



UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTA DI SCIENZE MATEMATICHE, FISICHE E NATURALI

DIPARTIMENTO DI TECNOLOGIE DELL'INFORMAZIONE

DOTTORATO DI RICERCA IN INFORMATICA (INF/01 INFORMATICA)

SCUOLA DI DOTTORATO IN INFORMATICA, XXIII CICLO

A Middleware-Independent and SEcure Peer-to-Peer SIP architecture (MISE-P2PSIP)

TESI DI DOTTORATO DI RICERCA DI:
HOUNGUE YENUKUNME PELAGIE ELYSE

Matr. R07911

RELATORE

Prof. Ernesto Damiani

Università degli Studi di Milano, Italy

CORRELATORE

Prof. Roch Glitho

Concordia University, Canada

DIRETTORE DELLA SCUOLA DI DOTTORATO

Prof. Ernesto Damiani

Anno Accademico 2010 – 2011

To my parents who have supported me all
the way since the beginning of my studies.

Abstract

A Middleware-Independent and SEcure Peer-to-Peer SIP architecture (MISE-P2PSIP)

The Session Initiation Protocol (SIP) is the de facto standard for multimedia multiparty sessions signaling in Next Generation Networks (NGN). It is at the basis of a wide range of IP multimedia services. SIP specification and current usage relies on centralized servers. However, research has recently started on the integration of Peer-to-Peer (P2P) principles into SIP for harnessing the benefits of decentralization. The contribution of this thesis is fourfold.

Firstly, this thesis contributes to this research by proposing a novel architecture for P2P SIP. Our architecture is an overlay composed of a set of self-organized proxies and distributed registrars. Unlike other architectures proposed so far, our proposal does not require an extension to SIP messages and is P2P middleware-independent. This eases implementation, interoperability with legacy, and ensures portability.

Secondly, the thesis discusses the routing issues related to such environment. Indeed, introducing proxies in a P2P SIP overlay raises two important issues namely the *proxy topology building* and *proxy-level routing*. Thanks to *proxy topology building*, a proxy joining the P2P SIP finds its neighbors in the network of proxies. *Proxy-level routing* enables messages to be correctly routed in the network built by *proxy topology building*. This part of the thesis proposes a new framework for proxy topology building and proxy-level routing in our proposed architecture. Our framework is P2P infrastructure independent and general enough to be used by any P2P SIP architecture that meets a minimal set of requirements. It relies on a simple algorithm that builds the network of proxies as a ring, and on routing algorithms specially designed for the ring topology.

Thirdly, the thesis handles the Network Address Translation (NAT) traversal problem. Whereas P2P SIP architectures come with several benefits, they inherit NAT traversal issues from SIP world. NAT traversal issues occur because SIP messages must carry important communication parameters, including the IP ad-

dress and port number to be used for signaling and media streams. SIP clients behind NAT device are not aware of how they are seen from the public network. Consequently, SIP packets sent by a client behind a NAT device, contain private IP addresses in the message headers and in the message body. These addresses being private, cannot be used by the destination node for answering. Then, we propose in this thesis, an efficient solution that enables nodes behind a NAT device to participate in the P2P SIP network.

Fourthly, effective operation of our architecture relies on collaboration between the nodes playing important roles such as, proxy and registrar servers. Therefore, we provide solutions for identifying and alleviating non-cooperative behavior. We focus on proxy servers because they perform an important role in the transmission of signaling messages. Proxy servers can misbehave by misrouting the signaling messages or by hijacking SIP call sessions. This thesis proposes techniques to secure the routing of SIP signaling messages.

Keywords: SIP protocol, P2P computing, P2P SIP networks, Network topology, NAT traversal, Secure routing.

Résumé

A Middleware-Independent and SEcure Peer-to-Peer SIP architecture (MISE-P2PSIP)

SIP (Session Initiation Protocol) est un protocole standard de signalisation pour les sessions multimédia multi-parties dans les Réseaux de Nouvelles Génération (Next Generation Network). Il est à la base d'une large gamme de services multimédia pour Internet. La spécification SIP et son utilisation actuelle reposent sur des serveurs centralisés. Cependant, la recherche a récemment commencé sur l'intégration des principes des réseaux Pair-à-Pair (P2P) à la technologie SIP, pour exploiter les avantages de la décentralisation. La contribution de cette thèse est quadruple.

Premièrement, cette thèse contribue à cette recherche en proposant une nouvelle architecture P2P SIP. Notre architecture est un réseau de recouvrement composé d'un ensemble de serveurs mandataires ("proxy") auto-organisés et de serveurs d'enregistrement distribués. Contrairement à d'autres architectures proposées jusque là, notre proposition ne nécessite aucune extension pour les messages SIP et demeure indépendante de l'infrastructure P2P utilisée. Cela facilite la mise en œuvre, l'interopérabilité, et assure la portabilité.

Deuxièmement, la thèse aborde les problèmes de routage liés à un tel environnement. En effet, l'introduction de serveurs proxy dans un réseau de recouvrement P2P SIP, soulève deux questions fondamentales à savoir: la construction de la topologie des serveurs proxy et le routage par le biais de ces serveurs. Grâce à cette topologie, un nouveau nœud proxy joignant le réseau, pourra découvrir ses voisins dans le réseau des nœuds proxy. Le routage par l'intermédiaire des nœuds proxy permet aux messages d'être correctement acheminés dans le réseau. Cette partie de la thèse propose donc un nouveau système pour la construction de la topologie des proxys et le routage par le biais des proxys dans l'architecture que nous avons proposée. Notre système est indépendant de l'infrastructure P2P sous-jacente et est suffisamment général pour être utilisé par n'importe quelle architecture P2P SIP qui répond à un ensemble minimal d'exigences. Il s'appuie

sur un algorithme simple qui construit le réseau des proxys en anneau, et sur des algorithmes de routage spécialement conçus pour la topologie en anneau.

Troisièmement, la thèse traite le problème de la traduction d'adresses (Network Address Translation). En effet, bien que les architectures P2P SIP viennent avec plusieurs avantages, ils héritent le problème de traduction d'adresses propre à la technologie SIP. Ce problème survient parce que les messages SIP doivent transporter des paramètres de communication importants, y compris l'adresse IP et le numéro de port utilisés pour les flux de signalisation et de média. Les clients SIP se trouvant derrière un dispositif NAT, ne sont pas conscients de la façon dont ils sont vus depuis le réseau public. Par conséquent, les paquets SIP envoyés par un client derrière un périphérique NAT, contiennent des adresses IP privées dans les en-têtes et dans le corps des messages. Ces adresses étant privées, ne peuvent pas être utilisées par le nœud de destination dans sa réponse. Ainsi, nous proposons dans cette thèse une solution efficace permettant aux nœuds derrière un périphérique NAT de pouvoir participer au réseau P2P SIP.

Quatrièmement, le bon fonctionnement de notre architecture repose sur une bonne collaboration entre les nœuds jouant des rôles importants tels que, les serveurs proxy et les serveurs d'enregistrement. Par conséquent, nous proposons des solutions permettant d'identifier et d'atténuer les comportements non coopératifs. Nous nous concentrons sur les serveurs proxy, car ils jouent un rôle très important dans la transmission des messages de signalisation. Les serveurs proxy peuvent mal se comporter en optant pour un mauvais acheminement des messages de signalisation ou en détournant les sessions d'appels SIP. Cette thèse propose des techniques pour sécuriser l'acheminement des messages de signalisation SIP.

Mots clés : Protocole SIP, Systèmes pair-à-pair, Réseaux P2P SIP, Topologie de réseaux, Traduction d'adresse NAT, Routage sécurisé.

Acknowledgments

I would like to use the occasion of this thesis to thank all the people who helped me to reach this important goal.

First of all, I would like to sincerely thank my supervisor, Ernesto Damiani. It has been an honor for me working with him. I would like to express him all my gratitude for giving to IMSP Institute in Africa the opportunity to promote young students by allowing them undertaking new and great experiences in research field abroad, especially in the famous University of Milan.

I would like to thank my co-supervisor Roch Glitho for giving to me, an opportunity of staying in Concordia University in Canada for a useful period of research allowing me to work on my research topic and collaborate with other students by greatly learning from their academic system. Thank also for his constant guidance and great contributions on all my Ph.D thesis work.

I would like to thank Stelvio Cimato, for his noticeable contribution on the security aspects of my work.

I would like to thank Hechmi Khlifi and Fatna Belqasmi for their great contributions in the various important aspects of my work.

I would like to thank Nadia Bennani, Jajodia Sushil and Jean-Charles Grégoire, for having dedicated their precious time to review the thesis and whose valuable feedback helped improving the presentation and the scientific content of this work.

I would like to thank my husband for his constant support, guidance, help and encouragement over the years, which made my Ph.D experience more pleasant and productive.

Last, but not least, I would like to thank my family. All the pages of this thesis would not be sufficient to express my gratitude to them; their teaching, their support and their love have been and will always be a fundamental reference point for me.

Contents

Abstract	iv
Résumé	vi
Acknowledgments	viii
Part I Introduction and Background	2
Chapter I Introduction	3
I.1 Problem statement and goals	4
I.2 Original contributions	4
I.2.1 Interoperable and portable P2P SIP architecture	4
I.2.2 Topology building and proxy-level routing	5
I.2.3 NAT traversal solutions in P2P SIP	5
I.2.4 Secure SIP signaling message routing	5
I.3 Thesis structure	5
Chapter II Background	7
II.1 P2P overlay	7
II.1.1 Overview	7
II.1.2 P2P overlay architectures	9
II.1.3 Distributed Hash Table	16
II.2 SIP Protocol	17
II.2.1 Overview	17
II.2.2 SIP components	18
II.2.3 SIP messages	19
II.2.4 SIP Protocol operation	22
II.3 P2P SIP	25
II.3.1 Overview	25
II.3.2 High level description	26
II.3.3 Reference model	26
II.4 Current candidate projects for P2P SIP implementation	27
II.4.1 CISCO P2P SIP Project	27
II.4.2 Columbia P2PP Project	28
II.4.3 SIPDHT2 Project	28
II.4.4 Kademlia dSIP	28
II.4.5 Huawei's SEP Peer and Client Protocol	28

II.4.6	P2PNS: A Secure Distributed Name Service for P2P SIP	28
II.4.7	39 peers project	29
II.4.8	Peer-to-Peer SHip	29
II.4.9	RELOAD	29
II.5	Conclusion	29

Part II P2P SIP architecture 30

Chapter III Related work on existing P2P SIP architectures 31

III.1	Requirements	31
III.2	Related work and analysis	31
III.2.1	SIP-using-P2P	32
III.2.2	P2P-over-SIP	35
III.3	Conclusion	36

Chapter IV MISE-P2PSIP architecture 37

IV.1	Assumptions and overall architecture	37
IV.2	Self-organization	38
IV.2.1	Groups' creation	39
IV.2.2	DC node' setting up procedure	39
IV.2.3	Node joining procedures	39
IV.2.4	Node leaving procedures	40
IV.3	Call procedures	42
IV.4	Call scenarios	42
IV.4.1	The two communication parties are in the same group	43
IV.4.2	The two communication parties are in different groups	43
IV.5	Multi-domains communication	43
IV.6	Implementation	44
IV.6.1	Software architecture	44
IV.6.2	Prototype	46
IV.7	Evaluation	52
IV.7.1	Simulation Scenario	52
IV.7.2	Results and discussions	52
IV.8	Conclusion	54

Part III Routing issues in P2P SIP overlay 56

Chapter V Topology building and routing in MISE-P2PSIP 57

V.1	Motivation	57
V.2	Requirements	57
V.3	Related work and analysis	58
V.4	Proposed framework	60
V.4.1	Architectural principles	60
V.4.2	Topology creation and maintenance	60
V.4.3	Routing algorithms	62
V.4.4	Example of routing table update	64

V.4.5	Call scenario	68
V.4.6	Evaluation	69
V.5	Conclusion	72
Chapter VI	NAT traversal solution for signaling messages and media streams	73
VI.1	Definition and background	73
VI.2	Motivation	74
VI.3	Requirements	74
VI.4	Related work and analysis	75
VI.4.1	Standardized techniques	75
VI.4.2	Non-standardized techniques	76
VI.5	NAT traversal solution	77
VI.5.1	Used techniques	77
VI.5.2	Node group creation	77
VI.5.3	NAT traversal for SIP signaling messages	79
VI.5.4	NAT traversal for media flows	79
VI.6	Call scenarios	81
VI.6.1	The two Client devices are behind the same NAT and are connected to the same proxy	82
VI.6.2	The two Client devices are behind different NATs and are connected to different proxies	82
VI.7	Implementation plan	83
VI.8	Experimental results	84
VI.9	Conclusion	87
Part IV	Secure routing	88
Chapter VII	Related work on secure routing protocols and mechanisms	89
VII.1	Motivations	89
VII.2	Security requirements	89
VII.3	Related work on secure routing protocols and mechanisms	90
VII.3.1	Secure routing in mobile Ad hoc networks	90
VII.3.2	Security in conventional SIP	92
VII.3.3	Secure routing in Peer-to-Peer systems	95
VII.4	Conclusion	96
Chapter VIII	Attack models and secure schemes for signaling messages routing in MISE-P2PSIP	97
VIII.1	Call hijacking	97
VIII.1.1	Background	97
VIII.1.2	Using HTTP Digest authentication	100
VIII.1.3	Our Secure scheme against call hijacking attack	100
VIII.2	Signaling message misrouting	105
VIII.2.1	The misrouting attack	105
VIII.2.2	Misrouting attack tests	106
VIII.2.3	Our secure routing mechanism	111

VIII.3 Conclusion	113
Part V Conclusions and future works	114
Chapter IX Conclusions and future works	115
IX.1 Summary of the problems and contributions	115
IX.2 Future directions	116
IX.2.1 Additional experimentations	116
IX.2.2 SIP service reliability	116
IX.2.3 Locality-aware self-organization	117
Publications	118
Bibliography	119
Appendix List of abbreviations	125

List of Figures

Figure II.1	Traditional C/S model	7
Figure II.2	A general P2P model	8
Figure II.3	Lookup in the pure P2P network architecture	11
Figure II.4	Centralized indexing in hybrid P2P network	11
Figure II.5	Decentralized indexing in hybrid P2P network	12
Figure II.6	Chord ring	13
Figure II.7	JXTA architecture	14
Figure II.8	SIP components	19
Figure II.9	SIP requests format	19
Figure II.10	SIP responses format	20
Figure II.11	SIP registration scenario	23
Figure II.12	Basic call flow with servers	24
Figure II.13	P2P SIP Reference model	27
Figure IV.1	MISE-P2PSIP Overall architecture	38
Figure IV.2	Joining procedures flow charts	40
Figure IV.3	SIP Call in the same group	43
Figure IV.4	SIP Call between different groups	44
Figure IV.5	Software architecture	44
Figure IV.6	Launching interface	47
Figure IV.7	Destination node IP address discovery	47
Figure IV.8	INVITE message sent by the source node	48
Figure IV.9	SIP messages on destination node side	48
Figure IV.10	Example of published IP address/port advertisement	49
Figure IV.11	Example of Chord network	49
Figure IV.12	Real Chord network	50
Figure IV.13	Real Chord network: call interface	50
Figure IV.14	INVITE message sent by the caller	51
Figure IV.15	INVITE message received by the callee	51
Figure IV.16	Final dialog state on the callee side	51
Figure IV.17	Proxy based vs. DHT based P2P SIP RTT	53
Figure IV.18	Effect of proxy caching on RTT	54
Figure IV.19	Effect of caching on network load	54
Figure V.1	New topology building proxies in ring	60
Figure V.2	Routing table creation/update for other proxies when a new proxy joins	63
Figure V.3	Creation of routing table of joining proxy	63
Figure V.4	Update of remaining proxies routing tables upon leaving	64

Figure V.5	Initial state of the proxy ring in the scenario	64
Figure V.6	Proxy ring after P5 joining	65
Figure V.7	Proxy ring after P6 joining	66
Figure V.8	Proxy ring after P7 joining	67
Figure V.9	Call establishment scenario	68
Figure V.10	Proxy level routing algorithm vs. OSPF: network load	70
Figure V.11	Proxy level routing algorithm vs. OSPF: convergence time	70
Figure V.12	Proxy level routing algorithm vs. DHT-based routing: Network load . . .	71
Figure V.13	Proxy level routing algorithm vs. DHT-based routing: RTT	71
Figure V.14	Proxy level routing versus “Right alternative”: RTT measurements . . .	72
Figure V.15	Proxy level routing versus “Right alternative”: Network load	72
Figure VI.1	Example of nodes’ topology behind NATs	75
Figure VI.2	Role assignment procedure flow chart	78
Figure VI.3	Piggybacked 200 OK response	78
Figure VI.4	NAT traversal for signaling messages	80
Figure VI.5	NAT traversal procedure for media flows	81
Figure VI.6	Call scenario for two nodes behind the same NATs	82
Figure VI.7	Call scenario for two nodes behind different NATs	83
Figure VI.8	Experimental environment	85
Figure VI.9	SIP signaling flows	86
Figure VI.10	RTP packets through media relay	87
Figure VIII.1	Call hijacking attacks based on requests redirection	98
Figure VIII.2	Encryption based on Identity	99
Figure VIII.3	Identity Based Encryption/Signature scheme at Alice side	101
Figure VIII.4	Decryption/verification scheme at Bob side	102
Figure VIII.5	Call procedure against call hijacking	104
Figure VIII.6	Example of MISE-P2P SIP architecture	106
Figure VIII.7	Completed transactions with static behavior	109
Figure VIII.8	Delayed transactions with probabilistic behavior	109
Figure VIII.9	Completed/in time transactions in scenario A	110
Figure VIII.10	Completed/in time transactions in scenario B	110
Figure VIII.11	Completed/in time transactions in scenario C	111
Figure VIII.12	Completed/in time transactions in scenario D	111

List of Tables

Table III.1	State of art summary	36
Table V.1	P2 initial routing table	64
Table V.2	P2 routing table after P5 joining	65
Table V.3	P2 routing table after P6 joining	66
Table V.4	P2 routing table after P7 joining	67
Table VIII.1	Rogue nodes dispositions	108

Part I

Introduction and Background

This part of the thesis is composed of two chapters. Chapter I defines the problem to solve and underlines the original contributions in the thesis. Subsequently, it outlines the thesis structure. Chapter II introduces the background on the main concepts discussed in our work such as VoIP, P2P, P2P SIP, etc.

Chapter I

Introduction

The worldwide success of the Internet and of its technologies has overthrown the world of telecommunications. Today, most Internet users have become familiar with Internet Protocol (IP) telephony (also known as Voice over IP or VoIP) thanks to the popularity of applications like Skype [2]. Thanks to IP telephony, voice, data, and video can be exchanged using the same communication infrastructure. Recently, many capabilities targeted to enterprises have been added to VoIP, supporting using IP telephony on corporate Wide Area Networks (WAN). An important benefit of IP telephony is cost savings, especially for corporations with large data networks. VoIP relies on a variety of protocols, including H.323 [20], MGCP (Media Gateway Control Protocol) [21], and SIP (Session Initiation Protocol) [1] for signaling and RTP (Real-time Transport Protocol) [18] for media stream transport. Among the existing open signaling protocols, the most widespread are H.323 and SIP. The H.323 protocol was originally created for multimedia sessions on Local Area Networks (LAN). It is at the same time a protocol and a whole architecture, including a family of other protocols and standards. SIP is a system of signaling specifically designed for IP networks. For this reason, in spite of the fact that SIP is less wealthy in services provided by its direct competitor, it has raised a great interest in Internet and telecommunication community.

P2P VoIP refers to the jointly usage of the P2P technology and VoIP. This method is used to carry out voice using Internet in a decentralized manner. The absence of a central server allows communications to be faster because users are directly connected to each other, so that voices do not pass through an intermediate server. Skype is the first implementation of this technology. Other important P2P VoIP systems are Google talk, Google voice, Yahoo Messenger, etc [71].

In order, to circumvent all drawbacks of centralized systems that rely on SIP, a Peer-to-Peer (P2P) SIP-based communication system was proposed, since conventional SIP-based communications are strongly based on centralized servers. Initially, P2P SIP was the name of the Internet Engineering Task Force (IETF) Working Group (WG) in charge of the standardization of decentralized SIP architectures. Many research activities are being carried out on the integration of P2P principles into SIP to capitalize on the benefits of decentralization. In 2003, the SIPpeer project at Columbia University [10, 19] and the SOSIMPLE project at William & Mary College [6] did some pioneering works in the field of P2P SIP communication systems. In the following years, P2P SIP research has attracted great attention both from academia and industry (e.g. Cisco, Nokia, Ericsson, HuaWei, etc). Many other solutions have been proposed [22, 23, 24]. However, P2P SIP cannot yet be considered mature. Many of the proposed P2P SIP techniques to date are not interoperable and portable because they extend SIP standard protocol and are based on specific P2P middleware. In this thesis, we contribute to P2P SIP research by proposing a novel

architecture for P2P SIP. Our architecture is an overlay composed of a set of self-organized proxies and distributed registrars. Unlike other architectures proposed to date, it does not extend SIP messages and is P2P-middleware-independent. This eases implementation, interoperability with legacy systems and ensures portability. The following sections detail our research question and provide the motivation of our original contributions to this topic.

I.1 Problem statement and goals

The Session Initiation Protocol (SIP) is an IETF (Internet Engineering Task Force) protocol for the establishment, modification and tearing down of multimedia sessions [1]. It has been selected by most Next Generation Network (NGN) forums as the standard signaling protocol for multimedia multiparty sessions.

Peer-to-Peer (P2P) networks are logical overlay networks built on top of physical networks. P2P computing is an alternative to the centralized client/server computing on which SIP is currently based. Research work has recently begun on the integration of P2P principles into SIP. The objective is to bring to the SIP world the benefits generally associated with P2P systems (e.g. self-organization, scalability, decentralization).

Interoperability is an important feature in P2P SIP because it allows developers to use existing SIP toolkits. In order to ensure interoperability at SIP level, SIP extensions should be avoided or at least kept to a minimum. Portability is also an important feature of P2P SIP as it eases the reuse of existing P2P SIP systems on various P2P middleware.

This thesis contributes to research activities on P2P SIP by proposing a novel architecture that achieves interoperability with legacy by not extending SIP messages and ensures portability by being P2P-middleware-independent. Our architecture is an overlay where SIP entities such as clients, proxies and registrars communicate using the SIP protocol as currently specified. Proxies and registrars are self-organized and registrars are distributed. Moreover, our proposal contains efficient proxy topology building and routing mechanisms. Furthermore, we define a NAT traversal solution to allow effective transmission of SIP signaling messages and media streams in presence of NAT devices. Finally, we propose a technique to secure signaling messages routing in our P2P SIP framework. The next section will explicitly detail our original contributions.

I.2 Original contributions

As argued in the previous section, this work mainly aims to describe a P2P SIP-based communication system which enables effective end-to-end communication by being fully compliant to SIP standard and allows portability by being middleware-independent. The proposed system uses its own routing algorithm among its proxy servers and handles NAT traversal issues. In addition, we propose new mechanisms to secure the routing of SIP signaling messages.

I.2.1 Interoperable and portable P2P SIP architecture

Unlike other proposals, our P2P SIP architecture is P2P middleware-independent and does not extend standard SIP messages. We review the P2P SIP architectures proposed to date and pinpoint their shortcomings. Then, we illustrate the principles of our new architecture and the self-organization (Join/Leaving) procedures underlying P2P SIP network set up. We provide a proof-of-concept prototype using JAIN SIP [17], JXTA (Juxtapose) [16] and Chord implemented upon

ProActive [67] middleware. In addition, we provide more performance results through simulation and discuss them.

I.2.2 Topology building and proxy-level routing

In P2P SIP architecture, the service for locating SIP nodes on the network is executed by peer entities in the P2P overlay. Having SIP proxies as first-class entities raises two important issues: proxy topology building and proxy-level routing. Thanks to proxy topology building, a proxy joining the P2P overlay knows how to find its neighbors in the network of proxies. Then, proxy-level routing enables messages to be routed in the SIP network built by proxy topology building.

This part of the work proposes a novel framework for proxy topology building and proxy-level routing. The framework is general enough to be used on any P2P overlay that meets a minimal set of requirements. Independence from the P2P infrastructure is aimed at supporting seamless integration of multiple SIP communication environments. Our framework relies on an algorithm that builds the network of proxies in a ring and uses its own routing algorithm to route calls. We describe our novel routing algorithm used between proxies on the ring and evaluate its performance through simulations.

I.2.3 NAT traversal solutions in P2P SIP

Whereas P2P SIP architectures come with several benefits, they inherit Network Address Translation (NAT) [77] traversal issues from SIP world. Indeed, NAT traversal issues occur because SIP messages contain important communication parameters including the IP address and port number to be used for signaling and media streams. However, SIP clients are not aware of how they are seen from the public network. Consequently, SIP packets sent by a client behind a NAT, contain private IP address in the message headers and in the message body. These addresses being private, they cannot be used by destination for answering. Many NAT traversal solutions have been proposed to solve this problem. However, they are not suitable for our framework because they do not fit our performance and portability requirements. We describe a technique designed for our P2P SIP architecture that provides efficient NAT traversal.

I.2.4 Secure SIP signaling message routing

Within the P2P SIP overlay, where no central authority is present, each node acting as a server can misbehave. More servers can also collude to misbehave. For instance, a proxy node could drop, wrongly alter, delay or misroute a message. There is a need to secure the routing of SIP signaling messages in such a way that those messages are delivered correctly, since misrouting may disrupt call set up. In this part of our work, we focus on attacks performed by a malicious proxy server because the proxy node has a critical and important role in SIP networks. We have performed different misrouting attack tests considering different scenarios. In addition, we propose secure schemes in order to secure the routing of SIP signaling messages.

I.3 Thesis structure

The thesis is structured in several parts.

1. The remainder of part I includes Chapter II which introduces the overall background on relevant topics covered by P2P SIP research area. Basically, we describe VoIP system, P2P computing and SIP protocol specification. We finally focus on P2P SIP communication systems. Basically, we introduce the P2P SIP network, the P2P SIP reference model and the current available projects.
2. In part II on P2P SIP architecture, chapter III describes the main requirements and provides the analysis of the related work on the existing P2P SIP architectures. Moreover, chapter IV gives a full description of our novel P2P SIP architecture. Indeed, the chapter includes the assumptions and the main principles of this architecture. Furthermore, it describes the self-organization procedures, useful for nodes joining and leaving and shows the call procedures. Finally, the prototype and other relevant simulation results have been discussed.
3. In part III on routing issues, chapter V presents the motivation of proposing a proxy topology building and proxy-level routing algorithm for our framework. It also makes a survey of related work on existing routing algorithm. Subsequently, it describes our proposed framework on topology building and routing by emphasizing the requirements, the routing algorithm and the simulation results. Finally, chapter VI highlights the motivation and the state of art on NAT traversal solutions. In addition, it describes our NAT traversal solutions for SIP signaling messages and for media streams. Relevant experimental results have been provided.
4. In part IV on security challenges, chapter VII gives the motivation and point out the requirements on security in P2P SIP. Moreover, it discusses the related work on secure routings protocols and mechanisms. Finally, chapter VIII underlines two main attack models which are misrouting attack and call hijacking attack and defines secure approaches to tackle those attacks in our P2P SIP framework.
5. Part V includes the last chapter IX which concludes the thesis and suggests future works.

Chapter II

Background

This chapter provides some background to our research. It emphasizes the key concepts regarding the P2P overlay networks. Subsequently, we describe the basics of the SIP signaling protocol including its components, the kind of messages used and its operations. Finally, the main concepts behind P2P and SIP are underlined and the P2P SIP reference model released by IETF is described following by the introduction of the current available P2P SIP projects implementation.

II.1 P2P overlay

This section gives a basic background on P2P overlay. First, it provides an overview of P2P paradigm. Subsequently, we classify the different types of P2P networks according to their structure. Finally, we discuss the Distributed Hash Table concept.

II.1.1 Overview

In Client-Server (C/S) models, each node plays the role of either client or server. The resource such as storage or computing capability can be shared between client and server. The server in C/S model is factually a central control point. A traditional C/S model is illustrated in figure II.1. Instead, in Peer-to-Peer (P2P) model, each node plays role of client and server. As a client, it can query and download its wanted objects from other peers. As a server, it can provide objects to other nodes at the same time.

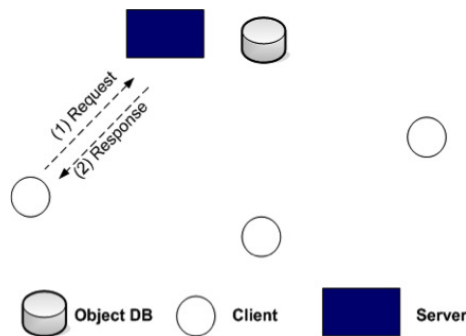


Figure II.1: Traditional C/S model

A simple definition of P2P has been given by Oram in [25] as: “P2P is a class of applications that take advantage of resources storage, cycles, content, human presence available at the edges

of the Internet. Peer-to-peer nodes must operate outside the DNS (Domain Name System) and have significant or total autonomy of central servers” [25], because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses. In other word, P2P is a special distributed system on the application layer, where each pair of peers can communicate each other through an application layer routing protocol. The participants in the P2P network can act as a server and a client at the same time. They are accessible by other nodes directly, without going through intermediary entities. A general P2P network is pictured in figure II.2. Each peer keeps an object (such as file, music, MP3, MPEG, etc.) database. Each peer can request an object from other peers through a logical connection on P2P layer.

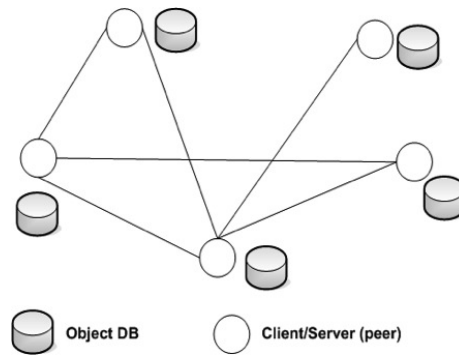


Figure II.2: A general P2P model

There are several concepts underlying P2P systems [123]: resources sharing, decentralization and self-organization. Resource sharing implies that applications cannot be set up by a single node. The resources shared can be physical resources such as disk space, CPU or network bandwidth, as well as, logical resources such as services or different kinds of information. Decentralization feature is an immediate consequence of resources sharing. Decentralization is in particular interesting in order to avoid single point of failure and bottlenecks. When a p2p system is fully decentralized, there exists no longer a node that centrally coordinates its activities or a database that stores global information about the system in centralized way. Therefore, nodes have to self-organize themselves, based on whatever local information is available and interacting with locally reachable nodes that are their neighbors. Self-organization also allows the overlay to be built in a distributed manner and be robust enough to support dynamic changes of the peers which take part in the overlay. Moreover, the overlay should continuously adapt to changes in the network such as bandwidth and latency variances. The system should be self-improving, meaning that the overlay should evolve towards a better structure as more information becomes available.

There are in general four phases for a P2P node according to lifetime cycle: join, query, download, and leaving. First, a just entering node must actively join the P2P system. During joining procedure, it may get some basic information (such as its neighbors) to start up, and simultaneously publish information about the objects it holds. Secondly, the node can issue query for objects it wants. At this time, P2P location protocol will allow the node to determine destination node, while P2P routing protocol can route query messages to destination node. Third, the node can directly download the object from the destination node if the query successes. Finally, the node will announce its departure in the ideal case. Thus, three important components of P2P are: neighbor finding, location protocol, and routing protocol. These components use the P2P topology, which is structured or unstructured, and self-organized by P2P nodes themselves.

II.1.2 P2P overlay architectures

Overlay networks are built on top of one or more existing networks. The links in the overlay network are virtual and can correspond to a multi-hop path on the underlying IP network. Some examples of overlay networks are cloud computing, Peer-to-Peer networks, and client-server applications since their nodes run on top of the Internet. P2P overlay networks aim to implement services that are not available in the underlying IP network [119].

Because of the virtual nature of the links between the peers, the topology of the P2P overlay network differs from the IP network it is built on. The P2P overlay network topology is determined by a specific algorithm that defines which peers should have a virtual link between each other. P2P networks are distributed systems in nature, without any hierarchical organization or centralized control.

As anticipated in the previous section, the literature identified two classes of P2P overlay network [52]: Unstructured and Structured.

a) Unstructured architectures

Unstructured P2P networks are so many and each P2P network has so many different properties, that there is no single classification criterion. Therefore, the classification is usually made according to the differences of search mechanisms and logical topology. Unstructured P2P networks do not impose any structure on the overlay networks. Peers connect in random way. As discussed previously, decentralization is one of the major concepts of P2P systems and is related to distributed storage, processing, information sharing and also control information. Ideally, unstructured P2P systems would have absolutely no centralized system, but in practice there are several types of unstructured systems with various degrees of decentralization. Thus, the current unstructured P2P architectures can be classified in two types [26]: Decentralized or pure P2P architecture and hybrid architecture.

1) Decentralized or pure P2P architecture

Decentralized P2P systems are serverless networks where every peer has equal functionality and responsibilities for routing messages. Object query is executed hop-by-hop, on the P2P network till success/failure or timeout. Searches in pure P2P networks are often done in a broadcast manner with a flooding algorithm where the query messages have a Time To Live (TTL) field defining the number of hops, through which the query message can travel.

Pure P2P systems are inherently scalable. Scalability in the system is usually restricted by the amount of centralized operation necessary. These systems are inherently fault-tolerant, since there is no central point of failure and the loss of a peer or even a number of peers can easily be compensated. They have a greater degree of autonomous control over their data and resources. However, such systems present slow information discovery and there is no guarantee about quality of services. In addition, it is difficult to predict the system behavior because of the lack of a global view of the system.

An example of the lookup process in a decentralized unstructured P2P network is depicted in figure II.3. Peer A initiates a lookup and the requests are flooded in the network with the TTL field value of 2 in the first message. The requested resource is found in peer B.

As we can see from figure II.3, only four messages (phases 1-4) are needed to complete the lookup and data is exchanged directly between peer A and B at phase 5. However, the lookup sends lots of additional messages that do not reach any peer holding the desired resource. In

large networks with thousands of peers, the unstructured decentralized approach does not scale well because of the amounts of traffic it produces. The lookup process is also slow, because when queries are flooded, only the neighbor peers can be contacted using one routing hop. To reach a more distant peer, the number of hops becomes very large. Some decentralized P2P networks use random walk queries where the targets of the query messages are chosen randomly. This improves the search efficiency but even with random walk, efficiency still remains lower than in the centralized P2P networks [29]. Freenet and early implementations of the Gnutella protocol [28] are examples of the decentralized model.

- Example of Gnutella

In Gnutella [28], there is neither a centralized directory nor any precise control over the network topology or object placement in such architecture as the typical “Gnutella”. Gnutella is a decentralized file-sharing system whose participants self-organize a virtual mesh network running in a P2P fashion for distributed file search. In order to participate in Gnutella, a node first must connect to a known Gnutella node to get lists of some existing Gnutella nodes for start-up. To find a file, a node issues queries to its neighbors. The most typical query method is flooding, where the query is broadcasted to all neighbors within a certain radius or constrained by P2P TTL mechanism. This unstructured architecture is very resilient to nodes entering and leaving the system. However, the current flooding-based lookup mechanism is un-scalable, since it generates large loads on the network participants.

Recently, some works [26] try to overcome this disadvantage proposing two mechanisms: “dynamic TTL setting or expanding ring” and “k-walker random walk”. But “k-walker random walk” may have large lookup length (latency). So, object replication mechanisms [39][40] (such as uniform replication, proportional replication, square-root proportional replication, and log-form replication) are proposed at the same time to reduce the lookup length. Lookup messages amount and length can be simultaneously decreased using cache mechanisms such as: cache some objects in the reverse path of queries.

2) Hybrid architectures

In hybrid P2P systems, there is a central server that maintains directories of information about registered users to the network, in the form of meta-data. The data exchange is done between two peer clients. Two kinds of hybrid systems exist: centralized indexing and decentralized indexing.

- Centralized indexing

In this type of hybrid P2P systems, a central server maintains an index of the data or files that are currently being shared by active peers, in the form $\langle \text{object-key}, \text{node-address} \rangle$. Each incoming node needs to actively notify this server about the object it holds. Each peer maintains a connection to the central server, through which the queries are sent. Such systems are simple and operate quickly and efficiently for discovery information. Searches are comprehensive and can be guaranteed. However, because of central servers, those systems have a single point of failure. They are not inherently scalable, because of limitations on the size of the database and its capacity to respond to queries.

An example of this P2P architecture is Napster [27]. Figure II.4 illustrates the lookup process in a centralized indexing architecture. The requesting peer sends the query to the server pool (phase 1) which holds the information about the connected peers and their resources. The server

pool then performs the search for the requested resources (Phase 2) and responds with the address of the peer holding the requested resource (Phase 3). Only the actual data transfer (data exchange) is made directly between the peers (Phase 4).

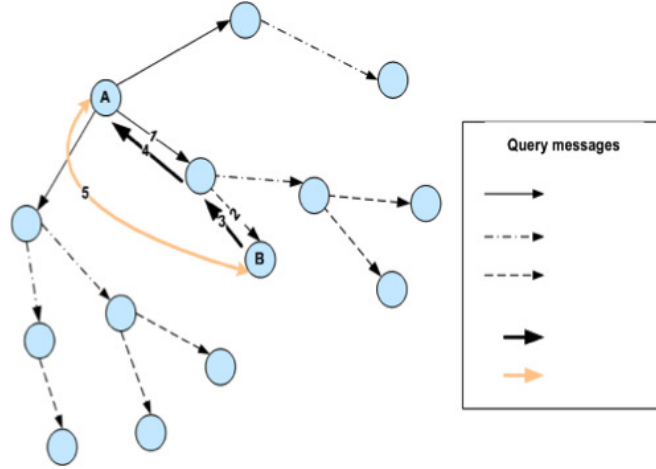


Figure II.3: Lookup in the pure P2P network architecture

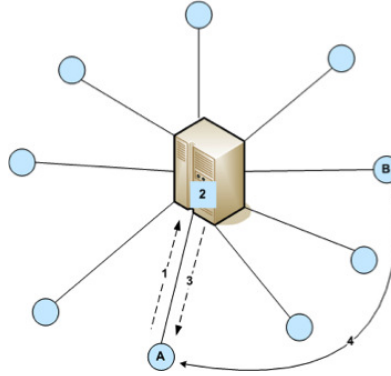


Figure II.4: Centralized indexing in hybrid P2P network

- Decentralized indexing

In decentralized indexing, a central server registers the users to the system and facilitates the peer discovery process. In these systems some of the nodes assume a more important role than the rest of nodes. They are called “supernodes”. Supernodes maintain the central indexes for the information shared by local peers connected to them and forward search requests on behalf of these peers. Thus, queries are sent to Supernodes and not to other peers. Only the supernodes participate in the peer and resource lookup [30]. Modern Gnutella implementations (e.g. Gnutella2) as well as Kazaa [124] and Morpheus [123] are examples of decentralized indexing systems. In such systems, peers are automatically elected to become Supernodes, if they have sufficient bandwidth and processing power. A central server provides to new joining peers, a list of one or more Supernodes to which they can connect.

Figure II.5 illustrates the lookup process in a decentralized indexing architecture. As it can be seen, from the perspective of an ordinary peer, the decentralized indexing is similar to the centralized one. The difference is that in the decentralized indexing, the supernodes perform the search for the requested resources among themselves in a decentralized way (phases 2-5).

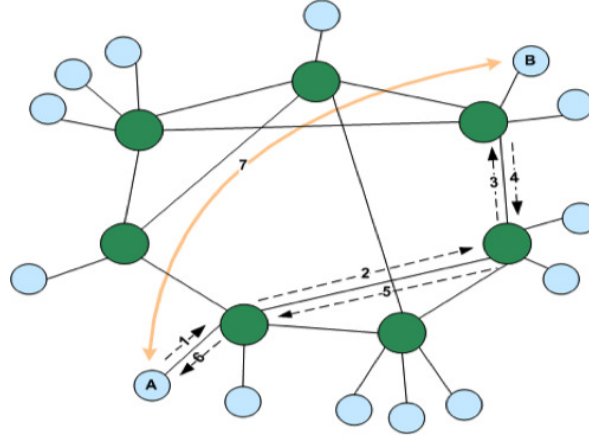


Figure II.5: Decentralized indexing in hybrid P2P network

In comparison with purely decentralized systems, hybrid decentralized indexing systems reduce the discovery time and reduce the traffic on messages exchanging between nodes. With respect to centralized indexing, hybrid decentralized indexing systems reduce the workload on central server but they present slower information discovery. In addition, decentralized indexing systems do not present unique point of failure and are inherently scalable. If one or more supernodes fail, the nodes connected to them can open new connection with others, and the network will continue to operate. In the case a large number or even all supernodes go down, the ordinary peers become supernodes themselves.

In conclusion to this sub-section on unstructured P2P architectures, best choice of architecture depends on what the P2P network will be used for and who will be using it. If an entity wants or must be in control of the network, as it may happen where the P2P network is used to deliver commercial services, decentralized indexing architecture is out of the question. For hybrid networks, some degree of operator control can be included because of their dependency of servers or super-peers [31].

b) Structured architectures

This type of architecture has no central directory server, but has a significant amount of structure. “Structure” means that the P2P network topology is tightly controlled (such as Mesh [32] [33][34][41], Ring [35] [43] [45], d-dimension Torus [36], K-ary tree [42], SkipList [46] and butterfly [37][38]), and files are placed not at random nodes but at specified locations that will make subsequent queries easier to satisfy. Such structured P2P systems often support a hash-table-like interface, and it is currently quite prevalent in the research literature. It uses precise placement algorithms and specific routing protocols to make the searching efficient. The object query is also executed hop-by-hop through the structured topology and is sure to be successful after some deterministic hops under ideal case. Each peer is given an identifier when it joins the network. The identifier defines the logical location of the peer in the overlay and thereby also the set of other peers it connects to. The resources that are stored in the network also get identifiers. With these identifiers the address of the node holding that resource can be found. The algorithms that are used for these purposes are called Distributed Hash Tables (see next section) [33][36].

Peer and content lookup is efficient in the structured P2P networks because the search mechanisms can be made simple, as based on the identifier being searched the querying node already has an idea of the location of the searched resource. Structured architectures can guarantee location

of a given target within a bounded number of hops. This guarantee holds also for queries for resources that do not exist in the network. Non-existence can also be verified within the same bounded number of hops.

Examples of P2P middleware that belong to this category are: Plaxton [18], Tapestry [33], Pastry [34], Chord [35], CAN (Content Addressable Network) [36], JXTA [54], and Kademlia [55]. Following, we give some details on Chord, JXTA, ProActive [67] and Kademlia middleware which have been used in the thesis to carry out our simulations and implementations.

i. Chord

Chord is a scalable protocol for lookup in a dynamic Peer-to-Peer system with frequent node arrivals and departures. It provides a decentralized P2P lookup service that stores key/value pairs for distributed data items. The Chord protocol supports just one operation: given a key, the node responsible for storing the key's value can be determined using a hash function that assigns an identifier to each node and to each key (by hashing the node's IP address and the key) [35]. Chord uses a variant of consistent hashing [53] to assign keys to Chord nodes. Consistent hashing tends to balance load, since each node receives roughly the same number of keys, and involves relatively little movement of keys when nodes join and leave the system. Each key k is stored on the first node whose identifier ID is equal or follows k in the identifier space. In Chord, active nodes will form a connected ring topology under ideal case. An example of a Chord ring is depicted in figure II.6.

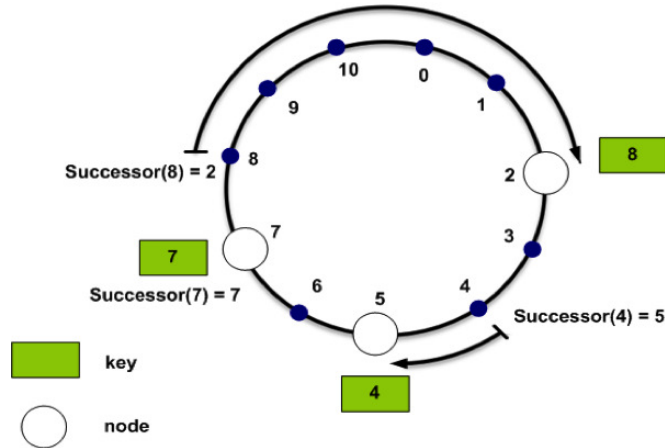


Figure II.6: Chord ring

A key k is assigned to the node whose identifier is equal to or greater than the key's identifier. This node is called $\text{successor}(k)$ and is the first node clockwise from k . Each node maintains a routing table with information for only about $O(\log N)$ nodes. In fact, a Chord search is similar to a binary one, where the searching space is reduced by half after each search/routing-hop. So the number of nodes that must be contacted to resolve a query in a N -node network is $O(\log N)$. As an extension of Chord, K -ary tree search [42] has been proposed. Then, the search hop will decrease to $O(\log_k N)$, while the items of routing table in each node will increase to $O((k - 1) * \log_k N)$. There are some other variants of Chord, such as Viceroy [43], Multi-Ring [44][45], etc.

ii. JXTA

The JXTA middleware is a set of open Peer-to-Peer (P2P) protocols that allow any connected device (Cell phone to PDA, PC to Server) on the network to communicate and collaborate [54][16]. JXTA is an open source project that was originally conceived by *Sun Microsystems, Inc.* and

designed with the participation of a small number of experts from academic institutions and industry. The goal of project JXTA is to develop and standardize basic building blocks and services to enable developers to build and deploy interoperable P2P services and applications. The term JXTA is short for juxtapose, as in side by side. It is a recognition that P2P is juxtaposed to client-server or Web-based computing, which is today's traditional distributed computing model. JXTA provides a common set of open protocols and an open source reference implementation for developing P2P applications. The three (3) main objectives were to achieve interoperability, platform independence and ubiquity. JXTA defines a set of six protocols that can be used to construct P2P systems using a centralized, brokered or decentralized approach but its main purpose is to facilitate the creation of decentralized systems. Below, we provide a list of the main JXTA protocols:

- The Peer Resolver Protocol (PRP) is the mechanism by which a peer can send a query to one or more peers, and receive a response (or multiple responses) to the query.
- The Peer Discovery Protocol (PDP) is the mechanism by which a peer can advertise its own resources, and discover the resources from other peers (peer groups, services, pipes and additional peers).
- The Peer Information Protocol (PIP) is the mechanism by which a peer may obtain status information about other peers, such as state, uptime, traffic load, capabilities.
- The Pipe Binding Protocol (PBP) is used to connect pipes between peers.
- The Endpoint Routing Protocol (ERP) is used to route JXTA Messages.
- The RendezVous Protocol (RVP) is the mechanism by which peers can subscribe or be a subscriber to a propagation service.

Each peer operates independently and asynchronously from all other peers, and is uniquely identified by a Peer ID (identifiers). Peers publish one or more network interfaces (advertisements) for use with the JXTA protocols, which are passed around the network in datagrams (messages). Peers can form transient or persistent relationships (peer groups). Each published interface is advertised as a peer endpoint, which is used to establish direct point-to-point (but not fixed) connections between two peers (pipes). Figure II.7 shows the JXTA architecture [54].

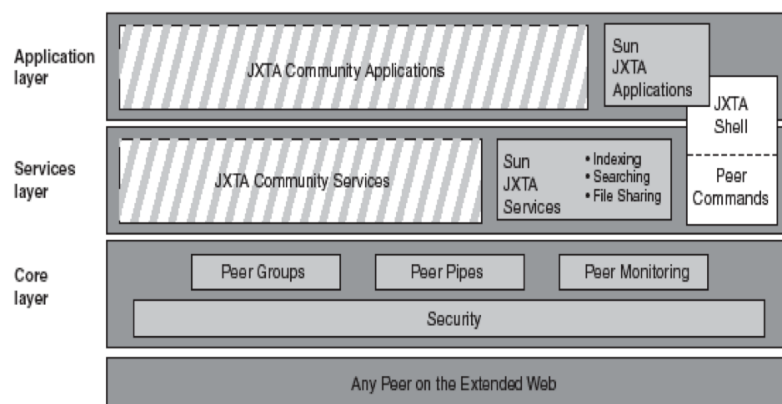


Figure II.7: JXTA architecture

iii. Kademlia

Kademlia is a P2P network protocol that was originally proposed to decentralize other file sharing P2P networks. It has been conceived by Petar Maymounkov and David Mazières in 2002 [55]. Kademlia nodes communicate using UDP protocol. Inside Internet network, Kademlia creates a new network in which every node is identified by an ID (160 bits). After a joining phase which consists of contacting a node of the overlay in order to get an ID, a mathematical operator computes the “distance” between two nodes and asks many nodes according to this distance in order to find the looking information. The eXclusive OR operator, also called XOR, allows using the notion of distance between two nodes giving a result as an integer number: the $\ll \text{distance} \gg$. This later does not have any relation with the geographical location of the participants, but model the distance inside a chain of IDs. Then, it may happen that a node in Italy and a node in Australia are neighbors. Nodes are considered as leaves of a binary tree. The XOR topology is symmetric unlike the ring topology used in Chord. The information in Kademlia is kept in $\ll \text{Values} \gg$, each value being associated to a $\ll \text{key} \gg$. For this reason, Kademlia is often called $\langle \text{value}, \text{key} \rangle$ network. Both keys associated to a given node are related to the address of the node. So knowing a key, the algorithm can determine the approximated distance that separates the requestor from the node holding the value associated to this key. In other words, in order to look for a key located in a node N, a node A needs to look for a neighbor B with $\text{Distance}(\text{B}, \text{N}) < \text{Distance}(\text{A}, \text{N})$ and asks for the information. If the latter node does not have the response, it will contact a neighbor, closest to the key and so on, until getting the value of the key (or until being sure that the key does not exist). The size of the network does not impact much the number of nodes contacted during the lookup phase. It can easily be shown that if the number of participants of the overlay doubles, the number of nodes that need to be consulted in each search increases by one only. Other advantages are inherent to the distributed network structure, increasing for instance the resistance to Denial of Service attack.

iv. ProActive

ProActive [67] is a middleware for easily programming and running java applications on Grids and P2P systems. Released under the LGPL license, ProActive is a Java library for parallel, distributed, and concurrent computing, also featuring mobility and security in a uniform framework. With a reduced set of simple primitives, ProActive provides a comprehensive API allowing to simplify the programming of applications that are distributed on Local Area Networks (LAN), on clusters, or on Internet Grids. ProActive does not require any modifications to Java or to the Java Virtual Machine (JVM), therefore allowing the deployment of applications using the ProActive API on any operating system that provides a compatible JVM. ProActive is based on the concept of active object, which is an entity with its own configurable activity. A distributed or concurrent application built using ProActive is composed of a number of entities called active objects. Each active object has one distinguished element, the root, which is the only entry point to the active object.

Communication between active objects is realized through method invocations, which are reified and passed as messages. These messages are serializable Java objects which may be compared to TCP packets. Indeed, one part of the message contains routine information towards the different elements of the library, and the other part contains the data to be communicated to the called object. The goal of ProActive is to deploy any application anywhere without having to modify the source code. The resources acquired through the deployment process are called nodes. Nodes are the containers of active objects. The second key principle is the capability to abstractly describe

an application, or part of it, in terms of its conceptual activities. The ProActive deployment framework relies on XML deployment descriptors to hold the infrastructure configuration.

II.1.3 Distributed Hash Table

Structured P2P networks tightly control their topology and place the indexing information for the advertised resources at specified locations. Thus, queries can be routed to these locations and lookup can be efficiently performed. The great majority of modern structured P2P systems use Distributed Hash Tables (DHTs) as a communication infrastructure. A DHT evenly distributes the data items and query load across the network. The data item is stored as $\langle \text{key}, \text{value} \rangle$ pairs and the node responsible for storing each data item is determined by the DHT algorithm. DHTs make it possible to find an object from a network of thousands of peers based on the object's key [49].

The keyspace is divided among the participating nodes. Each node has an identifier called node ID, which defines its logical location in the network. The stored data is also identified by a resource ID generated by a hash function. The key for a data value can be hashed for example from the file name or from the objects keywords. Indexing information is placed deterministically at the corresponding peers whose identifier is “closest” to the key. DHT algorithms vary in how they define this distance. DHTs provide a scalable way to store and retrieve data objects under given keys. Each key lookup is resolved in multiple steps, resulting in a multi-hop path to be taken in the overlay. Thus, the core operation that is provided by DHTs is the efficient routing of the query to the final destination, given a key.

The size of the keyspace a node is responsible for, is determined by the number of nodes in the network. Although the keyspace is divided evenly among participating nodes, there can be differences in the number of keys that nodes have to store. This is due to the fact that the hash algorithms are not always optimal, especially when there is churn affecting the network. DHTs aim to balance the responsibilities evenly on every node. Various load balancing methods have been proposed to address this problem [51].

DHT algorithms typically have four design constraints [50]. The “Few neighbours” constraint means that each node keeps routing information for usually only $\log N$ other nodes in a network of N nodes. By distributing the routing information evenly on every node, DHTs can handle the arrival and departure of nodes in a decent number of update messages. When the routing information is distributed, the node arrival or departure process affects only a small number of nodes and only those nodes need to update their routing table information. The “Low latency” constraint indicates that all nodes should reach any other node in the network in a small number of hops. Usually this means that the maximum hop count between two nodes is $\log N$. Nodes also need to be able to make their own routing decisions. This constraint called “greedy routing decisions” ensures that node lookups are efficient and every node can make its own routing decisions without the help of other nodes. Network should also be able to withstand the effects of churn and retain connectivity and ability to route packets correctly as the nodes arrive and leave. DHTs should balance the load evenly so that there would not exist any overloaded nodes and links. These characteristics are demanded by “robustness” constraint [50].

Various DHT algorithms have a different routing geometry for the keyspace. The keyspace can take the form of a hypercube, ring, tree or a butterfly. The routing geometry affects route and neighbor selection as well as the DHT's performance and resilience.

II.2 SIP Protocol

II.2.1 Overview

SIP (Session Initiation Protocol) is a signaling protocol defined by IETF (Internet Engineering Task Force) for the establishment, modification and release of multimedia sessions [1] (RFC 3261). The SIP protocol is an Application Layer protocol designed to be independent of the underlying Transport Layer; it can run on Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or Stream Control Transmission Protocol (SCTP).

SIP inherits some features of HTTP (Hyper Text Transport Protocol) used to browse the Web, and SMTP (Simple Mail Transport Protocol) used to transmit electronic messages like e-mails. SIP is based on a transactional client/server model as HTTP. The addressing uses the concept of SIP URL (Uniform Resource Locator) that looks like an e-mail address. Each participant in a SIP network is addressable by a SIP URL. Moreover, SIP requests are made by way of responses identified by a numerical code. Furthermore, most of SIP response codes were borrowed from the HTTP protocol. For example, when the recipient is not located, a response code “404 Not Found” is returned. SIP messages also consist of headers as an HTTP message. SIP is a textual protocol such as HTTP.

SIP is not a vertically integrated communication system. Rather, SIP is a component that can be used together with other protocols to build a complete multimedia architecture. Typically, this architecture includes protocols such as RTP (Real-time Transport Protocol) for transporting real-time data, and RTCP (Real-time Transport Control Protocol) designed to work in conjunction with RTP in order to convey on quality of service feedback and membership information. Moreover, there are RTSP (Real-Time Streaming Protocol) which is an application-level protocol used for controlling delivery of streaming media with real-time properties and RSVP (Resource ReSerVation Protocol) which is the network control protocol that allows data receiver to request a special end-to-end quality of service for its data flows and to reserve necessary resources (bandwidth, quality of service) at routers along the transmission paths. In addition, SDP (Session Description Protocol) is used for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation. Indeed, the SDP part of SIP messages includes the type of media (video, audio, etc.), the transport protocol (RTP/UDP/IP, etc.), the format of the media (H.261 video, MPEG video, etc.), and information to receive those media (IP addresses, ports, formats and so on). Furthermore, SIP makes use of “Media Control Protocols” responsible for the creation and tearing down of media connections. They are used to open and close media pin-holes on VoIP gateways and to process notifications coming from those gateways. The main “Media Control Protocols” are MGCP (Media Gateway Control Protocol) [21] and SIP-based media control protocols such as NetAnn (Network Announcement Protocol) [64], MSCML (Media Server Control Markup Language) [61] and the new Media Control Channel Framework [65].

Therefore, SIP should be used in conjunction with other protocols to provide complete services to the users. However, the basic functionality and operation of SIP do not depend on any of these protocols. SIP has been extended to support many other services such as presence, instant messaging (similar to SMS in mobile networks), call forwarding, conferencing, value-added services of telephony, etc.

In the next few sections dedicated to SIP, we will introduce the major SIP entities. Subsequently, we will give an overview of SIP messages and focus on SIP protocol functionalities.

II.2.2 SIP components

In SIP, there are two logical entities: User Agent Client (UAC) and User Agent Server (UAS), often just called User Agent (UA).

A UAC is a logical entity that creates a new request, and then uses the client transaction state machinery to send it. On the other hand, a UAS is a logical entity that receives a request and generates a response to that SIP request. The UAS accepts, rejects, or redirects the request. Therefore, all IP phones supporting SIP protocol can be either UAC or UAS depending on the direction of the call request. Figure II.8 depicts SIP network with the different entities.

It is possible to make a call directly between endpoints, but in most cases, servers are involved in the communication for authentication, call routing, advanced feature services, and so on. There are four servers in SIP: Registrar, Redirect and Proxy servers and a Back-to-Back User Agent (B2BUA). Here is the description of each server.

- Registrar: A registration server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles. It maintains a list of bindings (UA's SIP URI and the related IP address or addresses) that are accessible to proxy servers and redirect servers within its administrative domain.
- Proxy server: An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing, which means its job is to ensure that a request is sent to another entity "closer" to the target user. A proxy server is also useful for enforcing policy. It intercepts, and if necessary, rewrites specific parts of a request message before forwarding it.
- Redirect server: This is a server that accepts SIP requests, translates the SIP address of destination in one or more network addresses and returns them to the UAC. Unlike the proxy server, the redirect server does not route SIP requests. For instance, in the case of call forwarding, the proxy server has the ability to translate the called number in the SIP message received, into a new number (forwarding number) and route the call to the new destination, and this is transparent to the source UAC. For the same service, the redirect server returns the new number (forwarding number) to the source UAC that is responsible for establishing a new call toward the new destination.
- Back-to-Back User Agent: This is a logical entity that receives a request and processes it as a UAS. To determine how the request should be answered, it acts as a UAC and generates request. Unlike a proxy server, it maintains dialog state and must participate in all requests sent on the dialogs it has established. Because it is a concatenation of a UAC and UAS, no explicit definitions are needed for its behavior.

These servers could be separate entities physically, or integrated in a single device. The gateway in Figure II.8, allows connectivity between SIP network and PSTN (Public Switched Telephone Network). It interfaces to the PSTN on the one hand and to SIP network on the other hand [122]. The Gateway has two main functions:

- Translation of ISDN User Part (ISUP) signaling (the signaling protocol mostly used in PSTN) to SIP signaling and vice versa.
- Conversion of audio signals in RTP packets and vice versa.

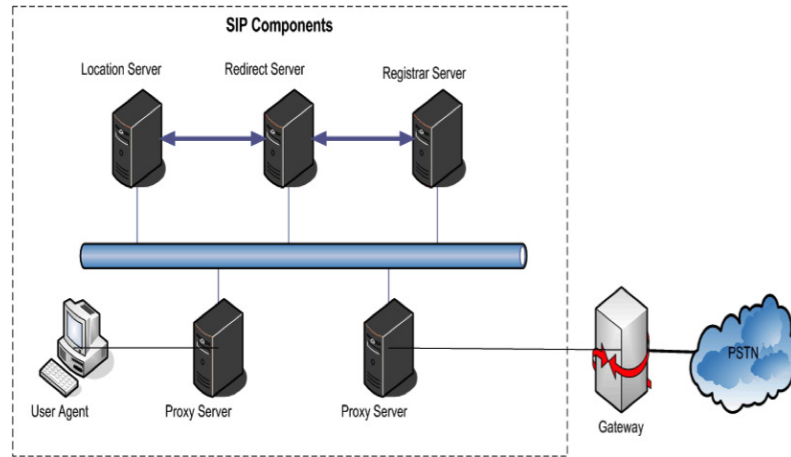


Figure II.8: SIP components

All messages sent from the PSTN and targeted to a UA in the SIP network, go through one or more proxy servers and vice versa.

II.2.3 SIP messages

As mentioned before, SIP is a text-based protocol. The formatting of SIP messages is based on the syntax of HTTP version 1.1. There are two types of messages: requests and responses.

a) Message format

The format of a request is shown in figure II.9.

Request Start Line Method, Request URI, SIP version
Headers General headers, or , Request headers
Blank Line (specifies end of headers fields and the begin of message body)
Message Body (carries the Session Description Protocol message)

Figure II.9: SIP requests format

In the message, end of line is always denoted with the two octets <CR><LF>. The format of the response is very similar to what has been shown above. The only difference is the first line. The response format is illustrated by figure II.10.

Below, we show a sample of SIP message:

```

INVITE sip:13@10.10.1.13 SIP/2.0
Via: SIP/2.0/UDP 10.10.1.99:5060;branch=z9hG4bK343bf628;rport
From: "Test 15" <sip:15@10.10.1.99>tag=as58f4201b
To: <sip:13@10.10.1.13>
Contact: <sip:15@10.10.1.99>
Call-ID: 326371826c80e17e6cf6c29861eb2933@10.10.1.99
CSeq: 102 INVITE

```

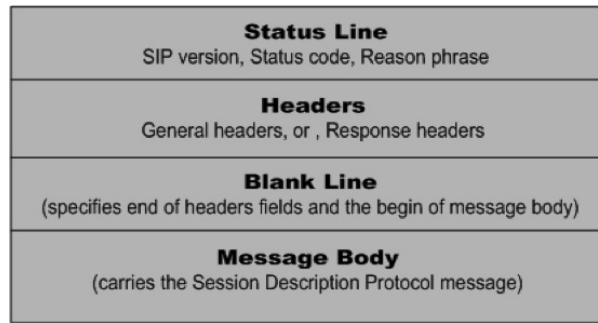


Figure II.10: SIP responses format

```

User-Agent: Asterisk PBX
Max-Forwards: 70
Date: Wed, 06 Dec 2009 14:12:45 GMT
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
Content-Type: application/sdp
Content-Length: 258

```

```

v=0
o=root 1821 1821 IN IP4 10.10.1.99
s=session
c=IN IP4 10.10.1.99
t=0 0
m=audio 11424 RTP/AVP 0 8 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=silenceSupp:off - - -
a=ptime:20
a=sendrecv

```

The above example shows an INVITE message. This is the message that a SIP endpoint needs to send in order to establish a call. The message was sent by an Asterisk PBX running at the IP address 10.10.1.99. It starts a call from extension number 15 to extension number 13 at the IP address 10.10.1.13.

b) Message headers

We now briefly describe some message headers used in the example above.

The request start line: The string “INVITE sip:13@10.10.1.13 SIP/2.0” tells that this is an invitation to a call. It also gives the SIP address of the receiving endpoint (sip:13@10.10.1.13) and identifies the version of the protocol (SIP/2.0).

- **Call-ID:** This is a unique identifier of a given SIP session. It usually consists of a random string and the IP address of the sender.
- **CSeq:** This is an ID that identifies the particular SIP transaction. The same CSeq: is always shared by a request and its related response(s).
- **From:** This field (‘From: “Test 15” <sip:15@10.10.1.99>;tag=as58f4201b’ in the example above) contains the address of the caller with an optional display name and with optional

tags. From: is a mandatory field in all SIP requests and responses. In SIP responses, From: is always a copy of the From: field in the related request message.

- To: This field contains the address of the called party. To: is a mandatory field. The To: fields in responses are copied from the related request message.
- Via: The Via: headers are used to record the route of the request. Each proxy server on the path of the message will add one Via: entry. Thanks to this, the replies can be routed back along the same path.
- Content-type: This field describes the media type of the message body. The type is usually “application/sdp”, denoting the Session Description Protocol. The message body can be sometimes empty (e.g. the REGISTER message) and then the Content-type: header is not present.
- Content-length: This is the length of the message body in octets. This header is always present but can be 0 (denoting there is no message body).

The message body carries a message of the SDP. This message contains a description of the audio (and possibly video) channel that the calling endpoint wants to establish.

c) SIP Requests

SIP originally included only six requests (also called methods). These requests have been a part of the standard since SIP 1.0. Below, we describe these core methods:

- INVITE: This is a request to establish a call (a session). We have seen an example of the message above.
- CANCEL: This method is used to stop an INVITE that is in progress (that is, the call has not been established yet).
- ACK: The ACK request is used to confirm that the endpoint has received a final response in a transaction. Typically, after the called party accepts a call, the caller confirms the receipt of the accepting response (200 OK) with the ACK method.
- BYE: The BYE method is used to end an established call (comparing with CANCEL that is used to stop the session before it has been established).
- REGISTER: The REGISTER method is used to register the SIP endpoint at the registrar server. In fact, this method does the same thing as the Registration Request (RRQ) in H.323 protocol.
- OPTIONS: This request is used to ask the other party for the list of SIP methods, it supports. The response may also contain the set of capabilities (i.e. audio/video codecs) of the responding party.

Later, many other SIP methods have been added, either in SIP 2.0 or in other RFCs. The INFO method was defined in IETF RFC 2976. It can be used to carry application-level information that are relevant to the session, for example participant images or account balance information. Moreover, the SUBSCRIBE, NOTIFY (RFC 3265), and MESSAGE methods extend SIP with

instant messaging features. The REFER method (RFC 3515) redirects the receiver to a resource identified in the method and the PRACK method (RFC 3262) has been defined to acknowledge the reception of provisional response of type 1XX. The UPDATE method (RFC 3311) allows a SIP terminal to update the parameters (e.g. media streams and their codecs) of a multimedia session. Finally, the method PUBLISH (RFC 3903) is used by an entity to publish its state.

d) SIP Responses

Like other IETF protocols, SIP uses 3-digit response codes. SIP responses fall into six (6) categories.

- Class 1xx: Information, the request has been received and is being processed.
Example: 100 Trying, 180 Ringing, 181 Call forwarded.
- Class 2xx: Success, the request has been received, understood and accepted.
Example: 200 OK
- Class 3xx: Redirection, the call requires further treatment before whether it can be done.
Example: 300 Multiple Choices, 301 Moved Permanently, 302 Moved Temporarily.
- Class 4xx: Client Error, the request cannot be interpreted or served by the server. The application must be modified before being returned.
Example: 400 Bad Request, 401 Unauthorised, 403 Forbidden, 404 Not Found.
- Class 5xx: Server Error, the server fails in processing a request apparently valid.
Example: 500 Server Error, 501 Not Implemented, 503 Service Unavailable, 504 Timeout.
- Class 6xx: Global Failure, the application cannot be processed by any server.
Example: 600 Busy Everywhere, 603 Decline, 604 Does not Exist, 606 Not Acceptable.

II.2.4 SIP Protocol operation

This section shows how registration occurs in SIP and how call can be established, modified and terminated through a proxy server.

a) Registration

Let's take a look at how SIP user agents register with a SIP registrar. The example explained in this section shows a situation where a SIP softphone (namely, the Ekiga client) registers with an Asterisk PBX (Private Branch eXchange). The Asterisk's IP address is 10.10.1.99, while the client is at 10.10.1.13 and wants to register the telephone number 13. Figure II.11 illustrates the registration scenario.

In order to register, the SIP telephone needs to send the REGISTER request. The registrar server will immediately reply with the provisional response "100 Trying". This indicates that the request has been received (and thus the client does not need to retransmit it) and that it is being processed. While processing the request, the registrar discovers that the user agent needs to authenticate itself. It therefore responds with "401 Unauthorized". For the user agent, this means that it has to send the REGISTER request once more, this time providing authentication.

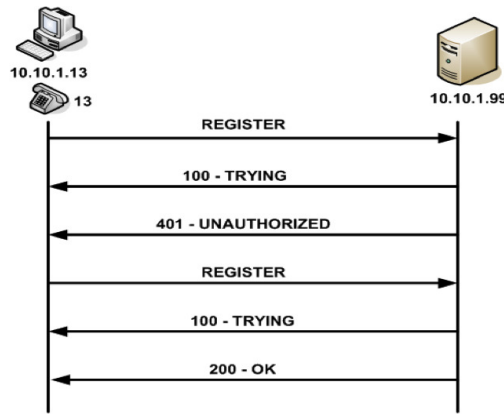


Figure II.11: SIP registration scenario

Following is an example of the REGISTER request:

```

REGISTER sip:10.10.1.99 SIP/2.0
CSeq: 1 REGISTER
Via: SIP/2.0/UDP 10.10.1.13:5060;
    branch=z9hG4bK78946131-99e1-de11-8845-080027608325;rport
User-Agent: Ekiga/3.2.5
From: <sip:13@10.10.1.99>
    ;tag=d60e6131-99e1-de11-8845-080027608325
Call-ID: e4ec6031-99e1-de11-8845-080027608325@vvt-laptop
To: <sip:13@10.10.1.99>
Contact: <sip:13@10.10.1.13>;q=1
Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,SUBSCRIBE,NOTIFY,REFER,MESSAGE,
    INFO,PING
Expires: 3600
Content-Length: 0
Max-Forwards: 70
  
```

The “401 Unauthorized” message is as follows:

```

SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 10.10.1.13:5060;
    branch=z9hG4bK78946131-99e1-de11-8845-080027608325;
received=10.10.1.13;rport=5060
From: <sip:13@10.10.1.99>;
    tag=d60e6131-99e1-de11-8845-080027608325
To: <sip:13@10.10.1.99>;tag=as5489aead
Call-ID: e4ec6031-99e1-de11-8845-080027608325@vvt-laptop
CSeq: 1 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
    SUBSCRIBE, NOTIFY
Supported: replaces
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk",
    nonce="343eb793"
Content-Length: 0
  
```

In the “401 Unauthorized” response, the important header is *WWW-Authenticate*:. It instructs the client to authenticate using the digest authentication (RFC 2617). The *nonce* (a short for “number used once”) is a “challenge string”. The client will combine the challenge string with the user’s password and compute the MD5 hash of the resulting string. The server will compute its own hash using the same method and compare it with the MD5 hash provided by the client.

The digest authentication is the most frequently used because the password is never sent over the network in plain text. The “basic” authentication method has been deprecated in SIP 2.0 as it is insecure. In general, sending a password in plain text over the method is obviously not secure.

Once the client computes the MD5 digest, it will re-send the REGISTER request. The message is similar to the previous REGISTER request but includes a new header called “Authorization” with the “CSeq” header set to 2 like this:

```
CSeq: 2 REGISTER
Authorization: Digest username="test13", realm="asterisk", nonce="343eb793",
uri="sip:10.10.1.99", algorithm=MD5,response="6c13de87f9cde9c44e95edbb68cbdea9"
```

The registrar server will again first respond with “100 Trying” and then compare the two MD5 hashes (the one provided by the client with the one computed by the registrar itself). If they match, the registrar will respond with “200 OK” and insert the endpoint to the location database. The database is usually shared between the registrar and the proxy server so that the proxy can use it to establish the calls.

Finally, the response “200 OK” is sent by the registrar and contains one important parameter, *Expires*. It tells the client that the registration will expire after the given number of seconds and the client will be required to register again.

b) Basic call flow

The call flows between SIP clients and servers are various depending on the service architecture. One of the common call flows is shown in figure II.12, assuming that servers share the registration information of users, and clients have to send INVITE to a redirect server first when initiating a call [56].

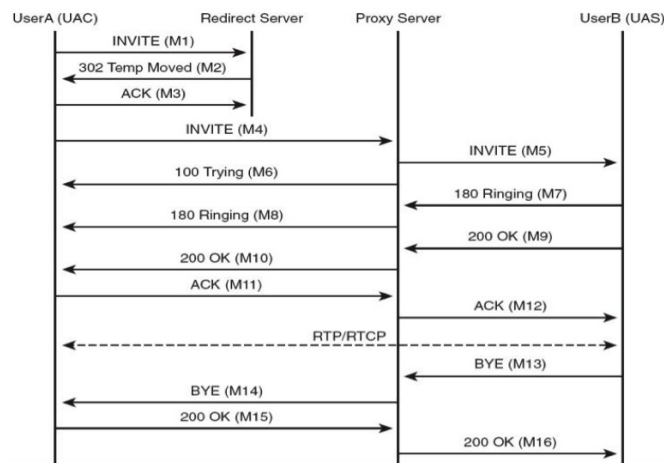


Figure II.12: Basic call flow with servers

Here is a brief comment for each message in the figure II.12.

- M1: User A (UAC) sends INVITE message to a Redirect Server, first to make a call to User B (UAS).
- M2: The redirect server returns 302 Moved Temporarily response containing a contact header with User B’s current SIP address.
- M3: Acknowledgement.

- M4 and M5: User A then generates a new INVITE with SDP and sends to User B via a proxy server.
- M6: This step notifies that the proxy server received the request and continues to process it.
- M7 and M8: User B sends 180 Ringing when the telephone is ringing.
- M9 and M10: User B sends 200 OK with SDP when picking up the phone.
- M11 and M12: Acknowledgment. After this, the media channel is opened.
- M13 and M14: User B sends BYE when hanging up the phone.
- M15 and M16: Confirmation of disconnection.

II.3 P2P SIP

This section presents some background related to the main concept of P2P SIP network. We provide first an overview of P2P SIP network and describe the P2P SIP reference model released by IETF. In addition, we introduce the current available candidate projects for P2P SIP implementation.

II.3.1 Overview

Most VoIP systems rely on permanent set of SIP servers and suffer from performance, bottlenecks, and single point of failure issues. As in a P2P system, there is no centralized server, such a system has greater robustness, scalability and fault tolerance. Then, it was thought that if SIP can be made to work over P2P systems, it will improve the performance of traditional SIP systems and eliminate the problems of using centralized SIP servers. However, SIP was already ready for P2P with little changes. SIP already uses symmetric, direct client-to-client communication and the proxies and the registrar only perform lookup and routing. Then, all that user agents lack to build a P2P network is lookup and routing. The lookup/routing functions of the proxies/registrar can be replaced by a DHT overlay built in the user agents. By adding join, leave and lookup capabilities, a SIP user agent can be transformed into a peer capable of operating in a P2P network.

Then, P2P SIP is the combination of a P2P network and SIP which provides an alternative solution to the session establishment that conventional client/server SIP offers. It replaces the somewhat fixed topology of SIP with a DHT-based structured Peer-to-Peer overlay network. The P2P overlay nodes, called peers, collaborate and provide the same location service function that maps Addresses of Records (public URIs) to overlay locations (SIP URIs) as conventional SIP does. In P2P SIP network, this is done in a distributed manner, every peer taking responsibility over routing and location information storing. To be able to provide the distributed location service function, P2P SIP offers distributed database and transport functions.

However, SIP and P2P can be combined using two approaches [3]: P2P-over-SIP and SIP-using-P2P. The former approach focuses on using SIP messages to maintain P2P overlay networks. SIP provides the P2P overlay with control functions and establishes multimedia sessions over unreliable P2P overlay networks. The disadvantage of the approach is that P2P-over-SIP needs to maintain many SIP dialog and transaction states during the overlay control procedure since all the P2P operations are carried by SIP messages. According to the statistics on [19], more than ten (10) SIP messages need to be exchanged from the node joining the DHT to the user registration,

and each SIP message has its own dialog states in user agents. The advantages of this approach include the use of existing SIP components and the independence from the external P2P networks.

Using the latter approach named SIP-using-P2P, the underlying P2P overlay networks such as Chord, CAN, Pastry, etc provides SIP with the peer locating service. This means that the traditional centralized SIP trapezoid is replaced by the decentralized P2P location service. Thus, the deploying cost of P2P SIP is reduced and the robustness of the network increases. One of the essential design targets of SIP is to find the peer anytime and anywhere using DNS. However, when the DNS and SIP Proxy are replaced with P2P overlays, connectivity problem needs to be addressed [4][60].

A comprehensive analysis of the related work on research activities belonging to the two approaches will be presented in the next chapter.

II.3.2 High level description

The P2P SIP Working Group (WG) was founded by the IETF in 2007 to develop standards for serverless use of SIP. P2P SIP is still largely under development. There are many IETF Internet Drafts discussing potential solutions for different mechanisms of P2P SIP. The current state of the general P2P SIP framework is documented in a WG draft [62].

In P2P SIP, P2P nodes are organized to make SIP based real-time communication possible. The P2P SIP network consists of P2P SIP peers and P2P SIP clients. P2P SIP peers participate in the P2P SIP overlay and provide storage and transport services to other peers in that overlay. The role of a P2P SIP client is still under debate. One approach is that the client is not itself part of the P2P overlay network but interacts with the overlay through an associated peer. Under this assumption, clients can store information in the overlay and retrieve information from it, but they do not contribute any resources to the overlay, and thereby do not route messages or store information for other nodes. The services offered by every P2P SIP node are needed to provide the location function which is a core function of the client/server SIP. This also causes a certain degree of interference (“cross-layering”) between the SIP protocol and the P2P layer, as the P2P overlay needs to be aware of the services its peers support, in order to know which functions it must provide [62].

A specific “Peer Protocol” is needed for P2P SIP to enable communication between the peers. The peer protocol routes messages within the P2P overlay, maintains the overlay, stores data in and retrieves data from the overlay. It was first suggested that SIP should be used for this inter-peer communication [62]. The P2P SIP WG is working towards a new protocol for this purpose. Several drafts for the peer protocol have been proposed. At the moment, the P2P SIP WG has one peer protocol draft as a working group item. It is called REsource LOcation And Discovery (RELOAD).

A “Client Protocol” to allow communication between clients and the peers is also needed. This protocol might be a logical subset of the peer protocol. This means that any operation supported by the client protocol is also supported by the peer protocol [62].

II.3.3 Reference model

It is expected that most P2P SIP peers and clients will be coupled with SIP entities [62]. The P2P SIP Reference Model illustrates this assumption in figure II.13. The scenario presented in the figure is only an example of a possible P2P SIP overlay. Other compositions are also possible.

Peer names refer to the SIP entity the peer is coupled with. Proxy peer is coupled with a SIP proxy, Redir peer with SIP redirect server and UA (User Agent) peers with a SIP UA. It is also possible that a peer is coupled with more than one SIP entity. A Gateway peer connects the overlay to other networks.

The nodes can connect to a P2P SIP overlay in several ways. In figure II.13, user agent peers A and B are directly connected to the overlay while UA peer C is behind a Network Address Translation (NAT) device. Peer D is an ordinary peer with no SIP capabilities. The client connects with the overlay via peer D using the P2P SIP client protocol. The Client uses client protocol to obtain information from the overlay, but has not inserted itself into the overlay, and therefore does not participate in routing messages or storing information.

The SIP user agent interacts with the overlay through the proxy peer using SIP to communicate with the proxy peer. SIP UA can also use the Redir peer as an adapter node to interact with the P2P SIP overlay. The Proxy peer and the Redir peer speak both SIP and the peer protocol used in the overlay (RELOAD for instance). The overlay can connect to other networks such as PSTN (Public Switched Telephone Network) through the gateway peer that speaks the appropriate protocols.

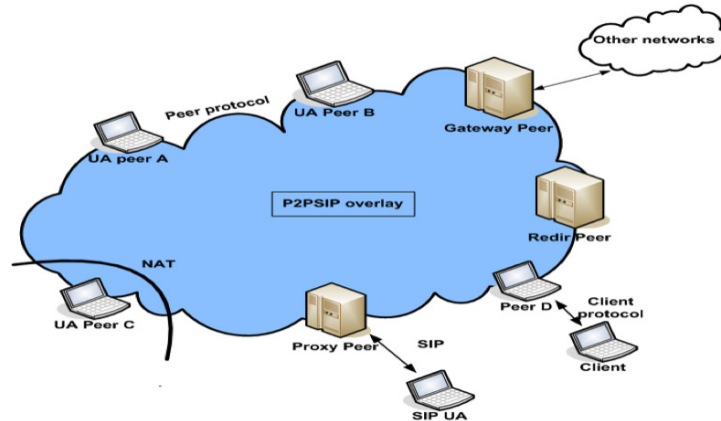


Figure II.13: P2P SIP Reference model

II.4 Current candidate projects for P2P SIP implementation

Several drafts of P2P SIP implementations have been proposed. These proposals are: Cisco's P2P SIP project, the P2PP project of the Columbia University, the SIPDHT2 project, Kademlia dSIP of the University of Parma, Huawei's P2P SIP implementation, P2PNS from the University of Karlsruhe, 39 Peers project, P2Pship project by Helsinki Institute for Information Technology and the P2P SIP RELOAD approach. Following, we shortly introduce each of these projects.

II.4.1 CISCO P2P SIP Project

The CISCO P2P SIP project uses a binary P2P signaling protocol, called Address Settlement by Peer-to-Peer (ASP). ASP supports Chord as a DHT algorithm, SIP for localization, STUN and TURN services, a security framework on the base of an abstract enrollment server and a protocol extensibility model [11]. The implementation is written in C++, it seems very clean and small but a good documentation for the code is not available. The project development seems to have been stopped since the last code update was done in July 2007.

II.4.2 Columbia P2PP Project

OpenVoIP (Open Peer-to-Peer VoIP and IM System) is an open source peer-to-peer VoIP and IM system of approximately 1000 nodes running on around 300 PlanetLab (a P2P simulator) machines. OpenVoIP runs Peer-to-Peer Protocol (P2PP) [130] which can be used to implement well-known DHTs or unstructured protocols (e.g. Kademlia, Bamboo, or Chord). Routing can be done in an iterative or recursive manner. Additional functions of P2PP are the support for four different hash algorithms (SHA1, SHA256, MD4, MD5) and the support for NAT traversal. OpenVoIP provides NAT traversal services using STUN, TURN, and ICE. OpenVoIP also provides a Google Map Interface to check the health of the system. Nodes provide routing and storage services to the overlay. Latest version (Version 0.21) of the project has been released on 11/03/09 [131].

II.4.3 SIPDHT2 Project

SIPDHT2 is an open source project and is a candidate solution for P2P SIP. The primary goal of the SIPDHT project is to provide a library to be used in applications for creating and using SIP based distributed hash tables. SIPDHT2 aims to develop a public API, which can be used by other applications to build Real Time Systems. It uses PCAN (Passive Content Addressable Network) [136] as the DHT algorithm which is a variant of CAN. PCAN provides a robust overlay even if most peers are behind a NAT system. SIPDHT2 uses SIP as its signaling protocol. The integrated SIP stack is the Sofia-SIP library. SIPDHT2 offers a good documentation and is written in pure C. A main concern with this project is the strong dependency on external Linux specific libraries like avahi and glib. STUN and ICE are used to connect peers behind NATs. The latest version has been released at June 2007 [132].

II.4.4 Kademlia dSIP

The dSIP [4] protocol uses the SIP syntax for signaling. dSIP provides STUN, TURN, and ICE for NAT traversal through the use of conventional SIP messages. Chord and Kademlia are used as DHT algorithms. Kademlia dSIP is implemented in the programming language JAVA, leading to a high availability on nearly all operating systems. The implementation is currently limited to SIP INVITE requests and development seems to have been stopped (last updates in 2007).

II.4.5 Huawei's SEP Peer and Client Protocol

Huawei's implementation uses the Service Extensible Protocol (SEP) for communication between P2P SIP peers and maintenance of the DHT service. SEP distinguishes between a peer [20] and a client [21] protocol. According to the P2P SIP specification, peers offer routing and storing services and clients will not offer these services. The client protocol controls the behavior between a client and its allocated peer. The code of SEP is written in C++ but it is very heavy and complex.

II.4.6 P2PNS: A Secure Distributed Name Service for P2P SIP

P2PNS (Peer-to-Peer Name Service) [13] is a distributed name service using a peer-to-peer network. The current focus of P2PNS is to provide a secure and efficient SIP name resolution for

decentralized VoIP. P2PNS is developed at the Karlsruhe Institute of Technology (KIT), Institute of Telematics (research group Prof. Zitterbart) within the scope of the ScaleNet project. An experimental implementation of P2PNS for OverSim is currently under development. There is also a live demonstration of P2PNS deployment in the PlanetLab and G-Lab research networks. The experimental P2PNS implementation consists of two parts: A modified OpenSER SIP proxy and the overlay framework OverSim. The SIP proxy connects to the OverSim P2PNS service via an XML-RPC interface. The implementation is still in an early stage and several security mechanisms are not implemented yet [133].

II.4.7 39 peers project

The 39 peers project aims at implementing an open-source P2P Internet telephony software using SIP. The 39 peers project is developed for student developers and researchers to experiment with new ideas. It is written in Python scripting language. It supports open protocols such as SIP and RTP. It is released by Kundan Singh under GNU/GPL license (an alternate commercial license is available as well) [134].

II.4.8 Peer-to-Peer SHip

P2Pship is a P2P framework for various applications. The P2Pship system is an experimental application that provides P2P communication capabilities for different applications. The system was originally designed for SIP-based communication applications (a SIP proxy), allowing users to make P2P voice / video calls, without the help of a centralized SIP infrastructure. The Host Identity Protocol (HIP) was used as data transport, making the connections secure and enabling features such as mobility and multihoming. P2Pship is developed inside a project called Secure Peer-to-Peer Services Overlay Architecture (SPEAR) of the Networking Research group at Helsinki Institute for Information Technology. Currently the P2Pship proxy supports applications like SIP with media proxying, P2P HTTP and P2P web caching. P2Pship is under GNU/GPL license and the last version (version 3.0) has been released the 30th September 2011 [135].

II.4.9 RELOAD

As mentioned before, P2P SIP working group of IETF is moving towards the development of applications that can use both P2P and SIP technologies in conjunction. RELOAD is a P2P signaling protocol, which is still strongly under development and there is currently no available implementation. RELOAD works in environments where there are NATs or firewalls. RELOAD can support various applications and provides security frameworks. RELOAD also allows the use of various Distributed Hash Table (DHT) algorithms in the form of topology plugins.

II.5 Conclusion

This chapter allows understanding the main concepts behind both P2P and SIP paradigms. Moreover, an overview of the P2P SIP telephony makes clear the main goals pursuing by the P2P SIP working group. Furthermore, current candidate projects for P2P SIP implementation have been shortly described. Next chapter will focus on the related work on the existing P2P architectures.

Part II

P2P SIP architecture

This part of the thesis proposes a new P2P SIP architecture called MISE-P2PSIP. MISE-P2PSIP architecture is an overlay composed of a set of self-organized proxies and registrars wherein clients, proxies and registrars communicate using the standard SIP protocol. It functions independently of the middleware used to map the logical P2P network onto the real physical network. This part of the thesis is composed of two chapters. The first chapter (Chapter III) points out the requirements of MISE-P2PSIP architecture and discusses the related work on the P2P SIP architectures that have been proposed to date by underlining their shortcomings. The second chapter (Chapter IV) presents the principles of the new architecture, the software architecture and the proof-of-concept prototype. It also provides a discussion of the evaluation results.

Chapter III

Related work on existing P2P SIP architectures

This chapter identifies the requirements of MISE-P2PSIP architecture and discusses the existing architectures in light of these requirements.

III.1 Requirements

In order to clarify the discussion of related work, we now briefly summarize some requirements that we think a P2P SIP architecture has to meet.

These requirements will be used to clarify the specific contributions of related approaches.

The first requirement is that the architecture should bring to the SIP world all the benefits inherent to the P2P paradigm. This means that it should enable self-organization, especially when nodes join and leave the overlay. It also should scale in terms of the number of peers participating in the overlay, and none of the peers should act as a permanently centralized node.

The second requirement is that the architecture should be independent from the P2P middleware and architectures used to map the logical P2P overlay onto the physical network. This will ensure portability, since today there is a plethora of non-interoperable P2P middleware.

A third requirement is that the architecture should be, as much as possible, compliant with existing SIP specifications. SIP extensions should be avoided whenever possible. This will ease implementation by allowing the re-use of existing SIP tool kits, and will also make interoperability with existing SIP implementations easier.

The fourth requirement is performance. SIP is used for multimedia session signaling, and session signaling messages are real-time messages. This implies for instance, that the discovery mechanism of the peers' contact information should be time efficient.

III.2 Related work and analysis

As we have seen, three core features of P2P networks are self-organization, distributed data storage and efficient location of data items. P2P SIP leverages these characteristics to support distributed data discovery and self-organization in SIP networks, eliminating (or at least reducing) the need for centralized servers [3]. Instead of storing the SIP peers' contact information (i.e. IP address and port) in a centralized server as per current SIP specifications, most of the existing P2P SIP networks replace the SIP location service by using the P2P overlay to store the information.

We refer to this technique as “SIP-using-P2P” in the rest of this section. While SIP-using-P2P architectures rely on location and distributed storage mechanisms provided by existing general purpose P2P architectures, other approaches use SIP as a protocol for managing the P2P overlay. We call these approaches “P2P-over-SIP”. We review the former approach first, and then the latter.

III.2.1 SIP-using-P2P

The P2P SIP proposals belonging to this category can be divided in two sub-categories: flat architecture and hierarchical architecture.

a) Flat architectures

The IETF P2P SIP Working Group has the goal to develop a protocol that can be deployed on any DHT overlay network. The working group has proposed REsource LOcation And Discovery (RELOAD) Base Protocol as P2P SIP protocol reference model [11]. RELOAD is a Peer-to-Peer (P2P) signaling protocol for use on the Internet which provides a generic, self-organizing overlay network service. RELOAD implementation is strongly based on Chord algorithm [15]. One important use of RELOAD is the SIP usage described in [12]. The SIP usage of RELOAD proposal’ goal is to replace SIP proxy and/or SIP registrar by RELOAD services implemented on the overlay. Indeed, the SIP usage of RELOAD involves two main functions: Registration and Rendezvous management. Registration allows SIP UA to use the RELOAD overlay to store a mapping between their SIP AoR (Address of Record) and their Node-ID (Node Identifier) in the overlay. The Rendezvous management function allows a SIP user agent to use the RELOAD message routing system to set up a direct connection with the destination user agent to exchange SIP messages, once it has identified the Node-ID of the user agent it wishes to call. Consequently, the SIP UA must be running on a RELOAD peer and must have a complete RELOAD implementation. For instance, a user Bob will store the mapping between his Node-ID, “1234”, and his AoR, sip:bob@dht.example.com into the overlay. When Alice wants to call Bob, she looks up for “sip:bob@dht.example.com” in the overlay and gets back Bob’s Node-ID which is 1234. Then, Alice uses the overlay to route an *AppAttach* (a RELOAD specification message) message to Bob’s peer. Bob responds with his own *AppAttach* and they set up a direct connection. At this step, Alice can send a SIP INVITE message to Bob using the connection.

While RELOAD provides a complete “SIP-using-P2P” solution, it raises some concerns with respect to our requirements. RELOAD proposal is not fully compliant to the standard SIP since the SIP messages are not sent using the node IP address but using the Node-ID.

Other techniques belonging to the “SIP-using-P2P” category have been proposed. For instance, the authors of [7] proposed a DHT-based architecture. The approach provides an extended location service for SIP entities registration and discovery in the P2P overlay. The architecture is composed of some of the nodes with high capacity (bandwidth, CPU, memory) and availability (uptime, public IP address) which act as supernodes and form the DHT, and other ordinary nodes which connect themselves to one or more supernodes without being part of the DHT. User location is obtained using the SIP module if the node is ordinary peer or using the DHT module, if the node is a supernode. The DHT module maintains the peer information and performs DHT operations such as find, join and leave. Peers should use DHT-based messages to interact with the DHT. Once the user location is done, the call setup or instant messages can be sent directly via the SIP module. Authors validated their approach using OpenDHT middleware [8]. The architecture can

easily work with other DHT-based middleware but cannot interoperate with non DHT-based P2P protocol. Then, the approach is middleware-dependent.

Another example of “SIP-using-P2P” solution, proposed by Holger Schmidt et al. in [9], uses JXTA-based P2P architecture. A JXTA-based SIP location service called JXTA-LOC has been deployed and used by standard SIP entities. This leads to SIP entities that externally behave according to standard SIP but internally use P2P mechanisms to interact with the JXTA-LOC database which resides in the JXTA overlay. It is used by the SIP registrar to store information about UAs and by the SIP proxy which discovers the target UA in the JXTA network and directly forwards the message to it. While this approach intends to be compliant to SIP standard, it is not middleware-independent.

Chuan Zhu et al. have also proposed in [14] an approach for a communication system for wireless networks through which various devices in mobile networks can communicate with each other. Their architecture is made up two logical overlays: the JXTA-based P2P overlay and the SIP overlay. Two main modules have been defined. The first module called JXTA-SIP bridging is responsible for the conversion between SIP messages and JXTA messages. The second module called SIP TU (SIP Transaction User) is responsible for the registration of nodes as well as the session establishment between them. The SIP TU module performs all SIP operations except the SIP location mechanism which is replaced by the JXTA location service. The main drawback of this approach is that the network building and functionalities are strongly based on JXTA network. This makes the approach being middleware-dependent.

In the architectures mentioned in [7], [9] and [14], the SIP REGISTER messages sent by the UACs result in the UAC contact information being published in the P2P overlay, using the mechanisms provided by the underlying P2P layer (i.e. DHT or JXTA). The contact information is stored according to the UAC’s SIP URI (Uniform Resource Identifier), which acts as the key in the overlay. SIP messages that require routing (e.g. INVITE, REGISTER), trigger contact information discovery in the overlay using the SIP URI as a key and employ the available overlay mechanism (i.e. DHT or JXTA). The discovered contact information is then used to send the SIP message to the targeted node. Both approaches are middleware-dependent.

The author of [13] has proposed a Secure Distributed Name Service for P2P SIP (P2PNS). The goal of the proposal is to provide a distributed name service on a DHT overlay to resolve AoRs (Address of Record) to Contacts URIs without relying on DNS (Domain Name System) and centralized SIP servers. The approach is based on a modular architecture composed of four separate layers. The Key Based Routing (KBR) layer allows messages routing to the nodeIDs while the Distributed Hash Table (DHT) layer provides distributed data storage. The P2PNS Cache layer provides name resolution service in addition to cache the AoRs records while P2P SIP proxy layer allows connecting legacy SIP UAs to the P2PNS service. The work includes a two-stage name resolution mechanism to handle frequent IP address changes in order to reduce communication cost in dynamic environments and provides security mechanisms in a decentralized environments. Indeed, the first name resolution stage resolves the node’s AoR in nodeID at DHT layer and the second name resolution stage resolves the nodeID in IP address at KBR overlay layer. According to the author, the motivation of the two-stage resolution is that modification of data records on DHT is expensive due to security mechanisms while IP address changes are efficiently handled on KBR layer. The approach ensures also uniqueness of the AoRs by preventing identity theft. When a Peer X wants to establish a call to the AoR A of Peer Y, firstly the UA of Peer X sends a SIP INVITE to its local P2P SIP proxy. Subsequently, the proxy queries P2PNS service to resolve

A. The P2PNS layer first fetches the corresponding nodeID for AoR A from the DHT. Then, the obtained nodeID gets resolved to the current IP address of peer Y from the KBR. Finally, the SIP INVITE message is forwarded to the UA of Peer Y through its proxy. While this approach could efficiently work on any DHT-based P2P infrastructures, it remains DHT-based middleware dependent.

b) Hierarchical architectures

Other approaches [4] and [60] belonging to SIP-using-P2P have been explicitly proposed to deal with the connectivity issue of heterogeneous P2P overlay and the overhead problem of SIP messages when SIP extension is used to maintain the overlays. In reference [4], various P2P overlays (such as Pastry, CAN) are respectively used as the underlying route discovery protocol. To connect peers from heterogeneous overlays, an upper level overlay (global overlay), is formed, to interconnect the local overlays. That is, each overlay elects one or more powerful peers to be the gateway-like nodes that route messages among heterogeneous overlays. The elected gateway-like node is defined as the P2P proxy. The benefits of introducing such hierarchical architecture are that peers in heterogeneous overlays can be interconnected and the balance between low signaling overhead and easy management can be achieved. The approach is middleware-dependent because the peer discovery is strongly based of the P2P infrastructure used. In addition, this approach adds two new headers to SIP messages in order to insert the node ID and the overlay ID. Due to its hierarchical architecture, the destination peer' contact information discovery could also lead to low performance of the communication if the SIP messages should go through many heterogeneous overlays.

Reference [60] proposed the H-P2PSIP (Hierarchical P2PSIP) approach similar to [4]. The user on the H-P2PSIP node acts as the SIP UA and uses the DHT service to locate other users. After that, nodes with higher capacity are elected to be Higher Level Overlay Peer (HiLO-Peers). The H-P2PSIP node with lower capacity acts as Lower Level Overlay Peer (LoLO-Peer). The LoLO is based on specific DHT algorithms such as Chord, Pastry, etc. An overlay within the same DHT could also split into several LoLOs based on the geographic information to reduce the session setup latency. However, the HiLO-Peer has two main functionalities. Firstly, HiLO-Peer acting as the SIP proxy invokes both local and upper level DHT services to interconnect various P2P SIP overlays. Secondly, it also performs operations of stateful SIP servers. A universal DHT algorithm for HiLO is required and Bamboo is suggested to perform the role. Then, the HiLO-Peer with multi-stack (LoLO DHT stack and HiLO DHT stack) performs the gateway-like behavior to connect heterogonous LoLOs. Peers within the same LoLO speak the private P2P protocol for DHT operations. Peers within HiLO and peers between HiLO and LoLO speak the SIP protocol. SIP elements such as SIP UA and SIP Proxy employ the DHT block as a resolver to determine IP address, port and transport protocol of the next hop element. Authors use DHT rather than DNS to implement the location function. To ease developers to port multiple DHT stacks to an H-P2PSIP node, authors leverage the common API to make various DHTs pluggable. While being DHT-based middleware-independent, this approach is not fully middleware-independent. In addition, as in [4], the session could be delayed due to the contact information lookup operation which is performed at many levels (caller overlay, interconnection overlay and the callee overlay). This could lead to performance issue.

To summarize, a major limitation of existing "SIP-using-P2P" architectures is that the architectures based on JXTA ([9] and [14]) are middleware-dependent by virtue of their design. As for the architectures proposed in [7] [13] [4] and [60], even though it may be possible to plug the

proposed mechanisms into different DHT-based middleware, they will not be able to work with non-DHT-based middleware and then are not middleware-independent. In addition, the performance requirement could be compromised in hierarchical architectures.

III.2.2 P2P-over-SIP

This category of architectures uses SIP messages to maintain the P2P overlay and/or to publish and discover contact information. A very few works belong to this category.

Bryan et al. proposed dSIP [5] which falls in this category. dSIP is a SIP-based system that uses P2P mechanisms to remove the need for central servers in SIP and SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) based communications systems. dSIP evolved from early work done by the same authors on the SoSIMPLE [6] P2P SIP project. We will focus only on giving an analysis of dSIP.

dSIP extends the SIP REGISTER message via additional headers to perform overlay maintenance and location service procedures. It organizes the overlay peers using a DHT P2P structure, where each peer or resource/data is assigned a Globally Unique Identifier (GUID), calculated using the DHT algorithm. The dSIP architecture requires the support of Chord, but other DHT algorithms can be added.

The overlay maintenance includes nodes joining and leaving and messages transfer between peers. When a peer wishes to join the overlay, it sends a REGISTER message to a bootstrap peer in the overlay. The REGISTER is then routed towards the peer closest to the joining peer's GUID. The message can be routed using different routing schemes. Following the iterative scheme, for instance, the bootstrap peer looks up the peer it knows, is nearest to the joining GUID and responds with "302 redirect" to this closest peer. The joining peer sends a REGISTER to that peer and the process repeats until the closest peer (according to the DHT algorithm) is found. The closest peer then sends to the joining peer, its DHT state information, which allows the joining peer to learn about other peers in the overlay (neighbors). This process is referred to as node registration. REGISTER messages are also periodically exchanged between peers to maintain information about the overlay as nodes join and leave.

After a peer joins the overlay, it publishes its contact information, making it discoverable by other peers. This process is referred to as resource registration. It publishes the contact information using the peer SIP URI as the key. The joining peer sends a REGISTER message, routed as in the node registration phase, to the peer closest to the resource GUID (calculated from the peer SIP URI). The closest peer stores the mapping between the resource ID and the contact information, and replies with a SIP OK message.

To discover the contact information of another peer, an overlay peer sends a REGISTER message to the peer closest (among its known peers) to the resource URI. The target peer answers with a SIP OK response including a special header that contains the required contact information. The location service in dSIP is either located in the SIP UACs or in an adapter. The adapter connects the standard SIP UACs to the P2P SIP overlay.

A major limitation of this category of architectures is that they explicitly extend standard SIP messages and then are not compliant to SIP standard.

Table III.1 summarizes the different approaches reviewed in this section with respect to our requirements.

Requirements P2P SIP approaches		Basic P2P features	Independence from P2P in- frastructure	Compliance to SIP specifications Or no SIP extension	Performance
SIP-using-P2P	RELOAD [11]	YES	YES	NO	N/A (Not Available)
	DHT-based architecture [7]	YES	NO	YES	Claim to be reasonable
	JXTA-based architectures [9]	YES	NO	YES	Good RTT but more traffic in comparison to standard SIP
	Communication system for wireless networks [14]	YES	NO	YES	N/A
	Secure distributed name service (P2PNS)[13]	YES	NO	YES	N/A
	Hierarchical P2PSIP[4][60]	YES	NO	NO	LOW
P2P-over-P2P	Dsip[5]	YES	YES	NO	N/A
	SoSIMPLE[6]	YES	YES	NO	N/A
	MISE-P2PSIP (our proposal)	YES	YES	YES	Reasonable

Table III.1: State of art summary

III.3 Conclusion

This chapter presented the requirements of our MISE-P2PSIP architecture and provided an analysis of the existing P2P SIP architectures. Unlike the existing architecture, our architecture aims to be fully middleware-independent and compliant to SIP standard. The next chapter will mainly describe the principles of our architecture followed by our prototype and performance simulation results.

Chapter IV

MISE-P2PSIP architecture

In this thesis, we follow an entirely different line of research with respect to existing proposals. Namely, we propose a middleware-independent P2P SIP architecture called MISE-P2PSIP (Middleware-Independent and SEcure Peer-to-Peer SIP architecture). Our solution is targeted to support VoIP signaling applications where user agents can be simple devices like IP phones, and the P2P overlay can be selected at deployment (as opposite to design) time, depending on which middleware is available in the specific deployment situation.

Our architecture integrates the location service into the proxy and distributes the registrar. Our P2P SIP proposal belongs to “SIP-using-P2P” category already discussed in the previous chapter. The main difference with the current SIP-using-P2P proposals on one side is that the user does not register directly its contact information in the overlay. In our architecture, the user registration happens at SIP layer as specified in RFC 3261. Then, the SIP client simply creates a SIP REGISTER that it sends to its Registrar server. Instead of keeping the user contact information (IP address/port) in a centralized database, the Registrar server stores the information in the overlay. This allows us to provide distributed storage of the contact information by being compliant to conventional SIP. On the other hand, the SIP location service is performed by the proxy at SIP layer as in conventional SIP. When a Registrar server gets a contact information request from a proxy, it resolves the request into overlay lookup request and retrieves the contact information from the overlay and sends the response to proxy as specified in RFC 3261. A major difference with the current “P2P-over-SIP” proposals is that during call establishment, we do not use the DHT or other overlay routing mechanisms to send the SIP messages and we do not perform any SIP message extensions. This gives us the advantage to be almost fully independent from the P2P infrastructure and being compliant to conventional SIP.

In this chapter, we first introduce the overall architecture and assumptions, followed by the self-organization procedures. The call procedures and an illustrative scenario have been also presented. Moreover, we fully describe the software architecture and present our prototype. Finally, some simulation results have been released.

IV.1 Assumptions and overall architecture

In this section, we assume our system functions in a single SIP domain. The domain may have many proxies and each proxy can manage a limited number of client peers. Our architecture is depicted in figure IV.1. It defines four functional entities: Domain Controller (DC), Group Registrar (GR_x), Proxy (P_x) and Client (C_x). The default DC is the first node that joins the overlay

and is a non-permanently centralized SIP registrar entity that acts as the overlay bootstrapping node. However, the default DC is replaced by a participating GR, when it leaves or fails. Upon joining the overlay SIP domain, each node should first contact the DC. We assume that the joining node knows the contact information of the current DC (e.g. preconfigured with the DC address). The DC assigns appropriate roles (i.e. Client, proxy, or GR) to the joining nodes and provides support functions for self-organization. In addition, the DC keeps the list of all participating servers (proxy and registrar) and stores it in the overlay. The GR node is SIP registrar entity that has the capability to store, in part or as a whole, the SIP client contact information. GRs provide for a distributed storage of client contact information in the overlay. The proxy peer is a non-permanently centralized SIP proxy server which has the role to transmit a message from a source client peer to a target client peer in the P2P overlay. It integrates the location service. For performance reasons, as a peer with higher capabilities, it integrates a caching mechanism to store previously obtained contact information to avoid querying before each call. The client peer is a SIP client participating in the P2P overlay.

The peer nodes are organized in several groups. Each group is composed of one proxy, one GR and 0 to N clients. Each domain has a single DC, which is responsible for creating and managing the groups. We assume for simplicity sake in this version of the architecture that every node can play any of the roles.

Our overlay is composed of two levels. The first level contains the contact information storage nodes, meaning the GRs, along with the entry point, i.e. the DC. The second level includes the client peers, which initiate and receive calls, and proxy peers, which act as application-level routers, just as in the SIP standard.

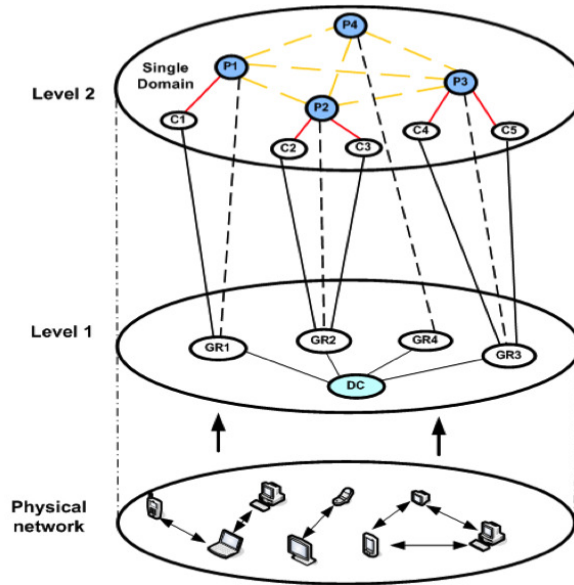


Figure IV.1: MISE-P2PSIP Overall architecture

IV.2 Self-organization

Self-organization is an important feature of P2P networks. In general, it ensures that the peers can easily join and leave the overlay. In our architecture, an important characteristic which is the group creation is added to allow the efficient management of nodes due to the coupling of peer

node with different SIP entities. This section describes the group creation, the DC election and the joining/leaving procedures. We take into account both voluntary and involuntary departures.

IV.2.1 Groups' creation

When a node (N_i) joins a domain, it contacts the DC. The DC checks if there is a group (G_i) with number of clients $< N$, where N is a configurable number. If it is the case, the DC adds N_i to that G_i . If there is already a proxy in the group, the DC sets the incoming node as a client. If there is no proxy, the incoming node is set as the proxy.

If no suitable group exists, the DC creates a new group and sets the incoming node as the GR. Each group is identified by its ID. The DC maintains the list of the groups and for each group, it updates and stores the GR and proxy addresses, and the number of clients.

IV.2.2 DC node' setting up procedure

The default DC is considered to be trusted. The default DC provides to each GR a certificate signed by itself. The certificate contains an ID which is the number of entering of the GR in the overlay. Then, at specific time all GRs should hold a certificate signed by the current DC. Each GR has public and private keys generated by the DC. DC provides its public key to all GRs. When a DC leaves or fails, it should be replaced by one of the attending GRs. Following the technique used in the "Web of Trust" [128], a GR that wants to become DC will be considered reliable or trusted only if it is able to provide a certificate signed by the old DC (the previous one). Thus, in order to avoid that unauthorized node becomes the new DC, the candidate DC should present the certificate it holds to the other GRs. The other GRs then verify the certificate using the public key of the DC (the old one). If the verification successes, the candidate DC receives a kind of agreement from each GR in a SIP *NOTIFY* message. The candidate DC answers with an *OK*. The candidate DC is elected when it receives agreements from all GRs. The new DC creates for each GR a new certificate that it signs and sends to it. Along with the signed certificate, the new DC sends also its public key to each GR. The new certificate replaces the old one and each GR that receives it, knows that a new DC has started working. Therefore, in the case where two candidates DC trigger together the election procedure, a given GR answers (by sending the SIP *NOTIFY*) to the candidate DC with higher ID.

IV.2.3 Node joining procedures

Figure IV.2 summarizes the joining procedures. When a new node joins the network, it sends a SIP *REGISTER* message to the DC. In a *200 OK* message, the DC notifies the joining node of its role (i.e. GR, Proxy, or Client). The information is inserted in the body of the *OK* message by piggybacking the optional attribute "u" (containing the URI of description). Then, no SIP extension is needed to do this. This SIP *REGISTER* message is only used to establish a preliminary communication of the joining node with the DC.

If the joining node is a client, it gets its role, its group' ID and the addresses of the proxy/GR of the group to which it is added (in the *200 OK*). The joining node should properly register itself to its own GR. In addition, the client might send a SIP *SUBSCRIBE* message to the DC, to its GR and its proxy in order to be notified about important events (e.g. proxy or GR leaving).

If the joining node is a GR, the node is also given its role and the ID of its group. The DC sends a SIP *SUBSCRIBE* message to the new GR. The GR registers to the DC and sends *SUBSCRIBE*

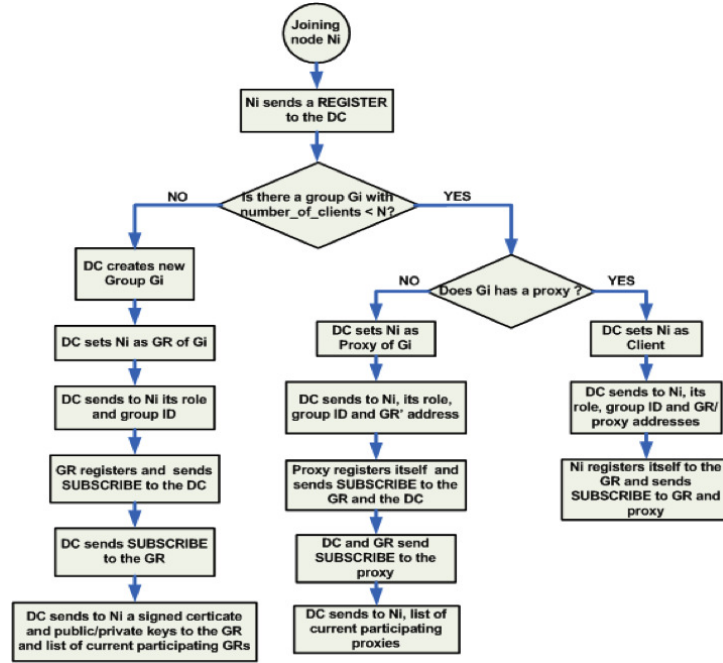


Figure IV.2: Joining procedures flow charts

message to the DC. In addition, the GR gets a signed certificate, its public and private keys along with the DC' public key from the DC. The DC sends also the list of the participating GRs to the new coming one. The list is updated by the DC every time that a GR joins or leaves.

When a proxy joins the network, it gets its role, its group' ID and the address of the GR of its group. The proxy registers itself and sends a *SUBSCRIBE* message to the GR and the DC. Subsequently, GR and the DC subscribe to that proxy. The DC gives to the proxy the list of participating proxies and updates it every time that a proxy joins or leaves.

IV.2.4 Node leaving procedures

This section describes first, voluntary nodes leaving procedures and subsequently the involuntary leaving procedures.

a) Graceful leaving

i. A client is leaving

The client sends a *REGISTER* message to the GR to unregister itself. The *REGISTER* message is the same as that used in the joining procedure except that the 'Cseq' is different and the 'expires' parameter in the header is set to 0 (zero). The GR sends a SIP *NOTIFY* message to the DC indicating that one client has left group Gi. This step allows the DC to know the current number of clients in each group.

ii. A GR is leaving

If the group includes at least one client, the GR (randomly) chooses one client (Ci) to set as the new GR (N-GR). The GR sends a *NOTIFY* message to Ci to inform it about its new role. It also sends a *NOTIFY* to the other clients to inform them about the substitution of the GR. The clients then register to the N-GR. To keep the DC informed, the leaving GR sends a *NOTIFY* message to the DC to announce that it is leaving and that Ci is the new GR. It then unregisters

from the DC that *SUBSCRIBES* to N-GR. The DC updates the list of GRs, it held and sends it to all GRs.

If the group only has a proxy, that proxy becomes the new GR. If the GR is the last node in the group, the GR sends a *REGISTER* message to the DC to unregister itself and the group is deleted.

iii. A proxy is leaving

The leaving proxy (Ni) sends a *REGISTER* message to the GR to unregister itself. If there is at least one client in the group, the GR chooses one client in the same group and sets it as the new proxy. The new proxy is informed about its new role via a SIP *NOTIFY* message. The GR also notifies all the clients in the group about the new proxy, and informs (using a SIP *NOTIFY*) the DC about the newly elected proxy and the departure of the old one. The DC updates the list of proxies, it held and sends it to all proxies.

If there is no client in the group, the GR notifies the DC that Ni has left.

iv. The DC is leaving

The DC contacts one GR in the overlay. This GR will become the new DC by using the DC setting up procedure described above. In this case, a new GR needs to be set up in the group, the old registrar was responsible for, using the procedure described above. The new DC should broadcast its contact information to the proxies.

b) Involuntary departure

Involuntary departure refers to the case the node has left without notifying its departure because of power failure, battery lost, etc.

The failure of a Client node should be detected by its proxy. The failure of a proxy is detected by the GR of its group and the GR failure is detected by the DC. We provide also mechanism to allow replacing the DC which acts as bootstrap node when it fails.

i. Client failure

The proxy sends periodically keep-alive messages to the client nodes under its responsibility. When the responses to these keep-alive messages are delayed the proxy infers that the concerned client node has failed. Then, the proxy sends SIP *NOTIFY* messages to the DC and the GR to inform them. The DC will update the number of client node of this group and the GR will unregister the node.

ii. Proxy failure

The GR sends periodically keep-alive messages to the proxy of its group. If the response to a given keep-alive message is not received, the GR infers that the proxy has suddenly left. The remainder of the procedure is the same as in the case of graceful leaving.

iii. GR failure

The DC sends periodically keep-alive messages to the GRs of its domain. When the DC does not receive response from a given GR, it triggers the procedure of replacement of this GR. The procedure is as following.

The DC informs the proxy of the group which will have in charge the setting up of the new GR. If the group includes at least one client, the proxy chooses randomly one client (Ci) to set as new GR (N-GR). The proxy sends a *NOTIFY* message to Ci to inform it about its new role. It

also sends a *NOTIFY* message to the other clients to inform them about the substitution of the GR. The clients should register to N-GR. The proxy should also register itself. When the N-GR is set, the proxy sends to the DC the address of the N-GR in the *NOTIFY* message. The DC subscribes to N-GR in order to get notifications in the future. N-GR might register to the DC and send also the *SUBSCRIBE* message to it.

If the group only has a proxy, the proxy becomes the new GR. If the GR is the last node in the group, the DC unregisters the old GR and deletes the group.

iv. DC failure

The first node that joins the overlay is configured as default DC which is like a bootstrap node for the overlay. However, all nodes acting as registrar servers can also act as DC when needed. In order to handle the DC failure, the other GRs send periodically keep-alive messages to the DC. When the DC does not answer the keep-alive messages, the first GR that detects the DC failure triggers the procedure to become the new DC of the SIP domain. This procedure has been described above.

Note that each node holds different kinds of SIP modules with are activated according to the role played by the node. Those modules will be described later in the chapter.

IV.3 Call procedures

During a call setup, the role of the proxy is mainly to forward the SIP *INVITE* message based on the SIP URI of the target's node. There are two communication scenarios:

- The source Peer Client and the target Peer Client are in the same group. The request should be sent to the proxy of the group. The proxy will request the contact information of the target client to the group's GR, if the information is not in its cache or has expired. The GR replies by sending the requested information. Note that the contact information is retrieved from the overlay and converted in non-SIP messages (i.e. database lookup) as specified in RFC 3261. The proxy forwards the *INVITE* request to the target node and caches/updates the contact information received from the GR.
- The source Peer Client (C_i) and the target Peer Client (C_j) are in different groups. After checking the local cache and GR, the proxy (P_i) sends the *INVITE* message to all proxies in the domain. When a proxy receives the *INVITE*, it processes it as in the previous case. The proxy (P_j) that is responsible for the target node will then forward the request to the target node. The response from the target node should take the same path back to reach the call initiator. When the proxy P_i receives a response from P_j , it forwards the response to the source Peer Client and stores in its cache that the target node (C_j) is accessible via the proxy P_j .

IV.4 Call scenarios

This section gives examples of the call scenarios that we mentioned in the previous section.

IV.4.1 The two communication parties are in the same group

Figure IV.3 shows a call scenario where a client C1 in group1 is willing to establish a SIP session with a client C2 in the same group. The two clients register each to the group registrar GR1 at steps 1 to 4. Subsequently, Client C1 creates the SIP *INVITE* request and sends it to its proxy P1 at step 5. P1 checks its cache (step 6) and does not find the IP address of C2 which is the target client. Then, P1 asks C2' IP address to GR1 and gets it since C2 is the same group as C1 (steps 7 and 8). P1 forwards the *INVITE* to C2 (step 9). C2 answers with the SIP *OK* response which is sent to C1 through P1 (steps 10 and 11). P1 stores the IP address of C2 for future uses at step 12.

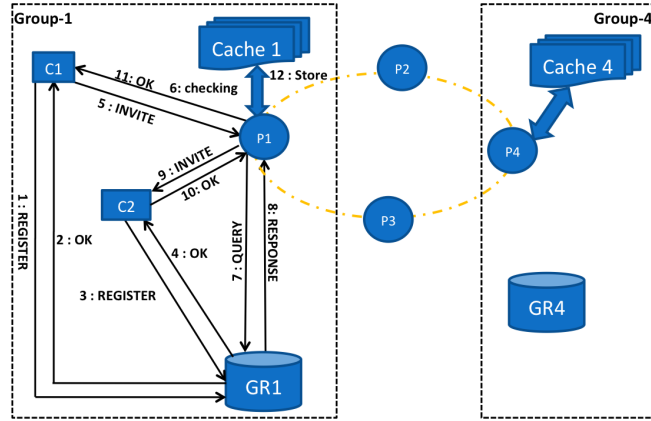


Figure IV.3: SIP Call in the same group

IV.4.2 The two communication parties are in different groups

Figure IV.4 shows a scenario where a client C1 in group1 is willing to establish a SIP session with a client C2 in group2. First, the two clients register to their respective GRs (steps 1 to 4). In step 5, C1 sends a SIP *INVITE* message to its proxy (i.e. P1). P1 checks its cache and its GR but does not get anything about C2' IP address (steps 6 to 8). Then, P1 forwards the request to the other proxies at step 9. When P2 gets the request, it checks its cache at step 10, but the cache does not include any information about C2. It then asks its GR for C2' contact information (steps 11 and 12) and forwards the request to it at step 13. The response of C2 is forwarded back to C1 via P2 and P1 (steps 14 and 15). P1 forwards the OK response to C1 (step 16) and stores in its cache the information that C2 is accessible via the proxy P2 (step 17).

IV.5 Multi-domains communication

Each SIP domain is mapped to a single overlay. Taking into account multiple SIP domains, the SIP call scenario might change. Due to our requirement on the SIP compliance, the inter-domain call cannot occur without the help of the DNS. According to RFC 3263, a specific host name could be added to the DNS for a given domain. Then, we record a specific DC' host name in the DNS. The query of the domain name will then display only the DC contact information for this domain. When a DC enters in a domain or is newly elected, it should add its information in the DNS so that DCs' list in the DNS is always updated. Actually, when a proxy is looking for a client which is not in its domain, the request is forwarded to the DC which resolves the domain

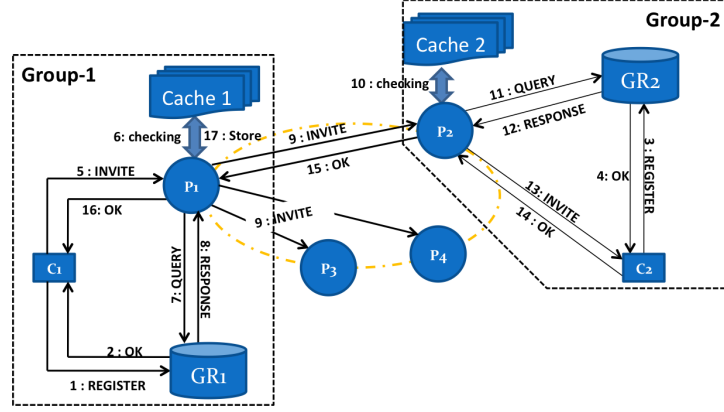


Figure IV.4: SIP Call between different groups

name in the DNS and gets the address of the DC of the destination domain. Thus, the request is directly forwarded to this DC. The DC of the destination domain sends the request to a random proxy in the domain. At this point, the request is handled as in a single domain. The response is forwarded following the same procedure. DCs' addresses are not locally caching in a single domain because of the constant change of the network.

IV.6 Implementation

This section focuses on the implementation of our P2P SIP network. We present the software architecture of the nodes and the prototype.

IV.6.1 Software architecture

The overall software architecture, including the interactions between the different types of nodes, is presented in figure IV.5. We define three (3) main modules: “P2P SIP”, “SIP to Overlay” and “Overlay Peer Management” that are described below.

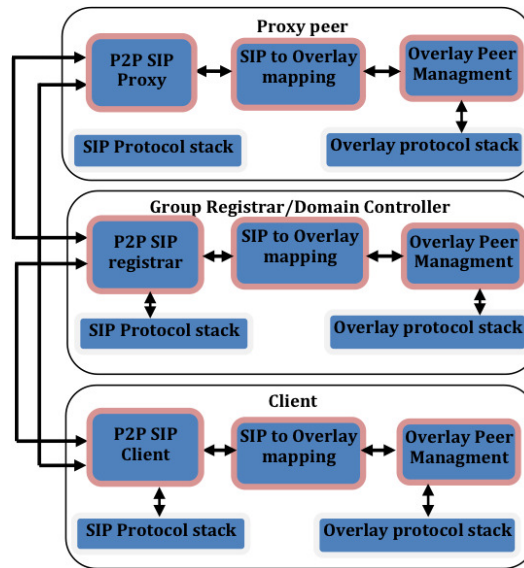


Figure IV.5: Software architecture

a) P2P SIP Proxy, P2P SIP Registrar and P2P SIP Client modules

P2P SIP proxy, P2P SIP registrar and P2P SIP client are modified standard SIP proxy, SIP registrar and SIP client in the sense that they implement the joining and the leaving procedures previously described in addition to the standard SIP multimedia sessions procedures.

From the joining perspective, this means, for instance, that P2P SIP proxy, P2P SIP registrar and SIP client modules start by creating a SIP *REGISTER* message that is sent to the DC. Then, the DC uses P2P SIP registrar module to process the request and to create an *OK* response with the assigned role to the joining node.

P2P SIP proxy, P2P SIP registrar, and P2P SIP client modules also interact with “SIP to overlay mapping” that interacts with the “overlay peer management”. The interaction with the “SIP to overlay mapping” is used to isolate the P2P SIP proxy, registrar and client from the specifics of the middleware. The P2P SIP modules contain many useful procedures which process SIP messages. Among them, there are:

- *Process_Request*: It processes SIP requests.
- *Process_Response*: It processes SIP responses.
- *Send_SIP_message*: It allows sending SIP messages between SIP nodes.

b) The Overlay Peer Management

“Overlay Peer Management” is the middleware-dependent part of the architecture and interacts with the “SIP to overlay mapping” module. It mainly includes procedures which allow joining nodes to become peers and then to get a Peer ID in the overlay. It also enables the storage and retrieval of contact information. Those procedures are:

- *Become_peer*: It allows a node to become a peer and to get a unique peer ID.
- *Store_contact* information: It allows the storage of contact information in the overlay.
- *Retrieve_contact* information: It allows the retrieval of contact information from the overlay.

c) SIP to Overlay mapping

The SIP to Overlay mapping segment understands/speaks SIP on one side, and understands/speaks the specific middleware primitives on the other side. It enables the mapping of SIP messages to overlay messages and vice versa. The main procedures in this module are:

- *Mapping_of_SIP_message_to_overlay_message*: it allows information to be extracted from SIP messages in order to create specific overlay messages. This procedure is used, for instance, when the GR receives a SIP *REGISTER* message from a Peer Client or a Peer proxy and has to store the contact information in the overlay. The appropriate overlay message is generated by this procedure and sent to the “overlay peer management”, which has in charge to store it using the overlay protocol stack.
- *Mapping_of_overlay_message_to_SIP*: it allows the creation of SIP messages containing information that is retrieved from the overlay. This procedure is for instance used by the GR to send the contact information requested by a proxy. The appropriate response message is generated by this procedure and sent to the “P2P SIP registrar” module which then sends it to the proxy using the SIP protocol stack.

IV.6.2 Prototype

We have prototyped a sub-set of our architecture on top of two different middleware. The prototype provides call establishment between two SIP clients. For each middleware implementation, the two communication parties run on two laptops with 1.6 GHz of processor frequency and 1Go of memory. The prototype is basic but the goal is to show the successful integration of P2P and SIP leading to a real SIP signaling messages exchange between two parties.

The P2P part of the prototype is implemented using JXTA middleware on one side and Chord-based ProActive middleware on other side, while the SIP part is implemented using JAIN SIP [17].

For both prototype implementations, the “P2P SIP”, the “SIP to Overlay” and the “Overlay Peer Management” described in the previous section, have been implemented on client side.

a) **Prototype on Jxta middleware**

JXTA [16] is used as the middleware. It is a set of XML (Extensible Markup Language) based protocols and a Java reference API that provides a generic implementation framework for P2P. The framework provides the basic elements for P2P computing, including peers, peer groups, discovery, service, and messages routing. All JXTA network resources are represented by an advertisement. An advertisement is created using language-neutral meta-data structures and is represented as XML document.

Two communication transport types have been implemented:

- Using SIP ports: The P2P node uses TCP/UDP as transport protocol to send signaling messages.
- Using JXTA Pipes: The P2P node uses JXTA pipes to send signaling messages. Indeed the pipes are like ports used in JXTA to create communication channels in order to send messages between peers. The pipes could be asynchronous, unidirectional and non reliable.

We have successfully tested the following functions:

- The client successfully enters in the overlay, publishes directly an advertisement containing its IP address in the overlay since the GR has not been implemented.
- A client node can successful initiate a SIP call (generating and sending an INVITE) toward a target client without knowing a priori its IP address.
- The IP address of the destination node is successfully discovered by the source node.
- The two clients can successful exchange SIP signaling messages (INVITE, OK and ACK).

Figure IV.6 shows the call interface provided by the prototype. As said before, there are the possibilities to initiate/receive call using SIP port or to initiate/receive call using JXTA pipes.

The peer successful joins the overlay, creates input JXTA pipe in order to be able to receive message and publish its IP address in the overlay.

In the example shown here, we choose to make SIP call using SIP ports. Then, a source peer pushes the Play SIP call button to initiate a call (by creating a SIP INVITE). Figure IV.7 shows important additional messages. At this step, the source peer has found the IP address of the destination node and sends the INVITE to it.

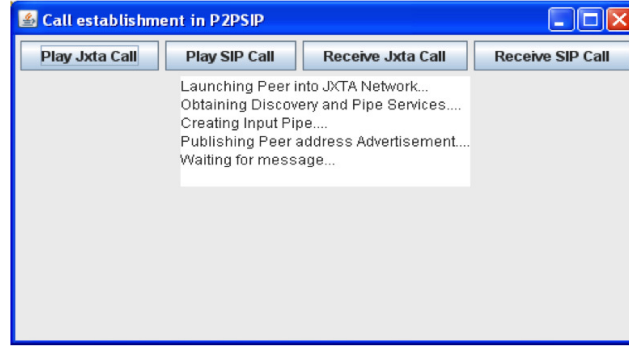


Figure IV.6: Launching interface

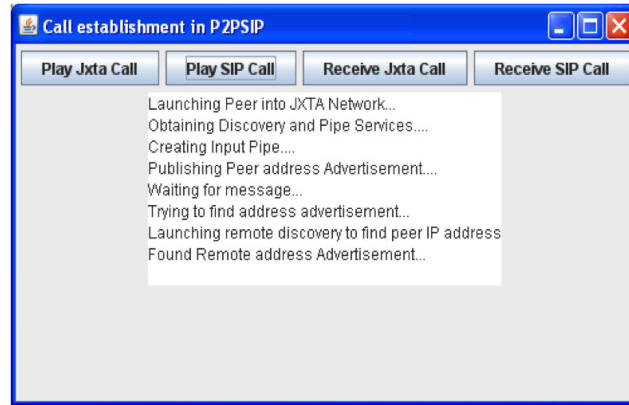


Figure IV.7: Destination node IP address discovery

The *INVITE* message created and sent by the source node to the destination node is shown in figure IV.8. Once the callee received the *INVITE*, it creates the OK that it sends to the caller and receives back the ACK message. Figure IV.9 depicts the rest of messages transaction on the destination node side.

Figure IV.10 depicts an example of the IP address/port advertisement sent by the source node into the overlay.

b) **Prototype on Chord-ProActive**

The P2P part of the prototype is implemented using the open source implementation of Chord based on ProActive middleware called Chord-ProActive [68]. Indeed, Chord network behaviors (architecture, functionalities, etc) are implemented using ProActive middleware. Chord-ProActive used the concept of tracker in order to maintain references of peers in a Chord P2P network. This allows having at any time entry points into the network. In theory, the references of all peers are not stored but a large set is kept to tolerate a certain number of failures. Each tracker is associated with a Chord network and keeps a reference to peers belonging to this network. The tracker contains node IDs calculated automatically when node joins and the key.

We suitably modify the available source code of Chord-ProActive to fit our needs. For instance, upon node registration in the overlay, the tracker is modified to contain the node IP address and port of each node in addition to the information mentioned above and an additional function is implemented to allow a physical node to join the Chord network.

The following functionalities have been successfully tested:

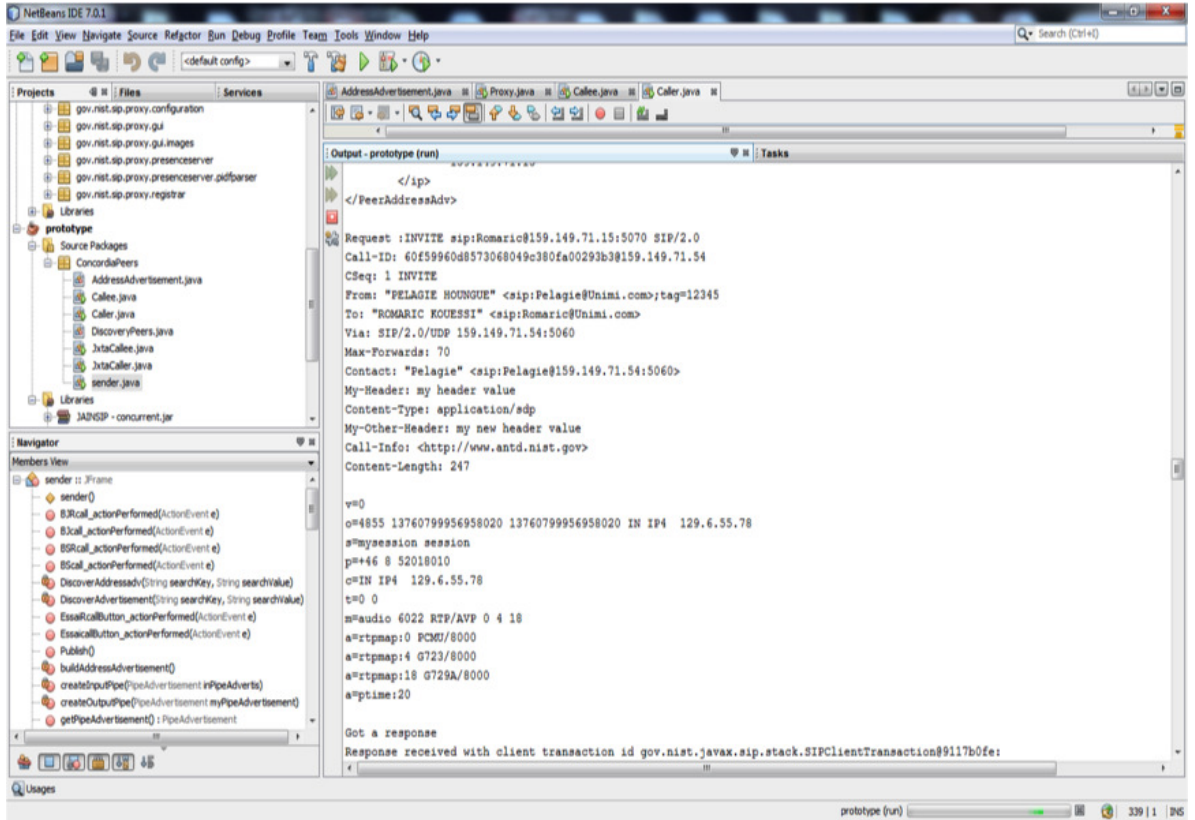


Figure IV.8: INVITE message sent by the source node

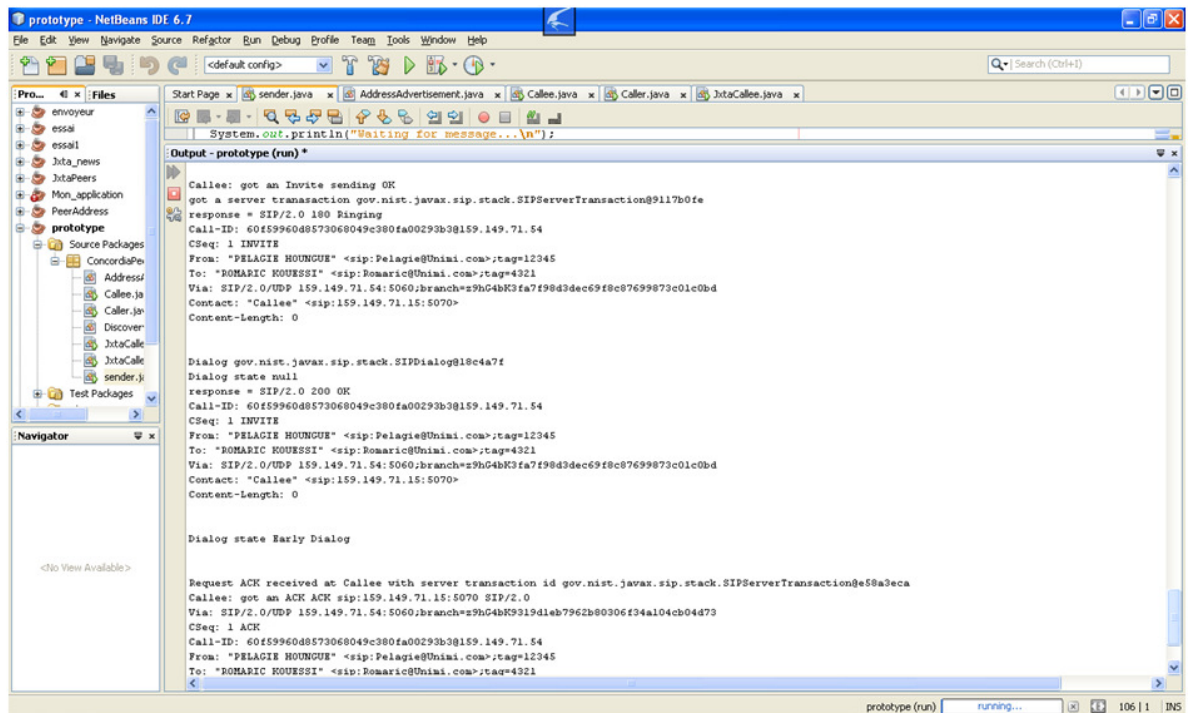


Figure IV.9: SIP messages on destination node side

- Nodes are able to join the network from different computers and become chord peer.
- Nodes can gracefully leave the network and the tracker is successfully updated.

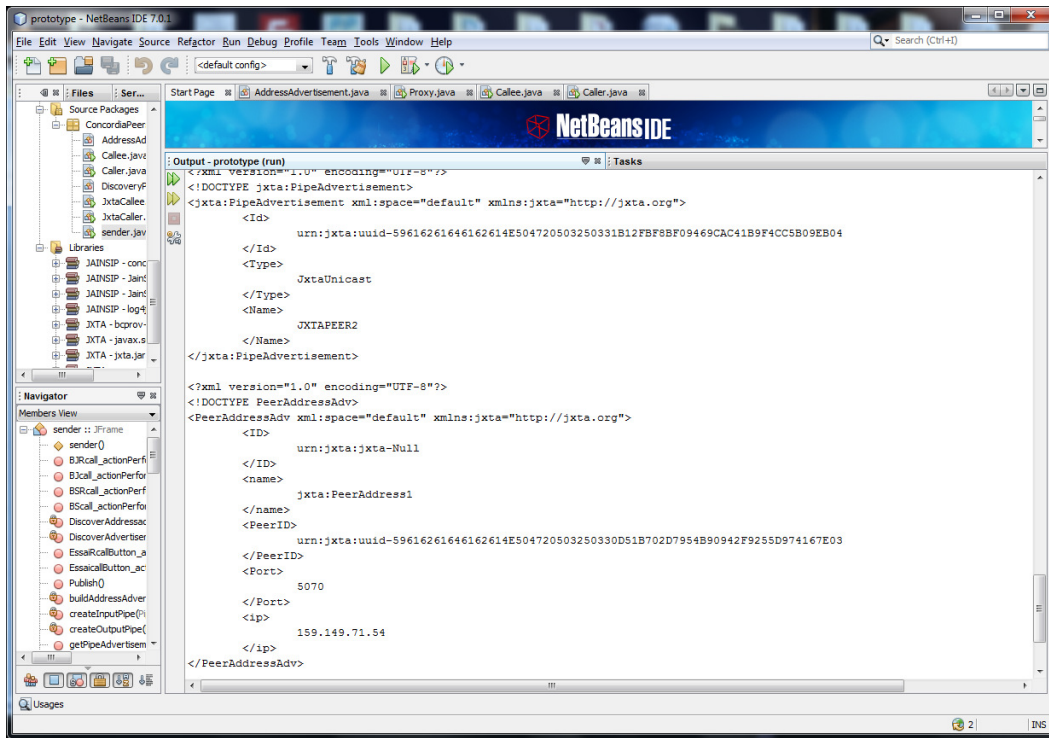


Figure IV.10: Example of published IP address/port advertisement

- Nodes in the chord network have a kind of contact list from which they choose the destination node.
- SIP signaling messages are successfully exchanged between the two nodes.

Figure IV.11 shows the application interface with an example of virtual nodes organizing in ring.

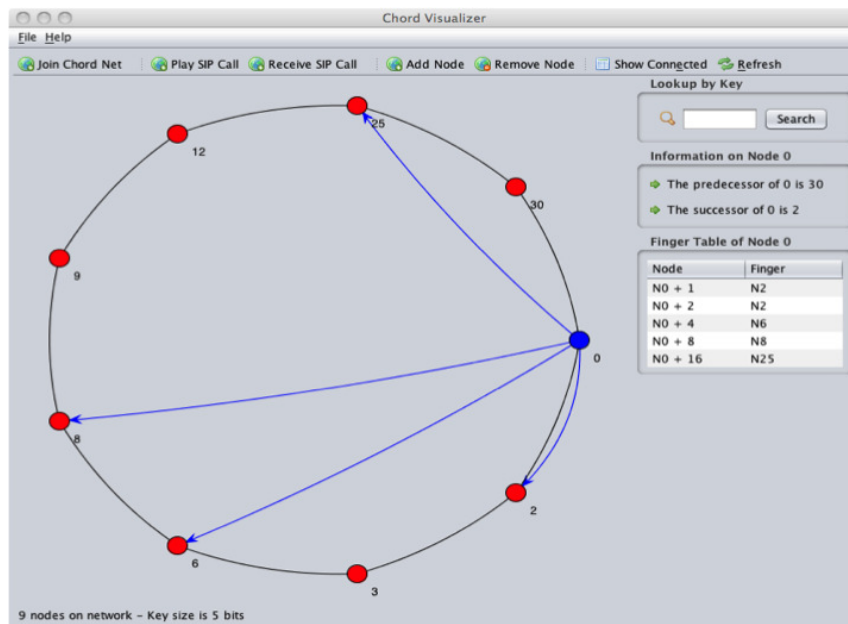


Figure IV.11: Example of Chord network

The interface makes available some buttons that allow performing important functionalities.

Button “Join Chord Net” allows physical (real) nodes to join the overlay. Buttons “Add Node” and “Remove Node”, respectively allow adding and removing virtual nodes from the network. Buttons “Play SIP Call” and “Receive SIP Call” allow initiating and terminating SIP call session. In addition, it is possible to view the finger table of a given node. Note that the finger table contains a list of keys and their successor’s IPs, and is organized such that each node holds the IP of an exponential sequence of nodes that follow it, (i.e. entry i of node k ’ finger table holds the IP of node $k + 2^i$ with i being the key size). In Figure IV.11, we can see the finger table of node with ID=0 that contains five rows because $i=5$.

Figure IV.12 shows the network, we effectively used to make SIP calls. The network contains two real Chord nodes.

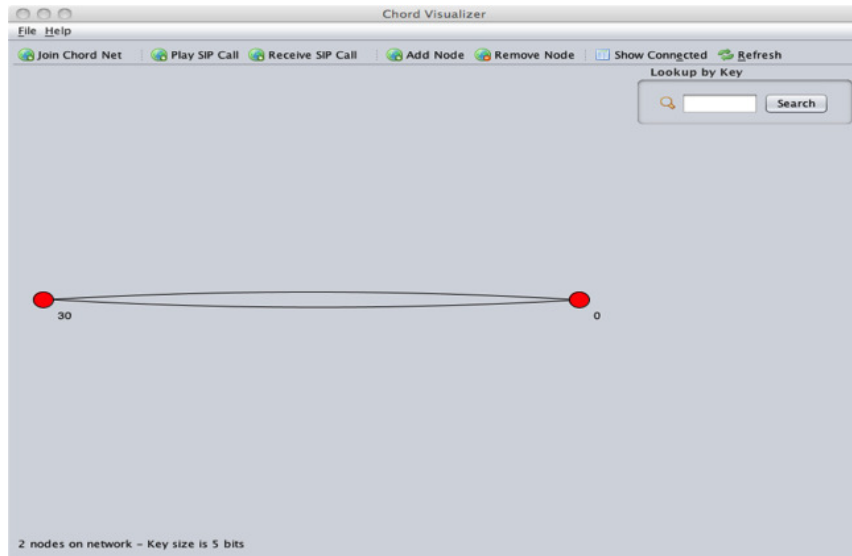


Figure IV.12: Real Chord network

The interface shown in figure IV.13 appears when the call is initiated by pushing “Play SIP Call” button. It shows the list of the connected peers and allows choosing the called peer. In our scenario node with ID 30 wants to call node with ID 0.

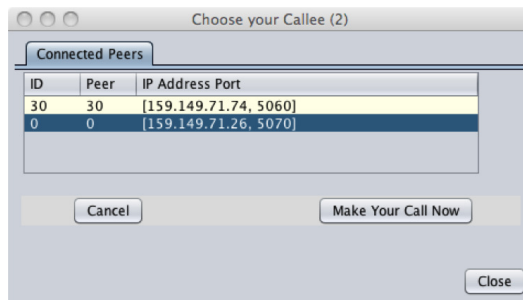
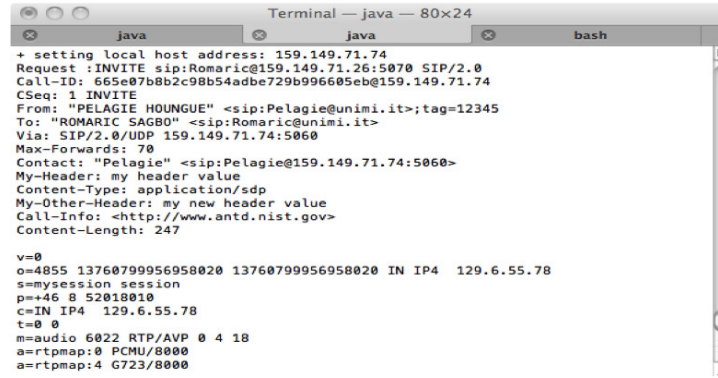


Figure IV.13: Real Chord network: call interface

Figure IV.14 depicts the SIP INVITE created by the caller and sent to the callee.

Figure IV.15 shows the SIP INVITE received by the callee.

When the caller received the OK, it creates a SIP ACK that it sends to the callee. Figure IV.16 shows the SIP OK message sent by the callee and the ACK message that it receives from the caller. The SIP final dialog state is also shown to prove the successful establishment of the call session.



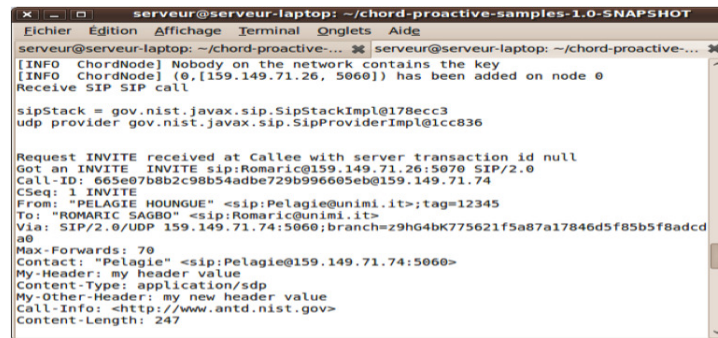
```

Terminal — java — 80x24
+ setting local host address: 159.149.71.74
Request : INVITE sip:Romaric@159.149.71.26:5070 SIP/2.0
Call-ID: 665e07b8b2c98b54adbe729b996605eb@159.149.71.74
CSeq: 1 INVITE
From: "PELAGIE HOUNGUE" <sip:Pelagie@unimi.it>;tag=12345
To: "ROMARIC SAGBO" <sip:Romaric@unimi.it>
Via: SIP/2.0/UDP 159.149.71.74:5060
Max-Forwards: 70
Contact: "Pelagie" <sip:Pelagie@159.149.71.74:5060>
My-Header: my header value
Content-Type: application/sdp
My-Other-Header: my new header value
Call-Info: <http://www.antd.nist.gov>
Content-Length: 247

v=0
o=4855 13760799956958020 13760799956958020 IN IP4 129.6.55.78
s=mysession session
p=+46 8 52018010
c=IN IP4 129.6.55.78
t=0
m=audio 6022 RTP/AVP 0 4 18
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000

```

Figure IV.14: INVITE message sent by the caller



```

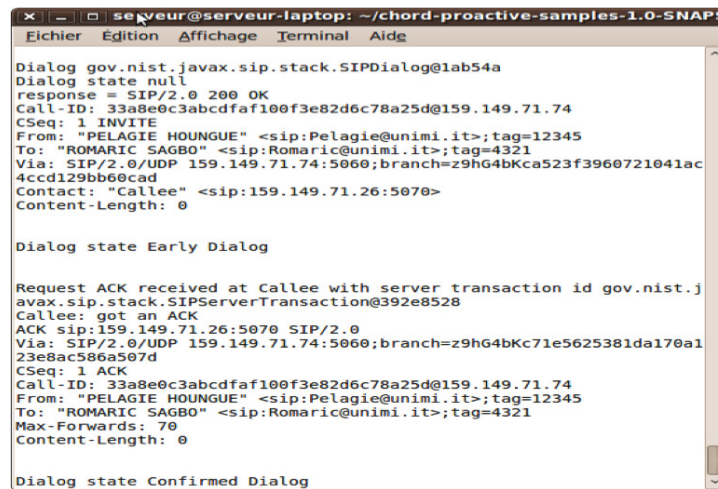
serveur@serveur-laptop: ~/chord-proactive-samples-1.0-SNAPSHOT
Echier Edition Affichage Terminal Onglets Aide
serveur@serveur-laptop: ~/chord-proactive-samples-1.0-SNAPSHOT
[INFO ChordNode] Nobody on the network contains the key
[INFO ChordNode] (0,[159.149.71.26, 5060]) has been added on node 0
Receive SIP SIP call

sipStack = gov.nist.javasip.SipStackImpl@178ecc3
udp provider gov.nist.javasip.SipProviderImpl@1cc836

Request INVITE received at Callee with server transaction id null
Got an INVITE INVITE sip:Romaric@159.149.71.26:5070 SIP/2.0
Call-ID: 665e07b8b2c98b54adbe729b996605eb@159.149.71.74
CSeq: 1 INVITE
From: "PELAGIE HOUNGUE" <sip:Pelagie@unimi.it>;tag=12345
To: "ROMARIC SAGBO" <sip:Romaric@unimi.it>
Via: SIP/2.0/UDP 159.149.71.74:5060;branch=z9hG4bK775621f5a87a17846d5f85b5f8adcd
a0
Max-Forwards: 70
Contact: "Pelagie" <sip:Pelagie@159.149.71.74:5060>
My-Header: my header value
Content-Type: application/sdp
My-Other-Header: my new header value
Call-Info: <http://www.antd.nist.gov>
Content-Length: 247

```

Figure IV.15: INVITE message received by the callee



```

serveur@serveur-laptop: ~/chord-proactive-samples-1.0-SNAPSHOT
Echier Edition Affichage Terminal Aide
Dialog gov.nist.javasip.stack.SIPDialog@1ab54a
Dialog state null
response = SIP/2.0 200 OK
Call-ID: 33a8e0c3abdcfaf100f3e82d6c78a25d@159.149.71.74
CSeq: 1 INVITE
From: "PELAGIE HOUNGUE" <sip:Pelagie@unimi.it>;tag=12345
To: "ROMARIC SAGBO" <sip:Romaric@unimi.it>;tag=4321
Via: SIP/2.0/UDP 159.149.71.74:5060;branch=z9hG4bKca523f3960721041ac
4ccd129bb60cad
Contact: "Callee" <sip:159.149.71.26:5070>
Content-Length: 0

Dialog state Early Dialog

Request ACK received at Callee with server transaction id gov.nist.j
avasip.stack.SIPServerTransaction@392e8528
Callee: got an ACK
ACK sip:159.149.71.26:5070 SIP/2.0
Via: SIP/2.0/UDP 159.149.71.74:5060;branch=z9hG4bKc71e5625381da170a1
23e8ac586a507d
CSeq: 1 ACK
Call-ID: 33a8e0c3abdcfaf100f3e82d6c78a25d@159.149.71.74
From: "PELAGIE HOUNGUE" <sip:Pelagie@unimi.it>;tag=12345
To: "ROMARIC SAGBO" <sip:Romaric@unimi.it>;tag=4321
Max-Forwards: 70
Content-Length: 0

Dialog state Confirmed Dialog

```

Figure IV.16: Final dialog state on the callee side

c) Conclusion

By implementing our prototype on two different middleware, it becomes evident the strength of our architecture. We implemented first the three modules (“P2P SIP”, “SIP to Overlay” and the “Overlay Peer Management”) for the JXTA-based prototype. In order to get completed the Chord-based prototype, it was not necessary to modify the P2P SIP module. Only the “SIP to Overlay” and the “Overlay Peer Management” have been suitably modified to work with the new middleware. This has been possible because, no overlay mechanism and message is implied during the SIP call establishment procedure. In addition, the JAIN SIP API has been used without

modifications to implement the SIP functionalities. Hence, the proof of our main requirements that are the middleware-independence and the SIP compliance features.

IV.7 Evaluation

We used OverSim [15], an open source P2P network simulation framework, to evaluate the performance of our architecture. OverSim was built by the Institute of Telematics, Universität Karlsruhe to work on top of the OMNeT++/OMNEST simulation environment. It includes several structured and unstructured P2P protocols such as Chord, Kademlia and Pastry. OverSim also provides several common functions, including a generic lookup mechanism and a remote procedure call interface to facilitate the implementation of additional protocols. Another important feature of OverSim is that it provides several models for generating churn (including a lifetime-based churn model supporting different distribution functions) and underlying network topology.

The goal of our simulations was to evaluate the performance of our architecture on top of different P2P overlay middleware. We measured the round trip time experienced by SIP messages and the traffic overhead introduced by the architecture for different network configuration scenarios. We compared the obtained results to those obtained for a flat P2P SIP network (i.e., a conventional DHT-based P2P SIP network without proxies). The goal of the comparison was to evaluate the impact of the introduction of proxies into the P2P SIP topology on the performance of the P2P SIP solution. We have also evaluated the effect of caching on the overall architecture.

IV.7.1 Simulation Scenario

We have designed a realistic network scenario to run our simulations. The network is made of a set of clients that run the P2P SIP application, a set of access routers and a set of backbone routers. The clients are connected to access routers using DSL (Digital Subscriber Line) connections (delay = 20ms and bandwidth = 1Mbps) and the peers are connected to access routers using Ethernet (delay = 1ms and bandwidth = 100Mbps;). Access routers are connected to backbone routers using fiberlines (delay = 1ms and bandwidth = 10Gbps) and the backbone routers are connected to each other using fiberlines (delay = 1ms; bandwidth = 10Gbps).

The lifetime of peers is randomly drawn from a probability function (Weibull distribution). When this time is reached, the node is removed and a new peer is created after a dead time drawn from the same probability function.

When a client joins the overlay, it constructs its contact list by randomly selecting 10 contacts out of the list of available contacts on the overlay. This assumption is a simple way to model the fact that users generally make their calls to almost the same list of contacts. During the simulation, each client randomly starts a SIP session with one of its contacts with a mean of one new session every five minutes. Each session lasts three minutes. Processing time for the peers is neglected and not considered in the RTT (Round Trip Time) delay, and each simulation scenario runs for at least three hours. The caching lifetime is set to one hour.

IV.7.2 Results and discussions

a) Round trip time

Figure IV.17 shows the SIP average end-to-end round trip delay (from request generation at the client to the reception of the final response) as a function of the number of nodes in the overlay

for both proxy-based architecture (our architecture) and DHT-based architecture. The simulations were run on top two P2P middleware: Kademlia and Chord. The results were collected for different numbers of overlay nodes.

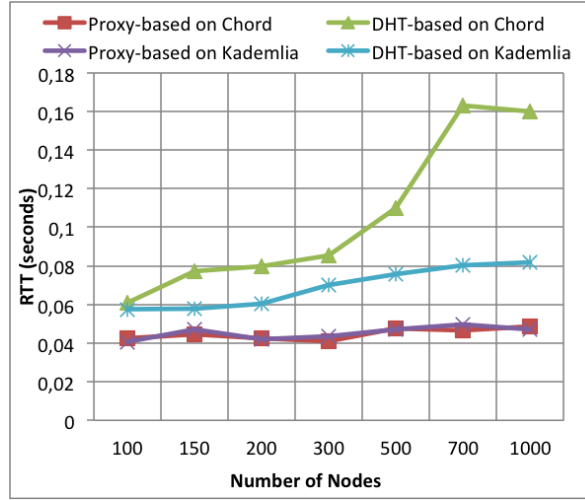


Figure IV.17: Proxy based vs. DHT based P2P SIP RTT

From figure IV.17, we can easily see that the average round trip delay for our architecture is always smaller than the average round trip for the DHT-based architecture. This means that the introduction of proxies into the P2P SIP architecture helps to reduce the delay involved in establishing a SIP session. This is a very important result, as one of the main determining factors for user experience is the time the user spends waiting to hear the ringing back signal.

A second result, we can draw from figure IV.17 is that the round trip time in our architecture shows almost no increase with the number of nodes, which shows that our architecture scales very well with the number of peers.

b) Effect of caching

The caching mechanism used by the proxy is important to reach our performance requirement. A given proxy' cache contains the SIP URI of a given node and its corresponding IP address when the node is in the same group as the proxy or a given node' SIP URI and the IP address of its corresponding proxy when the node is not in the same group as a given proxy. The cache is updated in two cases. The first case refers to when during a call set up a new destination is required. Then, the cache does not contain any information about this destination node. The information is added to the cache once it is received from the proxy of the destination node. The second case refers to when the destination node is in the proxy's group but the information about it, is already expired since SIP clients have to periodically refresh their registration. In this case, the cache is updated from the GR of the group.

In the second set of our experiments, we evaluated the effect of proxy caching on performance. We studied the effect of caching through two metrics: the round trip time and the network load (number of SIP applications and P2P overlay maintenance packets transiting through the network). Figure IV.18 shows the round trip measurements.

From figure IV.18, we can see that caching significantly reduces the round trip delay of SIP calls. This result was expected, as in many cases caching spares the proxy from fetching peers' contacts from the GR.

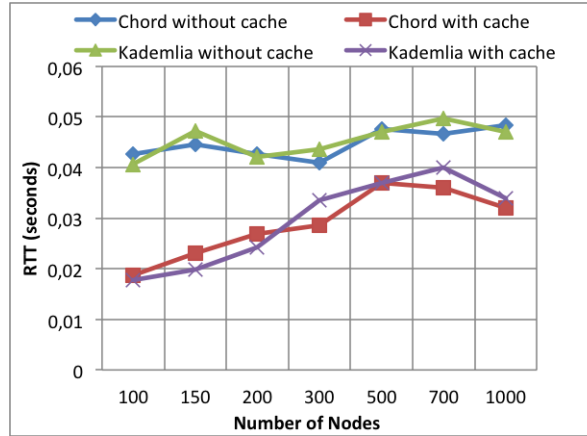


Figure IV.18: Effect of proxy caching on RTT

Another important result, made visible in figure IV.19, is that proxy caching also reduces network load. We can see that caching allows a network load gain, which tends to revolve around 30% when the number of peers increases.

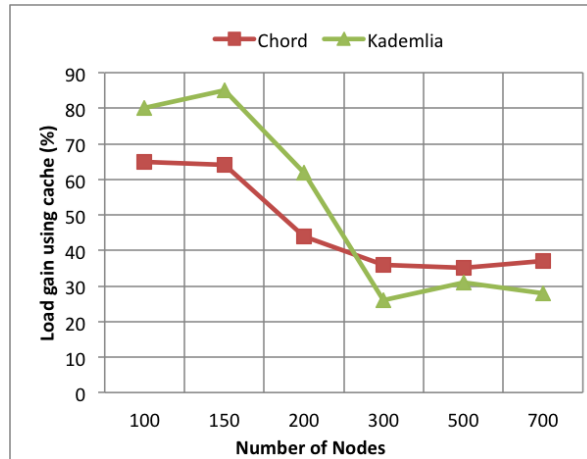


Figure IV.19: Effect of caching on network load

Based on the results presented here, we can conclude that the proxy-based P2P SIP approach, along with proxy caching, improves the round trip delay and network usage efficiency of P2P SIP architectures.

IV.8 Conclusion

In this chapter, we have proposed a novel architecture for P2P SIP. This novel architecture focuses on the case of a single domain, and assumes that the domain may have many proxies, each capable of managing a limited number of client peers. The peers are organized in several groups, where each group is composed of one proxy, one group registrar and a given number of clients. The domain has a bootstrapping node (i.e. the Domain Controller) that is responsible for creating and managing the groups.

The architecture defines procedures for self-organization, ensures independence from P2P middleware, requires no extension to SIP messages, and provides a high performance level by implementing a caching mechanism on the proxies. The proxies integrate a caching mechanism to store

previously obtained contact information, which avoids querying before each call.

Particular software architecture was also designed and a proof-of-concept prototype built using JXTA on one side and Chord-ProActive on other side, has been released. SIP functionalities have been implemented using JAIN SIP. The prototype implemented the Client side modules. The use of SIP messages for all of the procedures ensures middleware-independence. The same is applied to the software architecture that cleanly isolates the SIP modules from the middleware through mapping modules. We did not extend any SIP message.

The simulations show that the proxy-based P2P SIP approach along with proxy caching, improves round trip delay and network usage efficiency of P2P SIP architectures.

Part III

Routing issues in P2P SIP overlay

In the previous chapter, we have proposed an architecture to integrate Peer-to-Peer (P2P) principles into the Session Initiation Protocol (SIP) so that to take advantage of all benefits inherent to the two paradigms. In our P2P SIP architecture, the SIP location service is integrated in proxies that are peer entities in the P2P overlay. Therefore, having SIP proxies as first-class entities, raises two important issues: proxy topology building and proxy-level routing. Thanks to proxy topology building, a proxy joining the P2P overlay knows where it should insert itself in the network of proxies. Proxy-level routing enables messages to be routed in the SIP network built by proxy topology building.

This part of the thesis is composed of chapters V and VI. Chapter V discusses a novel framework for proxy topology building and proxy-level routing. The framework is general enough to be used on any P2P overlay that meets a minimal set of requirements. Our framework relies on an algorithm that builds the network of proxies in a ring and uses its own routing algorithms to route calls. We describe our novel routing algorithms used between proxies and evaluate their performance through simulation. Although those algorithms work efficiently to make communication effective between two endpoints, it is necessary to cope with NAT traversal issues inherent to SIP communications when needed. Chapter VI provides a suitable technique for NAT traversal in our P2P SIP framework for both SIP signaling messages and media streams.

Chapter V

Topology building and routing in MISE-P2PSIP

This chapter gives first the motivation of having a special topology for proxy servers in MISE-P2PSIP and appropriate routing algorithms. Subsequently, the related work as well as the requirements are discussed. Moreover, the framework is presented in some details. Finally, we compare our framework on one side with OSPF (Open Shortest Path First) routing algorithm and on other side with a DHT-based SIP message routing.

V.1 Motivation

In the architecture proposed in the previous chapter, the proxies are fully meshed. While fully meshing proxies increases call performance and makes routing straightforward, scalability is a major issue, making the architecture unfit for large-scale deployment. Topology maintenance cost may become prohibitive. This indicates the need of building a topology that scales and routing algorithms that remain simple. Our choice is based on ring topology because it is very simple and easy to implement. Therefore, the framework proposes two main functionalities: proxy topology building and proxy-level routing. Proxy topology building is the process that indicates where a new proxy joining the overlay will be inserted in the network of proxies. Proxy-level routing is the process that routes messages in the network resulting from proxy topology building. Thus, in this chapter, we improve our architecture by proposing a topology that scales and simple yet effective routing algorithms.

V.2 Requirements

The first requirement is that the framework should meet the key requirements of our overall architecture. The new architecture preserves our initial proposal's properties (independence from P2P infrastructure, SIP extensions kept to a minimum and self-organization). Also, the new framework supports four additional requirements.

- Firstly, our framework allows recovery from failure, automatically restoring broken routes between the different proxies.
- Secondly, it provides simple algorithms for topology building and routing. In terms of topology, the proxies' topology should be as simple as possible to ensure it can be created and

re-organized easily. Regarding routing, simplicity means that the maintenance procedure for the routing tables should be simple, not take much time to converge, and not generate too much traffic in the network.

- Thirdly, each proxy in a topology should be able to reach any other proxy in the same topology, meaning that the graph formed by the proxies should be connected.
- A fourth requirement satisfied by our proposal is that the routing protocol provides the basic properties of routing, namely efficiency, optimality, simplicity, robustness, stability, and scalability in the number of proxies. Routing efficiency means that the routing algorithm should always find a path to reach one node (if a path exists). Optimality refers to the algorithm’s ability to always provide the shortest path. The routing table should also be as smaller as possible. Robustness means that the routing algorithm should be able to cope with changes in the topology, including node failure. A stable algorithm reaches equilibrium and stays there.

V.3 Related work and analysis

We split the related work into two categories: P2P SIP architectures that include both topology building and routing, and general infrastructure-independent routing protocols. We do not consider the first category of the existing P2P SIP architectures that extend SIP [5], because we consider compatibility with current SIP standards as key requirement. We will rather focus on the architectures that do not extend SIP.

Several P2P SIP frameworks proposed to date like P2PNS [13], SIP usage for RELOAD [12] belong to the first category and are similar to our framework.

RELOAD’s routing [12] relies on two basic mechanisms: Symmetric recursive routing and Iterative routing. Symmetric recursive routing requires that a message follows the path through the overlay to the destination without returning to the originating node. This means that each peer forwards the message closer to its destination. RELOAD messages contains both “Via List” attributes (built hop-by-hop as the message is routed through the overlay) and “Destination List” attributes (providing source-routing capabilities for requests and return-path routing for responses). The return path of the response is then the same path followed in reverse. RELOAD also supports a basic Iterative routing mode where the intermediate peers merely return a response indicating the next hop, but do not actually forward the message to that next hop themselves. Iterative routing is implemented using the “Route.Query” method that allows a node to query a peer for the next hop it will use to route a message.

P2PNS [13] uses the Key-Based Routing layer (KBR) that provides a “route” method to efficiently route a message to an arbitrary key by successively forwarding the message to overlay neighbors which have a nodeID closer to the destination key. Indeed, a common service which is provided by all structured P2P networks is the KBR layer. This layer provides efficient routing to identifiers called keys from a large identifier space. Every participating node in the overlay chooses a unique nodeID from the same id space and maintains a routing table with nodeIDs and IP addresses of neighbors in the overlay topology. Every node is responsible for a particular range in the identifier space, usually for all keys close to its nodeID in the id space. P2PNS proposes to use the Kademlia protocol as KBR layer, then uses routing protocol released by Kademlia.

Both RELOAD and P2PNS proposals explicitly do not fill our specific requirement on P2P infrastructure-independence, by using the P2P routing mechanisms to send messages.

The second category includes infrastructure-independent routing protocols that are classic Internet routing protocols such as Open Shortest Path First (OSPF) [69] and Distance Vector Routing (DVR) [70].

OSPF is a link state protocol. The Shortest Path First (SPF) routing algorithm is the basis for OSPF [69] operations. When an SPF router is powered up, it initializes its routing-protocol data structures and then waits for indications from lower-layer protocols that its interfaces are functional. After a router is assured that its interfaces are functioning, it uses the OSPF Hello protocol to acquire neighbors, which are routers with interfaces to a common network. The router sends hello packets to its neighbors and receives their hello packets. In addition to helping acquire neighbors, hello packets also act as keep-alive packets to let routers know that other routers are still functional. On multi-access networks (networks supporting more than two routers), the Hello protocol elects a designated router and a backup designated router which among other things, is responsible for generating LSAs (Link-State Advertisement) for the entire multi-access network. Designated routers allow a reduction in network traffic and in the size of the topological database. When the link-state databases of two neighboring routers are synchronized, the routers are said to be adjacent. On multi-access networks, the designated router determines which routers should become adjacent. Topological databases are synchronized between pairs of adjacent routers. Adjacencies control the distribution of routing-protocol packets, which are sent and received only on adjacencies. Each router periodically sends an LSA to provide information on router's adjacencies or to inform others when a router's state changes. By comparing established adjacencies to link states, failed routers can be detected quickly and the network's topology altered appropriately. From the topological database generated from LSAs, each router calculates a shortest-path tree, with itself as root. The shortest-path tree, in turn, yields a routing table.

Many routing protocols fall in the category of DVR protocols. There are for instance RIP (Routing Information Protocol) [120] and IGRP (Internet Gateway Routing Protocol) [121]. The methods used to calculate the best path for a network are different for each routing protocol but the fundamental features of Distance Vector (DV) algorithms are the same across all DV-based protocols. Distance Vector means that routers are advertised as vector of distance and direction. Direction is simply the next hop address and exit interface and distance means hop count. Routers using distance vector protocol do not have knowledge of the entire path to a destination. The DV protocol is based on calculating the direction and distance to any link in a network. The cost of reaching a destination is calculated using various route metrics. For instance, RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth. Updates are performed periodically where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance vector routing protocol. Once a router has this information, it is able to amend its own routing table to reflect the changes and informs its neighbors of the changes. This process is usually called "routing by rumor" because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true.

OSPF and DVR protocols meet most of the routing requirements (e.g. efficiency and optimality). However, in the specific case of the topology we propose, they generate more network traffic.

V.4 Proposed framework

We assume that the P2P SIP nodes are in a single SIP domain. The domain may have many proxies, and each proxy can manage a limited number of client peers. However, our middleware-independent approach can support handling multiple domains even when they rely on heterogeneous P2P overlays.

In this section, we first introduce the architectural principles on which our framework is based, followed by the topology creation. The routing is then discussed and an example of application is presented in the last sub-section.

V.4.1 Architectural principles

The entities are the same as in the previous architecture (figure IV.1). So, the peer nodes are organized in several groups. Each group is composed of one proxy, one GR (Group Registrar) and 0 to N clients. The proxies are organized into a ring; we chose a ring topology because it is simple to implement and easy to maintain. Each domain has a single DC (Domain Controller), which is the overlay bootstrap node. Each node joining the overlay SIP domain should first contact the DC. The DC is responsible for creating and managing the groups, and for creating and maintaining the proxies' ring. The DC keeps the list of the groups and for each group, it maintains the GR address and the number of clients. To allow recovery from failure, we select the alternative which consists of putting the recovery functionality in the topology building instead of in the routing procedure. The ring architecture is depicted in figure V.1. Unlike many existing architectures with proxies, the algorithms used by our architecture for topology creation and for routing are P2P infrastructure-independent, thereby ensuring portability.

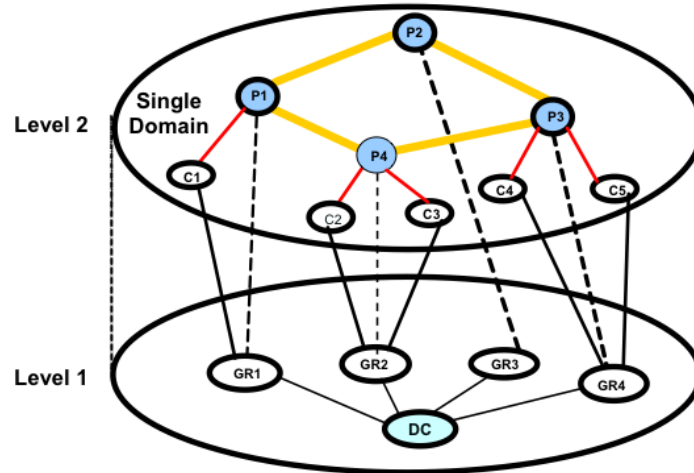


Figure V.1: New topology building proxies in ring

V.4.2 Topology creation and maintenance

The proxies' ring is organized following a clockwise direction, where the new node is inserted between the first and the last node in the ring. The first and the last nodes will be the successor to and the predecessor of the new node, respectively. We choose this alternative (clockwise direction) because it makes new node insertion simple and minimizes the information maintained by the DC.

The DC maintains a topology table that includes the first and the last node in the ring. When a new node is inserted, only the table entry corresponding to the last node is updated. Each proxy has also a topology table which contains its successor and predecessor in the ring. This section describes how a new proxy is added to or removed from the ring. The joining and leaving procedures for the other nodes (Client, GR and DC) are the same as those described in the previous chapter. It is the reason why they are not presented here.

a) **A proxy joins the network**

When a proxy (Pi) joins the network, it sends a *SIP REGISTER* message to the DC. The DC sends to Pi the address of the GR, the addresses of Pi's predecessor and successor, and the total number of the proxies in the ring (N), including the new incoming (Pi). This information is sent in the *200 OK* SIP message. The DC also forwards the Pi's registration to the GR.

Then, Pi sends a *SUBSCRIBE* message to the GR to subscribe to events notification, and the GR does the same. Pi subscribes to its neighbors as well, to notify its place in the ring (i.e. as predecessor or successor) and to receive future notifications from them. Subsequently, each of the two neighbors also subscribes to the Pi. Pi sets a TTL ($TTL=(N/2)-1$) variable in the message. TTL is used to control the number of times the information about the new proxy is forwarded inside the ring.

When a neighbor (Ni) receives the *SUBSCRIBE*, it sends a *SUBSCRIBE* to Pi and updates its routing table using the procedure "Routing table update" described in the next section. If $TTL > 0$, Ni notifies (via a *SIP NOTIFY*) its other neighbor (i.e. other than the one from which it receives the information) about Pi. It sends the following information: $new_node=Pi$; $distance=D(Ni \rightarrow Pi)$; $TTL=TTL-1$. The *SUBSCRIBE* messages sent by the first neighbors contains $D(Ni \rightarrow Pi)=1$.

At this step, the receiving proxy (Pj) updates its routing table by using the procedure "Routing table update". If $TTL-1 > 0$, the proxy forwards the information in a *NOTIFY* message to the next hop with: $new_node=Pi$; $distance=D(Pj \rightarrow Pi)$; $TTL=TTL-1$. If a proxy receives the information twice (from its successor and predecessor), it chooses the first path it receives.

b) **A proxy leaves the network**

The leaving proxy (Ni) sends a *NOTIFY* message to the GR of its group, along with its predecessor (Pr-i) and successor (Su-i), and then unregisters from the GR.

If there is at least one client in the group, the GR chooses one client in the same group and sets it as the new proxy (the clients *SUBSCRIBE* to the GR when they enter the network). The newly-elected proxy will have the same proxy-ID as the departing one. It is informed about its new role via a *NOTIFY* message and it is given its successor (i.e. Su-i) and predecessor (i.e. Pr-i). It then sends a *SUBSCRIBE* to its neighbors as in the joining procedure. Next, the neighbors update the address of the proxy in their routing tables. The GR also notifies all the clients in the group about the new proxy, and informs (using a *SIP NOTIFY*) the DC about the new elected proxy and the departure of the old one. The DC had subscribed to the GR when the GR joined the network. If the departing proxy is the first or the last entering in the ring, the DC updates its topology table.

If there is no client in the group, the GR notifies Pr-i that Ni has left and that its new successor is Su-i. Pr-i subscribes to Su-i and informs it to be its new predecessor. Then, the remaining proxies in the ring will use the algorithm described in figure V.4 to update their respective routing table. In this way, the proxy ring is always kept connected. The GR also notifies the DC that Ni has

left, and gives Pr-i and Su-i. DC updates its topology table if Ni was the last or the first proxy in the ring.

V.4.3 Routing algorithms

Each proxy maintains a routing table that includes an entry for every other proxy in the ring. Each entry has three columns: destination (i.e. the proxy to reach), send (i.e. right or left), and distance (i.e. the distance needed to reach the destination).

The distance is calculated as the number of hops towards the destination. The “send” column is set to “left” if the shortest path to reach the destination is in the clockwise direction. Otherwise, the column is set to right. This section describes how the routing tables of the different proxies are created and updated.

Basically three (3) routing algorithms have been proposed:

1. The procedure for an existing proxy to update its routing table when a new proxy joins the ring
2. The procedure for the joining proxy to create its routing table
3. The procedure for an existing proxy to update its routing table when a proxy leaves the ring

The following legend is applied to all algorithms described in this section:

- $D(p_i \rightarrow p_j)$: The distance between P_i and P_j ,
- $\text{Right}(P_j)$ and $\text{Left}(P_j)$: The list of the proxies that P_j reaches from its right or its left, respectively,
- $\text{Candidates}(P_j)$: equal to $\text{Right}(P_j)$ or $\text{Left}(P_j)$ depending whether P_j receives the message (i.e. a proxy is joining or a proxy is leaving) from its right or from its left.

Following is the description of each algorithm.

a) Creation/update of the routing tables when a proxy joins

i. Routing table update

When a proxy (P_i) joins the ring and sends a *SUBSCRIBE* message to its neighbors, it includes a TTL variable in the message. The TTL is equal to the integer value of: $(N/2)-1$, where N is the number of proxies in the ring. It is used to control the number of times the information about the new proxy is forwarded inside the ring. When a neighbor (N_i) receives a *SUBSCRIBE*, it updates its routing table using the procedure described in figure V.2. Among others, it adds a new entry to its local routing table as follows: destination= P_i ; distance=1; send=left (or right, depending on if P_i is the predecessor or the successor). If $\text{TTL} > 0$, N_i notifies (via a SIP *NOTIFY*) its other neighbor about P_i . It sends the following information: new_node= P_i ; distance= $D(N_i \rightarrow P_i)$; $\text{TTL} = \text{TTL} - 1$; $D(N_i \rightarrow P_i)$ is the distance between N_i and P_i .

Algorithm 1 The procedure for an existing proxy to update its routing table when a new proxy joins the ring

```

1: Begin
2: A proxy  $P_j$  receives information that a new proxy  $P_i$  is added to the ring (via SUBSCRIBE or NOTIFY),
   with the attributes:  $\text{new\_node}=P_i$ ;  $\text{distance}=d_i$ ;  $\text{TTL}=\text{TTL}_i$ ;
3:  $P_j$  updates  $N$  value:  $N=N+1$ 
4: If  $D(p_j \rightarrow p_i) \leq N/2$  then ::: there may be some proxies that  $P_j$  should now reach via  $P_i$ 
5:   For each proxy  $p_k$  in  $\text{Candidates}(P_j)$ 
6:     If  $(D(p_j \rightarrow p_k) \geq D(p_i \rightarrow p_j))$  then ::: If  $P_k$  is now reachable via the new proxy  $P_i$ 
7:       If  $(D(p_j \rightarrow p_k) + 1 \leq N/2)$  then ::: If the new distance to  $P_k$  is still less than the maximum distance
           that we can get inside the ring (i.e.  $\text{int}(N/2)$ )
8:         Change  $D(p_j \rightarrow p_k)$ :  $D(p_j \rightarrow p_k) += 1$  ::: update the distance, but keep the same path
9:       Else
10:        Switch the “send” column corresponding to  $p_k$  in the routing table ::: Keep the same distance
           but specify that  $P_k$  is now reachable via the opposite path (not via  $P_j$ )
11: End

```

Figure V.2: Routing table creation/update for other proxies when a new proxy joins

ii. Routing table creation for a new proxy

The receiving proxy (P_j) executes the same procedure as N_i (i.e. it updates its routing table and notifies its other neighbor if appropriate) and the procedure is repeated until $\text{TTL}=0$. If a proxy receives the notification twice (from its successor and from its predecessor), it chooses the first path it receives. This will happen if N is even, and the two paths will have the same length. The new proxy creates its own routing table using the procedure described in figure V.3.

Algorithm 2 The procedure for the joining proxy to create its routing table

```

1: Begin
2: When a proxy  $P_i$  is added to the ring, it gets its successor ( $S_u$ ) and predecessor ( $P_r$ ).
3:  $P_i$  adds  $S_u$  to its routing table, with “send”=left &  $\text{distance}=1$ .
4: When  $P_i$  sends a SUBSCRIBE message to its successor, it receives a NOTIFY message including the routing
   table of the successor ( $T_{s_u}$ ).
5:  $P_i$  updates its own routing table from its successor’s routing table.
6: For each  $P_k \in \text{Left}(S_u)$ 
7:   If  $(D(S_u \rightarrow p_k) + 1) \leq N/2$  then
8:      $P_i$  adds  $P_k$  to its routing table with the same “send” attribute as in  $T_{s_u}$ ,
       and with  $\text{distance} = D(S_u \rightarrow p_k) + 1$ 
9:   Else
10:     $P_i$  adds  $P_k$  to its routing table with a switched value of “send” attribute in  $T_{s_u}$ ,
      and with the same distance as in  $T_{s_u}$ .
11: For each  $P_k \in \text{Right}(S_u)$ 
12:    $P_i$  adds  $P_k$  to its routing table with the same “send” attribute and distance as in  $T_{s_u}$ .
13: End

```

Figure V.3: Creation of routing table of joining proxy

b) Update of the routing tables when a proxy leaves

When a proxy P_i leaves the ring and there is no client to replace it, its predecessor P_r is notified. P_r notifies its neighbors that P_i has left and the information is propagated inside the ring using the same procedure as for a node joining, by using $\text{TTL}=(N/2)-1$. When a proxy receives the

leaving notification, it removes the entry corresponding to P_i from its routing table and recalculates its distances to the other proxies using the procedure in figure V.4.

Algorithm 3 The procedure for an existing proxy to update its routing table when a proxy leaves the ring

```

1: Begin
2: A proxy  $P_j$  receives the news that a proxy  $P_i$  has left the ring with attributes:
   leaving_node= $P_i$ ; distance= $d_i$ ; TTL= $TTL_i$ ;
3: If  $D(p_j \rightarrow p_i) < N/2$  then :::: there may be some proxies that  $P_j$  is reaching via  $P_i$ 
4:   For each proxy  $p_k$  in Candidates( $P_j$ )
5:     If  $(D(p_j \rightarrow p_k) > D(p_j \rightarrow p_i))$  then :::: If  $P_k$  is currently reachable via the leaving proxy  $P_i$ 
6:       Change  $D(p_j \rightarrow p_k)$ :  $D(p_j \rightarrow p_k) -= 1$  :::: update the distance
7:   For each proxy  $p_m$  in Candidates =  $\{p, p \text{ is a proxy and } p \notin \text{Candidates}\}$ 
8:     If  $(D(p_j \rightarrow p_m) = N/2)$  then
9:       Switch the “send” attribute corresponding to  $P_m$  and switch direction
10:  $P_j$  updates  $N$  value:  $N = N - 1$ 
11: End

```

Figure V.4: Update of remaining proxies routing tables upon leaving

V.4.4 Example of routing table update

This section gives an example of the execution of the routing table update algorithm described above in figure V.2, upon node joining.

Figure V.5 shows the initial state of the ring in our scenario. We will follow step by step, the change of the routing table of proxy P_2 while nodes P_5 , P_6 and P_7 join the ring.

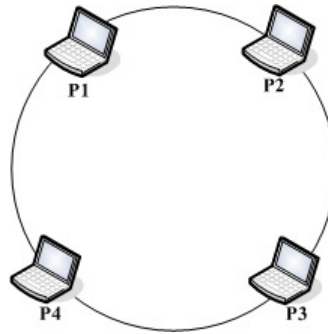


Figure V.5: Initial state of the proxy ring in the scenario

Table V.1 shows the content of the routing table of P_2 in the first three columns. The path column is added to make easy the understanding of the example and indicates the shortest path used to reach a proxy. So P_2 routing table contains the “Destination”, the “Send” and the “Distance” columns.

Destination	Send	Distance	Path
P1	Right	1	P2P1
P2	-	0	P2
P3	Left	1	P2P3
P4	Left	2	P2P3P4

Table V.1: P_2 initial routing table

✓ Step 1 : P5 joins the ring

Figure V.6 illustrates the ring with five participating proxies

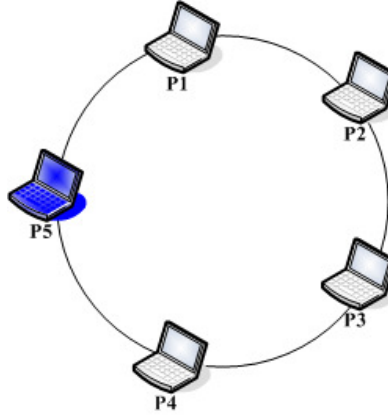


Figure V.6: Proxy ring after P5 joining

What happens is:

- P2 receives a notification that P5 is added to the ring from its right (P1) and updates N the number of nodes. $N=5$ (after new proxy joining)
- The distance between P2 and P5, $D(p2 \rightarrow p5)=2 < N/2$. $N/2$ is the maximum shortest distance that we can get inside the ring. When N is not even the maximum distance is equal to the integer part of $N/2$. In this case the maximum shortest distance is equal to 2.
- $\text{Right}(p2)=\{P1\}$; the list of the nodes that P2 reaches from its right according the P2' routing table. Then, the distance between P2 and each of these nodes is calculated and compared to the distance between P2 and the new joining node (P5).
- $D(p2 \rightarrow p1)=1 < 2$ ($D(p2 \rightarrow p5)$). This infers that no change on P1' entry in the routing table is needed.
- P2 adds to its routing table, a new entry about P5. The "Distance" is equal to "2" and the "Send" field is set to "Right" because P2 has received the notification from its right.

P2' updated routing table, looks like the first three columns of table V.2.

Destination	Send	Distance	Path
P1	Right	1	P2P1
P2	-	0	P2
P3	Left	1	P2P3
P4	Left	2	P2P3P4
P5	Right	2	P2P1P5

Table V.2: P2 routing table after P5 joining

✓ Step 2 : P6 joins the ring

Figure V.7 illustrates the ring with six participating proxies.

What happens is:

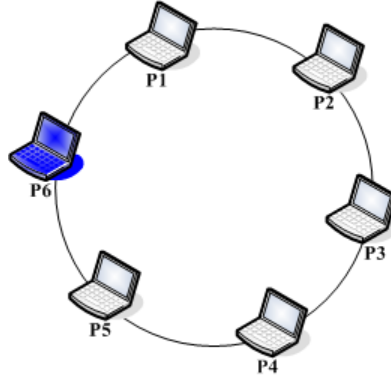


Figure V.7: Proxy ring after P6 joining

- P2 receives the message that P6 is added to the ring from its right and updates N . $N=6$, $N/2=3$.
- The distance between P2 and P6, $D(p2 \rightarrow p6)=2 < (N/2)$;
- $\text{Right}(p2)=\{P1, P5\}$ is the list of the nodes that P2 reaches from its right according to its routing table. The distance between P2 and each of these nodes is calculated and compared to the distance between P2 and the new joining node (P6).
- $D(p2 \rightarrow p1)=1 < 2$. This means that there is no change to P1 entry in the routing table.
- $D(p2 \rightarrow p5)=2 \geq 2$. This means that the distance may be changed. The new distance could be $D(p2 \rightarrow p5) + 1 = 3$. In order to know if the distance should be changed or not, it is useful to make a further comparison of this distance with the maximum shortest distance in the ring. Then, $D(p2 \rightarrow p5)=3 \leq N/2$. This infers that the distance effectively changes and there is no need to change the “send” column in the routing table. To summarize, the unique change to P5 entry in P2 routing table is the distance that is set to 3.
- P2 adds to its routing table a new entry about P6. The “Distance” is equal to “2” and the “Send” field is set to “Right” because P2 has received the notification from its right.

At this step, P2' routing table looks like the first three columns of table V.3.

Destination	Send	Distance	Path
P1	Right	1	P2P1
P2	-	0	P2
P3	Left	1	P2P3
P4	Left	2	P2P3P4
P5	Right	3	P2P1P6P5
P6	Right	2	P2P1P6

Table V.3: P2 routing table after P6 joining

✓ Step 3 : P7 joins the ring

Figure V.8 illustrates the ring with seven participating proxies. What happens is:

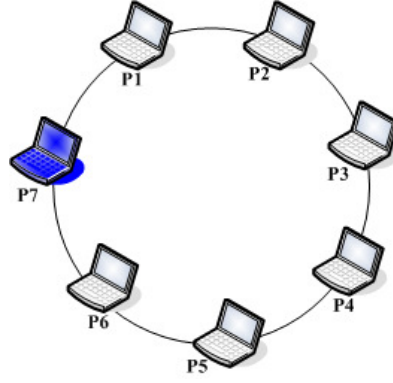


Figure V.8: Proxy ring after P7 joining

- P2 receives the message that P7 is inserted to the ring from its right and updates N . $N=7$, $N/2=3$ (integer part).
- The distance between P2 and P7 is $D(p2 \rightarrow p7)=2 < N/2$.
- $\text{Right}(p2)=\{P1, P6, P5\}$ is the list of the nodes that P2 can reach from its right. The distance between P2 and each of these nodes is calculated and compared to the distance between P2 and the new joining node (P7).
- $D(p2 \rightarrow p1)=1 < 2$. This means that no change to P1 entry in the routing table is needed.
- $D(p2 \rightarrow p6)=2 \geq 2$. This means that the distance may change. The new distance could be $D(p2 \rightarrow p6) + 1 = 3$. In addition, the new distance $D(p2 \rightarrow p6)=3 \leq N/2$. This means that the distance effectively changes but the value of the “send” column does not change.
- $D(p2 \rightarrow p5)=3 \geq 2$. This means that the distance may change. The new distance could be $D(p2 \rightarrow p5) + 1 = 4$. In addition, the new distance $D(p2 \rightarrow p5)=4 > N/2$. This means that the distance does not change but the “send” column in the routing table should be switched. Thus, for entry regarding P5 in the routing table of P2, the distance should be kept (i.e. $D(p2 \rightarrow p5)=3$) and the “send” column should be set to “Left”.
- P2 adds to its routing table a new entry about P7. The “Distance” is equal to “2” and the “Send” field is set to “Right” because P2 has received the notification from its right.

At this step, P2’ routing table looks like the first three columns of table V.4.

Destination	Send	Distance	Path
P1	Right	1	P2P1
P2	-	0	P2
P3	Left	1	P2P3
P4	Left	2	P2P3P4
P5	Left	3	P2P3P4P5
P6	Right	3	P2P1P7P6
P7	Right	2	P2P1P7

Table V.4: P2 routing table after P7 joining

V.4.5 Call scenario

Let's consider figure V.9. We assume that there are five node groups in the overlay and that Client C2 in group 2 wants to call Client C5 in group 5. First, the two clients register to their respective GR (Steps 1-4). In step 5, C2 sends a SIP *INVITE* message to its proxy (i.e. P2). P2 checks its cache and asks its GR but does not get C5's IP address (steps 6-8). Then, the proxy forwards the message to its successor and its predecessor. The successor and predecessor should forward the message to their neighbors except the node from which they receive the message. Each time that the message is received by a proxy, the proxy should check in its cache or in its GR (steps 9, 9'). If the target node's IP address is found, the message is stopped being forwarded. Otherwise, it is sent to the following proxy. In this scenario, P2 sends the *INVITE* message to P1 and P3. P1 will forward the message to P5. On the other side, the message will go from P3 to P5 through P4. But notice that P5 already got the same message from P1 and then should process only the first received message and should ignore the second one. Therefore, P5 checks its caching (step 9) or its GR and gets the target node IP address (steps 10-11) and then forwards the *INVITE* message to it (step 12). The targeted node should create a SIP *OK* message and send it to its proxy P5 at step 13. P5 then checks the OK message and sent it in the back way. Notice here that we don't need to check the routing table to know where to send the *OK* message because the proxies addresses are recorded in the message. When the *OK* message is being sent back to the source node, each proxy on the route stores in its cache, not only the binding between a SIP URI of a given node and its IP address, but also its related proxy (steps 14-16). This is useful for the next time the proxy will receive a SIP *INVITE* message targeted to the same node. In that case, the proxy will check its cache and find out that the target client is client *X* whose proxy is *Y*. Then, it will check in its routing table the way to reach this proxy. This will prevent the proxy to send the message to its two neighbors and the road chosen is surely the shortest one. When the source node's proxy receives the *OK* message, it sends it to the source node at step 17 and the rest of SIP signaling messages are sent following the same route.

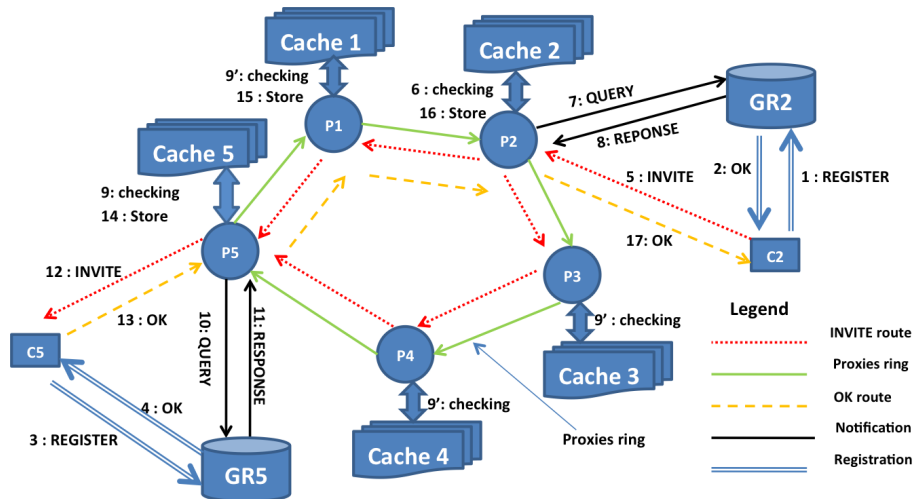


Figure V.9: Call establishment scenario

V.4.6 Evaluation

We used OverSim [15], an open source P2P network simulation framework, to evaluate the proxy topology building and proxy-level routing performance in our architecture. The goal of our simulations is to evaluate the performance of our routing algorithm and compare it to other techniques such as the usage of OSPF [69] routing algorithm in our architecture and the usage of DHT-based routing to send the SIP message to the destination. In addition, we show the benefit of implementing a routing table in each proxy node, instead of using for instance, the alternative to always route the message to one direction.

First, we discuss how OSPF can be used as the routing protocol of our architecture. Subsequently, we describe the DHT-based routing framework that has been simulated. Finally, we present and analyze the obtained results.

a) OSPF simulation on our architecture

This section describes how we adapted and implemented OSPF on our architecture using the simulation framework. In this simulation, we only focus on proxy node joining.

When a proxy P_j enters the ring, it creates a Link State Packet (LSP) containing its two neighbors and the distance to reach them (1 in this case) and sends it to all of the proxies in the ring. The LSP is sent following the same procedure, we described earlier for node joining, meaning that the LSP is sent to the neighbors using a *SUBSCRIBE* message and the neighbors forward it to all of the other proxies using *NOTIFY*. Each of the other proxies, creates a similar LSP and sends it to all of the other proxies (using *NOTIFY*).

After a proxy receives $N-1$ LSP (i.e. LSPs are received from all other proxies), it computes its routing table using the Shortest Path First (SPF) algorithm.

The main difference between this implementation of OSPF and our routing algorithm is in the number of messages exchanged on the ring, in order to allow the correct update of their routing tables when a proxy joins or leaves the P2P SIP network.

b) DHT-based routing

Let us describe now the framework used to allow DHT-based routing of the SIP message. OverSim has made available a simple application for testing the DHT called DHTTestApp [137]. Basically, DHTTestApp is an application that is used to demonstrate the functionality of adding, updating, and querying resources/contents using a DHT. We simply modify the DHTTestApp application and allow it to process the SIP message as any P2P message. The peer that creates a SIP message sends it to its neighbor. The message is then routed used Key-based Routing (KBR) strategy, based on the underlying DHT (Chord, Kademlia, etc), until it reaches its destination. Indeed, KBR is a lookup method used in conjunction with distributed hash tables and certain other overlay networks. While DHTs provide a method to find a host responsible for a certain piece of data, KBR provides a method to find the closest host for that data.

c) Experimentation Results

i. Comparison with OSPF

We evaluate and compare first the traffic overhead introduced by our algorithm and that generated using OSPF as the routing protocol of our P2P SIP architecture. Second, we compare the convergence time of the two algorithms. The convergence time, we measured is the time between the new proxy entering the ring and the time when all of the proxies participating in the overlay

receive the information about the new proxy and update their routing tables. The evaluations show that our protocol outperforms OSPF.

1. Network load

Figure V.10 shows the network load in terms of the number of messages transmitted in the network. The figure clearly shows that OSPF generates more messages than our algorithm. This can be explained as follows: When a proxy joins the overlay, with our algorithm, only the joining proxy sends an LSP to all of the other proxies. In OSPF, an LSP is sent by each of the proxies in the overlay. The difference between the two algorithms becomes more noticeable as the number of proxies in the overlay grows.

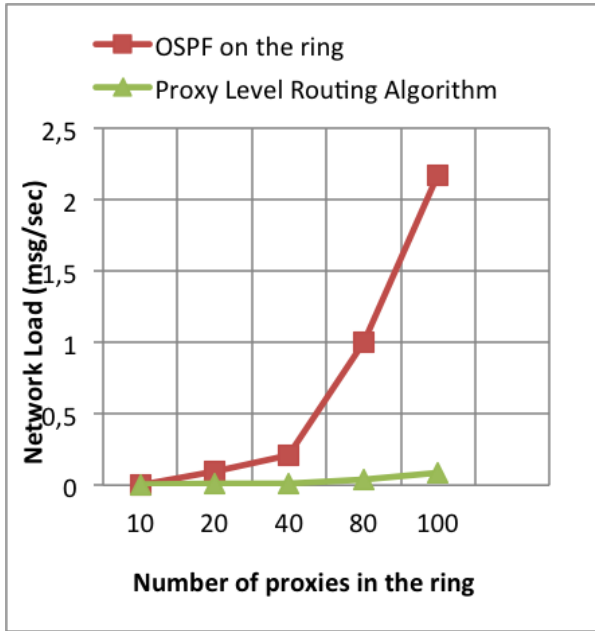


Figure V.10: Proxy level routing algorithm vs. OSPF: network load

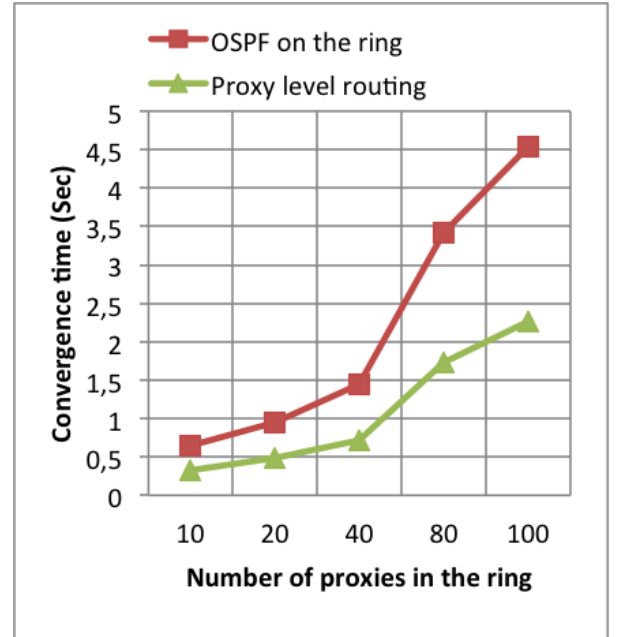


Figure V.11: Proxy level routing algorithm vs. OSPF: convergence time

2. Convergence time

The convergence time is the time that all nodes participating to the overlay spend to update their routing tables. Figure V.11 shows the convergence time of the two routing algorithms (i.e. our algorithms and OSPF) as a function of the number of proxies in the overlay. The simulations are run on top of the Chord P2P middleware. The results are collected for different numbers of overlay proxies.

We can easily observe that the convergence time of our algorithm is much smaller than the OSPF convergence time. This is because a large number of messages are exchanged in OSPF, and each proxy needs to wait until it gets an LSP from other proxies. Furthermore, each proxy needs to build the entire proxy graph in order to update its routing table. Instead, our algorithm encapsulates the distance between the new incoming proxy and all other proxies, in the proxy joining information packet.

ii. Comparison with DHT-based routing

The simulations have been run on top of Kademlia P2P middleware. We evaluate first the SIP average end-to-end round trip delay (two-way latency), from request generation (*INVITE*

message) at the client to the reception of the final response (*OK* message). Then, we compare the RTT registered with our algorithm and those getting using a DHT-based routing for SIP. Second, we compare the network traffic generated by each of the two frameworks. The network traffic measured includes all SIP messages and the overlay maintenance and lookup messages.

Figure V.13 shows the RTT as a function of the number of nodes in the overlay for both proxy-level routing and DHT-based routing. The results were collected for different numbers of overlay nodes. The network overload measurements are shown in figure V.12. The results show that our framework generates much less traffic than a DHT-based routing framework. However, in our framework, the RTT increases much quickly than in the DHT-based routing as long as the number of node increases. But the RTT provided by our algorithm is still reasonable (under 250ms) to offer a good quality of service for SIP calls with up to 700 nodes in the overlay.

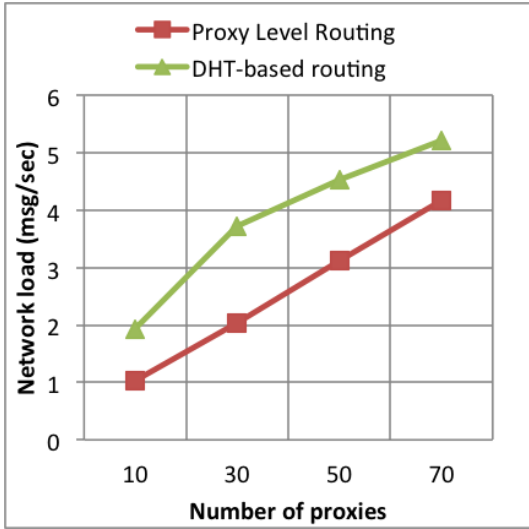


Figure V.12: Proxy level routing algorithm vs. DHT-based routing: Network load

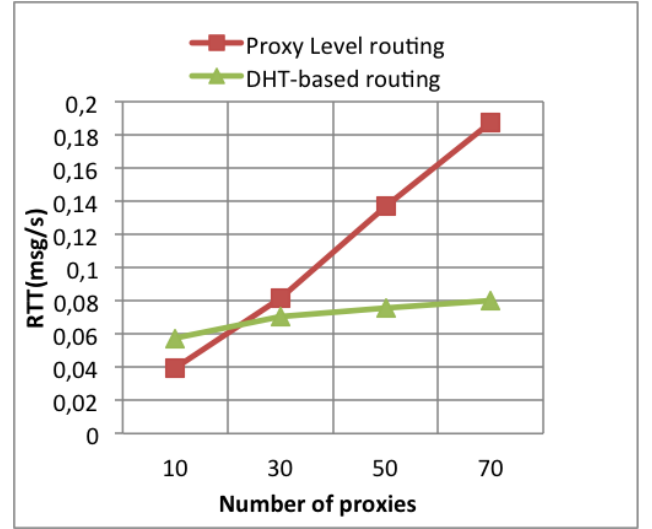


Figure V.13: Proxy level routing algorithm vs. DHT-based routing: RTT

iii. Comparison with “fixed direction” alternative

This simulation is performed to show the utility of having in the routing table the shortest path towards each destination. Our routing algorithm always uses the shortest path to send the *INVITE* message to the destination client (except the first time the client is called). Another alternative could be to send always the *INVITE* in one direction (left or right). Using this alternative, in some cases the message could take much time to reach the destination proxy. We have simulated the alternative of always sending the *INVITE* request to right (i.e. to the predecessor) and compare it to our framework by measuring the RTT of sending an *INVITE* and the network traffic that it generates. The results show that our framework generates much less traffic and significantly reduces the RTT delay with respect to the simulated alternative. RTT measurements are shown in figure V.14 and the network load is shown in figure V.15.

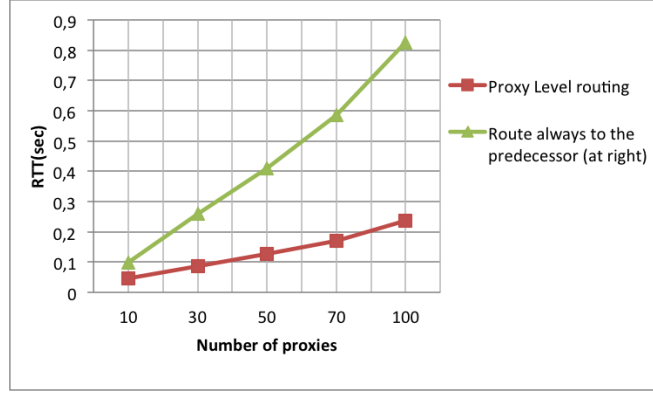


Figure V.14: Proxy level routing versus “Right alternative”: RTT measurements

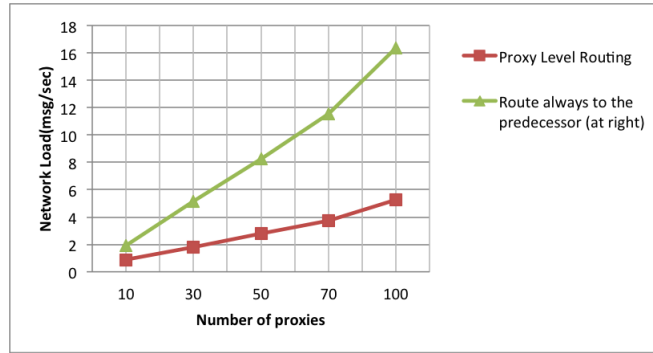


Figure V.15: Proxy level routing versus “Right alternative”: Network load

V.5 Conclusion

We have presented in this chapter a novel framework for proxy topology building and proxy-level routing on top of MISE-P2PSIP architecture. We have described an algorithm that builds the network of proxies in a ring, and our own routing algorithms that outperform existing routing algorithms such as OSPF when used on the ring topology. The novel algorithms are fully described and their performance evaluated via simulations. In comparison with a DHT-based routing scheme, our routing algorithms reveal to be much better in term of network load and provide reasonable RTT for SIP-based communications. Furthermore, we have shown the impact of having at proxy level, a routing table that always provides the shortest path instead of sending almost randomly the SIP message.

Chapter VI

NAT traversal solution for signaling messages and media streams

This chapter firstly defines NAT traversal issues and gives short background. Following, we outline the motivation and the requirements of our NAT (Network Address Translation) traversal approach. Moreover, the chapter discusses the related work and details the proposed solutions. Finally, we define an implementation plan and present some relevant experimental results.

VI.1 Definition and background

Network Address Translation (NAT) is a technology that has revolutionized Internet Communications. NAT allows multiple computers on a LAN to share a single public IP address for accessing to Internet. Without NAT, the IPv4 protocol's number of available addresses would reach its limits. NAT also provides some measures of “cloaking” of internal computers, since they are “hidden” from external (Internet) computers that can only “see” the NAT device through which they connect. The immediate benefit of NAT is that it allows a single internet connection with a single IP address to be shared. However, NAT has traditionally suffered from a big shortcoming. NAT breaks protocols/applications that require incoming connections and protocols that carry IP addresses in them. An example of these applications is VoIP. Actually, a VoIP client (computer with a “softphone” or VoIP phone) registers with a VoIP server, and then the server inform a VoIP client about an incoming call. The packets that carry the actual conversation are then exchanged directly between the calling parties with no involvement from the server. But, in order to connect calling parties, the server must be able to tell each VoIP client, where to send the VoIP packets. This must be a real, public address, and not the private address the VoIP application thinks it has. Each end VoIP client must be able to receive those incoming packets, which do not match a prior outgoing session in the NAT. The solution to this issue is NAT Traversal.

RFC 5389 [66] defines a terminology for different NAT types depending on their address binding schemes for UDP traffic. In particular, it uses the terms “Full Cone”, “Restricted Cone”, “Port Restricted Cone” and “Symmetric” to refer to different variations of NATs/firewalls. Following we describe the characteristic of each type.

- **Full Cone:** A full cone NAT is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Furthermore, any external host can send a packet to the internal host, by sending a packet to the mapped external address.

- **Restricted Cone:** A restricted cone NAT is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Unlike a full cone NAT, an external host (with IP address X) can send a packet to the internal host only if the internal host had previously sent a packet to IP address X.
- **Port Restricted Cone:** A port restricted cone NAT is like a restricted cone NAT, but the restriction includes port numbers. Specifically, an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P.
- **Symmetric:** A symmetric NAT is one where all requests from the same internal IP address and port, to a specific destination IP address and port, are mapped to the same external IP address and port. If the same host sends a packet with the same source address and port, but to a different destination, a different mapping is used. Furthermore, only the external host that receives a packet can send a UDP packet back to the internal host.

VI.2 Motivation

Traversing NAT is one of the major issues that hinder SIP communications. The reason is that, in SIP, many of the communication parameters are transmitted within the SIP messages; such parameters include the IP and port numbers used for signaling and media. A SIP device behind NAT does not know much about how it will be seen from the public network. It only knows its own private IP address and the ports where the SIP application runs. These addresses, being private, cannot be used by the destination node for answering. Therefore, SIP cannot work efficiently through a NAT device without using a NAT traversal mechanism.

In the previous chapter, we have proposed a scalable proxy topology building and routing in P2P SIP network, where SIP location service is integrated in proxies that run over peer entities. Although P2P SIP comes with numerous benefits, it should cope with NAT traversal issues. Our proposed architecture needs an efficient NAT traversal technique which can work easily and make end-to-end communication by meeting our requirements. We need to modify our P2P SIP framework in order to provide a suitable technique for NAT traversal for both SIP signaling messages and media streams.

Since in our P2P SIP framework, any P2P node can play any SIP role, taking into account networks with NAT can raise some roles incompatibility with respect to the location of the joining node. For sake of simplicity, we assume that only node located in the public network could play SIP server roles (Proxy or Registrar) in order to be easily and efficiently reachable without specific NAT traversal procedures. Consequently, node behind the NAT could only play the role of client. Thus, the role of a given node could not be any more assigned only by taking into account the node entering order in the overlay. Figure VI.1 shows an example of our architecture deployment over a NAT-ted network.

VI.3 Requirements

As described in chapter III, the main requirements of MISE-P2PSIP framework could be summarized into three key features that are Independence from P2P middleware, compliance to SIP standard and support for a “pure” P2P overlay.

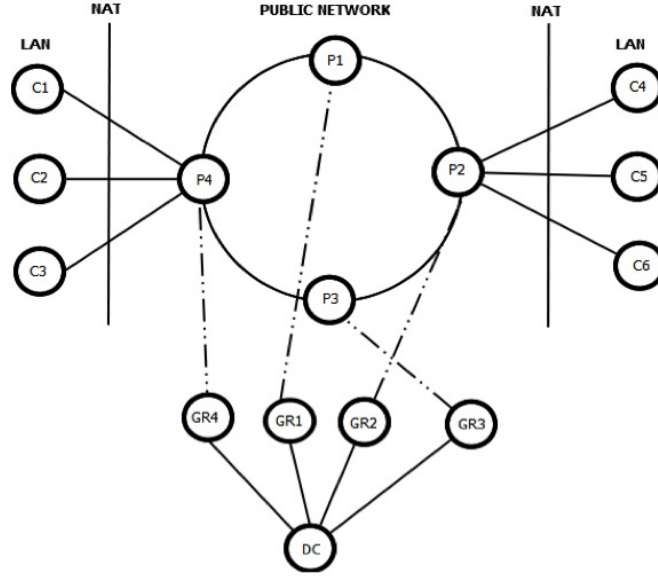


Figure VI.1: Example of nodes' topology behind NATs

In addition to the requirements cited above, in order to handle NAT traversal issue in MISE-P2PSIP architecture, we define two more relevant requirements. The first one is about the performance of the communication. This means that our proposed solution should not lead to communication quality degradation (e.g. in terms of latency or delay). The second requirement is that our proposed solution should not require modifications in SIP clients in order to allow interoperability, because many NAT traversal solutions require SIP client to support other protocols.

VI.4 Related work and analysis

NAT traversal is a true-honored problem for all VoIP-related protocols. The existing NAT traversal solutions could be classified in two categories. The first category includes standardized techniques. The non-standardized techniques include solutions proposed by some research activities and will be part of the second category. We review these solutions in light of our requirements.

VI.4.1 Standardized techniques

In order to allow NAT traversal for SIP, RFC 3581 [74] defines a SIP extension for symmetric response routing when SIP request operates over User Datagram Protocol (UDP). Indeed, this extension defines a new parameter for the Via header field, called “rport”, that allows a client to request that the server sends the response back to the source IP address and port from which the request originated. This solution does not fit our basic requirement on no SIP extensions.

STUN (Session Traversal Utilities for NAT) [66] allows a device behind the NAT to determine the NAT's behavior and bindings indirectly, and to modify the protocol messages appropriately. Using STUN different mappings are open in the NAT device for each new IP address: port combination. This renders the information provided by the STUN server useless for initiating communication to other addresses than the STUN server address. The defect of STUN is that it cannot work with symmetric NAT, which is widely used in today's enterprises. Therefore STUN cannot provide a complete solution. TURN (Traversal Using Relay NAT) [72] solves this problem by relaying data through a server that resides on the public Internet. A device behind NAT would

use TURN protocol to get the address and port on the TURN server. Then, the device uses them to invite its peers. TURN also assumes the clients have a trust relationship with a TURN server and requests session allocation based on shared credentials. This has scalability issues and requires complex changes in the SIP clients. Both STUN and TURN require that SIP client device supports them and do not fit our second requirement.

Another methodology called ICE (Interactive Connection Establishment) [73] allows direct communication. ICE allows the traversal of all devices and all topologies except the case where firewalls are used. ICE does not allow this crossing, since firewalls are devices that filter packages by static rules. Therefore, since ICE technique consists of identifying all possibilities of communication between a client's device and a public network and between two client devices, it increases the time needed to set up a call. This does not fit our performance requirement. In addition, client's device has to be modified to be able to negotiate dynamically a communication path. Thus, this does not fit our second requirement.

There are also Hole punching techniques using UDP or TCP [78][79] that belong to this category. Indeed, the hole punching NAT traversal mechanism of UDP packets, allows each peer behind a NAT to discover the presence and types of NATs and firewalls between them and the public network. This discovery includes the NAT's treatment of UDP traffic and the public IP address and port assigned to the peer. Afterward, the public address and the NAT behavior obtained are used to predict the address and port number for a subsequent session between the peers. Instead, using the NAT traversal Hole Punching of TCP, each host behind a NAT has to not only detect the presence of NAT and predict the public address/port assigned to it by the NAT, but has also to obtain the initial sequence number in the first SYN packet sent by the other hosts to establish a TCP connection. As STUN and TURN, Hole punching techniques requires modifications in the SIP device and do not fit our second requirement.

VI.4.2 Non-standardized techniques

Some research activities have provided solutions to handle NAT traversal issue.

The authors of [77] have proposed a technique called TAB (Triggering Address Binding). TAB technique deployed a proprietary server that allows SIP client to know whether it is behind a NAT device or not. TAB makes use of the NAT characteristic by triggering address bindings on the NAT and keeping these bindings alive for NAT traversal of SIP messages and media traffic. The TAB assigns to the client, different public transport addresses for signaling messages exchange and media stream transport. The SIP client and the TAB server communicate by using specific TAB messages. This means that the SIP client should support TAB messages. This solution is not suitable because it is proprietary and could lead to interoperability issues.

Another approach on distributed NAT traversal mechanism called SMBR (Selective-Message Buddy Relaying) for structured P2P [80] has been proposed. SMBR distinguishes control messages from data based on DHT messages types. For control messages, SMBR uses the method of buddy's relay while for data, a direct connection is built between the peer nodes with the help of a buddy. Indeed, a buddy is a peer in the overlay having a public IP address. A buddy of the source peer has the duty to communicate with the buddy of the destination peer, using specific messages, in order to allow direct connection between the two peers that wish to exchange data. Therefore, the framework is not middleware-independent as our NAT traversal approach should be, since it is strongly based on Kademlia protocol. Moreover, the technique is based on DHT messages types and is not suitable to SIP messages exchange through NAT.

VI.5 NAT traversal solution

This section handles NAT traversal issues in MISE-P2PSIP network. Firstly, we summarize the techniques that we use to handle NAT traversal issue. Subsequently, we define the procedure used for nodes' group creation. Moreover, the procedures used for signaling messages and media flows are described. Finally, we explain how SIP session can be established from and towards nodes behind a NAT, through some call scenarios.

VI.5.1 Used techniques

Our NAT traversal techniques reuse some solutions that are currently used in many enterprises and adapt them to our framework [75].

In order to allow efficient SIP signaling exchange in presence of NAT device, we simply implement a “smart registrar” which does not save the contact address as presented by the device in the *REGISTER* message but rather save the real IP:Port combination, the message originates from. Subsequently, the smart registrar server maintains the communication channel open with the SIP client by exchanging with it dummy keep-alive packets so that it will be always possible to ring the device behind the NAT and to start negotiating a SIP session. The only requirement, which fortunately is available in most of the SIP devices, is to use symmetric signaling, that requires the device to send and receive data on the same port number. This solution is currently used by OpenSER [129].

On the other side in order to allow media transport, we use “media relay” technique which allows exchanging media stream between the two SIP clients in presence of NAT through a trusted third party. However, this functionality is performed by one of the SIP servers already present in the framework. MediaProxy [59] is a distributed NAT traversal solution based on this technique.

VI.5.2 Node group creation

Some roles incompatibility can occur with respect to the location of the joining node by making more complex NAT traversal handling. Thus, we assume to assign node role by taking into account their location (i.e. behind a NAT device or not) in our architecture. Thereby, we need to revise our node group creation procedure proposed in the previous chapters. Based on this assumption, nodes behind a NAT device cannot play the role of GR (Group Registrar) server or Proxy server. Therefore, a node behind a NAT can act only as Client node. The nodes are also organized in several groups as explained in the previous chapters. A group is identified by its ID. Let's recall that a group (G_i) is composed of one GR, one Proxy and N client nodes where N is a configurable number. Role assignment procedure flow chart is shown in figure VI.2. The description is the following.

When a node (N_i) joins the overlay, it contacts the DC by sending to it a SIP *REGISTER* request. This SIP *REGISTER* message aims to get useful information from the DC. Afterward, the DC checks if the node is behind a NAT device. The checking is performed by comparing the IP address the packet came from and the one that is listed in the “via header”. If there is a difference, the node is assumed to be behind a NAT.

If the joining node is behind a NAT, the DC sets the node as client and sends it back the role in the SIP *200 OK* response. Notice that the *200 OK* response contains in addition to the node's role, the GR address, the Proxy address and the group ID of the group assigned to the Client node. Basically, the information is piggybacked in the optional attribute “u” (containing

the URI of description) of the body of the message. An example of a piggybacked *200 OK* message is shown in figure VI.3.

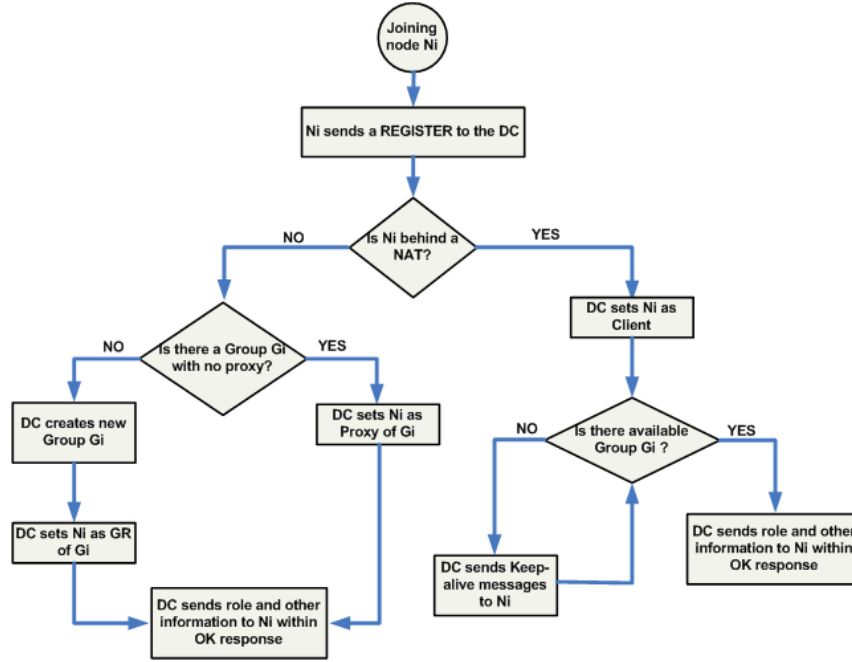


Figure VI.2: Role assignment procedure flow chart



Figure VI.3: Piggybacked 200 OK response

The *200 OK* message is sent automatically to the client node if a group is available (if there is a group with number of client less than N). Otherwise, it will be sent later when a new group is created. In this later case, the DC should send periodically keep-alive messages until it will be able to assign a group to the Client node. Of course waiting for a new group to become available, introduces a level of latency not known a priori. In order to alleviate this problem, a suitable SIP *202* message could be introduced to notify that the request is accepted but cannot be actually processed.

The DC would normally send the *200 OK* message to the port inserted in the “via header” of the SIP *REGISTER* request, but since the client is behind a NAT, it will use the real IP/Port, the packet is received from. After getting its Registrar server and Proxy server addresses, the client

node creates another SIP *REGISTER* request that it sends to its GR for real registration.

Let us now consider the case where the joining node is not behind a NAT device. The DC should set this node as GR or as Proxy and send it the *OK* response. Indeed, the joining node is set as proxy if there is a group with no proxy. Therefore, for a given group, the GR is set first and the Proxy later. The DC maintains the list of the groups and for each group, it updates and stores the GR and proxy addresses, and the number of clients.

Else if there is no group without proxy, the DC creates a new group and sets a new joining node (not behind a NAT device) as GR of this group.

VI.5.3 NAT traversal for SIP signaling messages

When a SIP device with private addresses (IP and port), wishes to be reachable from the outside of its private network, it must first initiate a connection to the public network. Many techniques like STUN [66] and ICE [73] can be used to allow the client knowing how it is seen from Internet. But, they do not fit our requirements.

In our architecture, the SIP client device does not need to know its public IP address and port. We simply implement a smart Group Registrar which does not save the contact address as presented by the Client node in the SIP *REGISTER* message during the registration phase. Instead, the registrar server (GR in our architecture), stores the real IP:Port (public) combination, the message originates from. The procedure used to exchange SIP signaling messages between two clients behind different NATs devices is summarized in figure VI.4. PR_A represents the proxy of the client device A and PR_B is the proxy of the client device B. The same definition is applied for the smart GR. Once, the GR has successfully registered the Client, it must maintain the communication channel open by sending periodically keep-alive packets to the Client node before the binding expires in the NAT device table [75]. The keep-alive packets could be a PING request sent by the GR to the Client node. The Client node answers by sending back a PING response. The goal of sending keep-alive packets is to maintain the public contact information (IP address and port) of a given Client node, on the NAT device table. In order to allow Client node to always receive SIP *INVITE* messages, its proxy should always check the Client node external IP address/port on the GR before sending the request to it. The delivery of the SIP *200 OK* response to the SIP *INVITE* request will follow the same procedure as the other signaling messages. The proxy queries its GR to get the destination node public IP as stated in SIP protocol standard specification [1]. In our architecture, the registrar server stores information in the overlay and retrieves it when needed.

VI.5.4 NAT traversal for media flows

Media consists of one or multiple flows which are negotiated in the SIP signaling messages. The media stream may be added or subtracted to the communication set between SIP devices. As this happens dynamically, one must be able to translate in real-time the mappings between the internal and public addresses.

Starting with the SIP *INVITE* message, the SIP devices negotiate a common media. The initial negotiation is performed by SDP (Session Description Protocol), a protocol used by SIP to convey information about the media stream (address where the media will be received, codec types, bandwidth and others). The problem is that, when a device is behind NAT, the SDP conveys information about the private IP of the SIP client device.

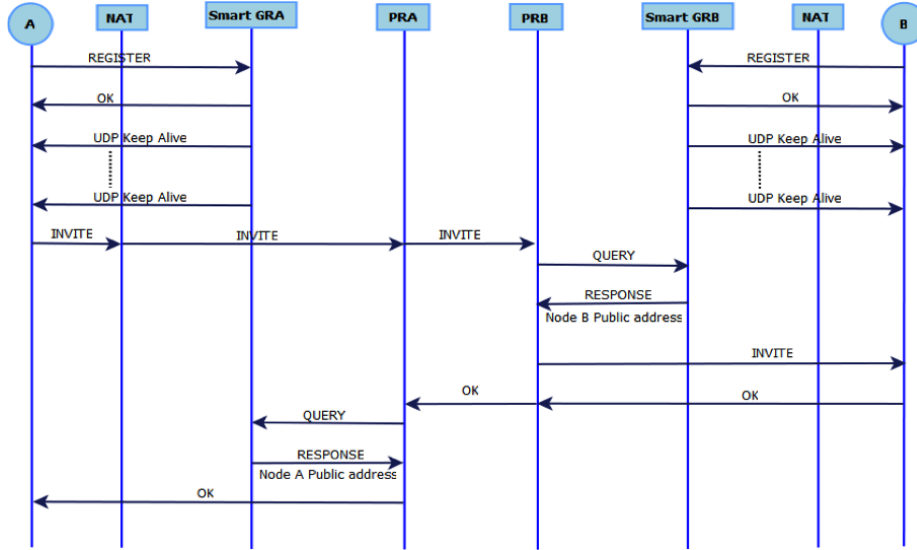


Figure VI.4: NAT traversal for signaling messages

Some solutions to this problem are already proposed in the literature. One solution is to improve the SIP devices to be able to negotiate dynamically a communication path for the media even after the initial SIP session has been setup. This can be achieved by ICE. In fact, before sending the *INVITE* message to the proxy, a Client inserts in the message, the list of available IP addresses/ports for media stream. This allows devices to probe for multiple paths of communication by trying to use different port numbers and STUN techniques. This media path discovery step by both caller and callee can introduce more delay in the communication.

Another alternative is to use a “Media relay”. With the Media relay technique, the media flows travel via a third party location on the Internet. Media relay technique reveals to be simple and efficient. Mediaproxy [59] and RTPProxy [57] are examples of applications that use this technique with Kamailio [48].

To enable efficient transport of media in our P2P SIP network, we define a suitable technique based on media relay technique. Our technique consists on relaying media through one of the servers in our P2P SIP network. We attribute the media relay role to the DC (Domain Controller) server because, we assume that it is trusted by the other nodes of the network. Let us recall that the DC and all servers (proxy and registrar) are located in the public network. Then, the address of the media relay is always known (being a public IP), so that the two endpoints know where to send RTP packets. Our technique allows a proxy node to perform the required modifications in the signaling message body in order to allow the two endpoints to exchange RTP packets through the DC server. To achieve this goal, we simply allow proxies to replace the private addresses provided by the originator of the signaling messages for media purpose, by the public addresses (IP address and available transport port for media) of the DC. Figure VI.5 shows our NAT traversal procedure for RTP packets during a SIP communication between two Client devices A and B.

Indeed, during the SIP signaling messages exchange, when the proxy of the caller receives a SIP *INVITE*, it asks to the DC to attribute to a given node an available port for media. The request is piggybacked in the body of a SIP *NOTIFY* message. The DC answers the proxy, by sending to it the port number in the *OK* response. Upon receiving the port number, the proxy of the caller inserts the port and the public IP of the DC in the attribute “o” (Owner/creator and session identifier) and/or in the attribute “c” (Connection information) of the body of the message,

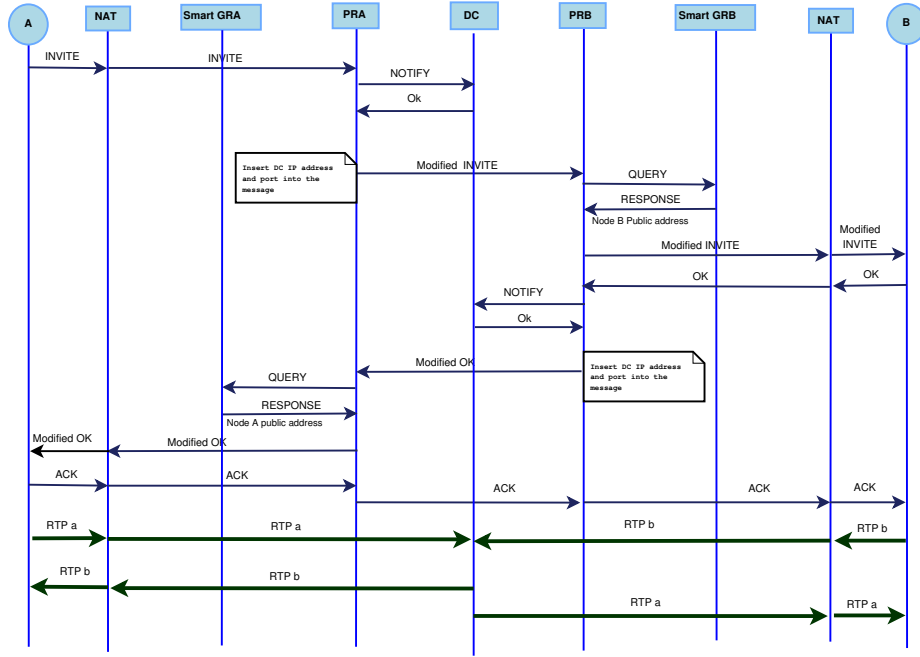


Figure VI.5: NAT traversal procedure for media flows

by deleting the private address before transmitting the message to the proxy of the callee node. By the same way, the proxy substitutes the private port number in the “m” (Media name and transport address) attribute of the body by the port number received from the DC. The proxy of the callee node performs the same procedure (it requests available port from the DC and inserts the DC’ public address and the port number in the message) upon receiving the SIP *OK* message from the callee. These operations are reasonably fast on modern CPUs and do not introduce additional delay. When the proxy of the caller device gets the SIP *ACK* message, it simply forwards it. In this way, the two endpoints will get the public addresses where the media streams should be sent. At the end of the signaling phase, the two nodes send media stream to the DC. In order to be able to receive the media from other endpoint, a node behind a NAT should send first media stream to the DC. Upon receiving the first media flows from both endpoints, the DC records the addresses they came from and will know where to forward RTP packets received from the other endpoint. This is an important step because the address/port, the NAT device will allocate for the media stream, is not known before the media flows are actually sent through the NAT device. Afterward, the DC learns their addresses and can efficiently forward packets between them.

When the two endpoints are behind the same NAT device, the media streams do not take through the DC, but are exchanged directly through the two endpoints. An example of this use case, is given in the next sub-section and depicted in figure VI.6.

VI.6 Call scenarios

Based on our group creation procedure, we distinguish four (4) different call scenarios in our framework.

- Caller and callee are behind the same NAT and are connected to the same proxy server.
- Caller and callee are behind the same NAT and are connected to different proxy servers.

- Caller and callee are behind different NATs and are connected to the same proxy server.
- Caller and callee are behind different NATs and are connected to different proxy servers.

In this chapter, we will explain two scenarios. The other scenarios can be easily derived from the ones, we explain.

VI.6.1 The two Client devices are behind the same NAT and are connected to the same proxy

Let's consider the scenario depicted in figure VI.6. Let assume, there are two SIP clients C1 and C2 behind the same NAT device and connected to the same SIP proxy located in the public network. Client C1 initiates a SIP session toward C2. C1 is not aware that C2 is in the same local network with him. C1 forwards the SIP *INVITE* to its proxy P1 through its NAT (steps 1 and 2). P1 checks its routing table and notices that the destination client C2 is located in its group and gets its public IP address from GR1 (steps 3 and 4). At this step, P1 should check if the source node C1 and the destination client are located behind the same NAT. To achieve this it suffices to compare the IP address it received from GR1 with the real address the SIP *INVITE* packet came from.

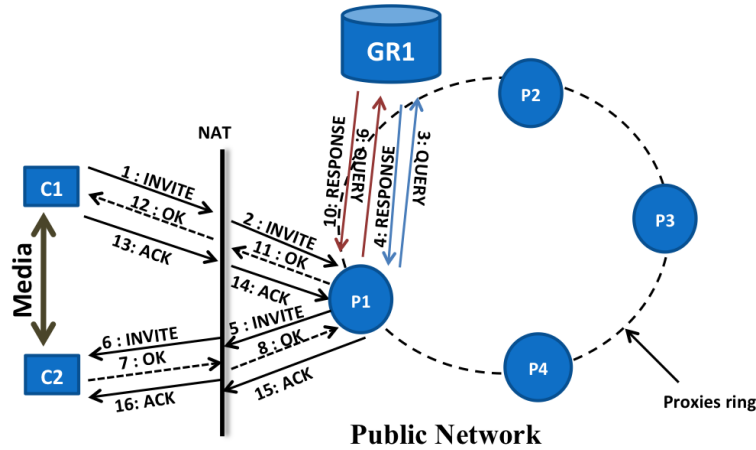


Figure VI.6: Call scenario for two nodes behind the same NATs

If the IP addresses are the same, the two clients are behind a common NAT. Subsequently, P1 forwards the SIP *INVITE* message to C2 through its NAT. But when P1 receives the *200 OK* response from C2 it does not need to make any modifications. At step 9 and 10, P1 checks again its GR to get the current public addresses (IP and port) of C1 in order to forward to it the SIP *200 OK* response. The rest of the signaling communication goes on, normally between C1 and C2 through P1. Thus, at the end of the signaling messages exchange, the media can flow directly between the two clients.

VI.6.2 The two Client devices are behind different NATs and are connected to different proxies

Let's consider now the scenario depicted in figure VI.7. There are two SIP clients C1 and C2 behind different NAT devices and connected to two different SIP proxies located in the public network. Client C1 wants to initiate a SIP session toward C2. C1 forwards the SIP *INVITE* to its

proxy P1 through its NAT device at steps 1 and 2. Then, P1 checks its routing table and notices that the destination client C2 is connected to proxy P3 which is reachable through proxy P2 along the shortest path. P1 sends a *NOTIFY* message to the DC to ask him to open a new port for media stream at step 3. The DC answers P1 using the OK response at step 4. P1 substitutes the private addresses (IP and port) provided by C1 in the body of the SIP *INVITE* message, in order to receive the media stream, through the IP address of the DC and the media port number on the DC, before sending the message to P3 through the proxies ring at steps 5 and 6. This substitution is useful to allow the DC relaying the media through the two endpoints. Otherwise, they will not be able to exchange RTP packets. When P3 gets the *INVITE*, it queries the public addresses of C2 to GR3 (steps 7 and 8). Once P3 gets the required information, it forwards the SIP *INVITE* to C2 (steps 9 and 10). C2 creates a *200 OK* message and sends it to P3 (at steps 11 and 12). At this step, P3 should check if C1 and C2 are behind the same NAT or not. To do this, P2 compares as before the public addresses of node C1 in the SIP *INVITE* with the public addresses of node C2. If these addresses are the same, it means that both nodes are behind the same NAT. Then, the *200 OK* message will not be modified. Otherwise both clients are behind different NATs and consequently the *OK* message needs to be modified. In this scenario, C1 and C2 are behind different NAT devices. Then, P3 requests a new port for media from the DC (steps 13 and 14) and inserts in the *200 OK* message received from C2, the public IP address of the DC and the new port before forwarding back the message (steps 15 and 16). Once it gets the message, P1 retrieves C1 public addresses from the GR (steps 17 and 18) and sends the *200 OK* message to C1 through its NAT device (steps 19 and 20). C1 answers with the SIP *ACK* message, that it sends to P1 at steps 21 and 22. Note that the modification of the *ACK* message is not needed here because it does not contain any information in its body. P1 forwards the *ACK* to C2 through P2 and P3 (steps 23 to 26). At this step, both clients are aware of where they might send RTP packets and are able to successfully exchange media streams through the DC server.

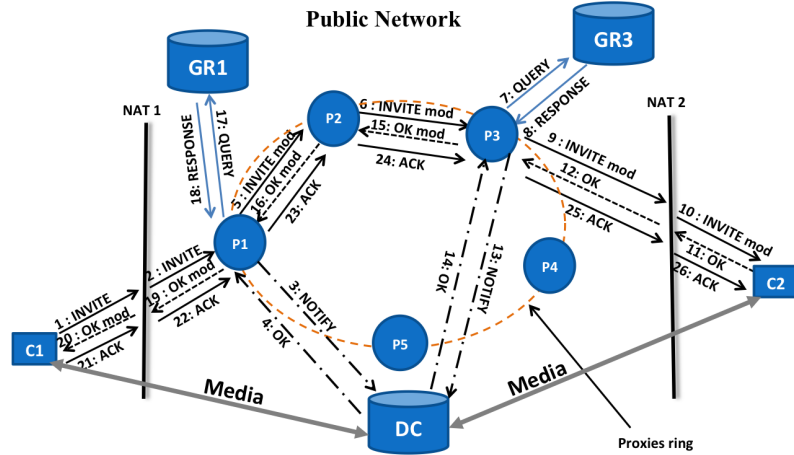


Figure VI.7: Call scenario for two nodes behind different NATs

VI.7 Implementation plan

This section aims to give the basic idea of an implementation plan for our NAT traversal solutions in P2P environments.

We will design a realistic network scenario to run our network with some nodes behind NAT

devices simulations with OverSim. The network is made of a set of nodes that run the P2P SIP application, a set of access routers and a set of backbone routers. The clients are connected to access routers using DSL (Digital Subscriber Line) connections (delay = 20ms and bandwidth = 1Mbps) and the peers are connected to access routers using Ethernet (delay = 1ms and bandwidth = 100Mbps). Access routers are connected to backbone routers using fiberlines (delay = 1ms and bandwidth = 10Gbps) and the backbone routers are connected to each other using fiberlines (delay = 1ms; bandwidth = 10Gbps). In order to test our NAT traversal solutions, some access routers will be configured with NAT support. The node linked to an access router with NAT support will have one private address attributed upon joining. A public address will be attributed to a given client by its access router when it attempts to contact a node outside of its private network. The nodes will run using Chord as P2P protocol. Any other P2P protocol can be used without modifying our NAT traversal procedures since our overlay is maintained by using only SIP messages.

Following, we define the basic procedures which will be implemented to simulate our NAT traversal solutions:

- Initialization: This procedure allows the Domain Controller (DC) to send to the joining node, useful information like Peer ID, SIP URI, Group ID, GR' IP, Proxy' IP, and role. This procedure calls the Group_management procedure to get this information.
- Create_Peer: Attributes to the new joining node, a Peer ID and a SIP URI.
- Group_management: This procedure stores in the cache, each group ID, the GR' IP, Proxy' IP and the current number of clients in each group. This procedure will attribute a role to the joining node accordingly to its related access routers (with NAT support or not). It also creates a new group when needed.
- SIP_request: This procedure allows the creation of all types of SIP request (*REGISTER*, *INVITE*, etc).
- SIP_response: allows answering suitably to a given SIP request.
- Store_IP_address: This procedure allows the GR to retrieve and store the public address of the client node upon registration.
- Retrieve_IP_address: This procedure allows a proxy to retrieve the public addresses of a given client from the GR in order to correctly deliver the SIP message.
- Keep_Binding: This procedure allows the GR of a given client to send periodically to it, keep-alive packets in order to maintain the public addresses of the client in the NAT device.
- Modify_SIP_message: In general, this procedure allows piggybacking the SIP messages. For instance, it allows to substitute in the SIP *INVITE* or in the SIP OK response, the private addresses of the message originator by the public IP address and media port of the DC.
- Send_SIP_message: this procedure allows sending SIP message from one node to another node.

VI.8 Experimental results

We have setup an experimental environment. The goal of this experimentation is to show that our solution for NAT traversal is simple and can enable two nodes inside two different private

networks to efficiently exchange SIP signaling messages and media streams. Our experimental environment is composed of seven (7) Virtual Machines (VM) running on Linux Operating System. Figure VI.8 depicts our experimental environment. Basically, we implemented a scenario of two SIP clients, each behind its NAT device. Each SIP client has one network interface with a private address and has its proxy server located in the public network. The DC server which acts as media relay is also located in the public network.

The NAT devices have one public network interface (192.0.2.X) and one private network interface (10.10.8.X or 10.10.9.X). Instead, the proxy servers and the media relay have each, a single network interface in the public network.

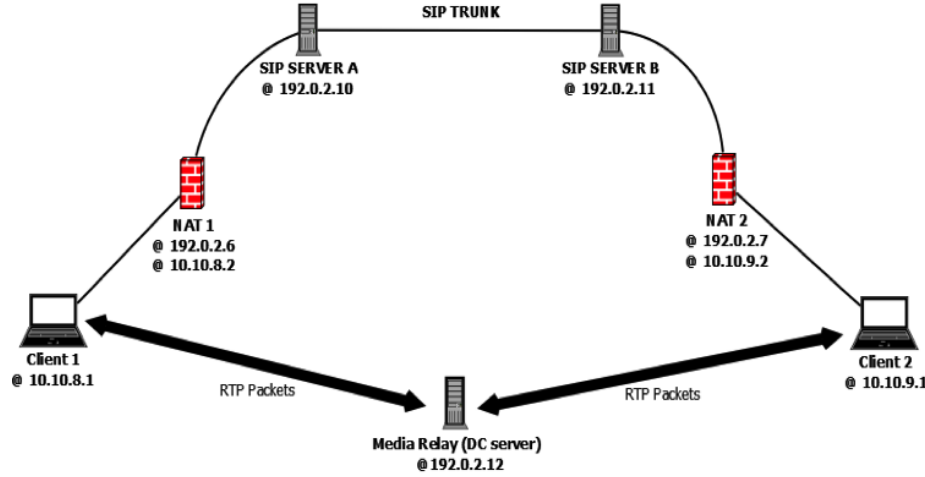


Figure VI.8: Experimental environment

To make effective our experimental test, firstly, we have installed and configure suitability Asterisk 1.8 [76] on each computer which might act as SIP proxy server. Asterisk is open source software which transforms easily a computer into a communication server. The two proxies communicate using a SIP trunk. In our test, the registration function is integrated into each proxy. The media relay is configured using Kamailio [48] with RTPProxy module [57] for NAT traversal. Kamailio is an Open Source SIP server released under GPL, able to handle thousands of call setups per second.

We have installed a Linphone softphone on each computer which might act as SIP client. Linphone is an open source VoIP (Voice over Internet Protocol) software which can be used to make computer to computer calls or computer to phone calls.

We implemented our NAT devices using Iptables. We have set up many reliable Iptables rules to allow each NAT device to accept SIP packets, to know where to redirect the SIP packets (port and IP address). In addition, those rules permit to the NAT device to change SIP packets' payload (Prerouting or Postrouting) when needed before forwarding the packets.

Following are two examples of rules that we have set up on NAT 1 to authorize SIP signaling packets delivery:

- `sudo iptables -t nat -A PREROUTING -i eth1 -p udp -dport 5060 -j DNAT --to-destination 10.10.8.1`

This rule called Destination NAT (DNAT) translation, allows all packets received from the public network and targeted to eth1 (public interface of NAT 1), with destination port 5060, to be redirected to the private address 10.10.8.1 (IP address of client 1).

- `sudo iptables -t nat -A POSTROUTING -o eth1 -p udp -s 10.10.8.0/24 -dport 5060 -j SNAT --to-source 192.0.2.6`

This rule called Source NAT (SNAT) translation, allows replacing the original source address of a SIP outbound packet from the network 10.10.8.0/24 with destination port 5060 by the public address of the NAT device (192.0.2.6).

We use Wireshark software to capture the packets on Client 1. Our tests reveal to be encouraging since each SIP client device inside a private network can successfully register itself to the proxy server (integrating the registration function) in the public network through its NAT device. Moreover, the SIP signaling messages are able to be efficiently exchanged between the two SIP clients behind different NATs by traversing their respective NAT devices and Proxies servers. Furthermore, the RTP streams are sent between the two SIP clients through the DC.

Figure VI.9 shows the SIP signaling flows on Client 1. In this test, SIP Client 1 initiates a call towards SIP Client 2. The SIP *INVITE* has been sent to the proxy of Client 1 (192.0.2.10). The SIP *OK* sent by SIP Client 2 has also been successfully received by Client 1 through its proxy server. The same applied to the SIP *ACK* message. Instead, figure VI.10 shows RTP flows on the media relay.

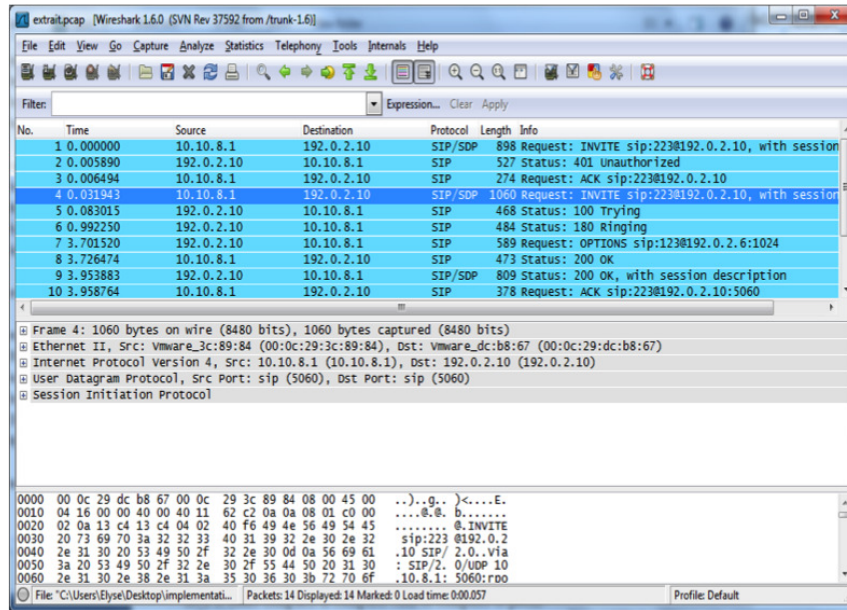


Figure VI.9: SIP signaling flows

No.	Time	Source	Destination	Protocol	Length	Info
49	13.236356	192.0.2.12	10.10.9.1	RTP	69	PT=speex, SSRC=0x188
50	13.253393	10.10.9.1	192.0.2.12	RTP	60	PT=speex, SSRC=0x382
51	13.253517	192.0.2.12	10.10.8.1	RTP	56	PT=speex, SSRC=0x382
52	13.268732	10.10.8.1	192.0.2.12	RTP	69	PT=speex, SSRC=0x188
53	13.268839	192.0.2.12	10.10.9.1	RTP	69	PT=speex, SSRC=0x188
54	13.284662	10.10.9.1	192.0.2.12	RTP	60	PT=speex, SSRC=0x382
55	13.284791	192.0.2.12	10.10.8.1	RTP	56	PT=speex, SSRC=0x382
56	13.295716	10.10.8.1	192.0.2.12	RTP	60	PT=speex, SSRC=0x188
57	13.295811	192.0.2.12	10.10.9.1	RTP	56	PT=speex, SSRC=0x188
58	13.305173	10.10.9.1	192.0.2.12	RTP	60	PT=speex, SSRC=0x382
59	13.305246	192.0.2.12	10.10.8.1	RTP	56	PT=speex, SSRC=0x382
60	13.325663	10.10.8.1	192.0.2.12	RTP	60	PT=speex, SSRC=0x188
61	13.325770	192.0.2.12	10.10.9.1	RTP	56	PT=speex, SSRC=0x188
62	13.344822	10.10.9.1	192.0.2.12	RTP	69	PT=speex, SSRC=0x382
63	13.344914	192.0.2.12	10.10.8.1	RTP	69	PT=speex, SSRC=0x382
64	13.366528	10.10.8.1	192.0.2.12	RTP	69	PT=speex, SSRC=0x188
65	13.366629	192.0.2.12	10.10.9.1	RTP	69	PT=speex, SSRC=0x188
66	13.387612	10.10.9.1	192.0.2.12	RTP	60	PT=speex, SSRC=0x382
67	13.387753	192.0.2.12	10.10.8.1	RTP	56	PT=speex, SSRC=0x382
68	13.406891	10.10.8.1	192.0.2.12	RTP	69	PT=speex, SSRC=0x188
69	13.406951	192.0.2.12	10.10.9.1	RTP	69	PT=speex, SSRC=0x188
70	13.424039	10.10.9.1	192.0.2.12	RTP	60	PT=speex, SSRC=0x382

Figure VI.10: RTP packets through media relay

VI.9 Conclusion

In this chapter, we proposed a simple solution to deal with NAT traversal issue in our P2P SIP framework. Our NAT traversal solution provides a way to efficiently set up end-to-end VoIP call. It also allows RTP streams to be exchanged through the DC server that acts as media relay and is located in the public network, precisely when the two SIP client's devices are behind different NATs. We have set up a virtual environment in which we implemented a full SIP-based communication system between two private networks and performed some relevant tests to show that our approach is feasible and can easily work in real world.

Although we did not perform our tests in P2P environment, the satisfying results, currently obtained in a pure SIP environment could be observed as well in P2P SIP overlay since the NAT traversal issue is only related to SIP world.

Part IV

Secure routing

In the previous chapters, we have proposed a P2P SIP architecture with specific requirements that we called MISE-P2PSIP and suitable proxy topology building and routing algorithms. Therefore, MISE-P2PSIP network can be subject to many attacks due to the fact that the node acting as server can misbehave. Many servers can also cooperate to misbehave. In particular, the proxy node which has the role to transmit signaling messages could drop, wrongly alter, delay or misroute the SIP messages. Those misbehaviors can severely affect the quality of service in P2P SIP environments. Then, it is important to secure the routing of SIP signaling messages in such a way that they are processed and delivered correctly. In this part of the thesis, we mainly focus on tackling two important attacks that can create unneglected inconveniences to SIP-based communications. Those attacks are called hijacking attack and misrouting attack. By performing misrouting attack, a given proxy server can send the signaling message on left instead of sending it to right and vice versa. This misbehavior can lead to the delaying of the transmission or the suppression of the message. On the other side, a malicious proxy can substitute a legitimate node during a SIP session establishment by performing call hijacking attack. We provide secure solutions for those two attacks.

This part of the thesis is composed of two chapters. Chapter VII outlines the motivations and the requirements and provides the related work on secure messages routing. Instead, chapter VIII gives the details of our secure schemes against attacks in MISE-P2PSIP.

Chapter VII

Related work on secure routing protocols and mechanisms

This chapter gives first the motivations for providing solutions to secure messages routing in MISE-P2PSIP. Subsequently, the requirements of our secure solutions are defined. Finally, we discuss the related work on the existing secure routing protocols and mechanisms.

VII.1 Motivations

In P2P SIP overlay, instead of servers, a distributed hash table can be used for registering and locating a user-ID. However, the lack of a central authority (proxy or registrar server) makes authentication of users and nodes difficult to do. Without trusted node that certifies users identities, malicious nodes can control a large fraction of a distributed system. Regarding our MISE-P2PSIP architecture, because of our requirement on independence from specific P2P middleware, many functions like node registration, location lookup, message routing are performed at SIP level. This leads to the assignment of many important duties to the proxy servers. However, a single proxy server can misbehave or many proxy servers can also cooperate to misbehave. For instance, a misbehavior proxy can drop, wrongly alter, delay or misroute the SIP messages. Then, nodes acting as servers need to be trusted in SIP environment. There is a need to secure the routing of SIP signaling messages in such a way that those messages are treated and delivered in right way because firstly, they convey sensitive information like caller and callee addresses, Call ID, etc, that needs to be protected. Moreover, SIP signaling messages should be delivered in real-time in order to provide acceptable quality of the call. SIP messages misrouting in general can lead to service disruption in P2P SIP overlay. In this framework, we focus on attacks performed by the proxy server such as signaling messages misrouting and call hijacking and propose secure schemes to mitigate those attacks.

VII.2 Security requirements

In addition to the overall requirements of MISE-P2PSIP architecture defined in chapter III, there are specific requirements to ensure a secure routing of SIP messages. These are the following:

- The proxy has to send the message to its right destination.

- The proxy could not misroute the SIP message in order to increase the transmission delay or to prevent the message to reach its destination.
- The proxy could not modify the SIP message beyond the SIP specification.

VII.3 Related work on secure routing protocols and mechanisms

This section will describe some of the existing works on secure routing protocols and mechanisms. We could classify those works in three groups:

- Secure routing in mobile Ad hoc networks.
- Security in conventional SIP.
- Secure routing in Peer-to-Peer systems.

VII.3.1 Secure routing in mobile Ad hoc networks

Several research activities have been undertaken to secure routing in Mobile Ad hoc NETWORKS (MANET). In general, existing strategies focus on securing routing by detecting misbehaving nodes or by making use of cryptographic algorithms. In the next sub-sections, we will discuss only a few part of the existing secure routing protocols in this area.

a) Detecting misbehaving nodes

i. Reputation and Trust based Models

The existing works in this category are based on reputation and trust based models in order to detect and isolate a misbehaving nodes in mobile Ad hoc Networks.

Several Reputation and Trust based Models (RTMs) for mobile networks have been proposed over the years. These models aim to provide information that allows nodes to distinguish between trustworthy and untrustworthy nodes and encourage nodes to be trustworthy. Malicious nodes are isolated, denied service and punished in some of these models [81, 82].

A Collaborative REputation (CORE) mechanism to enforce node cooperation in mobile ad-hoc networks model [86] was proposed by Michiardi and Molva. Reputation in this model is formed and updated with time by direct observations and information provided by other members of the network. Nodes have to contribute continuously to the community to remain trusted. Otherwise, their reputation will be degraded until they are eventually excluded from the network. Only positive information is shared and consequently, CORE prevents the distribution of false information about other entities.

Another scheme called the Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks (CONFIDANT) has been proposed by Buchegger *et al* [83]. The protocol aims at detecting and isolating misbehaving nodes, making it unattractive to any node to deny cooperation. In this model, each node maintains a reputation rating and a trust rating about every other node of interest. Only fresh reputation is propagated, with more weight given to current behavior of a node than the past. This prevents the possibility of a node from obtaining good reputation initially and subsequently misbehaving. Nodes monitor and detect misbehavior in their neighbourhood by means of an enhanced packet acknowledgment (PACK) mechanism where confirmation of acknowledgment comes indirectly by overhearing the next node forward the packet [84, 85].

Marti et al. [87] proposed a watchdog and pathrater schemes to improve the throughput of an ad hoc network in the presence of misbehaving nodes. Watchdog keeps track of misbehaving nodes. Pathrater avoids routing through those misbehaving nodes.

Awerbuch et al. [88] proposed a fault detection scheme to detect malicious links on a route between a source and a destination. The scheme is based on acknowledgements from some probe nodes on the route, which are specified by the source node. If the number of acknowledgement loss exceeds a particular threshold, a faulty link is considered to exist in the route. Then, a binary search can detect the faulty link.

ii. Sending packets via trustworthy routes

Yi et al. [89] developed a Secure Aware Routing (SAR) protocol for ad hoc networks, which extended the Ad hoc On-demand Distance Vector (AODV) routing protocol. In their protocol, the nodes in an ad hoc network have different security attributes and are classified into different trust levels. The trust level can be decided by an internal hierarchy of privileges in an organization. The nodes of the same trust level share a secret key. When a source constructs a route discovery message, it also specifies the required security level for the route. The route discovery message can also be encrypted by using the secret key shared by nodes of same trust level. Only the intermediate nodes that satisfy the required security level can process the message since only these nodes can decrypt the message. The other nodes just drop it. This protocol provides some protection to routing messages.

Papagiorgis and Haas [90] proposed a Secure Routing Protocol (SRP) for ad hoc networks. The assumption of SRP is the existence of a “Security Association” between a source node and a destination node, through which the source node and the destination node can authenticate each other. SRP is based on source routing. The source node broadcasts a route request to discover a route to the destination node. When an intermediate node receives the route request, it appends its identifier in the request packet and relays the request. When the destination node receives the route request, a route has been set up and carried in the route request. The destination node generates a route reply containing the route and sends it back to the source node along the reverse of the route. The most important secure measure used in SRP is called Message Authentication Code (MAC) and is calculated by using the shared secret key between the two ends. Both the unchanged fields of route request and the route reply are covered by a MAC so that modification and IP spoofing from non-colluding attackers can be prevented during the process of route discovery.

b) Public Key Cryptography

Venkatraman and Agrawal [91] proposed a protocol based on public key cryptography. They assume the existence of a governing authority for the distribution of public keys. A source node generates a route request and digitally signs it using its private key. When a destination node sends a route reply back to the source node, public key cryptography is used for pair-wise authentication to exclude malicious nodes. If a node does not know a forwarding node’s public key, they have to exchange public keys first. This pair-wise authentication is done by challenge and response process. The purpose of this protocol is to prevent external attacks.

A different approach called, Authenticated Routing for Ad hoc Networks (ARAN), was developed by Dahill et al. [92]. ARAN relies on public key cryptography for authentication. Authors assume that each node has a public/private key pair, and that there exists a trusted certificate server to issue a certificate to each node. The goal is to allow a source node to set up a route to a destination node. The source node broadcasts a route discovery packet, containing its certificate

and digitally signs it by using its private key. Upon receiving the packet, an intermediary node verifies the signature with the attached certificate. The node then removes the signature of the broadcasting node, signs the packet with its private key, attaches its certificate, and re-broadcasts the packet. Eventually, the destination node receives the packet and validates the signatures of the source node and the forwarding node with their certificates. The signature protects the routing information and prevents spoofing attacks. The destination node constructs a reply, signs it and unicasts the reply back to the source over the reverse path. The same process is applied by the intermediate nodes. Finally, the source node can receive the reply.

c) Analysis

The general purpose of securing ad hoc routing protocols is to protect the routing messages, to prevent attackers from modifying these messages or even injecting harmful routing messages into the network. So, integrity and authenticity of routing messages should be guaranteed. Confidentiality can be ensured easily by encryption. Therefore, those protocols cannot prevent a malicious node to misbehave for instance by misrouting the message. In addition, most of these secure routing mechanisms are designed for a specific routing protocol in ad hoc network.

VII.3.2 Security in conventional SIP

Privacy and security are mandatory requirements for any telephony system. SIP standard defines security mechanisms to be used for protecting SIP signaling messages and media transactions. We highlight in this sub-section the most important existing SIP mechanisms that provide authentication and security features for SIP signaling messages. These secure protocols comprise PGP (Pretty Good Privacy), S-MIME, IPsec, TLS and SIPS [97].

a) Authentication

SIP provides a stateless, challenge-based mechanism for authentication that is based on authentication in HTTP [93]. HTTP provides a simple challenge-response authentication mechanism that may be used by a server to challenge a client request and by a client to provide authentication information. Authentication is needed to prevent against several kinds of attacks, such as registration hijacking, call hijacking, and impersonation. It uses an extensible, case-insensitive token to identify the authentication scheme, followed by a comma-separated list of attribute-value pairs which carry the parameters necessary for achieving authentication via that scheme. The HTTP Authentication framework includes two authentication schemes: “Basic” and “Digest”. Both schemes employ a shared secret based mechanism for access authentication. When UA sends a request message to SIP server, the server answers with “401 Unauthorized” or “407 Proxy Authentication Required” asking for user’s credential. The “401 Unauthorized” header is used in SIP registrar or redirect server, and the “407 Proxy Authentication Required” is used in SIP proxy. In addition, to prevent replay attack, SIP server challenges the UA with a unique random nonce. This nonce is used only one time. Digest authentication also provides an optional integrity protection using an “auth-int” parameter. When the “auth-int” parameter is indicated, the entire body is hashed and attached to authentication messages.

Besides basic and digest authentication, SIP can also use digital certificate to authenticate SIP servers. For example, if TLS is used, UA can receive a server’s certificate during the handshake protocol and check whether the certificate is signed by a known authority. If the UA trusts the authority, the connection is then authenticated.

Note that due to its weak security, the usage of “Basic” authentication has been deprecated from RFC 2543. However, it is possible to use basic authentication with secure transport-layer or network layer tunnel such as TLS and IPsec.

b) SIP Signaling Security

i. Pretty Good Privacy

Pretty Good Privacy (PGP) [94] is proposed by Phil Zimmerman. Initially, PGP is designed to protect email privacy but it can also be used for other applications like SIP. PGP offers encryption, authentication, and integrity for SIP signaling messages using both symmetric and asymmetric encryption. Each PGP user requires a public/private key pair as well as a certificate. This certificate is signed by other PGP user which is called introducer of that key. PGP uses a concept of “web of trust” as an alternative to Public Key Infrastructure (PKI). If a user trusts the introducer, the certificate is verified. In PGP, before submitting a SIP message to a SIP server, UA signs the message with user’s private key to provide integrity. Moreover, PGP also provides an encryption function to conceal SIP message body with a generated session key. The session key is then encrypted with the recipient’s public key and attached to an encrypted message body. PGP has been replaced by S/MIME in SIP RFC 3261.

ii. Secure/Multipurpose Internet Mail Extension (S/MIME)

S/MIME [95] is designed as a standard for securing an email message in MIME (Multipurpose Internet Mail Extensions) format and later adopted for other applications such as SIP. RFC 3261 recommends S/MIME for securing SIP messages. S/MIME is based on RSA (Rivest-Shamir-Adleman) public key cryptography. It requires RSA for digital signature, SHA-1 for message digest, and AES (Advanced Encryption Standard) for symmetric encryption. In RFC 3261, the mandatory cipher suite requirement was Triple-DES. However, it has been changed to AES in RFC 3853 since it is faster and more efficient. Each participant in the system, needs to have a public/private key pair and a certificate which is signed by a trusted authority. The certificate is then distributed to other parties. Nonetheless, SIP does not specify how to manage public/private key and certificate. S/MIME provides integrity to SIP message. Before sending a message to a recipient, UA can sign the entire message using its private key and set the SIP “Content-Type” header to “multipart/signed”. Once the recipient UA receives a packet, it uses the sender’s public key to determine whether the message comes from the real originator. S/MIME also provides confidentiality to SDP and message body. UA can use the recipient’s public key to encrypt the SIP message body and set the “Content-Type” header to “application/pkcs7-mime”. This cipher message can only be decrypted using the recipient’s private key. In other words, only the recipient who possesses a legitimate key can have knowledge of the information.

iii. IP Security (IPsec)

IPsec acts at network layer by offering confidentiality, authentication, and integrity [96]. IPsec consists of three main components that are Authentication Header (AH) [100], Encapsulation Security Protocol (ESP) [101], and Internet Key Exchange Protocol (IKE) [102]. IPsec has two different modes of operation: transport and tunnel modes.

In AH service, an AH header of entire packet (i.e. all header fields and payload) will be generated and attached to every packet for source authentication and message integrity check. AH defines a set of cryptographic algorithms that can be used for computing an authentication value, such as Message Authentication Code (MAC) or one-way hash function like MD5, SHA-1,

and SHA-256. Besides the authentication value, the AH header also consists of a sequence number and index for recipient to determine which algorithm has been used. In general, AH is not widely deployed as it does not offer any confidentiality mechanism. Unlike, ESP covers all the services that AH has and also provides confidentiality of IP packets. In transport mode, only the payload will be protected and an ESP header will be interposed between an IP header and the encrypted payload. Meanwhile, in tunnel mode, the entire packet (i.e. IP header and payload) is encrypted. Thus, tunnel mode ESP provides packet confidentiality as well as protects against traffic analysis. Instead, Internet Key Exchange Protocol (IKE), is a key protocol in the IPsec architecture, that is a hybrid protocol using part of Oakley [103] and part of SKEME (Secure Key Exchange MEchanism for Internet) [105] in conjunction with ISAKMP (Internet Security Association and Key Management Protocol) [104] to obtain authenticated keying material for use with ISAKMP. ISAKMP provides a framework for authentication and key exchange but does not define them. ISAKMP is designed to be key exchange independent, which supports many different key exchanges. IKE's processes can be used for negotiating virtual private networks (VPNs). IPsec can be used to protect both SIP signaling and media stream. However, the protocol is not aware by the higher layer protocol. In other word, it is difficult for SIP application to detect whether the IPsec is operating or not, in contrast to TLS which is visible to SIP applications.

iv. Transport Layer Security (TLS)

TLS supports hop-by-hop security for SIP signaling messages [98]. It provides the ability to perform mutual authentication (client and server), confidentiality and integrity. The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol.

The TLS Record Protocol aims to maintain a secure connection between two endpoints (for example, client and server). The negotiation of the cryptographic properties (for example, cipher suites, encryption keys) for the corresponding connection is performed by the TLS Handshake Protocol, which is encapsulated within the TLS Record Protocol.

The TLS Handshake Protocol is used for mutual client/server authentication and to negotiate cryptographic properties (for example, encryption algorithms and keys) of the respective session. The TLS Handshake has to be completed successfully before transmitting any data.

TLS is designed to be used over a reliable transport such as TCP or SCTP [99] (Stream Control Transmission Protocol). This introduces a limitation for implementations that use UDP as their transport protocol because TLS cannot be used with UDP to protect SIP messages. IETF has published RFC 4347, "Datagram Transport Layer Security," to address this limitation. The RFC recommends the use of TLS to provide the necessary protection against attacks such as eavesdropping, message tampering, message replay, and so on.

Although TLS provides confidentiality between two endpoints (client/server relationship), it does not support direct end-to-end confidentiality between two users that are connected through intermediate SIP proxies. For each segment, a distinct TLS connection has to be established.

v. Secure SIP URI

RFC 3261 defines a secure SIP URI Scheme (SIPS) which offers end-to-end session confidentiality using TLS. When SIPS is used, TLS connections are established between two nodes at every hop along the routing path. Then, users can maintain a level of privacy, by using SIPS URI (SIP over TLS) to guarantee that secure, encrypted transport is used to protect the signaling messages between the two communication parties [98].

The SIPS message is similar to a SIP (unencrypted) message that is transported over UDP, TCP, or STCP. The major differences are as follows:

- The URI syntax is defined as sips:alice@domain-b.com.
- The transport is TLS, instead of UDP or TCP.
- The SIPS port is 5061, instead of 5060, which is reserved for UDP and TCP.

When SIPS is used, all SIP messages are transported over TLS, which provides an adequate level of protection against attacks such as eavesdropping, replay, and message manipulation. In addition, TLS provides the means for mutual authentication using certificates to protect against “man-in-the-middle” attack. The device can authenticate itself to the network, but it can also verify the authenticity of the SIP proxy (or SIP registrar).

The recommended cipher suite to be used with SIPS is AES, using a 128-bit key in CBC (Cipher Block Chaining) mode, and the message authentication code is SHA-1 to provide integrity.

Another added benefit of using SIPS is the ability to exchange encryption keys to encrypt the media stream using SRTP (Secure Real Time Protocol). For example, SDDescriptions can be used within a SIPS INVITE message to exchange the master key between two participants. The encryption key is provided in the SDP portion of the SIPS INVITE in the a=crypto attribute.

c) **Analysis**

Many efficient schemes have already been provided to secure SIP signaling messages. Those protocols provide confidentiality, integrity and authentication between participants but are not suitable to MISE-P2PSIP networks as they are currently defined. Indeed, the SIP Digest authentication mechanism cannot prevent proxies in MISE-P2PSIP to misroute the SIP signaling message or to perform call hijacking. Moreover, if SIP Digest authentication mechanism is used between proxies in our framework, it will generate too much traffic in the network. Also, the secure protocols using encryption (TLS, IPsec, etc) are not sufficient to avoid call hijacking attack. Therefore, our framework needs an appropriate combination of authentication mechanism, encryption and signature techniques to alleviate misrouting and call hijacking attacks.

VII.3.3 **Secure routing in Peer-to-Peer systems**

Many attacks can be targeted to DHT in structured P2P network, including routing attacks [106]. Some solutions have been proposed to handle secure routing issue in P2P systems. This sub-section summarizes some of them.

a) **Secure messages routing**

Existing works on secure message routing focused mainly on employing redundancy to increase the probability of a message being delivered successfully. The goal is to maximize the probability that at least one copy of the correct message reaches target node.

Wallach [108] presents a multiple redundant routing algorithms for Pastry. An enquirer sends a query to all of its neighbors in Pastry overlay. Then, each neighbor forwards the query towards the target node. If at least one copy of the query arrives, it is considered successfully delivered. However, the technique used will inevitably cause congestion and burden in the system, especially in a bandwidth sensitive system.

Artigas et al. in [109] had a similar idea. It provides disjoint paths between two nodes in DHTs. However, the routing success rate degrades greatly as the number of nodes in the system reaches a big value.

Miguel Castro et al. [107] consider security issues in structured P2P overlay networks. They identify attacks in which malicious nodes can prevent correct delivery of messages (by dropping, corrupting, misrouting messages or acting as its destination), propose and evaluate techniques to prevent such attacks. According to their approach “secure routing” requires tree main principles:

- A secure assignment of node identifiers.
- Secure routing table maintenance.
- Secure message forwarding.

Secure assignment of node identifiers and secure routing maintenance can be achieved by minimizing the probability that nodes are controlled by attackers. In order to provide secure message forwarding, their approach in [107] allows to route a message by performing a “failure test” to determine if routing worked. More expensive “redundant” routing is used when the failure test returns positive.

b) **Analysis**

The main issue of the proposed techniques to date to secure messages routing in structured P2P networks is the usage of many redundant routes that obviously cause congestion and burden in the systems, especially by increasing bandwidth consumption. Moreover, the solution does not prevent a malicious node to perform call hijacking attack by replaying the message. Furthermore, an adversary can prevent correct message from being delivered throughout the overlay. When one or more nodes between originator and target nodes are malicious, a message might be dropped, polluted or forwarded to the wrong place.

VII.4 Conclusion

This chapter provides the related work on the existing routing mechanisms and protocols for security purposes and reviews them in light of our requirements. The next chapter will mainly present our secure schemes for signaling messages routing in MISE-P2PSIP.

Chapter VIII

Attack models and secure schemes for signaling messages routing in MISE-P2PSIP

Our P2P SIP framework as per its topology setting up and call establishment scheme could be opened to many attacks. The Internet Draft at [110] defines a numerous number of vulnerabilities and attacks which can be targeted to P2P SIP network. Among those attacks, our proxy-level routing scheme could be essentially exposed to two kinds of attack that we will address in this chapter. In this work, we are more interesting on attacks generated by proxy servers in MISE-P2PSIP network since they are in the core of the communication and are more susceptible to misbehave. Basically, we will focus on the following two attacks.

1. Call hijacking,
2. Signaling message misrouting.

We will focus on the former first, and then the latter.

VIII.1 Call hijacking

This section describes the approach that we propose to alleviate call hijacking in MISE-P2PSIP. Our technique is based on Identity-based encryption (IBE) to perform part of SIP message encryption and signature during the call establishment in order to ensure messages integrity and authenticity in addition to mutual authentication of the two communication parties. This scheme efficiently mitigates call hijacking attack and replay attack by making use of one-time timestamp. Firstly, we give a short background on call hijacking attack and IBE technique. Next, we describe the usage of digest authentication mechanism among proxy servers by highlighting the limits of such technique. Finally, we describe our secure mechanism to mitigate this attack.

VIII.1.1 Background

a) Call hijacking attack definition

In VoIP, call hijacking attack refers to a situation where one of the intended endpoints of the conversation is exchanged with the attacker. Basically, attacker spoofs a SIP Response redirect-

ing the caller to a rogue SIP address and intercepts the call. Call hijacking can have common consequences with eavesdropping attacks (access to confidential information).

Different types of call hijacking attacks can be mentioned [111].

- Registration hijacking: Since SIP uses clear text, a hacker is able to read legitimate User Agent's registration message and then "replay" the same message using its own location information. This effectively registers the UA with the hacker's location.
- Session hijacking: This attack works much like registration hijacking, but is used to take over a session in progress.
- Requests redirection: The attack uses the 3xx SIP response codes which allow redirecting. These codes inform the requester that further actions have to be undertaken in order to successfully accomplish the request. The 3xx SIP based attacks rely on forged responses.
- Requests replay: Also known as replay attack, it occurs when a valid message sent by a node is captured and retransmitted for malicious purposes. A badly behaving node could replace any newer data with the old information present in this message and resend it in the network.

Let us give an example of classical call hijacking attack using requests redirection. A graphical illustration of the attack is given in figure VIII.1. The attack outline is the following:

- The victim issues a SIP request (an INVITE request for example) (step 1).
- The attacker sends a 3xx code response to the initiator (step 2). The attacker usurps the identity of either the callee UA or one of the SIP components (proxy, registrar, etc).
- The victim SIP client receives the forged 3xx response and redirects its communication through the attacker's system (step 3). The attacker then sends the request to the destination by impersonating the victim (step 4). The attack is then completed.

Basically, the attacker sends a 301 response (moved permanently response code) upon detecting that the victim has issued an INVITE request. The victim in turn contacts the attacker for achieving the SIP connection. Alternatively, for one-time attacks, 302 Moved Temporarily response code is used instead of 301.

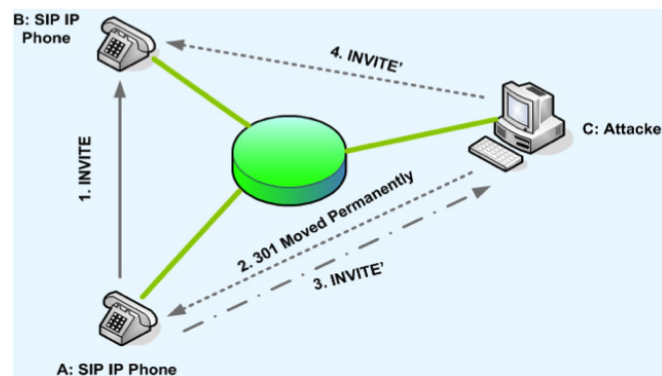


Figure VIII.1: Call hijacking attacks based on requests redirection

Attacks in call hijacking category, seek to compromise the message integrity of the conversation. To handle call hijacking, it is necessary to set up an efficient authentication mechanism in order to always verify that the sender and/or the receiver are legitimate and that the contents of the SIP message have not been wrongly altered in transit.

b) Background on Identity Based Encryption scheme

In 1984, Adi Shamir [112] has proposed the concept of identity-based cryptography. In this new paradigm of cryptography, instead of using digital certificates, the user identifier such as email or IP address can be used as a public key for encryption or signature verification. The identity-based cryptography reduces mostly the complexity and the cost to establish and manage the public keys known as Public Key Infrastructure (PKI).

While Shamir easily constructed the pattern of Identity-Based Signature (IBS) using the existing RSA (Rivest-Shamir-Adleman) algorithm, Boneh and Franklin [113] and Cocks [114] have built an encryption scheme based on identity. This has been done in 2001. IBE scheme is shown in VIII.2. This scheme follows the following steps:

- **The initialization:** the PKG (Private Key Generator) creates a key pair: master (private) and public named respectively sk_{PKG} and pk_{PKG} . The pk_{PKG} is provided to all interested parties and used as a constant parameter of the system for a long time.
- **The private key extraction:** Bob authenticates to the PKG and obtains the private key $sk_{ID_{Bob}}$ associated with its identity ID_{Bob} .
- **Encryption:** using the identity of Bob ID_{Bob} and the pk_{PKG} obtained from the PKG, Alice encrypts the plaintext message M and obtains the encrypted message C .
- **Decryption:** Upon receiving the message C , encrypted by Alice, Bob decrypts it, using its private key $sk_{ID_{Bob}}$ to restore the message M .

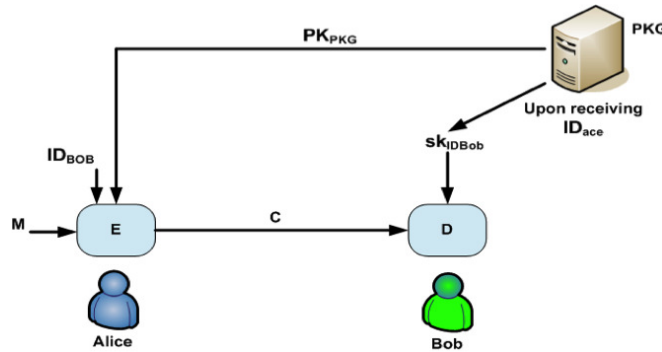


Figure VIII.2: Encryption based on Identity

The mathematical basis of the IBE is a special type of function called “bilinear map” [115]. The “bilinear map” is a pairing with the following property: $\text{Pair}(a.X, b.Y) = \text{Pair}(b.X, a.Y)$.

The operator “.” is the multiplication of a point on an elliptic curve by an integer. Therefore, the multiplication itself (e.g. the calculation of $a.X$) is simple, but the reverse operation which consists on finding “ a ” knowing X and $a.X$, is practically impossible.

Historically, bilinear mapping was first used to attack elliptic curve cryptosystems on super-singular curves in the early 1990’s [116]. However, in the recent application of bilinear maps to cryptography, they are used not for negative purposes (i.e., attacking cryptographic schemes) but for positive purposes (i.e., designing cryptographic schemes). Two examples of bilinear maps are the Weil Pairing and the Tate Pairing [117].

VIII.1.2 Using HTTP Digest authentication

While HTTP digest is more suitable to centralized SIP architecture, it can in principle be used between the proxies participating in the overlay, to authenticate the proxy servers in MISE-P2PSIP. HTTP digest authentication can also be used to provide authentication between two SIP proxies. Therefore, since proxies attending in MISE-P2PSIP are organized in ring, it is necessary that each proxy authenticates itself to its neighbors. More in detail, a joining node could get its credentials (e.g. password) from a Certificate Authority (CA) which could coincide with the Domain Controller in our framework. Subsequently, the node could send a challenge to its neighbors requesting authentication. After its neighbors answer successfully to the challenge, each of them will require authentication from the joining node. This scenario is repeated at every node joining. However, this approach could generate much traffic in the network. In addition, the authentication mechanism used is Digest Authentication which is known to be weak and prone to man-in-the-middle attacks. Hence, we propose a more efficient authentication mechanism suitable to MISE-P2PSIP network topology.

VIII.1.3 Our Secure scheme against call hijacking attack

In order to efficiently counter call hijacking attacks in MISE-P2PSIP, we propose an authentication mechanism based on Identity Based Encryption system during SIP call establishment. Identity-based authentication scheme for MISE-P2PSIP has the advantages of public key cryptography, provides mutual authentication without pre-shared secret and does not require PKI support.

Our secure scheme against call hijacking uses specially three (3) main concepts: Identity-based encryption, digital signature and one-time timestamp.

The identity-based encryption ensures user authenticity. The digital signature warrants message integrity while the one-time timestamp avoids message replay by malicious nodes.

We assume that the DC which is Central Authority, is trusted by all nodes and cannot misbehave. Let us assume that SIP Client Alice, located in the group of proxy P1, wants to initiate a call toward SIP Client Bob located in the group of proxy P3. The call will occur through proxies P1, P2, and P3. The authentication scheme follows three phases that are: initialization, encryption/signature and decryption/verification. The details of each phases is given below.

i. Initialization

The Domain Controller plays the role of PKG. Then, the DC takes a secret (the master) sk and a point P of an elliptic curve using a random number generator. Subsequently, the public parameters P and $s.P$ (product of $s = sk$ and P) are distributed to all users, typically through a certificate server. Each node has two basic keys and one-time special key. We denote the basic keys, the identity-based public key (Pub_{kx}) and the identity-based private key ($Priv_{kx}$). The special key is named one-time stamped private key ($Priv_{tskx}$). The identity-based public key is obtained by applying a hash function H to the node ID and can be computed by every node. Then, $Pub_{kx} = H(ID_x)$ is the public key of node x .

$Priv_{kx}$ and $Priv_{tskx}$ are calculated by the DC and are known only by their owners.

$Priv_{kx} = s.ID_x$ and $Priv_{tskx} = s.(ID_x \oplus \text{timestamp})$

The parameter " $ID_x \oplus \text{timestamp}$ " related to node X is therefore the concatenation of the ID of X and the timestamp. A given timestamp is related to a single SIP request/response between

two nodes in only one direction and then is unique. The timestamp is requested by the sender of the SIP request/response.

ii. Encryption and signature

The principle is that before sending an INVITE request to Bob, Alice requests the so called Bob's stamped ID (ID_{tsbob}) from the DC. ID_{tsbob} is ID_{bob} (Bob's SIP URI) concatenated with the timestamp.

$$ID_{tsbob} = ID_{bob} \oplus \text{timestamp}_{A \rightarrow B}$$

The DC computes the ID_{tsbob} and sends it to Alice. The DC keeps a copy of Bob's stamped ID and will use it later to calculate Bob's stamped private key, when Bob will request it. Upon receiving the ID_{tsbob} , Alice will create a regular SIP INVITE. Subsequently, she will sign and encrypt the body of the INVITE which contains important information about session negotiation, before sending it. Figure VIII.3 depicts the steps of encryption and signature procedure.

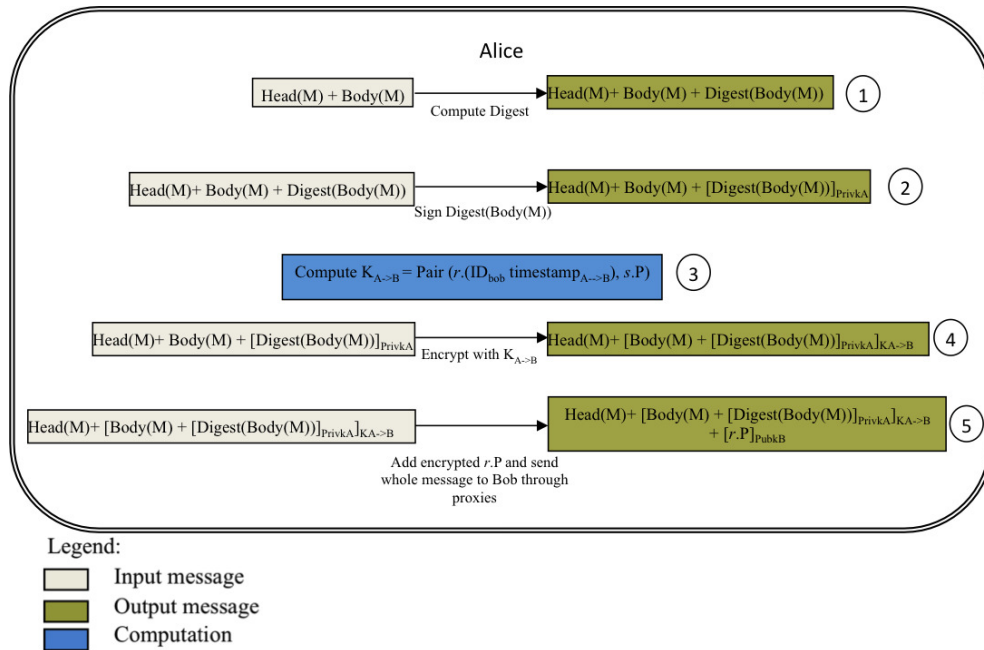


Figure VIII.3: Identity Based Encryption/Signature scheme at Alice side

The procedure is run in 4 steps:

- Alice calculates the digest of the body of the INVITE request: let us denote $Head(M)$ the header of the request and $Body(M)$ the body of the request.
 $Digest(Body(M)) = H(Body(M))$, meaning the hash of the message body.
- Alice signs the obtained $Digest(Body(M))$ with its identity-based private key $Priv_{kA}$. This signature will allow controlling the integrity of the message body.
- Alice will now apply a hash function on the stamped identity of Bob ID_{tsbob} (previously obtained from the DC) to get a point of the elliptic curve ID_{tsbob} . Then, she will choose a random number r and computes the key $K_{A \rightarrow B} = \text{Pair}(r.ID_{tsbob}, s.P) = \text{Pair}(r.(ID_{bob} \oplus \text{timestamp}_{A \rightarrow B}), s.P)$. The $K_{A \rightarrow B}$ is then used by Alice to encrypt the body of the INVITE message added to its digest.

- Next, Alice adds to the encrypted body, the product $r.P$ that she encrypts with Bob's public key (Pub_{kB}), and sends the new INVITE message to Bob through the proxies on the route.

Then, the final INVITE sent by Alice to Bob has the following structure:

$$\text{Head}(M) + [\text{Body}(M) + [\text{Digest}(\text{Body}(M))]_{Priv_{kA}}]_{KA \rightarrow B} + [r.P]_{Pub_{kB}}$$

Since we are encrypting part of the SIP message we will add the "Encryption" field to the headers as required by SIP specification [1] which mainly contains the encryption mechanism used and its version.

iii. Decryption and verification

When Bob receives the message, it should authenticate itself using Digest Authentication [3] to get its stamped private key from the DC. After Bob successfully authenticates itself, the DC computes $s.ID_{tsbob}$ and returns it to Bob. Note that ID_{tsbob} has been calculated and recorded by the DC upon Alice request. Let us call this value $Priv_{tskB}$. At the same time, Bob will request its identity-based private key $Priv_{kB}$ from the DC. $Priv_{kB} = s.ID_{bob}$. The DC deletes the record of the stamped ID of Bob since it will not be used anymore. The decryption and signature procedure is summarized in figure VIII.4.

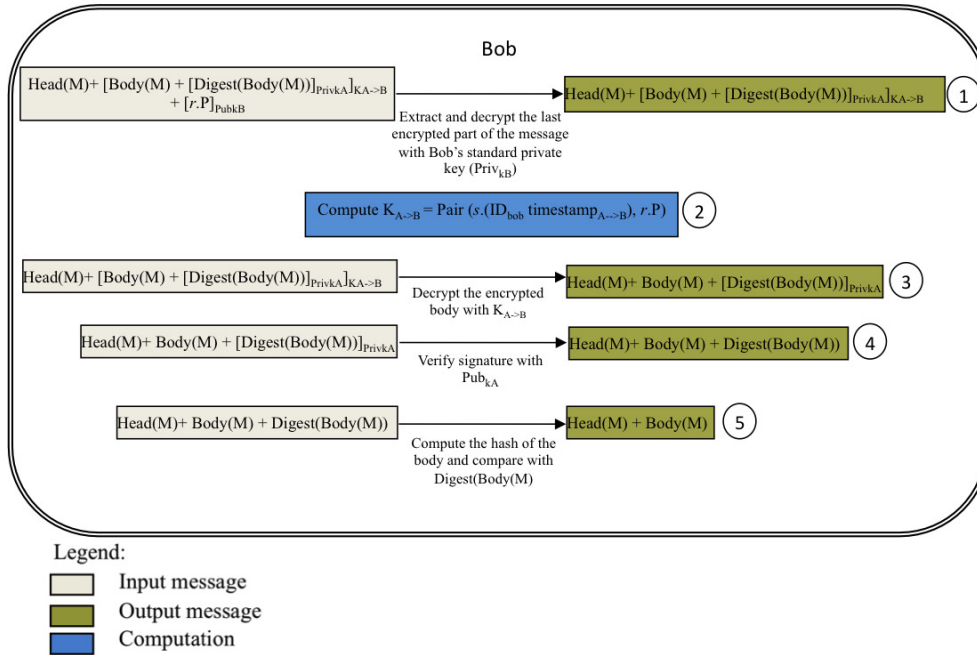


Figure VIII.4: Decryption/verification scheme at Bob side

Bob will follow the steps below in order to decrypt the message and verify the signature:

- Bob will first decrypt the last part of the message with its private key and obtain the parameter $r.P$.
- Next, Bob should decrypt the body of the INVITE but will need first to recover the key $K_{A \rightarrow B}$ by computing $K_{A \rightarrow B} = \text{Pair}(Priv_{tskB}, r.P) = \text{Pair}(s.ID_{tsbob}, r.P)$. Due to the property of "bilinear map", this value is the same as the key used by Alice to encrypt this part of the INVITE. With the computed $K_{A \rightarrow B}$, Bob will decrypt the body of the INVITE message. He will get the clear body and the signed digest body. Since Bob is the only one who knows his own stamped private key, no one else will not be able to calculate $K_{A \rightarrow B}$.

- Then, Bob will verify the signature by decrypting the digest using Alice public key (Pub_{kA}). Alice public key is obtained by computing the hash of the identity in the “From” header of the INVITE message. This step aims to verify that the message is really sent by Alice. At this step Bob gets in addition to the INVITE headers, the body in clear text and the digest body.
- Bob will check whether the body has not been altered by a malicious node. To do this, he computes the hash of the body in clear using the same hash algorithm as Alice and will compare the result to the digest body. If they are the same, the body has not been modified. If Bob decides to accept Alice’s call, he should answer with the OK message. The same encryption/signature procedure like those used by Alice will be used by Bob to encrypt and sign the body of the OK. On Alice side, the same decryption/verification procedure is used to decrypt and verify the integrity of the OK message.

Note that to be fully compliant to SIP specification [1] the node receiving a request (INVITE) or response (OK) containing an encryption part, decrypts the body and then concatenates the plaintext to the request line and headers of the original message. Moreover, the specification requires that if only the body of the message is being encrypted, the body has to be prefixed with CRLF to allow proper concatenation.

iv. Call procedure

Let us assume that Alice wants to establish SIP call toward Bob. SIP messages should go through three proxies (P1, P2 and P3). Figure VIII.5 illustrates a SIP call scenario based on our approach to mitigate call hijacking.

Following is the explanation of the secure call procedure depicted in figure VIII.5.

First of all, Alice and Bob register themselves to their registrar servers. Alice requests Bob’s stamped ID from the DC before creating the INVITE message targeted to Bob. Subsequently, Alice creates the INVITE, following the encryption/signature procedure described in the sub-section (ii) and sends it to Bob through proxies P1, P2 and P3. Once Bob receives the INVITE, he asks the stamped private key from the DC as described in the sub-section (iii). The DC computes this private key by using the timestamp generated when Alice was initiating the call toward Bob. Thus, Bob’s stamped private key could be used only by Bob in order to decrypt the body of the INVITE message. Hence, the proposed solution provides high protection against replay attack, since an attacker can use SIP RE-INVITE to spoof a legitimate user identity. Moreover, Bob verifies the signature of the sender in the body of the INVITE. This step allows authenticating the sender of the message that is Alice.

Afterward, Bob requires Alice’s stamped ID from the DC in order to send an authenticated SIP OK response to Alice. This requires that the DC generates a new timestamp, computes Alice’s stamped ID and sends it to Bob. Then, the mechanism used previously by Alice to process the INVITE request is also used by Bob to process the OK.

Finally, the SIP ACK is simply sent by Alice and the media streams are sent successfully in a secure way between Alice and Bob.

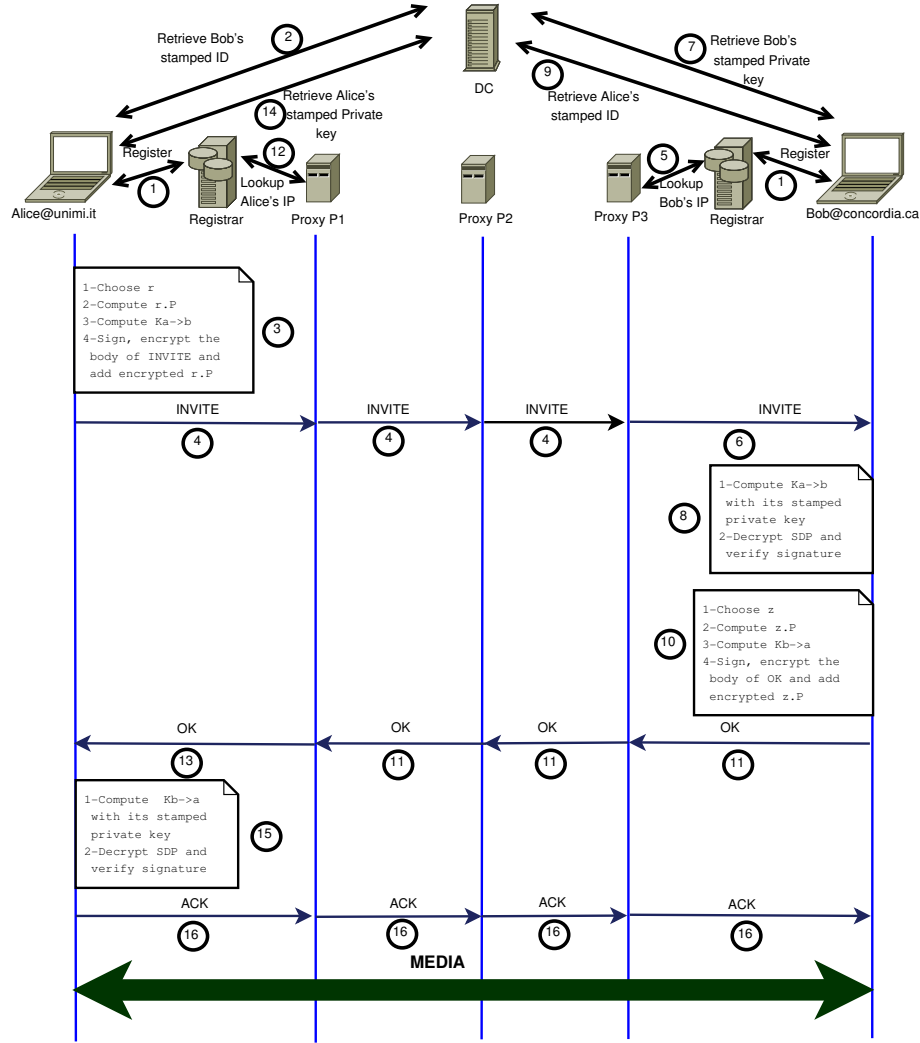


Figure VIII.5: Call procedure against call hijacking

v. Strengths of the proposed approach against call hijacking

Our secure scheme against call hijacking attack comes on with many strengths that will be brought to light in this section.

1. Mutual authentication

A successful processing of the INVITE message by the legitimate receiver, in our approach, gives the proof that the originator of the message is really who it claims to be and that also the receiver is a legitimate node. The same benefits are provided by the OK message. This scheme provides then a mutual authentication from the handling of the INVITE and the OK which are the most important signaling messages.

2. Identity based encryption and SIP message body encryption

Usually, SIP message is forwarded in clear text. Then, anyone having access to the network can get access to the message content by simply capturing the message using a number of sniffing tools. Once a malicious node has captured a message, the message body and the headers in the SIP message can be easily modified.

In our approach, the first part of the body of the SIP message is encrypted using a key

computed partially based on the receiver's identity and the second part is encrypted using the receiver's public key. However, using Identity-based encryption, our scheme provides double level of security and gives thereby, a strong insurance that the body of the message could only be decrypted by the legitimate receiver. On other side, the first advantage of encrypting the body is that a malicious node could not be able to gather confidential information like the IP address of the message originator. Secondly, the body could not be tampered.

For instance, if a rogue proxy receives an INVITE and modifies the FROM header to reflect his or her own SIP URI pretending to be the originator of the message, the receiver will detect it, because the signature inside the body will not match with the public key of the attacker.

3. Usage of one-time timestamp

The usage of timestamp is the main strength of this approach. The timestamp is held by the trusted DC and related data (stamped ID and stamped private key) are only delivered to the legitimate node upon authentication. Moreover, the first part of the encrypted body can only be decrypted by its legitimate receiver because the DC knows to which node belongs a given stamped private key and sends it only to him upon authentication. On the other side, a given timestamp is used one-time for a single SIP message between two communication parties. So, the proposed approach highly mitigates SIP signaling messages replay, since there is no way that a malicious proxy records a previous SIP message, impersonates another node in order to get a successful established SIP session.

VIII.2 Signaling message misrouting

This section focuses on misrouting attack in MISE-P2PSIP. Actually, this attack is related to the topology of our P2P SIP network. Indeed, a proxy can decide to not follow the routing algorithm specification and send the SIP messages in the wrong direction. This behavior can easily lead to SIP signaling message cancellation or delay on the message transmission which is not convenient in SIP communication. Our goal is to detect nodes which misbehave and circumvent them in the network in order to ensure effective delivery of the SIP signaling message in real-time. Before proposing our secure solution regarding misrouting issue, we have performed some simulation tests in order to evaluate essentially the number of messages successful delivered in function of the number of misbehaving nodes and in function of misrouting attack frequency. Firstly, we define how the attack can be performed and discuss the simulation results on misrouting attack tests. Finally, we describe our secure scheme.

VIII.2.1 The misrouting attack

A proxy can decide to misroute the SIP INVITE message received from another proxy or from a client. In our architecture, since the proxy has always two alternatives (e.g. send to right or to left) to route the message, the message misrouting means that its sends the message on left instead of right and vice-versa. Let us consider figure VIII.6 which shows an example of clients and proxies organizing in ring as in MISE-P2PSIP architecture.

Following are examples of two scenario of misrouting attack in MISE-P2PSIP.

Scenario 1: Let us assume proxy P1 receives SIP INVITE message targeted to C2 from P5. Instead to forward the message to P2, P1 decides to send it back to P5. P5 will check its routing

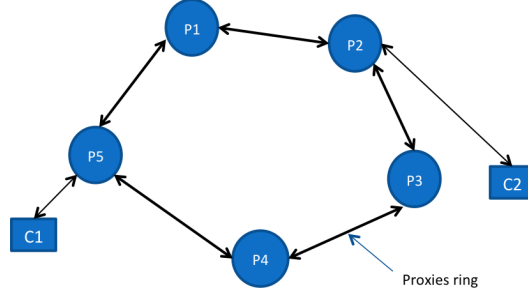


Figure VIII.6: Example of MISE-P2P SIP architecture

table and will send again the message to P1. In this scenario the message will be bounced between the nodes and will not be delivered to C2.

Scenario 2: Let us assume proxy P5 receives SIP INVITE message targeted to C2 from C1 and sends the message to proxy P4 instead of sending the message to proxy P1 (shortest path). P4 will send the message to P3 and the message will be stopped forwarded there ($N/2$ hops). Then, the message will not be delivered to P2.

Many nodes can misbehave in collaborative way or independently. In this thesis, we focus only on many nodes misbehaving independently.

VIII.2.2 Misrouting attack tests

The purpose of simulating misrouting attack tests is to evaluate the robustness of MISE-P2PSIP network in the presence of rogue nodes (malicious proxy). These nodes are characterized by not following the specification of the routing algorithm.

a) Assumptions

The various tests have been conducted using OverSim [15] simulation framework. Indeed, OverSim is a framework for simulating peer-to-peer environment for OMNeT++. The simulator contains several models for structured P2P systems (Chord, Kademlia, Pastry) and unstructured overlay protocols.

Four assumptions (compared to the procedures specified in our architecture) have been made:

- The rogue nodes show no abnormal behavior during the initialization of the ring, while nodes are joining.
- The number of nodes in the ring is fixed, no joins or leaves during the simulation.
- The nodes start to exchange messages only after the ring has been completely initialized.
- The malicious proxy always behaves properly about the messages it receives from clients, it is responsible for.

These assumptions are made to simplify the simulations. We think that the first and second assumptions are a bit strong and should be lifted in future work. Instead, the last two assumptions are less strong and should not decrease the reliability of the obtained results.

b) **How the attack is simulated?**

Basically, the attack consists in the modification of the routing algorithm by one or more proxies that take part of the ring in the same SIP domain. Being in a ring, the only wrong decision that the proxy could take is not forwarding a message to the side (logical) opposed to the receiving side. Rather, the rogue proxy bounces the message to the same side from which it was received.

Each test simulated 1000 seconds of activity in the ring, in which each node, every 10 seconds sends a SIP INVITE to a recipient chosen randomly among a list proxy participating to the ring. The structure of the message is very simple with only some of the basic SIP headers: From: SIP URI (sender), To: SIP URI (recipient), VIA: SIP URI (traversed node), and Call-ID: call identifier. The message is forwarded in the ring as specified by our routing algorithm until it reaches the recipient. When a proxy receives an INVITE message for which it is the recipient, it forwards the message to the target client. The client creates an OK message and sends the message back to its proxy. Instead, when a proxy receives a message for which it is not the recipient, it checks the routing table to know in which direction the message should be sent. However, this simulation proposed two routing policies which have been tested in several rounds:

- Default routing: The proxy reads the field “VIA” of the message, finds out from which direction the message has arrived and forwards it in the opposite direction.
- Table lookup: The proxy reads the “To” field of the INVITE message or the “From” field of the OK and checks the routing table to decide in which direction it has to forward the received message.

In all tests, it has been measured the number of messages sent (not the number of those forwarded) by each node, the number of completed transactions, the number of transactions completed in time and those delayed. Two series of simulations have been performed.

i. First category

In the first category of tests, all nodes have always applied the “default routing”. We have distinguished three possible behaviors of malicious proxy, based on the frequency with which the wrong decisions could be taken:

- Static: node always bounces messages to the direction from which they come;
- Alternating: node bounces towards the direction from which a message is received every two receptions;
- Probabilistic: node bounces messages received with a probability P .

The three analyzed behaviors can be simulated through the third, with appropriate choices of the value of probability P . In fact, putting $P = 1$ we obtain the static behavior, while setting $P = 0.5$, we obtain the alternating behavior. This reduces the first and second types of behavior to the particular case of probabilistic behavior.

The tests were conducted, with different behaviors defined on four rings of different sizes: 20, 60, 120 and 240 proxies. For each different size and for each behavior, the tests have been performed with different number of malicious proxies: 0, 1, 2 and 5 malicious nodes placed at the same distance in the ring. For the probabilistic behavior, the probability used in the tests is $P = 0.4$. In total 48 tests were then performed.

ii. Second category

In the second category, the number of proxies participating to the ring is set to ten (10) for all the tests. In addition, the probability value is fixed at 0.65 and then is a “pure” probabilistic behavior, the only that we consider being very interesting. Four scenarios have been tested, according to the routing policy used during the routing by good and rogue nodes. The scenarios are:

- Scenario A: all nodes apply “Default routing”.
- Scenario B: all nodes apply “Table lookup”.
- Scenario C: good nodes apply “Table lookup” and rogue nodes apply “Default routing”.
- Scenario D: good nodes apply “Default routing” and rogue nodes apply “Table lookup”.

For each scenario, we performed 12 tests with different numbers and arrangements of rogue nodes, for a total of 48 tests. The total number of possible dispositions, changes with respect to the number of rogue nodes. For example, with 2 rogue nodes there are 5 possible dispositions, but we consider sufficient to test 2 of them, while with 3 rogue nodes there are more than 8 possible arrangements and 4 have been tested.

Table VIII.1 displays the dispositions actually tested. The columns are:

- ID: identifies the disposition.
- N° Rogue: indicates the number of rogue nodes.
- Disposition: is a scheme of arrangement where b indicates a good node while r indicates a rogue one.
- Dist. Rogue: is the distance between the rogue nodes, measured in the number of good nodes interposed between two consecutive rogue nodes.

ID	N° Rogue	Dispositions	Dist. Rogue
a	0	bbbbbb bbbbbb	-
c	2	brbbb brbbb	4,4
d	2	brbrb bbbbbb	1,7
e	3	brrbb rbbbbb	0,2,5
f	3	brbrb rbbbbb	1,1,5
g	3	brbrb brbbb	1,2,4
h	3	brbbr brrbb	2,2,3
i	4	brrbr brbbb	0,1,1,4
j	4	brrbb rbrbr	0,2,2,2
k	4	brbrb brbrb	1,2,1,2
l	5	brrrr brbbb	0,0,0,1,4
m	5	brbrb rbrbr	1,1,1,1,1

Table VIII.1: Rogue nodes dispositions

c) Simulation results

i. First category

As it was easy to predict, in the case of static behavior, the consequence of the presence of rogue nodes, is the partitioning of the ring into groups of nodes that cannot in any way communicate with nodes of other groups. The only transactions that are completed are those between nodes belonging to the same group. Those transactions are all completed without delay. The graphic in figure VIII.7, shows the fraction of transactions completed according to the number of malicious nodes in all tested dimensions of the ring.

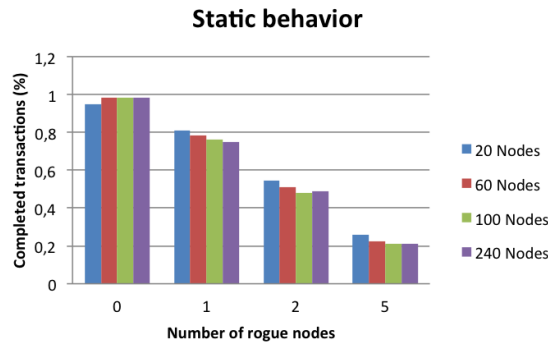


Figure VIII.7: Completed transactions with static behavior

From the graphic, it is clear that positioning rogue nodes at equidistant position, does not affect the results with respect to the size of the ring. The results are very similar with different sizes. It can also be noticed that increasing malicious nodes, decreases the number of completed transactions, with a fairly linear behavior. The graphics of alternating and probabilistic behaviors are not reported here because less significant. Indeed, the fraction of completed transactions and the percentage of transactions completed in time, are very similar regardless the size of the ring. In addition, the linear correlation between the number of completed transactions and the number of rogue nodes is noticed. For this reason, we only present in figure VIII.8, the graphic that shows the delayed transactions (in comparison with the total completed transactions) in the case of probabilistic behavior.

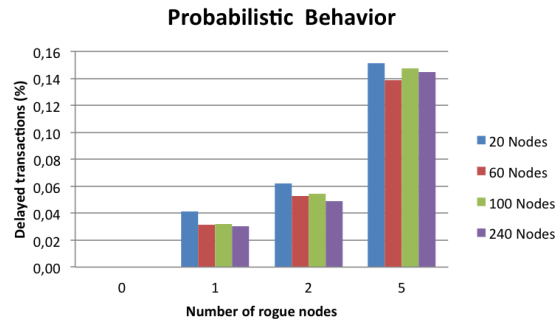


Figure VIII.8: Delayed transactions with probabilistic behavior

ii. Second category

The results obtained in the second category of simulation are certainly more interesting. We mainly focus on the number of completed transactions and the number of transactions delivered

in time. Figures VIII.9, VIII.10, VIII.11 and VIII.12 report respectively the graphics of scenario A, B, C and D.

In scenario A, where “Default routing” policy is used, the equidistant dispositions of rogue nodes as in “c”, “h”, “j” and “m”, provide the worst performance with respect to the other dispositions with equal number of rogue nodes. The fraction of completed transactions falls below 47% with the position “m”. The positive remark comes from the number of delayed transactions. Regardless the disposition of rogue nodes, the delayed transactions are always less than 13% of those completed (up to 87% transactions completed in time).

The results of scenario B clearly show that the “Default routing” policy is not the cheapest. Here, the fraction of completed transactions (compared to those initiated) falls below 70% only with the disposition “l”. Then, in general the performance of the ring is good. But, the dispositions that provide worst results are not only those that set the rogue nodes at the same distance (or almost) but also those that arrange rogue nodes at variable distance like a part “l”, “d”, “e” and “i”. Significant differences are also noticed about the delayed transactions. Delayed transactions are always lower than 26%, but always above 13% (the upper limit of the scenario A). Then, scenario B produces more delayed transactions compared to scenario A.

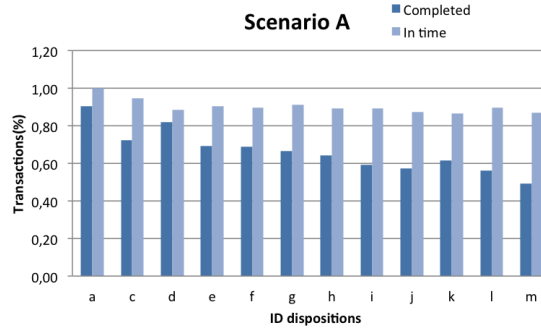


Figure VIII.9: Completed/in time transactions in scenario A

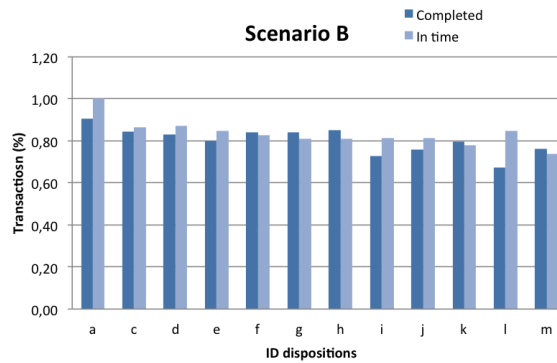


Figure VIII.10: Completed/in time transactions in scenario B

In scenario C, we got the best results. The worst disposition remains “l” with 74% of transactions completed with respect to those started. This is clearly a very positive result. The degradation of performance in this case is very limited (17% less than results obtained in disposition “a”), especially given the fact that 50% of the nodes of the ring is malicious (5 malicious nodes). The percentage of delayed transactions is compatible with those obtained in scenario B.

Finally, considering scenario D, even from a graphical point of view the similarities with the scenario A are obvious. The worst case for completed transactions (47%, provided by “m”), is slightly worse than scenario A (49%). The worst case for transactions concluded without delay (88%, provided by disposition “d” and “f”) is slightly better than scenario A (87%).

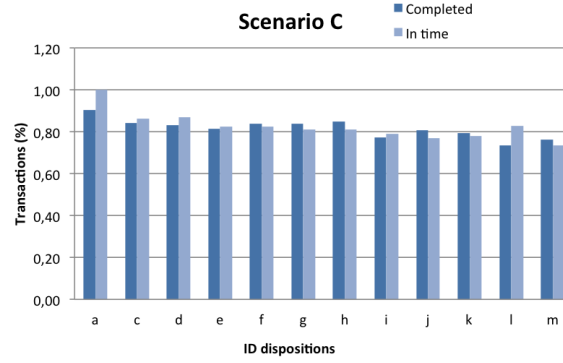


Figure VIII.11: Completed/in time transactions in scenario C

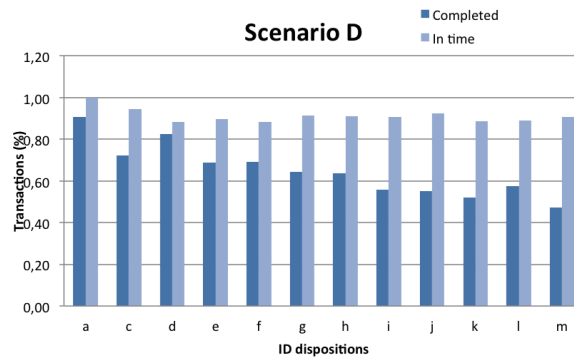


Figure VIII.12: Completed/in time transactions in scenario D

The results obtained after the performed tests on routing attack, clearly show that our P2P SIP framework is not secure with respect to proxy misbehavior. A malicious proxy in the ring can systematically bounce SIP messages or disregards the routing table, partitioning the proxies' ring and preventing SIP calls to be completed. This leads to SIP service disruption and could create severe discomfort to the user in our framework.

VIII.2.3 Our secure routing mechanism

We have performed some misrouting tests whose results show that it is fundamental to deal with this issue to ensure the successful achievement of call sessions in MISE-P2PSIP. To do so, the most efficient solution is to set up a secure mechanism which will detect a misbehaving node in order to prevent further misrouting from this node and exclude it from the proxies ring. Our secure scheme will use two main techniques: the “nonce transformation” usually used to secure route discovery in ad hoc network [118] and usage of “call ID caching” by a given proxy to create the backup of the message identifier.

a) **Nonce transformation**

We assume that the proxy of the source node (client) and the proxy of the destination node do not misbehave. We also assume that rogue proxies do not collaborate to misbehave and the message does not pass through two consecutive rogue proxies. If there are two consecutive rogue nodes the message is unavoidably bounced between them. Our technique works as the following.

When the proxy of the source client receives the INVITE request to transmit, it might append to the request a random nonce N_i . The nonce is piggybacked in the “Encryption” field (mentioned in the previous section about call hijacking) present in the SIP header.

Before forwarding the INVITE to the next hop, an intermediary proxy replaces the nonce N_i by nonce N_{i+1} that is the encrypted value of nonce N_i . As defined by our secure scheme to mitigate call hijacking, each proxy holds an identity-based public key and an identity-based private key. Therefore, the intermediary proxy used its identity-based public key to encrypt the nonce N_i in order to get N_{i+1} ($N_{i+1} = E_{puki}(N_i)$). Then, the proxy makes the backup of N_i and inserts N_{i+1} in the INVITE and sends it to the next hop according to its routing table. Once the proxy of the destination client receives the INVITE, it inserts its nonce and forwards the message. The recipient client processes the message and sends the OK to its proxy. The OK response still contains the last nonce value. The response should take back the same path as the request INVITE. Upon receiving the response, each proxy might decrypt the nonce value receiving in the message using its identity-based private key and replace the old value by the new obtained ($N_i = D_{privki+1}(N_{i+1})$). The new value of the nonce should be equal to the nonce value previously backed up during the INVITE transaction. In this way, the proxy of the source client should find its own nonce value in the OK response upon reception. If the nonce value does not match the previous one or if the decryption is not successfully performed then, the last proxy sending the message has wrongly forwarded the message.

This scheme allows to efficiently detecting the misbehaving proxy that sends the OK message in the wrong direction, at the next hop or at the source proxy. The proposed technique works well regardless the case the proxy forwards the message without putting the nonce. Therefore, it works only if the INVITE message is delivered correctly along the predefined path (according to proxies’ routing tables). Then, we need an extension to be able detecting the misbehaving node during the INVITE message forwarding. To achieve this, we propose a deployment of a call ID caching in the proxy.

b) **Call ID caching**

In order to detect misrouting proxy from the INVITE message, a given proxy needs to know whether a given message already went through itself or not. Then, each proxy needs to cache information that can identify in unique way the INVITE message before inserting a nonce value (as described in the previous section) and forwarding the INVITE. The cache contains the Call ID (that uniquely identify a given invitation of a particular client) and the SIP URI of the initiator of the call. Then, upon receiving an INVITE, an intermediate proxy checks whether the binding of call ID and the originator of the call is already present in its cache. If the binding is found then the message has been misrouted by the last proxy. Notice that based on our assumption on trusting the outbound and inbound proxies, the misrouting attack is summarized to bounce the message to the proxy that sent it.

c) **Denunciation procedure**

Using the techniques, we have described above, both misrouted INVITE and OK messages can be detected. The misbehavior proxy is easily identified and can be excluded from the overlay. The proxy that discovers the misbehaving proxy should send to the DC a SIP NOTIFY that contains in the body of the message, the address of the misbehaving node. The DC will then inform the successor and the predecessor of the misbehaving node, that their neighbor has left, allowing them to establish direct connection (virtual link) between them, bypassing the misbehaving node. This operation will lead to the update of the routing tables of all proxies attending to the ring.

VIII.3 Conclusion

The chapter has proposed secure solutions to mitigate both call hijacking and misrouting in MISE-P2PSIP. The secure scheme against call hijacking, uses the combination of three powerful techniques that are the identity-based encryption, digital signature and the usage of one-time timestamp. The identity-based encryption ensures user authenticity. The digital signature warrants message integrity while the one-time timestamp prevents message replay by malicious nodes. Moreover, we have performed misrouting tests on MISE-P2PSIP in order to test the resistance of the network. Many performance measurements have been taken and analyzed. Finally, we have proposed efficient solution to detect nodes that misroute SIP signaling messages and bypass them in the overlay.

Part V

Conclusions and future works

This part presents the summary of our main contributions and the future works.

Chapter IX

Conclusions and future works

The goal of this thesis was to give useful contributions to SIP-based communications in P2P environment by proposing a secure P2P SIP architecture that is independent from the P2P protocol and does not extend SIP standard. In this conclusion, we provide the summary of our contributions and formulate some relevant future works.

IX.1 Summary of the problems and contributions

The thesis describes P2P SIP-based communication system which enables effective end-to-end communication by being fully compliant to SIP standard and allows portability by being middleware-independent. The proposed system uses its own routing algorithms among its proxy servers and handles NAT traversal issue. In addition, the thesis defines new mechanisms to secure SIP signaling messages routing. The main contributions are fourfold:

- **Interoperable and portable P2P SIP architecture:** A P2P middleware-independent system for SIP-based communication has been proposed. The maintenance of the system is performed by using only standard SIP messages. It thereby overcomes the main drawbacks of existing P2P SIP architectures by providing interoperability and portability. We have provided a prototype upon two different middleware to show how easy the implementation is when no P2P message is involved in the SIP client lookup and the network maintenance. In addition, we have demonstrated through simulations the benefits of relaying SIP messages through proxy instead of using the overlay mechanisms (used in conventional DHT-based P2P SIP architectures). Finally, we have brought out the impact of setting up a caching mechanism in our proxy entity, on the system performance.
- **Topology building and Proxy level routing:** Having SIP proxies as first-class entities raises two important issues in our P2P SIP architecture: proxy topology building and proxy-level routing. Thanks to our proxy topology building, a proxy joining the P2P overlay knows how to find its neighbors in the network of proxies. In addition, our proxy-level routing enables messages to be routed in the SIP network built by proxy topology building. Then, suitable routing algorithms have been defined for our framework. We have evaluated our approach and shown that the well-known routing algorithms are not suitable to be used in our proxies' network to perform routing.
- **NAT traversal solutions in P2P SIP:** Our P2P SIP architecture comes up with several benefits, but also inherits Network Address Translation (NAT) traversal issues from SIP

world. Indeed, SIP clients are not aware of how they are seen from the public network. Consequently, SIP packets sent by a client behind a NAT, contain private IP addresses in the message headers and in the message body. These addresses being private, cannot be used by destination for answering. We have described techniques adapted to our P2P SIP architecture, that provide efficient NAT traversal solutions and have shown the feasibility of our solution through experimentations.

- **Secure SIP signaling message routing:** In P2P SIP overlay, each node acting as a server can misbehave because of the absence of centralized authority. More servers can also collude to misbehave. A malicious proxy node could drop, wrongly alter, delay or misroute a message. It is then important to secure the routing of SIP signaling messages in such a way that they are delivered correctly, because misrouting may disrupt call set up. Our simulations have shown that the system could become useless under misrouting attack. For this reason, we have focused on proposing secure schemes to alleviate misrouting attack and call hijacking.

IX.2 Future directions

The thesis has given several important contributions to P2P SIP research. However, it can be improved and extended along several directions. This section discusses a non-exhaustive list of works that will be the continuation of the thesis.

IX.2.1 Additional experimentations

We plan to provide full implementation of MISE-P2PSIP by also taking into consideration nodes departure. Moreover, we will experiment our NAT traversal solution in P2P SIP environment by considering firewall and cascade layers of NATs. Furthermore, an evaluation of the impact of using encryption/decryption techniques (in the SIP signaling messages) on the Quality of Service of calls could be provided along with full implementation of proposed secure solutions.

IX.2.2 SIP service reliability

The dynamicity of large-scale P2P SIP networks can make SIP service more unreliable than traditional SIP networks. Service nodes are very likely to fail or leave the P2P SIP networks when they are offering SIP service. The departure or breakdown of the service node has side effect on P2P SIP networks, corrupting the reliability of SIP service. Even if some procedures have been proposed to allow SIP services recovery from failure in MISE-P2PSIP, they do not handle the case when failure occurs during SIP call establishment. A typical example is when a Proxy server (in the transaction path) leaves or fails after caller and callee have exchanged a SIP INVITE message. This could lead to three issues: first, the downstream SIP elements (SIP proxies and client) cannot release the memory of the corresponding SIP transactions; second, the SIP transactions information is lost, and third, the SIP signaling cannot be forwarded to the destination correctly. Conventional SIP telephony networks address SIP service reliability by using the traditional redundancy and failover methods, such as reliable server pooling [125] and IP address takeover [126]. However, these solutions may not be suitable for large-scale P2P SIP networks because the cost of the two techniques is too high for service providers. To deal with this issue in MISE-P2PSIP, we need an algorithm which replicates SIP transactions information (mainly client side SIP transactions) among the nodes in the P2P overlay networks and selects one node between the neighboring proxies

(successor or predecessor) of the failed or departed node to act as the takeover proxy. The takeover proxy should rebuild the SIP message based on the stored transactions information in order to ensure the reliability of SIP service.

IX.2.3 Locality-aware self-organization

Locality-awareness guarantees that applications are not shipped across long distances when nearby resources are available. In MISE-P2PSIP, nodes are currently organized under a specific group based on their order of entering in the overlay and their location (within public network or not). This technique is not optimal since two logically distant nodes (distant between groups in number of hop) can be geographically closer. Then, it is desirable that those nodes being able to communicate without relaying their message through other distant nodes. The notion of locality-aware DHT [127] already exists but is not suitable for us, since the proposed solutions in this field are middleware-dependent. Our goal is to organize nodes in the same locality (country, province, etc) in the same group. The challenge here is to identify the locality of users and create one or more P2P SIP groups in each locality. On other hand, the proxies' ring will be build by taking into account proxies' locality so that the SIP messages take through the physical shortest path. This will significantly improve call establishment performance since it will reduce routing hop (in the underlying physical network) and latency specifically for intra-group calls. A locality prefix could be defined and inserted in the SIP REGISTER message when the user registers itself. Then, a group will be assigned to a user based on its locality prefix at registration phase.

Publications

1. P. Houngue, E. Damiani, and R. Glitho, “Overcoming NAT traversal issue for SIP-based communication in P2P networks”, WMNC’2011 IFIP/IEEE Wireless and Mobile Networking Conference, October 26-28, 2011, Toulouse, France.
2. P. Houngue, E. Damiani, F. Belqasmi, and R. Glitho, “A Novel Framework for Proxy Topology Building and Proxy-Level Routing in a Peer to Peer SIP Architecture”, Poster presented as part of 4th International Conference on Internet Multimedia Systems Architecture and Applications (IMSAA-10), December 15-17, 2010, Bangalore, India.
3. P. Houngue, E. Damiani, F. Belqasmi, and R. Glitho, “Proxy topology building and routing in P2P SIP architectures”, 5th International Symposium on Telecommunications (IST’2010), IEEE, December 4-6, 2010, Tehran, Iran.
4. P. Houngue, R. Glitho, and E. Damiani, “A novel architecture for a Peer-to-Peer Session Initiation Protocol”, IEEE Symposium on Computers and Communications (ISCC), June 22-25, 2010, Riccione, Italy.
5. D. Benetti, E. Damiani, P. Houngue, “VoIP Echo Suppression in Critical Environments”, Digital Ecosystems and Technologies, 2008. DEST 2008. 2nd IEEE International Conference. Feb 26-29, 2008, Phitsanulok, Thailand.

Bibliography

- [1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "RFC 3261: SIP: Session Initiation Protocol".
- [2] S. Baset, H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. Technical report, Columbia University, New York, NY, 2004.
- [3] X. Zheng¹, V. Oleshchuk¹, "A survey on peer-to-peer SIP based communication systems", Springer Peer-to-Peer Networking and Applications, ISSN 1936-6442 (Print) 1936-6450 (Online), January 2010
- [4] Shi, J., Y. Ji, H. Zhang, Y. Li. 2006, "A hierarchical P2P-SIP architecture", IETF draft, draft-shi-p2psip-hier-arch-00, August 2006.
- [5] D. Bryan, B. Lowekamp, C. Jennings, "dSIP: A P2P Approach to SIP Registration and Resource Location", IETF draft-bryan-p2psip-dsip-00, February 2007.
- [6] D. Bryan, B. Lowekamp, C. Jennings, "SOSIMPLE: Towards a serverless, standards-based, P2P communication system", In International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA), Orlando, FL, June 2005.
- [7] K. Singh, H. Schulzrinne, "Using an External DHT as a SIP Location Service", Tech. Report CUCS-007-06, Columbia University, 2006.
- [8] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, H. Yu, "OpenDHT: A public DHT service and its uses", In SIGCOMM '05, 7384, New York, NY, USA: ACM.
- [9] H. Schmidt, B. Aksoy, F. Hauck, A. Kessler, "How well does JXTA fit Peer-to-Peer SIP?", IEEE, Communications, 2008. ICC '08 proceedings
- [10] K. Singh, H. Schulzrinne, "Peer-to-Peer Internet telephony using SIP", Tech. Rep. CUCS-044-04, Department of Computer Science, Columbia University, New York, NY, Oct. 2004.
- [11] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, H. Schulzrinne, "Resource LOcation And Discovery (RELOAD)", draft-ietf-p2psip-base-19, October 29, 2011. <http://tools.ietf.org/html/draft-ietf-p2psip-base-19>
- [12] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, H. Schulzrinne, "A SIP Usage for RELOAD", draft-ietf-p2psip-sip-06, July 7, 2011.
- [13] I. Baumgart, "P2PNS: A Secure Distributed Name Service for P2PSIP", Proceedings of the Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008), Hong Kong, China, p. 480-485, March 2008.
- [14] C. Zhun, G. Chang, W. Ning, "A JXTA-Based SIP Communication System for P2P Wireless Networks", Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference.
- [15] OverSim. www.oversim.org, July 2010
- [16] JXTA project, <https://jxta-jxse.dev.java.net/>
- [17] JAIN SIP Specification, <https://jain-sip.dev.java.net/>
- [18] RFC 3550, RTP: A Transport Protocol for Real-Time Applications
- [19] K. Singh, H. Schulzrinne, "SIPpeer: A Session Initiation Protocol (SIP) based Peer-to-Peer Internet Telephony Client Adapt", white paper, January, 2005.

- [20] Recommendation ITU-T H.323, Packet-based multimedia communications systems
- [21] B. Foster, C. Sivachelvan, RFC 3661, "Media Gateway Control Protocol (MGCP) Return Code Usage", <http://tools.ietf.org/rfc/rfc3661.txt>
- [22] D. Bryan, B. Lowekamp, M. Zangrilli, "The Design of a versatile, secure P2PSIP communications architecture for the public internet", IEEE international Symposium on Parallel and Distributed Processing, IEEE, Miami, USA, April, 2008, pp. 1-8.
- [23] F. Cao, D. Bryan, B. Lowekamp, "Providing Secure Services in Peer-to-Peer Communications Networks with Central Security Servers", International Conference on Internet and Web Applications and Services (ICIW), IEEE, Guadeloupe, French Caribbean, Feb, 2006, pp. 105-110.
- [24] A. Fessi, H. Niedermayer, H. Kinkelin, G. Carle, "A cooperative SIP Infrastructure for Highly Reliable Telecommunication Services", Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTCOMM'07), ACM, New York, USA, July, 2007, pp. 29-38.
- [25] A. Oram, "Peer-to-peer: Harnessing the power of disruptive technologies," O'Reilly&Associates, March 2001. [online]: <http://www.oreilly.de/catalog/peertopeer/>
- [26] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, "Search and replication in unstructured peer-to-peer networks," in Proceedings of 16th ACM International Conference on Supercomputing (ICS'02), New York, USA, June 2002.
- [27] Napster: <http://www.napster.com/>
- [28] Gnutella: <http://www.gnutella.com/>
- [29] Gkantsidis C, Mihail M, Saberi A. Random Walks in Peer-to-Peer Networks. In Proceedings of IEEE INFOCOM, 2004
- [30] E. Harjula, M. Ylianttila, J. Ala-Kurikka, J. Riekk, J. Sauvola, "Plug-and-Play Application Platform: Towards Mobile Peer-to-Peer", In: Third International Conference on Mobile and Ubiquitous Multimedia (MUM2004). College Park, MD, USA, pp. 63-69, 2004.
- [31] J. Lehtinen, "Design and Implementation of Mobile Peer-to-Peer Application", M.Sc. thesis, Department of Communications and Networking, Helsinki University of Technology, Finland, 2006
- [32] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, B. Wiley, "Protecting free expression online with Freenet," IEEE Internet Computing 6(1), pp40-49 2002.
- [33] B. Y. Zhao, J. Kubiawicz, A. Joseph. "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department, 2001. [online]: <http://www.cs.berkeley.edu/~ravenben/tapestry/html/papers.html>
- [34] A. Rowstron, P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany, November 2001.
- [35] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications", in Proceedings of ACM SIGCOMM'2001, August 2001.
- [36] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A scalable content addressable network," in Proceedings of ACM SIGCOMM'2001, August 2001. [online]: <http://www.icir.org/sylvia/>
- [37] A. Fiat, J. Saia, "Censorship resistant peer-to-peer content addressable networks", in Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms, San Francisco, USA, January 2002. [online] <http://www.cs.unm.edu/~saia/>
- [38] M. Datar, "Butterflies and peer-to-peer networks", Technical Report, January 2002.
- [39] E. Cohen, S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in Proceedings of ACM SIGCOMM'02, Aug. 2002.
- [40] J. Kangasharju, K. W. Ross, D. A. Turner, "Optimal content replication in peer-to-peer communities," Manuscript, 2002.

- [41] C. G. Plaxton, R. Rajaraman, A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in Proceedings of ACM SPAA. ACM, June 1997.
- [42] S. E. Ansary, L. O. Alima, P. Brand, S. Haridi, "A framework for peer-to-peer lookup services based on k-ary search", SICS technical report T2002-06, 2002
- [43] D. Malkhi, M. Naor, D. Ratajczak, "Viceroy: A scalable and dynamic emulation of the butterfly", in Proceedings of PODC'2002, 2002.
- [44] M. Junginger, Y. Lee, "The multi-ring technology-High performance group communication in peer-to-peer networks", in Proceedings of the Second International Conference on Peer-to-Peer Computing (P2P'02), 2002.
- [45] J. Considine, T. Florio, "Scalable peer-to-peer indexing with constant state," Technical Report, September 2002
- [46] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, A. Wolman, "SkipNet: A scalable overlay network with practical locality properties", Accepted for publication to the Fourth USENIX Symposium on Internet Technologies and Systems (USITS'03), September 2002.
- [47] G. Pandurangan, P. Raghavan, E. Upfal, "Building low-diameter P2P networks", in Proceedings of IEEE Symposium on Foundations of Computer Science 2001
- [48] <http://www.kamailio.org/w/>, retrieved on September, 2011.
- [49] S. Sibakov, "Simulating a Mobile Peer-to-Peer Network", Master's thesis in Science of Technology, November, 2009.
- [50] J. Hellerstein, "Toward Network Data Independence", ACM SIGMOD Record 32 (3) 2003.
- [51] P. B. Godfrey, I. Stoica, "Heterogeneity and Load Balance in Distributed Hash Tables", In Proceedings of the IEEE INFOCOM, Miami, FL, Mar.2005.
- [52] C. Wang, B. Li, "Peer-to-peer overlay networks: a survey", Tech. Rep., Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong, 2003
- [53] D. Karger, E. Lehman, F. Leighton, M. Levine, D. Lewin, R. Panigrahy, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web", in proceedings of the 29th annual ACM symposium on theory of computing, may 1997, pp. 654663.
- [54] I. Taylor, "JXTA", Lecture 5: Distributed Systems, 2003
- [55] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes". IEEE Communications Surveys and Tutorials 7 (2005), 7293.
- [56] Voice over IP security, Patrick Park, Cisco Press, September 2008
- [57] <http://www.rtpproxy.org/>, retrieved on September, 2011.
- [58] L. Ong, I. Rytina, M. Garcia, H. Schwarzbauer, L. Coene, H. Lin, I. Juhasz, M. Holdrege, C. Sharp, "Framework Architecture for Signaling Transport," RFC 2719, IETF, Oct. 1999.
- [59] <http://mediaproxy.ag-projects.com/>
- [60] J. Shi, Y. Wang, L. Gu, L. Li, W. Lin, Y. Li, Y. Ji, P. Zhang, "A Hierarchical Peer-to-Peer SIP System for Heterogeneous Overlays", Interworking, IEEE GLOBECOM 2007 proceedings.
- [61] J. Van, E. Burger, A. Spitzer, "Media Server Control Markup Language (MSCML) and Protocol", RFC 5022, September 2007.
- [62] D. Bryan, P. Matthews, E. Shim, D. Willis, S. Dawkins, "Concepts and Terminology for Peer to Peer SIP", draft-ietf-p2psip-concepts-03 <http://tools.ietf.org/html/draft-ietf-p2psip-concepts-03>
- [63] E. Harjula, M. Heikkinen, A. Salinas, J. Hautakorpi, N. Beijar, Z. Ou, "DECICOM: Decentralized Inter-Service Communications State-of-The-Art Study", 2007.
- [64] J. Van, E. Burger, A. Spitzer, "Basic Network Media Services with SIP", RFC 4240, December 2005.
- [65] C. Boulton, T. Melanchuk, S. McGlashan, "Media Control Channel Framework", RFC 6230, May 2011.

- [66] J. Rosenberg, R. Mahy, P. Matthews, D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [67] D. Caromel, C. Delbe, A. di Costanzo, M. Leyton, "ProActive: an Integrated Platform for Programming and Running Applications on Grids and P2P systems", Computational Methods in Science and Technology (2006), Volume: 12, Issue: 1, Pages: 69-77
- [68] <http://code.google.com/p/chord-proactive/>, retrieved on October 2011
- [69] J. Moy, RFC 2328, OSPF Version 2, 1998
- [70] G. S. Malkin, M. E. Steenstrup, "Distance-vector routing, Routing in communications networks", Prentice Hall International (UK) Ltd., Hertfordshire, UK, 1995
- [71] VoIP: wireless, P2P and new enterprise voice over IP by G. Samrat, B. Sudeept, 2008
- [72] R. Mahy, P. Matthews, J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
- [73] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010
- [74] J. Rosenberg, H. Schulzrinne, "An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing", RFC 3581, August 2003.
- [75] A. Georgescu, "Best practices for SIP NAT traversal", <http://www.recursosvoip.com/docs/english/NATtraversal-BestPractices.pdf>, retrieved on September, 2011.
- [76] Asterisk: The Open Source Telephony Projects, <http://www.asterisk.org/>, retrieved on September, 2011.
- [77] Y. Yu, C. Shieh, W. Hwang, C. Hsu, C. Ho, J. Chiu, "An efficient NAT traversal for SIP and its associated media sessions", Int. Computer Symposium, Dec. 15-17, 2004, Taipei, Taiwan.
- [78] B. Ford, P. Srisuresh, D. Kegel, "Peer-to-peer communication across network address translators", Proceedings of USENIX Annual Technical Conference, April 2005.
- [79] H. C. Phuoc, R. Hunt, A. McKenzie, "NAT Traversal Techniques in Peer-to-Peer Networks", ZCSRSC 2008, April 2008, Christchurch, New Zealand.
- [80] P. Yang, J. Li, J. Zhang, H. Jiang, Y. Sun, E. Dutkiewicz, "SMBR : a novel NAT traversal mechanism for structured peer-to-peer communications", IEEE Symposium on Computers and Communications (ISCC '10) : June 22-25, 2010, Riccione, Italy, p.535-539
- [81] H. Li , M. Singhal, "A Secure Routing Protocol for Wireless Ad Hoc Networks", Proceedings of the 39th Hawaii International Conference on System Sciences 2006
- [82] T. V. P. Sundararajan, A. Shanmugam, "Performance Analysis of Selfish Node Aware Routing Protocol for Mobile Ad", ICGST-CNIR Journal, Volume 9, Issue 1, July 2009
- [83] S. Buchegger, and L. J. Boudec, "Performance analysis of the confidant protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks)", In: Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing, Mobi-Hoc., pp. 226-236 (2002)
- [84] S. Buchegger, L. J. Boudec, "Self-policing mobile ad hoc networks by reputation systems", IEEE Communications Magazine 43(7), 101-107 (2005)
- [85] A. Srinivasan, J. Teitelbaum, H. Liang, J. Wu, M. Cardei, "Reputation and Trust based Systems for Ad Hoc and Sensor Networks", In: Algorithms and Protocols for Wireless Ad Hoc and Sensor Networks. Wiley & Sons, Chichester (2006)
- [86] P. Michiardi, R. Molva, "Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks" In Advanced Communications and Multimedia Security, vol. 100, pp. 107-121 (2002)
- [87] S. Marti, T. Giuli, K. Lai, M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in Proceedings of MOBIKOM 2000, August 2000.

- [88] B. Awerbuch, D. Holmer, C. Nita-Rotaru, H. Rubens, "An on demand secure routing protocol resilient to byzantine failures," in Workshop on Wireless Security (WiSe'02), September 2002.
- [89] S. Yi, P. Naldurg, R. Kravets, "Security-aware ad hoc routing for wireless networks, Tech. Rep. UIUCDCS-R-2001-2241, August 2001.
- [90] P. Papadimitratos, Z. Haas, "Secure routing for mobile ad hoc networks", in SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), January 2002.
- [91] L. Venkatraman, D. P. Agrawal, "Security scheme for routing in adhoc networks", in Proceedings of the 13th International Conference on Wireless Communications, July 2001, pp. 129146.
- [92] B. Dahill, B. Levine, C. Shields, E. Royer, "Secure routing protocol for ad hoc networks," U Mass, Tech. Rep. UM-CS-2001-037, 2001.
- [93] J. Franks, P. Hallem-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", IETF RFC 2617, 1999
- [94] RFC 2440 OpenPGP Message Format, 1998
- [95] RFC 3851, Secure/Multipurpose Internet Mail Extensions (S/MIME)
- [96] RFC 2411 : IP Security Document Roadmap
- [97] Securing VoIP Networks, Peter Thermos and Ari Takanen, August, 2007
- [98] RFC 3546, Transport Layer Security (TLS) Extensions
- [99] RFC 4960 Stream Control Transmission Protocol
- [100] RFC 4302, IP Authentication Header
- [101] RFC 4303, IP Encapsulating Security Payload (ESP)
- [102] RFC 4307, Cryptographic Algorithms for Use in the Internet Key Exchange Version 2
- [103] RFC 2412, The OAKLEY Key Determination Protocol
- [104] RFC 4306, Internet Security Association and Key Management Protocol (ISAKMP)
- [105] Krawczyk, H., "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", from IEEE Proceedings of the 1996 Symposium on Network and Distributed Systems Security.
- [106] S. Emil, M. Robert, "Security Considerations for Peer-to-Peer Distributed Hash Tables", In Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS), March 2002.
- [107] M. Castro, P. Duschel, A. Rowstron, D. S. Wallach, "Secure routing for structured peer- to-peer overlay network" In Proceedings of the fifth Symposium on Operating System Design and Implementation, Dec 2002.
- [108] D. Wallach, "A survey of peer-to-peer security issues", In Proceedings of International Symposium of Software Security - Theories and Systems, Nov 2003.
- [109] M. S. Artigas, P. G. Lopez, A. F. G. Skarmeta, "A Novel methodology for constructing secure multipath overlays", IEEE Internet Computing, 9(6):50-57,2005.
- [110] H. Song, M. Matuszewski, D. York, "P2PSIP Security Overview and Risk Analysis", draft-matuszewski-p2psip-security-requirements-06 (work in progress), September 2009.
- [111] H. Al-Allouni, H. Rohiem, A. E. Hashem, M, El-moghazy, A.E.-A. Ahmed, "VoIP Denial of service attacks classification and implementation", Radio Science Conference, NRSC 2009, March 17-19, 2009, Faculty of Engineering, Future Univ., Egypt
- [112] A. Shamir, "Identity-based cryptosystems and signature schemes", In Advances in Cryptology - Crypto '84, Springer-Verlag LNCS 196, 1984, pp. 47-53.
- [113] D. Boneh, M. Franklin, "Identity-Based Encryption from the Weil Pairing", Proceedings of CRYPTO 2001, LNCS 2139, Springer-Verlag, 2001

- [114] C. Cocks, "An Identity Based Encryption Scheme Based on Quadratic Residues", Proceedings of the 8th IMA International Conference on Cryptography and Coding, 2001
- [115] T. Okamoto, "Cryptography Based on Bilinear Maps", in Marc P. C. Fossorier; Hideki Imai; Shu Lin & Alain Poli, ed., 'AAECC', Springer, , pp. 35-50 .
- [116] A. Menezes. T. Okamoto, S. Vanstone, "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field, IEEE Transactions on Information Theory 39, pp.16391646 (1993).
- [117] S. Galbraith, "Pairings", in Advances in Elliptic Curve Cryptography, New York: Cambridge Univ. Press, 2005, London Math. Soc. Lecture Note Ser. 317, pp. 183213.
- [118] K. Frank, G. Alfred, S. Stefan, W. Michael (2005), "Secure Dynamic Source Routing", In HICSS 2005 - 38th Hawaii International Conference on System Sciences 3-6 January, 2005, Big Island, HI, USA.
- [119] C. Plaxton, R. Rajaraman, A. Richa, "Survey and comparison of Peer-to-Peer overlay network schemes", IEEE Communications Surveys & Tutorials, Second Quarter 2005
- [120] C. Hedrick, "RIP (Routing Information Protocol)", RFC 1058, June 1988
- [121] Internet Gateway Routing Protocol, http://www.cisco.com/en/US/docs/ios/12_0/np1/command/reference/1rigrp.html, retrieved on October 2011
- [122] A. Vemuri, J. Peterson, "SIP for Telephones (SIP-T): Context and Architectures", RFC 3372, IETF, September. 2002.
- [123] B. Pourebrahimi, K. Bertels, S. Vassiliadis, "A Survey of Peer-to-Peer Networks", Computer Engineering Laboratory, ITS, TU Delft, The Netherlands.
- [124] S. N. Good and A. Krekelberg. "Usability and privacy: a study of kazaa p2p file-sharing", CHI'03 Proceedings of the conference on Human factors in computing systems, pages 137-144. ACM Press, 2003.
- [125] P. Lei, L. Ong, M. Tuexen, T. Dreiholz, "An Overview of Reliable Server Pooling Protocols", IETF, RFC5351, September 2008.
- [126] Setting Up Redundant Networking, http://horms.net/projects/redundant_content/related/linux-ha/High-Availability-HOWTO-8.html
- [127] L. Le, G-S. Kuo, "Hierarchical and Breathing Peer-to-Peer SIP System", The 2007 IEEE International Conference on Communications, June 2007.
- [128] Phil Zimmermann, PGP User's guide Volume I: Essential Topics, october 1994
- [129] OpenSER, <http://www.voip-info.org/wiki/view/OpenSER>, retrieved on January 2012
- [130] S. Baset, H. Schulzrinne, M. Matuszewski, "Peer-to-Peer protocol (P2PP)," Network Working Group, Internet-Draft, November 2007.
- [131] <http://www1.cs.columbia.edu/~salman/peer/>, retrieved on January 2012
- [132] SIPDHT project, <http://sipdht.sourceforge.net/sipdht2/index.html>, retrieved on January 2012
- [133] P2PNS project, <http://www.p2pns.org/>, retrieved on January 2012
- [134] 39 peers project, <http://www.39peers.net/>, retrieved on January 2012
- [135] P2PSHIP project, <http://code.google.com/p/p2psip/>, retrieved on January 2012
- [136] E. Marocco and E. Ivov, "XPP Extensions for Implementing a Passive P2PSIP Overlay Network based", draft-marocco-p2psip-xpp-pcan-01, November 2007.
- [137] Tier 2 Applications Layer, <https://sites.google.com/site/oversimdemystified/contents/3-4-tier-2-applications>, retrieved on January 2012

APPENDIX

List of abbreviations

Abbreviation	Meaning
1G	First Generation
2G	Second Generation
3G	Third Generation
AES	Advanced Encryption Standard
AH	Authentication Header
AODV	Ad Hoc On-demand Distance Vector
AoR	Address of Record
API	Application Programming Interface
ARAN	Authenticated Routing for Ad hoc Networks
ASP	Address Settlement by Peer-to-Peer
B2BUA	Back-to-Back User Agent
C/S	Client-Server
CA	Certificate Authority
CAN	Content Addressable Network
CBC	Cipher Block Chaining
CODEC	COmpressor-DECompressor
CONFIDANT	Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks
CORE	Collaborative REputation
CRLF	Carriage Return Line Feed
DC	Domain Controller
DHT	Distributed Hash Table
DNS	Domain Name System
DSL	Digital Subscriber Line
DV	Distance Vector
DVR	Distance Vector Routing
ERP	Endpoint Routing Protocol
ESP	Encapsulation Security Protocol
GR	Group Registrar
GUID	Globally Unique IDentifier
H-P2PSIP	Hierarchical P2PSIP
HiLO-Peer	Higher Level Overlay Peer
HTTP	Hyper Text Transport Protocol
IBE	Identity-Based Encryption
IBS	Identity-Based Signature

Abbreviation	Meaning
ICE	Interactive Connectivity Establishment
IETF	Internet Engineering Task Force
IGRP	Internet Gateway Routing Protocol
IKE	Internet Key Exchange Protocol
IP	Internet Protocol
ISAKMP	Internet Security Association and Key Management Protocol
ISUP	Integrated Services Digital Network User Part
JVM	Java Virtual Machine
JXTA	Juxtapose
KBR	Key Based Routing
LAN	Local Area Network
LGPL	Lesser General Public License
LoLO-Peer	Lower Level Overlay Peer
LSA	Link-State Advertisement
LSP	Link State Packet
MAC	Message Authentication Code
MANET	Mobile Ad hoc NETworks
MGCP	Media Gateway Control Protocol
MIME	Multipurpose Internet Mail Extensions
MISE-P2PSIP	Middleware-Independent and SEcure Peer-to-Peer SIP architecture
MSCML	Media Server Control Markup Language
NAT	Network Address Translation
NetAnn	Network Announcement Protocol
NGN	Next Generation Networks
Node-ID	Node Identifier
OSPF	Open Shortest Path First
P2P	Peer-to-Peer
P2PNS	P2PSIP Name Service
P2PP	Peer-to-Peer Protocol
P2PSIP	Peer-to-Peer SIP
PACK	Packet Acknowledgment
PBP	Pipe Binding Protocol
PBX	Private Branch eXchange
PCAN	Passive Content Addressable Network
PDA	Personal Digital Assistant
PDP	Peer Discovery Protocol
PGP	Pretty Good Privacy
PIP	Peer Information Protocol
PKG	Private Key Generator
PKI	Public Key Infrastructure
PRP	Peer Resolver Protocol
PSTN	Public Switched Telephone Network
RELOAD	Resource LOcation And Discovery
RFC	Request For Comments

Abbreviation	Meaning
RIP	Routing Information Protocol
RRQ	Registration Request
RSA	Rivest-Shamir-Adleman
RSVP	Resource ReSerVation Protocol
RTCP	Real-Time Control Protocol
RTMs	Reputation and Trust based Models
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming protocol
RTT	Round Trip Time
RVP	RendezVous Protocol
S/MIME	Secure/Multipurpose Internet Mail Extension
SAR	Secure Aware Routing
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SEP	Service Extensible Protocol
SIP	Session Initiation Protocol
SIP TU	SIP Transaction User
SIPS	SIP URI Scheme
SKEME	Secure Key Exchange MEchanism for Internet
SMBR	Selective-Message Buddy Relaying
SMS	Short Message Service
SMTP	Simple Mail Transport Protocol
SPF	Shortest Path First
SRP	Secure Routing Protocol
SRTP	Secure Real Time Protocol
STUN	Session Traversal Utilities for Network Address Translation
TAB	Triggering Address Binding
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTL	Time To Live
TURN	Traversal Using Relay NAT
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WAN	Wide Area Network
WG	Working Group
XML	Extensible Markup Language
XOR	eXclusive OR