCCESSFUL === Peer-Id: 0xeca518ecf59cad01c270c49cf725713975412d1b
OnAppCallBack: 200
Hash size: 20 Hash alg: SHA1
PublishObject: user Sacha hash is: 0x86820759a36fe886f2bc9d4488469131e79273
RManager::PublishUserInfo...We store the object. No need to send
```

6. Client is running

Appendix C

Accompanying CD-ROM

The accompanying CD-ROM contains the following:

ErasmusTyapa.pdf This thesis in pdf format.

/**References** Electronic and Jabref files for the reference material cited in this thesis.

/**Executables** OpenWengo-P2PP and P2P.exe for Windows

/**SourceCode** The source code of the standalone OverCord peer-to-peer layer, the modified JAIN SIP Applet Phone that embeds OverCord, SIP2P, 39 Peers, PJNATH and P2PP