

Convergence of web and communication services

Sivasothy Shanmugalingam

▶ **To cite this version:**

Sivasothy Shanmugalingam. Convergence of web and communication services. Other [cs.OH]. Institut National des Télécommunications, 2012. English. NNT : 2012TELE0007 . tel-00997697

HAL Id: tel-00997697
https://theses.hal.science/tel-00997697

Submitted on 28 May 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Ecole Doctorale EDITE

Thèse présentée pour l'obtention du diplôme de Docteur de Télécom & Management SudParis

Doctorat conjoint Télécom & Management SudParis et Université Pierre et Marie Curie

Spécialité : Informatique et Télécommunications

Par Sivasothy Shanmugalingam

Convergence du Web et des Services de Communication

Soutenue le 30, Avril, 2012 devant le jury composé de :

Olivier Festor	**Directeur de recherche, INRIA**	Rapporteur
Mika Ylianttila	**Professeur, University of Oulu**	Rapporteur
Woo Seop Rhee	**Professeur, Hanbat National University**	Examinateur
Peter Reichl	**Key researcher, Forschungszentrum Telekommunikation Wien (FTW)**	Examinateur
Marie-Pierre Gervais	**Professeur à l'Université Paris Ouest Nanterre la Défense and Chercheur au Laboratoire d'Informatique de Paris 6**	Examinatrice
Nazim Agoulmine	**Professeur, University of Evry Val d'Essonne**	Examinateur
Paul Labrogere	**Department Head, Alcatel Lucent Bell Labs**	Encadrant
Noel Crespi	**Professeur, Télécom SudParis**	Directeur de thèse

Thèse n°: 2012TELE0007

DEDICATION

To my father, Kandiah Shanmugalingam, who taught me the meaning of sacrifice and perseverance.

ACKNOWLEDGEMENT

My Ph.D thesis has been jointly carried out at HybridCom Department, Applications Domain, Alcatel Lucent Bell Labs France and Service Architecture Group,Wireless Networks and Multimedia Services Department, Institut Telecom/Telecom SudParis, France since 2008. I indulged in my passion for research at these two institutions while refining my technical prowess.

This thesis is the result of collaborative work. My Ph.D journey created many opportunities to work with many people at different locations. I interacted with many individuals who encouraged me in many ways, such as through discussion on ideas and new concepts, brainstorming and editing my various drafts. I want to acknowledge them here.

First and foremost, I would like to thank my supervisors, Paul Labrogere, and Noel Crespi for their invaluable guidance and support during the research. We have enjoyed many vehement discussions in research problems, solutions and results. They have given me an incredible breadth of perspective on numerous topics. I am and will be profoundly thankful to them.

All my colleagues at HybridCom have lent a helping hand throughout my stay. I have often sought and received insightful advice from Vincent Verdot, Nicolas Bouche, Mathiew Boussard, and Dohy Hong. My deepest heartfelt thanks to all of them.

I express my sincere thanks to the members of the Service Architecture Group who inspired and encouraged me by way of their discussions on various topics at each team meeting. I owe my special thanks to Gyu Moung Lee, a post-doctoral scholar in the team, whose motivation has enabled me to create papers and RFC drafts.

During the process of my research, I got a wonderful opportunity to work in the SERVERY project. All colleagues in the SERVERY project have taught me something unique and meaningful. I offer my sincere thanks.

I would like to express my eternal gratitude to my father, Kandiah Shanmugalingam, who passed away in March 2011. I will always be indebted for his wisdom, guidance and support. I also wish to thank my mother, Eswary Shanmugalingam, for her love and support throughout my life. Their dream and sacrifices become true today. Last but not least, I want to thank my siblings, Kugneswaray, Genga, and Pathma, for unconditional love and help. It is not possible without my close friends who gave energy to stand up during the difficult times.

ABSTRACT

Different communication services from delivery of written letters to telephones, voice/video over Internet Protocol(IP), email, Internet chat rooms, and video/audio conferences, immersive communications have evolved over time.

A communication system of voice/video over IP is the realization of a two fundamental layered architecture, signaling layer and media layer. The signaling protocol is used to create, modify, and terminate media sessions between participants. The signaling layer is further divided into two layers, service layer and service control layer, in the IP Multimedia Subsystem (IMS) specification.

Two widely used communication systems are IMS, and Peer-to-Peer Session Initiation Protocol (P2P SIP). Service providers, who behave as brokers between callers and callees, implement communication systems, heavily controlling the signaling layer. These providers do not take the diversity aspect of end users into account.

This dissertation identifies three technical barriers in the current communication systems especially in the signaling layer. Those are:

I. lack of openness and flexibility in the signaling layer for end users.

II. difficulty of development of network-based, session-based services.

III. the signaling layer becomes complex during the high call rate.

These technical barriers hinder the end-user innovation with communication services.

Based on the above listed technical barriers, the first part of this thesis defines a concept and architecture for a communication system in which an individual user becomes the service provider. The concept, My Own Communication Service Provider (MOCSP) and MOCSP system is proposed and followed by a call flow. Later, this thesis provides an analysis that compares the MOCSP system with existing communication systems in terms of openness and flexibility.

The second part of this thesis presents solutions for network-based, session based services, leveraging the proposed MOCSP system. Two innovative services, user mobility and partial session transfer/retrieval are considered as examples for network-based, session-based services. The network-based, session-based services interwork with a session or are executed within a session. In both cases, a single functional entity between caller and callee consistently enables the media flow during the call initiation and/or mid-call. In addition, the cooperation of network call control and end-points is easily achieved.

The last part of the thesis is devoted to extending the MOCSP for a high call rate and includes a preliminary comparative analysis. This analysis depends on four factors - scalability limit, complexity level, needed computing resources and session setup latency - that are considered to specify the scalability of the signaling layer. The preliminary analysis clearly shows that the MOCSP based solution is simple and has potential for improving the effective usage of computing resources over the traditional communication systems.

Résumé

Les services de communication, du courrier postal à la téléphonie, en passant par la voix et la vidéo sur IP (Internet Protocol), la messagerie électronique, les salons de discussion sur Internet, les visioconférences ou les télécommunications immersives ont évolué au fil du temps.

Un système de communication voix-vidéo sur IP est réalisé grâce à deux couches architecturales fondamentales : la couche de signalisation et la couche média. Le protocole de signalisation est utilisé pour créer, modifier et terminer des sessions multimédias entre des participants. La couche de signalisation est divisée en deux sous-couches - la couche de service et celle de contrôle - selon la spécification de l'IP Multimedia Subsystem (IMS).

Deux systèmes de communication largement utilisés sont l'IMS et SIP Pair-à-Pair (P2P SIP). Les fournisseurs de services, qui se comportent en tant qu'intermédiaires entre appelants et appelés, implémentent les systèmes de communication, contrôlant strictement la couche signalisation. Or ces fournisseurs de services ne prennent pas en compte la diversité des utilisateurs.

Cette thèse identifie trois barrières technologiques dans les systèmes de communication actuels et plus précisément concernant la couche de signalisation.

I. Un manque d'ouverture et de flexibilité dans la couche de signalisation pour les utilisateurs.

II. Un développement difficile des services basés sur le réseau et les sessions.

III. Une complexification du la couche de signalisation lors d'un très grand nombre d'appels.

Ces barrières technologiques gênent l'innovation des utilisateurs avec ces services de communication. Basé sur les barrières technologiques listées ci-dessus, le but initial de cette thèse est de définir un concept et une architecture de système de communication dans lequel chaque individu devient un fournisseur de service. Le concept, "My Own Communication Service Provider" (MOCSP) et le système MOCSP sont proposés, accompagné d'un diagramme de séquence. Ensuite, la thèse fournit une analyse qui compare le système MOCSP avec les systèmes de communication existants en termes d'ouverture et de flexibilité.

La seconde partie de la thèse présente des solutions pour les services basés sur le réseau ou les sessions, mettant en avant le système MOCSP proposé. Deux services innovants, user mobility et partial session transfer/retrieval (PSTR) sont pris comme exemples de services basés sur le réseau ou les sessions. Les services basés sur un réseau ou des sessions interagissent avec une session ou sont exécutés dans une session. Dans les deux cas, une seule entité fonctionnelle entre l'appelant et l'appelé déclenche le flux multimédia pendant l'initialisation de l'appel et/ou en cours de communication. De plus, la coopération entre le contrôle d'appel réseau et les différents pairs est facilement réalisé.

La dernière partie de la thèse est dédiée à l'extension de MOCSP en cas de forte densité d'appels, elle inclut une analyse comparative. Cette analyse dépend de quatre facteurs - limite de passage à l'échelle, niveau de complexité, ressources de calcul requises et délais d'établissement de session - qui sont considérés pour évaluer le passage à l'échelle de la couche de signalisation. L'analyse comparative montre clairement que la solution basée sur MOCSP est simple et

améliore l'usage effectif des ressources de calcul par rapport aux systèmes de communication traditionnels.

List of Publications

The following original articles are produced during my Ph.D journey. Those are:

Articles in International Journals

I. Shanmugalingam, S.; Crespi, N.; Labrogere, P.;, "Scalability and signaling architecture," International Journal of Research and Reviews in Next Generation Networks (IJRRNGN), Vol. 1, No. 2, December 2011, ISSN: 2046-6897.

II. Hongguang Zhang, Zhenzhen Zhao, Shanmugalingam Sivasothy, Cuiting Huang, and Noël Crespi, "Quality-Assured and Sociality-Enriched Multimedia Mobile Mashup," EURASIP Journal on Wireless Communications and Networking, vol. 2010, Article ID 721312, 11 pages, 2010.

Papers in International Conference Proceedings

I. Shanmugalingam, S.; Verdot, V; Crespi, N.; Labrogere, P.; , "A Solution for Partial Video Voice over IP Session Transfer and Retrieval," The 14th International Symposium on Wireless Personal Multimedia Communications , 3-7 Oct. 2011

II. Shanmugalingam, S.; Crespi, N.; Labrogere, P.; , "User mobility in a Web-based communication system," Internet Multimedia Services Architecture and Application(IMSAA), 2010 IEEE 4th International Conference on , vol., no., pp.1-6, 15-17 Dec. 2010

III. Shanmugalingam, S.; Crespi, N.; Labrogere, P.; , "My Own Communication Service Provider," Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on , vol., no., pp.260-266, 18-20 Oct. 2010

IV. V. Verdot, M. Boussard, N. Bouché, S. Sivasothy, "The bridging of two worlds: A Web-IMS Communication Solution", Internet and Multimedia Systems and Applications (EuroIMSA 2009), Cambridge, United Kingdom,2009

V. Zhang, Hongguang; Nguyen, Hang; Crespi, Noël; Sivasothy, Shanmugalingam; Le, Tien Anh; Zeghlache, Djamal; Wang, Hui; , "A Metadata-Based Approach for Multimedia Service Mashup in IMS," Communication Networks and Services Research Conference (CNSR), 2010 Eighth Annual , vol., no., pp.356-360, 11-14 May 2010

VI. S. Sivasothy, G. M. Myoung, and N. Crespi. A unified session control protocol for IPTV services. In Proceedings of the 11th international conference on Advanced Communication Technology, pages 961-965, Gangwon-Do, South Korea, February 15-18, 2009.

VII. Sivasothy, S.; Gyu Myoung Lee; Crespi, N.; Bertin, E.; , "An enabler gateway for service composition using SIP," Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on , vol., no., pp.1-6, 26-29 Oct. 2009

Internet Drafts

I. S. Shanmugalingam, GM. Lee and N. Crespi, “SIP extensions for media control”, draftsiva- sip-media-00.txt, March 2009.

II. S. Shanmugalingam, G.M. Lee, N. Crespi, “SIP extensions for media control” draft-sivasip- media-01.txt, IETF Sept 2009.

Patent

I. Shanmugalingam S., Verdot V, Bouche N. : “PROCEDE DE TRANSFERT PARTIEL D’UNE SESSION DE MEDIA”, Filed in France on 14-Jan-2011 , n°1100131

Contents

List of Figures **13**

List of Tables **15**

1 Introduction **17**
1.1 Motivation 17
1.2 Problem Statement 18
1.3 Contributions 21
1.4 Organization of the Thesis Manuscript 22

I State of the Art **23**

2 IP Multimedia Subsystem **24**
2.1 Background 24
2.1.1 Session Initiation Protocol 24
2.1.2 IP Multimedia Subsystem 25
2.1.3 IMS Session Establishment Call Flow 27
2.2 Openness and Flexibility 28
2.2.1 Openness 28
2.2.2 Flexibility 31
2.3 Network-based, Session-based Services 32
2.3.1 User Mobility 33
2.3.2 Partial Session Transfer and Retrieval 37
2.4 Scalability 40
2.4.1 Proposed Definition of Scalability 40
2.4.2 Interested Architecture 43
2.4.3 Existing Solutions 44
2.4.4 Scalability Analysis 47
2.4.5 Indirect Work 48
2.5 Conclusion 50

3 Other Communication Systems **52**
3.1 P2P SIP 52
3.2 Scalability 53
3.3 XMPP/Jingle 54
3.4 Web Service Initiation Protocol (WIP) based Service Oriented Communication 55

3.5 Conclusion . . . 56

4 Descriptive Model of IP Media Services 57
4.1 Introduction : IP Media Services . . . 57
4.2 A Descriptive Model of Media Aspect of a Service . . . 58
4.2.1 Signaling Path . . . 58
4.2.2 Media Channels . . . 59
4.3 Signaling Protocol . . . 60
4.3.1 Protocol Requirements . . . 60
4.3.2 Protocol Definition . . . 60
4.3.3 Properties of the Protocol . . . 62
4.4 Protocol Comparison . . . 62
4.5 Conclusion . . . 62

II Contribution 64

5 My Own Communication Service Provider (MOCSP) 65
5.1 From Concept to System . . . 65
5.1.1 Concept . . . 65
5.1.2 System . . . 65
5.1.3 Definition of Communication Services . . . 66
5.1.4 Communication Hyperlink (CH) . . . 66
5.1.5 Architecture of the MOCSP System . . . 67
5.1.6 MOCSP vs I-Centric Communications . . . 69
5.2 Control Session in the MOCSP System . . . 69
5.3 Impact on End Users . . . 71
5.3.1 Openness . . . 71
5.3.2 Flexibility . . . 72
5.3.3 Privacy . . . 72
5.4 Conclusion . . . 73

6 User Mobility 75
6.1 Descriptive Model . . . 75
6.1.1 Role of Network Box . . . 76
6.1.2 Usage of Network Box in the MOCSP system for the user mobility use case . . . 77
6.2 Call Flow for User Mobility . . . 77
6.3 Phenomenon of Missed Call . . . 78
6.4 Conclusion . . . 79

7 Partial Session Transfer and Retrieval 83
7.1 Architecture . . . 83
7.1.1 Network Box . . . 83
7.1.2 Description of a Medium Device, Medium Sink and Medium Source . . . 84
7.1.3 Caller Box/Callee Box . . . 85
7.2 Protocol . . . 86
7.3 Call Flow . . . 87
7.3.1 Network-Initiated Partial Session Transfer/Retrieval . . . 87

7.3.2 User-Initiated Partial Session Transfer/Retrieval 89
7.3.3 High Mobility Situation 90
7.3.4 Limitations 91
7.4 Conclusion 91

8 Scalability **95**
8.1 MOCSP based Signaling Architecture 95
8.2 Impact of DNS 97
8.2.1 Basic concepts in DNS 97
8.2.2 DNS lookup and Caching 98
8.2.3 Conclusion: DNS lookup delay and Scalability of DNS . . 100
8.3 Calculation of Number of Servers 101
8.3.1 Performance of the Web server 101
8.3.2 Number of Network boxes in a Web server 101
8.3.3 Discussion on the Calculation 102
8.3.4 Virtual Web Hosting 102
8.4 Scalability Analysis 103
8.4.1 How Scalable 103
8.4.2 Complexity 103
8.4.3 Needed Computing Resources 104
8.4.4 Session Setup Latency 105
8.5 Conclusion 105

9 Discussion **106**
9.1 Transfer Call Control 106
9.1.1 SIP based Solution 107
9.1.2 MOCSP and Transfer Call Control 108
9.2 IP Mobility 109
9.2.1 SIP based Solution 109
9.2.2 MOCSP and IP Mobility 109
9.3 Implementation Choices 110
9.3.1 Web Browser 111
9.3.2 Web Server 116
9.3.3 Sample Code for the MOCSP system 119
9.3.4 Summary 126
9.4 SDP Offer/Answer vs Media Description 126
9.4.1 Session Description Protocol 127
9.4.2 Offer/Answer Mechanism 130
9.4.3 Media based Negotiation 133
9.5 Conclusion 134

III Conclusion **136**

10 Conclusion **137**

11 Future Work **140**
11.1 Prototype and Further Validation 140
11.2 Scalability Validation 141
11.2.1 Experimentation Setup 141
11.3 Contribution to Standard Bodies 145
11.4 Possible Extensions 145

Bibliography **147**

A French Summary **157**
A.1 Motivation 157
A.2 Enoncé du problème 158
A.3 La solution proposée 162
A.3.1 MOCSP: concept et système 162
A.3.2 Le flux d'appels 163
A.3.3 L'ouverture et flexibilité 163
A.3.4 La mobilité utilisateur 164
A.3.5 Partial session transfer and retrieval 165
A.3.6 La montée en charge 165

List of Figures

2.1 IMS architecture with interested entities in the service control layer and service layer 26
2.2 IMS session establishment call flow 27
2.3 High level view of IMS based communication system. Callees and Callers are user agents. ISC stands for IMS Service Control and SP stands for Service Provider. 29
2.4 Existing abstractions for end user service creation 29
2.5 Diagram for the user mobility use case 33
2.6 IMS architecture connecting two different administrative domains (suffix 1 and 2 denote two domains.). 43
2.7 A load balancer between callers and P-CSCF1. Network admin 1 has many physical nodes for a single functional entity, P-CSCF 46

4.1 General practice of implementing IP media services 57
4.2 Programming abstractions in a signaling path 58
4.3 The user interface at one end of a media channel shows message transmission between two ends. The events preceded by exclamation marks are chosen by the user, while the events preceded by question marks are chosen by the other end of the channel. Commas separate unrelated transition labels with the same source and sink states. 59
4.4 Specification of a protocol endpoint. ? means received, ! means sent. ?oack / !select means send select if and when oack is received. !oack / !select means send the two signals in sequence. Commas separate unrelated transition labels within the same source and sink states 60
4.5 Protocol usage between two endpoints 63

5.1 A formal definition of communication services 67
5.2 Simplified architecture of MOCSP system; thick line shows the control path and dotted line shows the media path 68
5.3 A call flow diagram for a session in the MOCSP; A caller initiates the session and a callee terminates the session. 74
5.4 High level view of the MOCSP system. Callee and callers are Web browser based clients. 74

6.1 Diagram for descriptive model for user mobility 75
6.2 Call flow during the session mobility 80
6.3 Diagram for descriptive model for missed call situation 81

6.4 Call flow during the voice mail and call 82

7.1 Architecture of the partial session transfer and retrieval. Thick lines show the connectivity established by the network box to medium sources and sinks. Dotted lines are possible paths for media flow 84
7.2 The classification of medium 85
7.3 Protocol stack for medium devices 85
7.4 Sample user interface for partial session transfer and retrieval. . . 86
7.5 A call flow for session establishment between caller and callee and for partial session transfer at the caller side. 88
7.6 A call flow for partial session transfer at the callee side. 89
7.7 A call flow for partial session retrieval at the caller side and session termination. 90
7.8 A call flow for user-initiated partial session transfer and retrieval 93
7.9 A sample scenario showing both ends sending a message for the same purpose. 94

8.1 A scalable architecture for communication services. Each small circle is represented by each user/callee. 96
8.2 Hierarchical Namespace in DNS. 98
8.3 Resolution of http://siva.mocsp.com in the Internet. 99
8.4 A sample deployment scenario. 102

9.1 SIP call flow for call transfer with consultation hold, protecting the transfer target 107
9.2 Entities involved in transfer with consultation hold (protecting Transfer Target) call in MOCSP 108
9.3 A typical call flow in the MOCSP system with important points that need to be considered during IP mobility 111
9.4 Architecture of WebRTC including MOCSP client side. 114
9.5 The high level architecture of Node.JS 118

11.1 Lab experimentation setup. Each thick line shows the TCP connection to be used for sending (as well as receiving) the signaling messages. Each connection indicates each client connection to the MOCSP instance. 142

List of Tables

2.1 List of SIP Methods 25

7.1 List of abstractions for the auxiliary protocol 86

8.1 Description of each step in Figure 8.3 99

9.1 List of Events defined in [1] 113
9.2 Terminology list that is originally available as presented in [2] . 116

10.1 Comparison of different signaling architectures based on the proposed scalability evaluation framework. Here, N is the number of super-nodes in a P2P overlay 138

Chapter 1

Introduction

1.1 Motivation

The Web has changed the way people perform their day to day activities and has evolved rapidly (e.g. Web 2.0 , HTML5) since the invention of the Web in 1989 by Sir Tim Berners Lee [3] [4]. The reason for this success is that "The Web adopts relatively simple technologies with sufficient scalability, efficiency and utility" [5]; other two factors that also contributed to the success of the Web are: its openness and flexibility. These factors serve as the key means for high user participation in creating applications. As a consequence, the Web paves the way for innovative applications in the Internet era where end-users become producers and consumers of contents and services.

Various communication services from postal delivery of letters to telephony, voice/video over Internet Protocol(IP), email, Internet chat rooms, and video/audio conferences, immersive communications [6] have evolved over time. Like other communication services, voice/video over IP is an essential need of the people. For example, around the first quarter of 2011, the number of Skype users signed in is reached 30 million [1] - There were 30 million people online on Skype at the same time. There is a pressing requirement for providing communication services to the people in the planet because, mobile subscription and the number of individuals connected to the Internet at home keep on increasing as of today. Approximately, two billion people are connected to the Internet in the first quarter of 2011 [2]. As more and more people are connected to the Internet, individuals become part of a networked world.

Voice/video over IP communication system is based on the famous layered architecture, consisting of a signaling layer and a media layer. The signaling protocol is used to create, modify, and terminate media sessions between participants. Existing communication systems can be built by either IP Multimedia Subsystem (IMS) [7] based on Session Initiation Protocol (SIP) [8] , Peer-to-Peer (P2P) SIP[9], XMPP/Jingle (like gTalk) [10] , Web Service Initiation Protocol (WIP) [11] based Service Oriented Communication (SOC), Service oriented VoIP (SOVoIP) [12] , or Web personal communication systems [13]. Generally,

[1] http://blogs.skype.com/en/2011/03/30_million_people_online.html

[2] http://www.eg8forum.com/fr/documents/actualites/McKinsey_and_Company-internet_matters.pdf

communication systems take a bottom-up approach to provide communication services for mass users without much personalization, thus serving as brokers between callees and callers. Callee is the receiver of calls and Caller is the originator of calls.

However, two widely used communication systems are IMS, and P2P SIP, where SIP is used as the signaling protocol. The IMS architecture divides the signaling layer into a service control layer and a service layer. The service control layer has many functional entities between caller and callee. Except for user agents, all other functional entities are controlled by the operator. This means that the operator has to deploy the required functional nodes. The nodes mostly perform rendez-vous functions for signaling messages of a session. Similarly, services such as presence are deployed in the service layer, relying on the common service control layer. The important objectives of IMS specifications are to reduce capex and opex of the provider, to encourage telecom and Internet convergence and to shorten the time to market with new services. Remarkably, the thought of end user innovation is underestimated in the initial design of the IMS communication systems.

Much research has been devoted to develop the functional nodes and protocols (SIP, Real-Time Transport Protocol (RTP)[14], Session Description Protocol(SDP)[15]) for the IMS architecture. Extended work such as providing the click-to-dial function for web users is performed in order to lure end user contributions. However, these approaches do not drive the end user innovation. For example, according to ProgrammableWeb, there are around 200 mashups that are communication services related out of listed 5100 mashups [16]. A mashups is a service by combining existing services. The main factor behind these few mashups of communication services is that these communication systems could not accommodate diverse needs of end users. This triggers another issue that service provider controlled infrastructure should change from middleware for researchers and geeks to a day-to-day reality for billions of people.

1.2 Problem Statement

The overall research question this thesis tries to answer is:

"How to enable users to contribute to/develop different communication service-based mashups? Or When will users get freedom (or without having to wait for permission) to innovate with communication services ?"

This thesis considers that users are not naive regarding information and communication technologies. In order to be able to answer the overall research question, I defined a set of important research questions that address the problem in detail.

I. The existing communication infrastructure, IMS, does not have enough openness and flexibility to attract third party or end users for developing applications based on communication services.

   I define the openness and flexibility as follows: 1) Openness means that users develop their applications or implement their new ideas (for new

services) with less dependability. 2) Flexibility means that developed applications can be easily modified.

Key features of the communication services are session/call control, routing between caller and callee, and state management. While defining the protocol and controlling the functionalities, the service provider (SP) initially offers a call control Application Programming Interface (API) in order to empower third party developers.

Communication service related functionalities that are now defined by SP are given to end users by relevant APIs. These functionalities/APIs are in the form of SIP API or HTTP API (e.g. ParlayX [17]). IMS Service Control (ISC) interface is SIP API, standardized for developers or third party. Therefore, services based on a Third Party Call Control (3PCC) controller can be developed. However, service development is cumbersome for developers because of the complexity of the SIP protocol and architecture. A lot of research has been carried out to hide the complexity of the operator controlled infrastructure by providing abstractions or APIs. A high level abstraction is defined by the Parlay Group (e.g. IMS network) for Web developers. For example, click-to-dial application is based on 3PCC [18]. Based on a click-to-dial application, a user in the web site can initiate and terminate a session. This means that these APIs provide limited functionality. These approaches do not meet the individual needs of an end user (i.e. there are not many different composed services based on this API).

Essentially, many of these APIs (e.g. Session Data Types [19]) are at the research level and are not deployed in the network for end user usage because they are injecting complexity into the platform. The level of complexity is also associated with granularity of APIs. Furthermore, third party or end users strongly depend on a particular service provider.

Moreover, the design pattern of the communication system is similar to the one-size-fits-all model where each session should behave similarly. The core aspects of the signaling protocol are defined by the standards. Changes (for new features) in the control protocol should be analyzed with more care. Until changes are added into the prospective places, these new features will not be enabled. For example, if one client does not have the feature and establishes a session, this new feature will not be enabled. Typically, changes for the features take long time to roll out during standardization and deployment.

Providing enough openness and flexibility for end users will enable them to get control on the signaling layer. This freedom encourages end users to compose new services based on their needs.

II. Network-based, session-based services are very difficult to develop in the existing infrastructure. The network-based, session-based services interwork with a session or are executed within a session. For example in Call Forwarding on Busy, if callee is busy, call will be forwarded to the predefined destination by the callee. This service is executed only during call establishments. Many efforts have been devoted to compose this kind of services. This kind of problem is referred as feature interactions. However, there is a special case where service should be executed during call

establishments and/or mid-calls. Such a situation is not considered by the research community. Moreover, developing a solution based on the SIP protocol and IMS architecture brings more complexity and requires deep knowledge of interdependency of the functional nodes in the IMS network.

Moreover, network-based session-based services require the cooperation of network call control and endpoints, as in Partial Session Transfer and Retrieval (PSTR). In PSTR, service logic should be deployed in the SIP Application Server (AS) and end points for supporting network-initiated and user-initiated PSTR. In the SIP or IMS specifications, this kind of cooperation is overlooked. Instead, placing the logic independently in the network call control and endpoints is encouraged. This deployment is not efficient in the IMS architecture because independently deployed services logic in the intermediate and end points can not be easily coordinated. In addition, this PSTR creates deployment complexity. Both user and network side approaches need to deploy the 3PCC mechanism. This means that for PSTR at the user side, two instances of 3PCC must be executed and for the PSTR at the network side, one instance of 3PCC should be executed in the SIP AS. Therefore, this separated deployment adds complexity in developing a solution.

Besides, Eric et al focus on the composition of multiple SIP-based 3PCC controllers [20]. If multiple SIP-based controllers are in a signaling path, how does the system behave? Their research is in the direction of having many call controllers in the signaling path. This aspect is not considered in this thesis.

PSTR enables mobile users to transfer and retrieve media session partially to devices that are located in their vicinity. Similarly, services that can be deployed in the network enrich user experience.

III. The signaling infrastructures such as IMS and P2PSIP become complex when a high call rate is experienced.

The signaling layer in IMS has many functional entities such Proxy-Call Session Control Function (P-CSCF), Serving-CSCF (S-CSCF) and AS. The two aspects listed below should be taken into account when a high call rate is experienced. First, capacity of a node that implements the functions, P-CSCF, S-CSCF, etc is finite, therefore, it becomes saturated at one point. Second, the signaling protocol is a state-based protocol unlike Hyper Text Transfer Protocol (HTTP). These states are kept in these functional entities. Therefore, messages between caller and callee should be consistent. This means within a session all messages should follow the same path.

The two aspects listed above hinder an effective solution for the scalability problem. One possible solution is to accommodate a load balancer within the signaling layer as an additional functional entity.

In [21], H.Jiang et al propose a finer-grained load balancer that has knowledge of SIP such as, different transactions in SIP and processing cost for different transactions. This solution helps increasing the scalability of the service control layer by forwarding the traffic to proxies based on the capacity. However, this load balancer falls recursively into the problem of

scalability. Furthermore, no work has been reported on how to solve the problem of scalability when a large number of users are connected to services (e.g. service logic in SIP AS) in the service layer.

When using load-balancers, the service provider should be aware of congestion. This means that each node should not be overloaded during the high call rate. If overloaded in one or many physical nodes, there should be a throughput collapse in the IMS/SIP network [22]. In addition, the overloaded network will not recover easily. Controlling overload in a network of SIP servers is a widely discussed topic in research and standard organizations (e.g. IETF). Existing solutions for congestion avoidance are inefficient and complex, demanding computing resources for the congestion avoidance algorithm.

This higher level of complexity that is injected by the load balancer and congestion avoidance mechanism demands additional computing resources and adds latency to the session setup. This inefficient usage of computing resources also undermines the green IT efforts.

1.3 Contributions

A major contribution of this thesis is the specification of a communication system architecture in the Web platform for enabling user innovation.

The specific contributions of this thesis work are the following:

1. Proposal of a new concept called My Own Communication Service Provider (MOCSP) and a MOCSP system in the Web platform for providing the control of the signaling protocol to end users. This natural convergence of Web and communication services facilitates the end user innovation as it is witnessed in the Web. Moreover, this model enables diversity while managing the complexity.

2. Development of the solution for the user mobility use case based on MOCSP system. This solution simplifies the development, because it reduces the interdependency nodes between caller and callee into one that is called Network box. Fundamentally, this thesis work merges two existing services (that are sometimes called features)- which are complementary to each other - into one service.

3. Development of the solution for the partial session transfer and retrieval use case based on MOCSP system. The central node can accomplish the goals of users (caller and callee) such as transfer and retrieval of partial media. This solution permits any number of session transfers and retrievals at callee and caller side effectively regardless of the originators (caller, callee, or network side application).

4. Preliminary evaluation of the MOCSP system for scalability based on four parameters: scalability limit, complexity level, needed computing resources and session setup latency. This preliminary evaluation shows that the MOCSP based system is a simple solution for a high call rate compared to the existing communication systems.

1.4 Organization of the Thesis Manuscript

In this thesis, I contribute to the architecture of the communication system that facilitates the innovation with communication services. In addition, this thesis contains a survey of existing communication systems towards the research problems, a description of the work and an overview of the analytical results and contributions.

This thesis is organized as follows: after the introduction, a review of the existing literature that is relevant to the research problem at hand is presented. Based on maturity and uniqueness of the communication systems, this thesis initially considers the IMS architecture and relevant work in Chapter 2. Chapter 3 discusses other communication systems such as P2P SIP, XMPP/Jingle and Web service initiation protocol. The P2P overlay natively supports scalability and therefore, related work on the scalability aspect is reviewed in Chapter 3. This thesis is built, leveraging the idea of the descriptive model of IP media services that is presented in Chapter 4

The contribution part presents five chapters:

Chapter 5 presents the main contribution of the thesis, a new concept and system of My Own Communication Services Provider (MOCSP). Next, I present the solution for user mobility and partial session transfer and retrieval based on the MOCSP system in Chapter 6 and 7, respectively. Then, Chapter 8 presents analytical results for the scalability problem based on MOCSP. After all, it is important to verify whether some different important use cases(that are common in SIP) can be developed in the MOCSP. Chapter 9 identifies the ability of the MOCSP by considering two new use cases. Additionally, a possible path for implementation is examined in Chapter 9.

Finally, I summarize and discuss future directions for this thesis work in Chapter 10.

Part I

State of the Art

Chapter 2

IP Multimedia Subsystem

Before answering the research questions, it is important to show the weakness of existing solutions. IP Multimedia Subsystem (IMS) is an important architecture for this thesis. Therefore, this chapter surveys the literature of IMS that attempted to answer the listed research problems of this thesis. Initially, background of Session Initiation Protocol (SIP), IMS and call flow is presented. Then, three sections are devoted to review the mechanisms used for the listed three research problems in Section 1.2, respectively.

2.1 Background

2.1.1 Session Initiation Protocol

Session Initiation Protocol (SIP), an application layer protocol, is initially standardized in 1999 as RFC 2543 for controlling multi-media sessions (set up, modify and tear down) in the IP network. Later, SIP becomes a de facto signaling protocol, used widely by telecom service providers. Today, the main draft for SIP is RFC 3261 that obsoletes RFC 2543.

This protocol co-exists with other protocols; two important protocols are Session Description Protocol (SDP) [15] and Real-time Transport protocol (RTP)[14]. SDP is used to describe multimedia sessions, working together with SIP. Real time data such as audio, video, messages are sent between two endpoints using RTP. Media capability negotiation between endpoints are performed using offer/answer exchange as specified in RFC 3264.

Design of HTTP influenced on the SIP messages. This means SIP employs text-based encoding and syntax as used in HTTP. However, SIP differs from HTTP in terms of using the transport service. HTTP runs only over Transmission Control Protocol (TCP), but SIP is designed to run over User Datagram Protocol (UDP) or TCP. Reliability mechanisms and message retransmission timers are natively built in SIP. Therefore, SIP can be used over UDP.

Basic six methods are specified in RFC 3261. However, over time, there are around eight other methods defined as extensions. The complete list of methods and their descriptions are included in Table 2.1.

The another important aspect is a rendez-vous function that helps to locate a given user. Rendezvous function or routing of SIP messages is carried by a SIP server/proxy that can redirect to other servers or users or fork (replicate)

Method	Description	RFC No
INVITE	(Re-)Initiation of new session	RFC 3261 [8]
ACK	Acknowledge a response to INVITE	RFC 3261 [8]
CANCEL	Cancel a pending request	RFC 3261 [8]
BYE	Terminate a existing session	RFC 3261 [8]
OPTIONS	Query another UA or SIP server for discovering its capabilities	RFC 3261 [8]
REGISTRATION	Temporary binding of SIP URI to an address-of-record (or collect IP address from UA)	RFC 3261 [8]
UPDATE	Updating session parameters, but unlike RE-INVITE method. No impact on the state of dialog	RFC 3311[23]
PRACK	Providing reliable provisional response messages	RFC 3262 [24]
SUBSCRIBE	Part of event notification framework and requesting state updates in future	RFC 3265 [25]
NOTIFY	Part of event notification framework and Responding the current state information after SUBSCRIBE	RFC 3265 [25]
PUBLISH	Publishing event state	RFC 3903 [26]
REFER	Act upon the information in REFER and inform the outcome of action to originators	RFC 3515 [27]
MESSAGE	Transfer of messages between users	RFC 3428 [28]
INFO	Carrying application level information between endpoints	RFC 6086 [29]

Table 2.1: List of SIP Methods

the session INVITE method to many locations of an user. In SIP, messages are routed in a hop-by-hop fashion. SIP used e-mail-like identifiers, called SIP URI.

2.1.2 IP Multimedia Subsystem

Third generation Partnership Project (3GPP) has specified the IMS as an architectural framework for delivering multimedia services. The specification of IMS is initially available in 3GPP Release 5 [30] and is then continuously updated in the following releases. The following objectives are taken into account when designing the IMS: 1) convergence of telecom and Internet, 2) delivering rich multimedia services, and 3) shortening the time to market with new services.

The IMS is intended to separate functions in a layered fashion. This thesis concerns with two layers, service control and service layer as shown in Figure 2.1. Services in the service layer do not need to own control functions. 3GPP chooses SIP, an IETF protocol as a signaling protocol in the IMS architecture.

The service control layer consists of three functional components, Proxy-Call Session Control Function (P-CSCF), Serving-CSCF (S-CSCF), and Interrogating-CSCF (I-CSCF) [7]. Splitting the SIP servers into S-CSCF, P-CSCF and I-CSCF is considered as an innovative idea of IMS. I-CSCF is not shown in Figure 2.1. Home Subscriber Server (HSS) is a database for subscriber and service re-

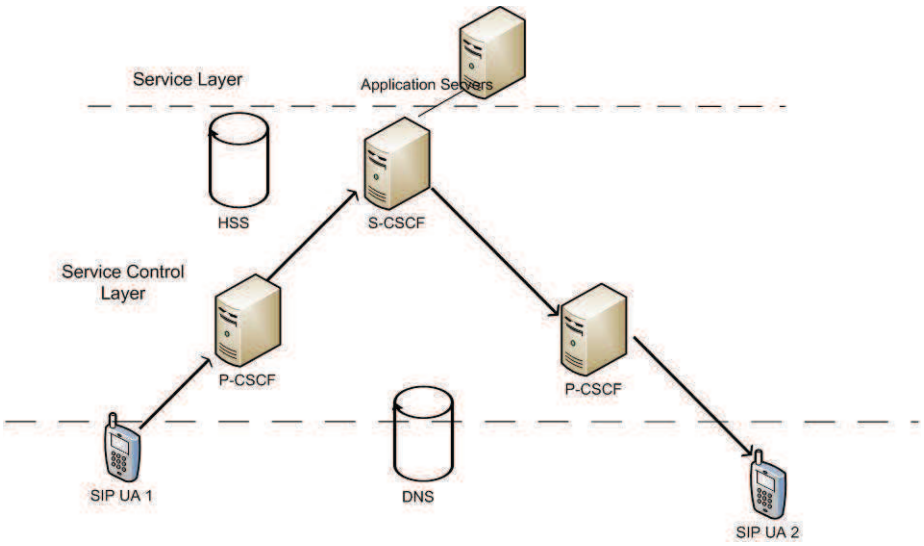


Figure 2.1: IMS architecture with interested entities in the service control layer and service layer

lated information that is used for Accounting, Authorization and Authentication (AAA). Even though the signaling layer (i.e. functional entities) is associated with AAA, the AAA feature is not in the scope of this thesis.

Application Servers (AS) host the services for execution and is linked to S-CSCF via IMS Service Control Interface (ISC). ISC is typically a SIP interface. The services in the ASs involve in originating, terminating and transiting the sessions. In addition, presence server, instant messaging server, and group list management server are some important application servers.

Different roles of SIP Proxies are as follows:

P-CSCF The first contact point for any UA to the IMS network is P-CSCF. For sake of scalability and redundancy, an IMS network has a number of P-CSCFs; each P-CSCF serves a number of UAs depending on capacity of the proxy. Important roles for P-CSCF are lookup, forwarding, and session management.

I-CSCF Main role of I-CSCF is to route SIP messages to proper destinations, mostly an S-CSCF. This entity is the main contact point for messages coming from outside.

S-CSCF It stores the binding of user location and the user's SIP address of record (also known as Public User Identity). In addition, it performs session control. Beyond session control, S-CSCF decides to route SIP messages to application servers based on information in the form of initial Filter Criteria (iFC).

Some functional entities that are defined in the IMS specification are not deeply discussed here as they are not relevant for the thesis. For media streaming, only end-to-end media flow between participants is considered throughout the thesis.

1. Gateway Control Function (GCF) is a gateway for controlling signaling messages to the circuit switch networks.

2. Media Resource Function Controller (MRFC) performs controlling the media resource functions.

3. Media Resource Function Processor (MRFP) supports media capabilities such as playing or recording short announcements, conferencing, transcoding, and DTMF detection.

4. Policy and Charging Rules Function (PCRF) maintains the allocation of the quality of service and applies the required policies on the transport and the access network and is responsible for charging.

2.1.3 IMS Session Establishment Call Flow

IMS provides many functionalities such as service control, security, routing , registration, QoS and charging. However, this thesis deals primarily with session management - routing between SIP UAs. This subsection shows a call flow which includes SIP messages and IMS functional entities which are discussed earlier. The relevant call flows are shown in Figure 2.2. More details about call flows are available in 3GPP TS 24.228 [31]. Additionally, behavior of each functional entity for SIP messages is specified in 3GPP TS 24.229[32].

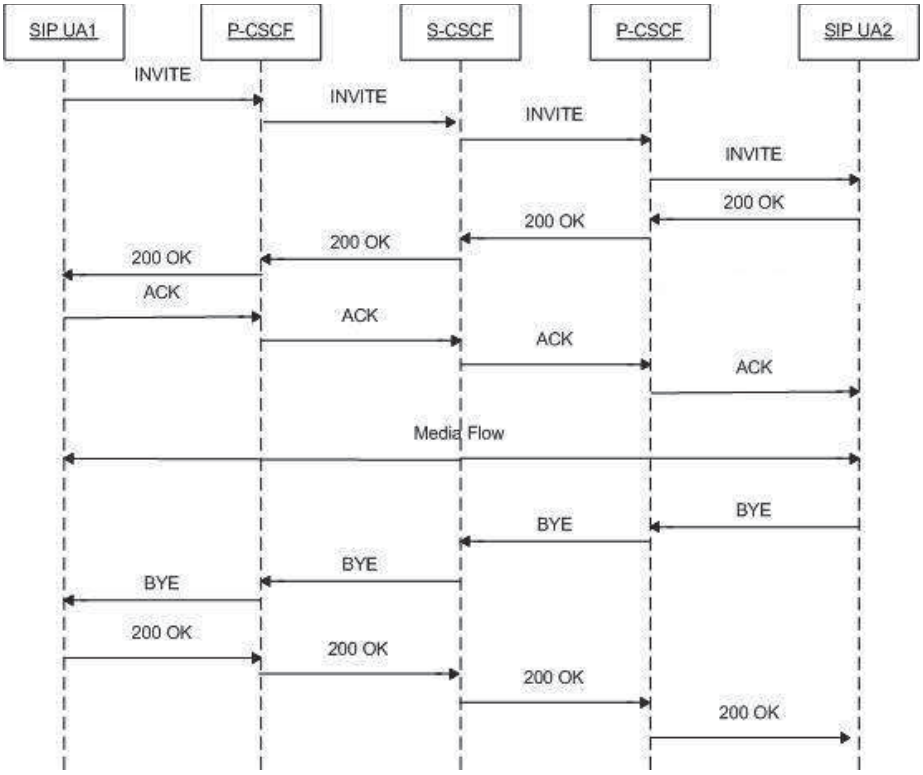


Figure 2.2: IMS session establishment call flow

Here, SIP UA (3GPP refers to it as an User Element (UE)) used to refer to an endpoint. In Figure 2.2, three methods (INVITE, ACK and BYE in Table 2.1) are used in the call flow. Initially, SIP UA1 sends an INVITE message that passes through P-CSCF, S-CSCF, again P-CSCF and ultimately SIP UA2. Similarly, other two messages (200 OK and ACK) are transferred. Generally, SIP UA2 has to accept the INVITE message. If accepted, 200 OK will be sent back to SIP UA2. This kind of response is well defined in the standards. Once ACK reaches SIP UA2, both endpoints (SIP UA1 and SIP UA2) transport the media (e.g. video or audio) as specified in INVITE/OK/ACK.

There is another important message flow between SIP UA1 and SIP UA2, by which both sides allocate resources in advance. The message flow takes places between INVITE and 200 OK. However, this aspect is not discussed in the call flow. Moreover, each message has many SIP headers that are sometimes important for P-CSCF, and S-CSCF. These details are available in [31] and [32]. Interactions between S-CSCF and Application server are not also discussed here.

Finally, SIP UA2 wishes to terminate the session, so that SIP UA2 sends the BYE message. In response, SIP UA1 sends the 200 OK message.

2.2 Openness and Flexibility

I assume the meaning of the *openness* and *flexibility* as follows: *Openness* means that users develop their applications or implement their new ideas (i.e. new services) with less dependability; *Flexibility* means that developed applications can be easily modified. In this subsection, the present researcher reviews the existing techniques in providing openness and flexibility and evaluates their limitations.

2.2.1 Openness

It is possible to map the high level IMS architecture as in Figure 2.3 with more emphasis on different roles. As shown in Figure 2.3, a service provider controls all the intermediate routing entities (P-CSCF, S-CSCF and I-CSCF) between caller and callee. Since services are hosted in the services layer, ISC interface is provided to the services by the SP.

Communication systems evolve from the walled-garden manner that does not include third parties, to the paradigm of open service marketplace for communication world that is influential for openness in communication systems [33]. It means that SPs open their communication system for third party usage, making end users active contributors. In this analysis, the present researcher considers only IMS based communication system that has adopted many methods to provide openness for end users or developers. The service control layer and service layer of IMS can be modeled, based on the roles of caller, callee and SP as shown in Figure 2.3.

In IMS, SPs build communication system and services using open standards, but users (tech savvy) can create and gain control over their clients, but need to highly comply with the standard (e.g. 200 page SIP standard - RFC 3261 [8]).

IMS-based SPs can provide APIs to third party or end users for new service creation. The basic way to expose the communication service functionality is via

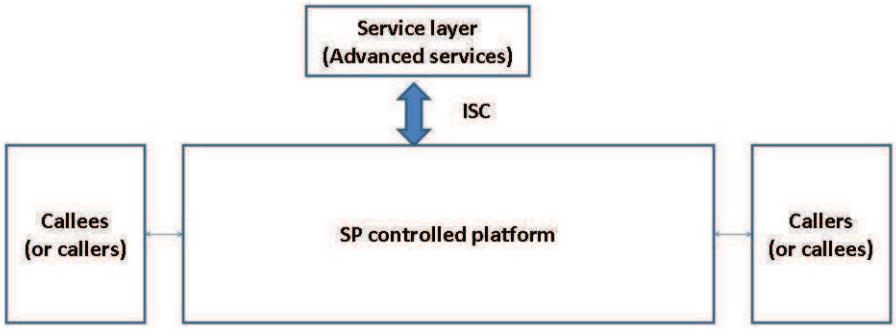


Figure 2.3: High level view of IMS based communication system. Callees and Callers are user agents. ISC stands for IMS Service Control and SP stands for Service Provider.

ISC. This API is fine-grained and makes very difficult for Web developers to develop applications. In order to empower the Web developers, many approaches have been proposed by different researchers by abstracting the existing ISC interface. I group the existing abstraction approaches into three groups: 1) Web Programming level APIs, 2) Session Data Type (SDT) API, and 3) Domain Specific Languages (DSLs).

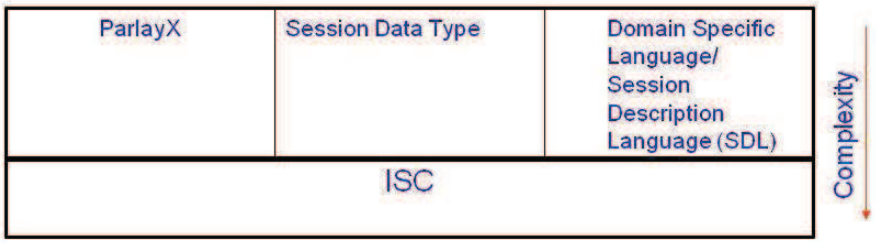


Figure 2.4: Existing abstractions for end user service creation

For more clarity, the present researcher shows all the available open interfaces in Figure 2.4. The details of each block in Figure 2.4 are given below.

Protocol level APIs (e.g. SIP): IMS provides the ISC interface to application servers (e.g. SIP Servlets) where users can implement their services. A typical example is 3PCC that can be developed based on ISC interface. Since the ISC interface is based on SIP, users should be strongly familiar with the concepts of SIP and IMS such as dialog, transaction, session or require more knowledge to manage the intricacies of the underlying technologies [34]. In addition, changes to the ISC interface need to undergo a lengthy standardization process and require proper changes in the SP controlled platform in Figure 2.3.

More importantly, this interface only provides a call control function based on a finite-state machine and does not guarantee effectiveness of routing and consistency between many interdependency nodes.

It is worth mentioning here that SIP Servlet API [35] is a programming abstraction and standardized architecture for SIP application servers that is built on the SIP standard.

Web Programming level APIs: The ParlayX web service specification [17] defines how to expose communication functions using the Web standards. This allows software developers to use the communication functions that are high level abstractions and simple to use. The available functions are Third Party Call, Call Notification, Call Handling and Multimedia Conference (not an exhaustive list).

In the Third Party Call function or the click-to-call service, the HTTP request is translated into a request to a third-party call controller to establish a SIP call between caller and callee so that it sets up a media session between them.

This simplified API supplies less capability than Protocol level APIs, but is more Web developer friendly thanks to the functional abstractions. However, in the IMS network, Parlay gateway is newly added to provide abstraction in a HTTP fashion from the signaling protocol. Moreover, to the best of the present researcher's knowledge, there is no real deployment using ParlayX standard to provide fine-grained APIs.

WIMS 2.0 proposes another way of exposing the IMS session capability (REST APIs) in order to support interactions from both Worlds, Internet and Telecom. However, users who are using these exposure APIs need to take care of the interactions between user and exposure framework. Moreover, this method does not advance further [36].

Session Data Type (SDT) API: The paper [19] argues that telco should provide a simple and expressive model that represents telco services. Therefore, many telco+web mashups can be created.

SDT has a number of primitive concepts:

1. Bubble refers to a shared conversion over a single medium.
2. Session refers to a group of related bubbles, possible over different media.
3. Event triggers are used to communicate device actions back to an application.

This approach tries to reduce effort and deep knowledge of telecommunications network compared to the ISC interface. In fact, developer users should know about abstraction of session, bubble, finite state machine and notification along with a more integrated view of multimedia communication. End users can create particular classes of advanced services in this way more easily than doing it with the ISC interface.

Examples of a shared-experience service are active conference call, family chat-room and coffer-room experience.

The SDT prototype reveals that SDT can be deployed to any communication systems (e.g. IMS based). In fact, SDT focuses on the functional abstraction (session, bubble) for end user service creation (e.g. shared-experience services).

However, for IMS service providers, this feature brings more complexity to the platform. For example, SDT framework manages explicitly the finite state machine that describes the behavior of the involved session's end points. This complexity is hidden into the session manager which provides session, bubble, finite state machine and notification to the applications.

Domain Specific Languages (DSLs): Domain specific approach offers high level abstraction that describes the domain.

In telephony domain, many domain specific modeling languages such as Call Processing Language (CPL) [37], VisuCom, Session Processing Language (SPL) [38] and Language for End System Services (LESS) [39] are available. These languages are scripting languages and are used to create call routing logic only.

DiaSpec [40] [41] is an approach to separate architecture description and implementation. In this case, users develop programs at the architecture/design level. The compilers will transform these programs for the implementation. Hence, complexity of developing new applications will be reduced. This approach is also used to develop new telephony applications [40] [41]. The new layer with high level abstractions is added on top of SIP.

In fact, these languages are created for end user services creation since abstraction reduces complexity in service development. However, these languages consider abstraction of call control features. These services can be deployed in the 3PCC controller, SIP AS, or endpoint.

Furthermore, like SDT, many of these languages do not reach the end users from research labs.

2.2.2 Flexibility

Flexibility means that changes to semantic/functionality of the signaling protocol should be an easy task. Nevertheless, achieving flexibility in the IMS architecture for new session based services is a challenging task due to three reasons.

Firstly, one-size-fits-all model is chosen for the service control layer. This means that behavior of all sessions should be similar. The core aspect of the protocol and many extensions have been defined in the standard organization based on one-size-fits-all model not based on diversity. The key reason is complexity for not considering the diversity.

The second factor is derived from the above reason. In this case, signaling protocol is static and strictly implemented in order to avoid interoperability issues.

Thirdly, the IMS architecture has symmetric peers - end points. Fundamentally, all the communication systems deliver communication services to mass users based on the end-to-end paradigm – less knowledge at the core and more knowledge at the end points, resulting in symmetric peers. For example, SIP

end points are in charge for complete message creation and processing. If there is a new header, all clients have to understand this new header to avoid interoperability issues or simply message discarding. This problem can be viewed as a version control issue.

In the end-to-end paradigm, network protocol design needs strong analysis in order to prevent the race conditions created by peers when sending and receiving messages [42]. Consequently, changes to clients require a huge amount of time and work.

Finally, changes or extensions need an agreement at the standard body (i.e. IETF) (and operators) as specified in RFC 5727 [43]. This will take a long time before rolling out new versions ubiquitously.

2.3 Network-based, Session-based Services

The network-based, session-based services interwork with a session or are executed within a session. For example, Call Forwarding on Busy (CFB) is a feature/service for existing telephone calls enabled by callee by providing the third party number. If callee/destination is busy, i.e. generally in another call, call will be forwarded to another destination provided by callee. The Call Forwarding on Busy is executed only during the call establishment. Many efforts have been devoted to compose this kind of services. This kind of problem is referred to as feature interactions [44].

There are a few techniques that discuss feature interactions during the initial media connections in different situations such as features in a single application server (e.g.Distributed Feature Composition (DFC)[45]), features in distributed SIP call control services [46], and features in Internet services and Telecommunication services [47]. Recently, the research community has looked into feature interaction problem during the mid-call [48]. However, all solutions are not general. This means the solution depends on the case.

This thesis examines two different cases that are not similar to feature interactions. The first one is that a service should be executed during call establishments and/or mid-calls. And the second one is that network-based session-based services require the cooperation of network call control and endpoints. These different cases are not discussed widely by the SIP community. Besides, developing a solution based on the SIP protocol and IMS architecture brings more complexity and requires deep knowledge of interdependency of the functional nodes in the IMS network.

This section starts explaining two unique requirements with the help of two innovative use cases for building network-based session services. The first use case is user mobility in which service logic should consistently be executed during the initial media connection and the mid-call. Another use case is partial session transfer and retrieval in which network call control and endpoints should be cooperated.

The organization of this section is as follows: initially, the user mobility subsection presents the detailed description of the use case and available solutions for user mobility based on the SIP protocol and IMS architecture. Likewise, the next subsection elaborates on the use case description and available solutions for partial session transfer and retrieval.

2.3.1 User Mobility

Introduction

The present researcher describes a use case, followed by requirements in this sub-section. An elderly inhabitant is living in a smart home which can detect the context of inhabitant (e.g. location). He receives a video call on his TV screen while sitting in the living room. During the call, he feels sleepy and walks to the bedroom. Once he reaches the bedroom, he is able to continue his video call on his bedroom TV screen without any disruption or issuing any commands. Once he finishes the call, he issues a cancel command using remote button while in the bed. When a caretaker calls, the call is directed to the bedroom TV screen. His smart house has many TV screens installed in each room and has the ability to receive and make a video call. Figure 2.5 visually presents the above-described use case.

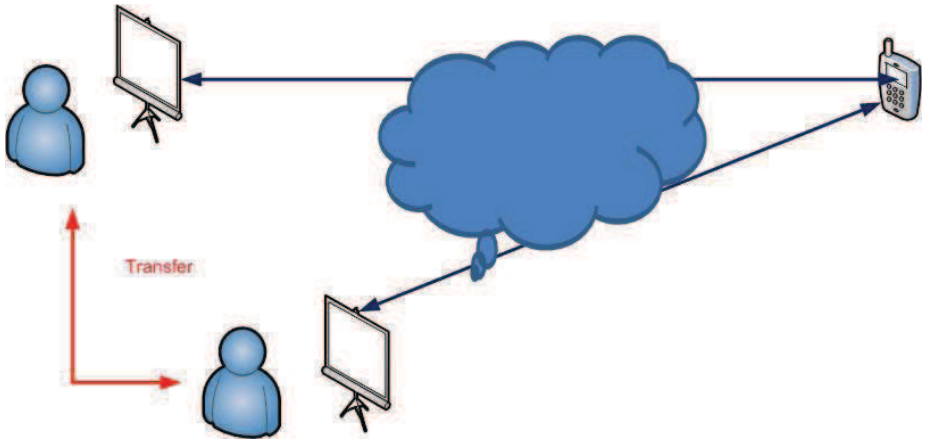


Figure 2.5: Diagram for the user mobility use case

From this use case, the following requirements are derived before discussing different solutions. For clarity, the present researcher mentions that the communication end points have audio/video processing capabilities and are connected to the Internet.

Multiple registrations: A user ('callee') should register in advance from different locations where the user moves around. This registration indicates the possible locations for transferring the calls by the network. When registering from different locations, the user uses the same username and password.

Network-initiated session transfer: During the user's movements, after receiving ambient knowledge information, the network (or application) informs the user (here, the 'callee') about a new call. If the call is accepted by the user, then media will be established. During the call, the user moves to a new communication end point. In this situation, the network informs the callee that call can be forwarded to his/her new location. Until the user accepts the call, media flow will not be disturbed. If the user accepts the call at the new end point, the network asks the caller to transfer the

media to the new end point and terminates the session with the callee's previous end point. For this use case, network-initiated session transfer gives much user experience compared to user-initiated session transfer.

Ambient knowledge: This means user context detection, distribution and derivation of high level knowledge. This use case depends on context information of a person. The context information is the location, e.g. the room the user is in. This low-level information should be derived to make a decision about which communication end point is near to the user. The framework that handles the ambient knowledge informs the communication system about the changes dynamically and provides high level information such as details of communication end points near to a user.

Existing Solutions based on SIP

There are four kinds of mobility (personal mobility [49], terminal mobility, service mobility [50] and session mobility) discussed in the literature. This use case closely involves personal mobility and session mobility. Personal mobility refers to the user's ability to access personalized mobility services that they are subscribed to from anywhere, at any time and using any terminal. Session mobility refers to the user's ability to maintain an active session while moving across networks or switching between terminals. This literature review considers the case where devices are located in the IP network, not in the circuit switched network.

Session transfer across multiple devices can be performed in two ways: partial session mobility [50] and complete session mobility. "Transferring parts of a multimedia session (e.g. video part in one device and audio part in another device) between different devices is defined partial session mobility" [50]. Complete session mobility means that signaling and media parts of a session completely transfer from one device to another device. The user mobility use case focuses only on a complete session transfer.

The SIP specification has defined mechanisms related to personal and session mobility. The present researcher considers personal mobility at session initiation and session mobility during a session. In order to deliver an integrated solution, the existing standards recommend to deploy service intelligence in a forking proxy, end points and a SIP AS separately. The service developer should consider a proper coordination mechanism among these entities.

Personal Mobility at session initiation SIP network has a stateful proxy (Forking proxy) [8] that implements three different mechanisms: sequential, parallel and pipeline [51] for personal mobility at the session initiation. This approach pre-requires multiple registrations of user agents to the SIP network and needs to resolve the address of record with different contacts. This service poorly utilizes the resource (many SIP messages over the network) in a parallel search and involves a long delay for call set up in a serial search. Integration of knowledge of user mobility is important for delivering this service. The work reported in [51] models the user mobility and derives the values for making the best decision in the forking. The other work related to making a decision in forking is that receiving devices

are grouped into active and standby in order to reduce delay and amount of message traffic [52].

The articles [53][54][55] indicate that getting the context information and integrating with communication services are feasible and useful. For example, in [53], using Radio Frequency Identification (RFID), a solution for personal mobility (not session continuity) is proposed for a SIP network that resolves RFID with SIP user identification. This solution avoids multiple registrations and reduces changes in the IMS network. In [56] [57], the authors propose a special proxy (loosely coupled application interaction proxy [56], context-aware SIP proxy [57]) for the SIP network. The efforts behind these new proxies are orthogonal to this thesis because they involve semantically enhanced interaction models and context aware service adaptation.

In the present researcher's proposal, context enabler is proposed separately and it provides a functional interface to the communication system. The information via the functional interface is used to make decisions in routing calls at session initiation and during a session. Therefore, two different entities, forking server and AS do not need to interact with the context enabler.

Session Mobility during a session Complete session mobility can be performed by either a user agent (UA) or a network entity. The present researcher provides the literature review based on user-initiated transfer and network-initiated transfer.

UA- initiated transfer: REFER method [27] is proposed to transfer calls from one device to another device by an action of the user. The user should know the identity of the target UA and make the transfer using REFER method and Refer-To header. This method allows moving signaling and media paths together and has join and replace headers. The REFER mechanism is hostile to ASs in the network. It means that ASs in the old and new signaling paths are different.

This method does not improve the user experience since it requires the user's involvement in knowing the target device addresses and in performing the transfer action.

In UA-initiated transfer in SIP, all the devices use a dedicated Address-of-Record (AOR), such as sip:user1@example.com. In this use case, users log into the system using a single AOR (username). Therefore, mid-call session transfer gives an undesirable result as the initiator sends a REFER message with the same AOR for REFER method and contact header. Alternatively, if instances for the same AOR can be differentiated using terminal contact or Globally Routable UA URI (GRUU) [58], performing the session mobility will become easier.

Network-initiated inter UE session transfer: IMS network deploys SIP ASs for initiating and managing sessions across many devices. In this case, one SIP AS performs as Back-to-Back User Agent (B2BUA) or executes third party call control by sitting in the signaling path.

This approach relieves users from dealing with session transfer. Besides, network based application is easy to integrate with other services such as context information (presence, location) in order to provide rich communication services. However, scalability is a show stopper in the B2BUA approach. It means that single application can not serve for all the users who can access simultaneously. Further details about scalability are available in Section 2.4 and Chapter8.

In addition, B2BUA manages the session transfer based on SIP addresses. For the proposed use case in this thesis, a single user logs in from different user agents using the same user ID. As a result, B2BUA applications have to depend on the forking functionality from the SIP network. Due to this dependency, a B2BUA application needs to first close the session and then it has to restart the session during the session transfer, so a hard hand off. Otherwise, B2BUA should depend on terminal contact or GRUU to support the soft hand-off.

Alternative Solution As discussed earlier in this section, solutions for personal and session mobility are working separately. In fact, current IMS specifications do not provide clear design guidelines for the user mobility use case and as a consequence, this thesis has to develop a workaround solution for the IMS architecture. A possible alternative solution is to implement the logic for personal and session mobility in B2BUA of IMS. It means that B2BUA is able to do forking and/or session continuity.

One assumption is made that each user in the network has a unique B2BUA application that implements the personal and session mobility. Thus, this solution is free of the scalability problem. In this alternative solution, S-CSCF in the IMS architecture should scan the incoming session request (e.g. INVITE) messages and forward them to the right B2BUA applications. Hence, this method requires at least a B2BUA application, and an S-CSCF. In this case, S-CSCF should process at least six SIP messages for address binding and message routing - three (INVITE/ACK/OK) messages between B2BUA and callee and between B2BUA and caller, as well. In addition, S-CSCF can be stateful (depending on the implementation) and well-defined routing decisions should be implemented in the IMS network. As indicated in [59], each stateful transaction request requires additional memory and CPU cycles compared to a stateless transaction request.

As explained in the previous paragraph, this alternative solution suffers from inefficient SIP routing or going through S-CSCF to reach the B2BUA application. The solution presented in this thesis eliminates the overhead (delay and processing power, etc) in S-CSCF and does not worry about routing decisions. Rather, it depends on simple HTTP. In addition, this thesis intends to reduce session set-up time and session transfer time (session mobility).

This alternative solution and the thesis proposal compare in terms of development and complexity. In addition, I provide an evaluation of scalability in Section 2.4 and Chapter 8.

2.3.2 Partial Session Transfer and Retrieval

Introduction

Ubiquitous digital devices with rich media processing and networking capabilities open an avenue for enriching user experiences, especially in video voice over IP communication sessions because a communication session can be partially transferred to multiple devices that are in the vicinity of caller and (or) callee. Then, the transferred partial session can be retrieved back to the original place. For supporting partial video voice session transfer and retrieval, the solution needs a proper orchestration for the media channels. In this case, the initiator of partial session transfer and retrieval is a network entity that has service intelligence or users (caller/callee). For example, a session can span (or transfer) over many devices for different media streams as presented in [50] (i.e. partial session mobility).

The solution for the above-mentioned complex scenario of Partial Session Transfer and Retrieval (PSTR) should include a proper orchestration mechanism for media streams, service/device discovery and negotiation [50]. An orchestration mechanism establishes right media streams across media devices regardless of the originator (network side or users).

This thesis focuses on the orchestration mechanism. The existing solutions are divided into either user side [60] or network side [50] mechanism. In this sub-section, a review of the existing solutions for PSTR based on Session Initiation Protocol (SIP) is presented. Then, the present researcher highlights the weakness of the integration of universal plug-and-play (UPnP) devices with SIP user agent for PSTR.

Existing Solutions for PSTR based on SIP

The current solutions for PSTR do not provide a single orchestrator to support network-initiated and user-initiated PSTR (based on initiator). Therefore, this thesis classifies the existing solutions into two types: network-initiated, and user-initiated.

Network-initiated: In this kind of solution, a network entity (e.g. application server) should take care of partial transfer and retrieval at both sides of a session (caller/callee). To the best of the present researcher's knowledge, there is only one solution [50] that addresses the use case (not all its features). The detailed comparisons of the solution are as follows:

In [50], the authors propose a SIP B2BUA application, called a sub-session controller (SSC) that performs partial session mobility based on a modified SIP mechanism (INVITE-based). This approach defines two new headers called pst-to and pst-call-id; these headers are used with the INVITE method to inform the user about the partial session transfer (PST). If new headers are not implemented in the UAs, it is difficult to benefit from this kind of service. In the approach used in this thesis, changes can be easily made because user agents (which are identified as caller and callee boxes) are downloaded at run-time [13] [61].

When a Mobile Node (MN) retrieves the session, MN sends the INVITE message to the Correspondent Node (CN). Throughout the document, MN and CN stand for caller and callee or vice versa. In this case, the SSC

may not be aware of this retrieval. The MN closes the opened connections by sending the BYE message to the auxiliary devices that are involved in partial session mobility on the MN side. This logic should be implemented in the MN in addition to the SSC.

This approach [50] does not separate the different aspects (goals of PSTR and session initiation/modification). Instead, it uses (RE-)INVITE to inform users about session-transfer approval. Therefore, this approach adds complexity to UE development and further maintenance.

The SSC approach does not provide adequate partial session mobility at both MNs and CNs. In my approach, the network box performs all the media control for the PSTR (Ref Section 7). The present researcher's solution leverages the protocol [62] for media control and defines its own messages that instruct the transfer and retrieval between the network and the caller/callee boxes.

Furthermore, services developed based on the B2BUA method are encountered with the scalability problem [50].

User-initiated: In this kind of solution, caller/callee should take care of partial transfer and retrieval at both sides of a session (caller/callee). To the best of my knowledge, the basic idea is presented in RFC 5631 [60]. RFC 5631 proposes two different transfer modes: Session Handoff (SH) and Mobile Node Control (MNC). The focus is on enabling the media transfer based on the end-point (SIPUA) [60]. Nevertheless, some efforts are devoted to enhance the session handoff method further. This thesis reviews all the techniques presented in RFC 5631, followed by extensions based on the session handoff method.

For the SH mode, an MN sends a REFER request to a local device (SIPUA) that can participate in the session. For transferring partial sessions to many local devices, the MN should send multiple REFER messages. However, the sending of many REFER methods is not supported by the existing SIP standards. Hence, [63] proposes a new entity called Multi Device System Manager (MDSM) that acts as a 3PCC controller between the CN and the local devices. As a result, MDSM encounters the same problems as the MNC.

In MNC, an MN implements the 3PCC [18] for PSTR. When the MN and the CN perform 3PCC, it will increase transfer/retrieval delay during the PSTR. This means that MN and CN need to send a RE-INVITE message for every session transfer and retrieval (i.e. new media end-point descriptions). If any of these events simultaneously occur on both sides, there will be a race condition as reported in [64]. Moreover, MN and CN should implement UA and B2BUA, and thereafter, MN and CN can support PSTR. Obviously, this approach is very complex for development and implementation. In [63], it is not explained how a user interface is designed to show the transfer and retrieval based on the 3PCC approach.

Similar to the SH mode, [65] and [66] present different solutions for transferring and retrieving a partial SIP session over multiple devices using the modified REFER method. To split a SIP session, the authors in [65] propose a new header called "Mobility" and a new concept called "Association". In this approach (i.e SSIP), MN establishes a session with CN.

During the splitting, the MN sends a REFER request to a free node with a mobility header. Later, the free node sends a new invite message with the mobility header to the CN. In this case, the CN should identify that there are many sub-sessions within a single session. For terminating the session, the CN sends an individual BYE message based on association.

The mobility header in the REFER method allows the referee to be informed about the session medium by the mobile node. The main drawback in the SSIP method is that it does not support the case when both the caller and callee sides start transferring the partial session. This means that session transfer and retrieval is considered only in the MN and not in the CN. In the approach presented in this thesis, both end-points can perform session transfer and retrieval at any time; the present researcher explicitly defines the free nodes based on media capabilities (Ref subsection 7.1.2).

For another example, the SSIP concept is extended to support retrieval using the nested REFER method [66] [67]. To retrieve a transferred session back to the mobile node, four messages must be sent between the mobile node and the free node because of the nested REFER message. In my approach, only two messages need to be sent between a mobile node (caller and callee boxes) and the network box.

Integration of UPnP devices and SIP UA

UPnP: Interaction between control point and media servers/renderers: UPnP supports ad-hoc networking of devices and interaction of services [68]. The UPnP protocol stack includes addressing, discovery, description, control, eventing, and presentation. To the best of the present researcher's knowledge, UPnP has only defined the interaction between a control point, and media servers/media renderers for media sharing in the UPnP AV specification. Even though ConnectionManager (CM) Service and an optional AVTransport Service (AVT) (depending on the supported transfer protocols) are defined in the UPnP standards, these standards do not have solutions for the integration of media servers and renderers for PSTR (in real-time communication services). One work has been reported by integrating the UPnP devices and SIP UA for PSTR. The following paragraph discusses the work.

Integration of UPnP devices and SIP UA: In Sub-Section 2.3.2, the media orchestration when all end points are SIP UA, was initially presented. Now, the present researcher reviews the integration of UPnP devices and SIP UA for PSTR. The UPnP protocol enables control of the devices connected to the network. For this integration, a new device called VVoIP - (a SIP UA and a UPnP control point for interacting with UPnP media servers and UPnP media renderers) is proposed [69].

This solution [69] has five drawbacks.

1. During the PSTR, a SIP UA should send the Re-INVITE message. If both ends send the re-invite message at the same time, then there is a race condition [64]; it has the same problem as MNC.

2. Development of VVoIP requires a very large work investment - the proper linking of the SIP UA and the UPnP control point.
3. The UPnP AV standard was not developed for the real-time communication services. For example, changing the codec during a session, VVoIP requires a work-around solution.
4. This solution is available only in UPnP-enabled networks.
5. UPnP devices have synchronous interfaces (HTTP/SOAP), therefore, they are not better positioned for real-time communication services.

In this thesis proposal, devices (e.g. UPnP media servers and media renderers) implement the media control logic (open, modify and close). A central entity manages the session based on the method in [62], among the devices, caller and callee. The solution developed in this thesis simplifies integration in which all media servers and renderers are media end-points (Ref Chapter 7).

2.4 Scalability

The last two sections (Section 2.2 and 2.3) present the related work for the first two research problems of this thesis. In order to discuss the third problem and the related work, this thesis starts with a new definition of scalability associated with the signaling layer. Next, the present researcher introduces an interested IMS architecture and discusses possible solutions in order to support the scalability in the two following sections. Then, based on the proposed definition, architecture and possible solutions, a detailed analysis for scalability is provided. Finally, there are some complimentary work. Since they are not mature and efficient, they are not taken into account for the comparison.

2.4.1 Proposed Definition of Scalability

Scalability is an important attribute and is taken into account when designing a network, system or process/algorithm. Poor scalability of any network or system is considered to provide poor system performance.

The term scalability has been defined based on the context in the literature [70], [71] because of different intrinsic properties and behavior of the systems. Different systems (e.g. information retrieval system/communication system) offer different definitions.

Determining factors for scalability are not always clear, and may vary from one system to another. As a consequence, scalability is not defined properly in literature by definition. To fill this gap, [71] attempts to classify scalability at a high level, reflecting the four different aspects: structural scalability, space scalability, space-time scalability and load scalability. According to [71], "Structural scalability is the ability of a system to expand in a chosen dimension without major modifications to its structure." Load scalability is the ability of a system to perform gracefully as the offered traffic increases. 'Performing graceful' means performing without undue delay and with unproductive resource consumption or resource contention at different loads (light,moderate and heavy)and making good use of available resources.

The main question here is: Is it possible to model the communication system to suit a situation where the number of calls increase? In other words, it is a question of load scalability. Therefore, this thesis concerns of load scalability. A system with poor load scalability has different states such as graceful function and overloaded. When an overload situation arises, it is possible to have a deadlock. This kind of situation may happen in an IMS with poor scalability. More relevant information on overload in IMS has been given in Section 2.4.4. The ineffectiveness of IMS in handling growing number of calls takes toll on the resources.

One general idea presented in [71] is that performance and scalability are closely intertwined and need not be considered separately. In fact, this thesis takes this bottom line idea when defining the scalability for signaling architecture. In addition, when exploring different scalability, it is possible to see that they are independent completely and sometimes overlap, as well.

The scalability aspect can be further explained with help of operating system and local area network. Ethernet and Token Ring are good examples of poor load scalability. For instance, effective usage of bandwidth in an Ethernet decreases at heavy loads, because of high collision rate within Ethernet.

Another example of load scalability is information retrieval system. A large-scale information retrieval system has many parameters that should be considered while dimensioning [70]. Google search engine is an example of information retrieval system which crawls the Web and indexes pages against words. Users pose requests and retrieve relevant answers from the system.

The important parameters are:

1. Number of documents indexed: The number or size of the documents is indexed.

2. Queries/sec: The number of queries that must be handled per second by the system.

3. Index freshness/update rate: Sometimes pages become old and not relevant. Additionally, some pages are expired and newly developed. Index refresh rate is important to tackle this kind of situations.

4. Query latency: The amount of time taken by the system to respond a user's request.

5. Information kept about each document: The number of key words used to describe each document. Some words are not necessary to describe the document.

6. Complexity/cost of retrieval algorithms: The type of ranking algorithm used for retrieval and its associated cost/complexity.

Another work related to this thesis is reported in [72] that attempted to evaluate SIP server clusters developed by different software vendors for availability and scalability. Here SIP servers are either a proxy, a user agent client, a user agent server, or a B2BUA. Three criteria are considered to validate scalability. They are:

1. Scaling Increment: Resources should be added at small increments, so, it permits fine-grained tuning of the cluster.

2. Operational cost of scaling: The operational cost incurred to deployers should be minimized while scaling a cluster.

3. Scaling efficiency: Efficiency of usage of additional resource should be high.

The above mentioned three parameters are used to evaluate the scalability of a SIP server cluster. In fact, this thesis ascertains scalability across any communication system (IMS, P2P SIP, etc). On the other hand, these three parameters can not be used to evaluate the scalability of the signaling layer as they do not reflect the global view of the signaling layer. All the same, the core aspect is taken into account. For instance, the operation cost of scaling is considered as complexity in the present research.

In this thesis, scalability means that a communication system is able to cope with a concurrent high number of calls. It is simply load scalability according to [71]. To the best of the present researcher's knowledge, there is no proper definition for the scalability that is linked with a signaling architecture. This thesis intends to establish a common framework for characterization of scalability that is linked with a signaling architecture. In other words, considering a signaling architecture as a whole, but not the scalability of an individual component in the signalling architecture, is the goal of this thesis

Proposed Definition for Scalability

The following four intrinsic properties of the signaling architecture are considered as requirements for the definition of scalability.

1. The number of users that is connected by the architecture: Typically this means total number of users that are registered to the network.

2. Which entities, except endpoints, handle session management: This is important to create network-based session services, for the reason that these entities keep session state information and demand computing resources.

3. The call rate: The number of successful calls that can be served by a system at a time.

4. Session duration: The length of each session.

Based on these four different requirements, I propose four properties - *how scalable*, *complexity level*, *needed computing resources* and *session setup latency* - to the characterization of scalability. The details of each properties are presented as follows:

How scalable means that up to which point a signaling architecture performs without failure. For example, architecture will work as long as any number of users is added into the architecture. This refers to the factor limits scalability.

Complexity level means that when a call rate varies, any new functional entity should be added into the architecture and it involves the level of easiness/difficulty in coordinating all the entities to accomplish goals.

Needed computing resources refer to the costs towards the overall system in terms of computing resources (e.g. CPU, memory).

Session setup latency is an important parameter for communication services. This parameter enables to show how latency is influenced during the scale up and down.

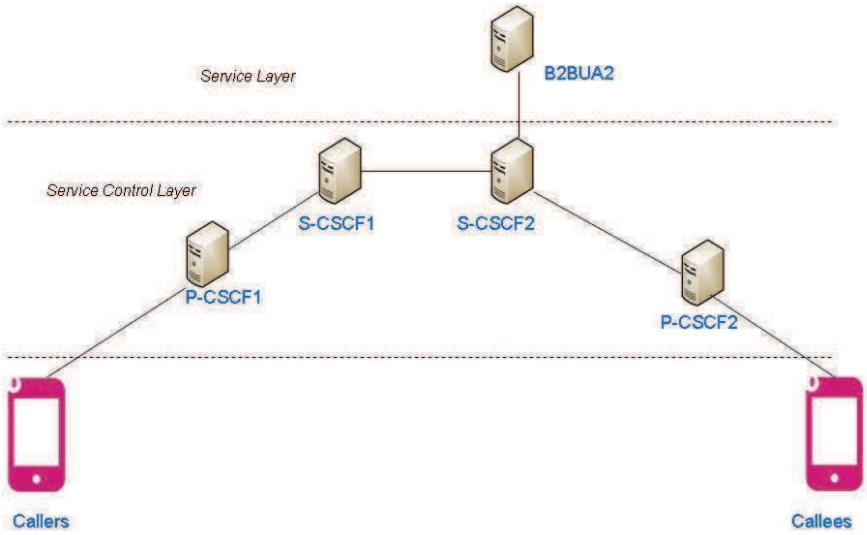


Figure 2.6: IMS architecture connecting two different administrative domains (suffix 1 and 2 denote two domains.).

In any case, there are some interesting properties (availability/reliability) of the system, which are not explicitly considered with scalability.

2.4.2 Interested Architecture

Based on the above listed four parameters, I compare the signaling architecture when it experiences a low to high call rate (billion per second). This signaling architecture is assumed to serve for the entire population of the world (assuming 6+ billions of people).

The IMS signaling architecture is grouped into two layers, a service control layer and a service layer. This thesis considers the architecture for the scalability analysis shown in Figure 2.6 in which entities are involved in executing network-based session services, for example user mobility. Figure 2.6 also shows two different administrative regions that are marked with suffix 1 and 2 in proxies.

The proxies in the service control layer is responsible for routing, address resolution and session management. These functions are deployed in proxies that are shown in Figure 2.6. Even though some other factors (for example, location server) may impede the scalability in the service control layer, only the routing aspect is considered by the present researcher, here.

To route SIP requests from SIP User Agent (UA)1 to SIP UA 2 (similar to caller and callee in Figure 2.6), IMS specifications have defined three proxies, P-CSCF, I-CSCF, and S-CSCF. I-CSCF is not shown in Figure 2.6. These SIP proxies perform host discovery, routing, maintaining states (for retransmissions, accounting etc.) and authentication.

During a UA registration process, I-CSCF assigns the S-CSCF. Thereafter, P-CSCF sends the SIP messages directly to the S-CSCF. In this case, I-CSCF acks as a load balancer. This is different from the dynamic assignment (see

Section 2.4.3) that performs load balancing during the session initiation.

For routing messages, the following operation modes demand different computing resources (CPU resources) in a proxy [59]:

1. Stateless with No Lookup: During the handling of session messages, session state is not kept. This operation depends on the message that has sufficient information (IP address within the SIP URI of the endpoint) to be forwarded.

2. Stateless with Lookup : Handling of message takes places as in the first case, but for forwarding messages, lookup is performed to map the URI to an IP address.

3. Transaction Stateful with Lookup: Session state is kept only for individual transactions. Also, lookup is performed to map the URI to an IP address.

4. Dialog Stateful with Lookup: Session state is kept during the call, consisting of multiple transactions. Also, lookup is performed to map the URI to an IP address.

5. Dialog Stateful with Authentication: While session state is kept during the call, proxy needs to validate the credentials of the clients.

In addition, a proxy needs additional computing resources to maintain states [59]. Therefore, the present researcher assumes that S-CSCF only performs dialog stateful, and that the other proxies (P-CSCF and I-CSCF) are stateless. This means that each call needs one dialog stateful with lookup and one or more stateless dialogs with lookup actions. This helps to reduce the needed computing resources in the intermediate entities for a single session.

2.4.3 Existing Solutions

In IMS, state of a session can be memorized in intermediate proxies while forwarding the SIP messages. Essentially, messages in a session should traverse via the same proxy in order to be consistent. These two aspects are reported in Sub-Section 2.4.2. Considering the above mentioned two aspects, this subsection presents two solutions in IMS, pre-defined and dynamic assignment, which can support a higher call rate. The details of each solution are provided hereunder.

Pre-defined Assignment

Based on the capacity of a physical node (like one in [73]), users are pre-defined to one physical node. It will exist for a long period whether the user makes calls, receives calls or not. This can be achieved in two ways. One is by making settings in DNS. This is done by allocating a set of users to one server, by default. For example, from user1@domain1.com to user500@domain1.com is statically defined to one physical node (assuming that to be the maximum capacity). The other is based on the unique domain names for a set of users, like pre-defined-numbersofusers@domain1.com and pre-defined-numbersofusers@domain2.com.

These servers only serve for assigned users and session set up delay is constant (with an average of three hops). Since call arrival rate varies (low to high

over time), keeping the physical nodes for signaling purpose is not an economically viable solution (over-provisioned). To address this problem, assigning a server is dynamically considered when user requests arrive.

Dynamic Assignment

When many requests arrive simultaneously, all the requests should be divided into different servers based on the servers' capacity. The users are dynamically assigned to proxies with the help of a Load Balancer (LB). In this setup, there are many proxy physical nodes. Based on the algorithm, each user is assigned to a particular server by the load balancer. User requests go first to a load balancer that forwards the request to a right proxy. It is important that transactions corresponding to the same call must be routed to same server due to the session-oriented nature of the SIP. This aspect should be taken into account when employing the load balancer.

The LB is either fine-grained or coarse-grained. In a session, all the requests and responses go through the LB; then it is referred to as fine-grained. Figure 2.7 shows load balancer deployed between users and P-CSCF1. In coarse-grained LB, the initial message (e.g. INVITE) goes through the LB and then the subsequent messages for the same session go directly to the proxy that is selected by the LB; a main challenge is to collect instantaneous information from proxies for effective decision making in LB.

As per the literature, load balancing across many nodes can be achieved by many different algorithms, such as Hash, FNVHash, Round Robin, response-time Weighted Moving Average (RWMA), Call-Join-Shortest-Queue (CJSQ), Transaction-Join-Shortest-Queue (TJSQ), and Transaction-Least-Work-Left (TLWL)[21]. Information regarding the above algorithms is as follows:

1. Hash and FNVHash: This approach is static where the central entity, let's say load balancer, assigns a new call to server (Hash(x) mod N), where Hash(x) is a hash function of Call-ID x and N is the number of servers. More details of FNVHash are available in [74].

2. Round Robin: The central entity assigns a new call to each server rotationally. Hence, load is distributed equally across servers. Based on the previous call assignment to server M, the next call is assigned to server (M + 1)modN, where N is again the number of servers in the cluster. In the Hash method, it is not possible to guarantee equal load across the servers.

3. Response-Time Weighted Moving Average: This method helps to make load balancing decisions based on server response times. [75] provides details of Response-time Weighted Moving Average (RWMA) algorithm that assigns calls to the server with the lowest weighted moving average response time of the last n response time samples. The main idea of this method is to make load balancer responsive to dynamically changing loads. The load balancer can determine the response time for a request based on the time when the request is forwarded to the server and the time the load balancer receives a 200 OK reply from the server for the request.

4. Call-Join-Shortest-Queue: The central entity tracks the number of calls and assigns a new call to the server that has the least number of active calls.

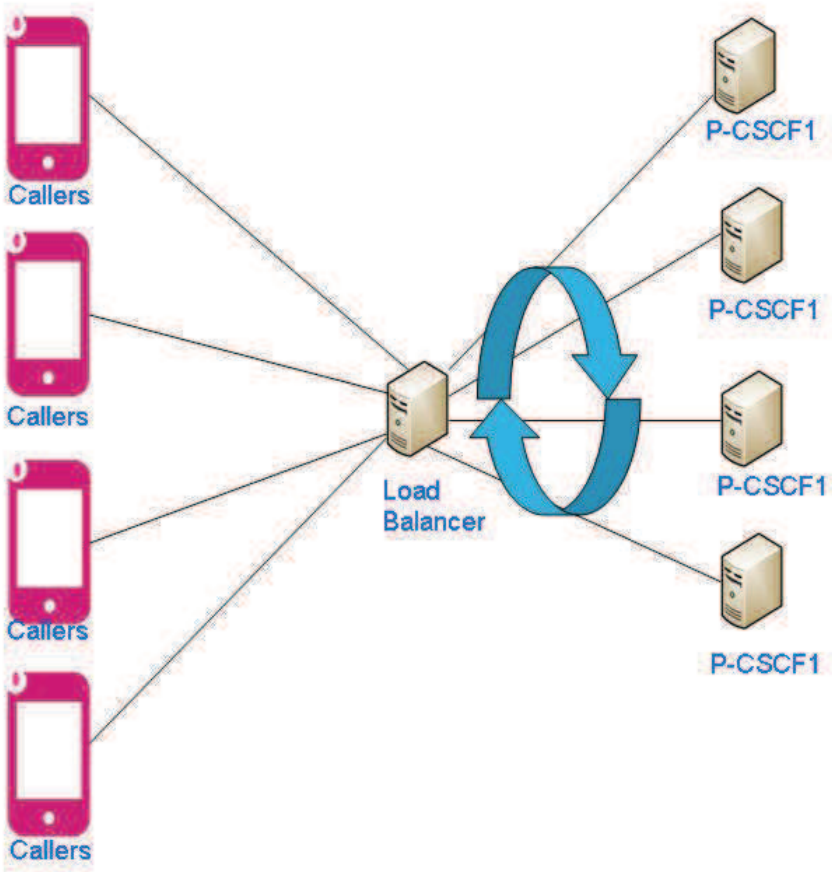


Figure 2.7: A load balancer between callers and P-CSCF1. Network admin 1 has many physical nodes for a single functional entity, P-CSCF

5. Transaction-Join-Shortest-Queue: SIP has different transactions such as INVITE and BYE. In this case, instead of considering a complete call, the central entity assigns a new call to the server that has the fewest active transactions.

6. Transaction-Least-Work-Left: In this case, each transaction is given relative estimation of a transaction cost. Typically, an INVITE transaction needs more work than a BYE transaction. The central entity assigns a new call to the server that has performed least work. This means that the central entity should know the relative overhead of each SIP transaction.

The present researcher describes two different load balancers and the algorithms employed before getting into the scalability analysis. This way, readers can understand the complexity when supporting scalability in IMS.

1. Fine-grained Load Balancer:

   The authors in [21] propose an algorithm (select a server for handling requests) called Transaction-Least-Work-Left (TLWL) and deployed with a

fine-grained load balancer. Based on this algorithm, a new call is routed to the SIP proxy that has the least work. In this case, the load balancer knows in advance the number of transactions each server can handle and the cost of each transaction. The authors found that 1.75:1 is the cost ratio between INVITE and BYE in a SIP session.

H. Jiang et al claim that TLWL-1.75 can linearly increase throughput when the number of nodes increases (up to 10 nodes) [21]. In the TLWL approach, the load balancer is fine-grained (i.e. all requests and responses pass through the load balancer) and is compatible with the SIP (can deal with variability in call lengths, distinguish transactions within calls, and can assess different processing costs for different SIP transaction types).

A fine-grained load balancer requires high computing resources (processes all incoming and outgoing messages) and increases the session setup delay (by one hop). Another problem is the scalability of a load balancer. This problem becomes recursive; a solution for increasing the capacity of a load balancer must be found. Consequently, the scalability presents a major bottleneck for real-life deployment with high call rates (in thousands).

2. Coarse-grained Load Balancer:

   In this case, after the first message, all the messages are routed to the proxy without going through the LB. For effective decision making at the load-balancer, the LB should monitor all the available servers over the time whether assigned jobs in proxies are finished or not. In this case, the heartbeat mechanism is used to convey information between proxies and the LB as explained in [76]. There is a considerable overhead in sending information between proxies and the LB. The effectiveness of the heartbeat mechanism in the context of signaling architecture remains a question. More importantly, the present researcher has not found any work (with results) that is linked to any algorithm (like round-robin) with a coarse-grained load balancer including in [76].

   Coarse-grained LB can scale well compared to fine-grained because the coarse-grained LB only processes the first message of each session. However, both approaches consider only in balancing the originating requests from UAs. As per Figure 2.6, callees are assigned to a particular B2BUA, S-CSCF and P-CSCF. In this case, callees are willing to receive call via this particular setup. It is not clear how to use (both) load balancing techniques in this situation.

2.4.4 Scalability Analysis

The proposed scalability evaluation framework has four parameters as presented in Section 2.4.1. Additionally, two different solutions based on IMS are presented in Section 2.4.3. This sub-section presents the scalability analysis based on the proposed scalability evaluation framework.

How Scalable

The scalability can be achieved by deploying needed resources in advance based on the pre-defined approach. But scalability is a bottleneck with a load balancer in the dynamic assignment approach especially during a high call rate. The

capacity planning for the IMS signaling architecture is an important aspect in both approaches. If the capacity of an IMS network (i.e. capacity of each proxy) is not sufficient during the high call rate, the IMS network experiences overload and is likely to cause throughput collapse [22]. In addition, recovering the throughput is also very slow.

To avoid throughput collapse in overload situations, several suggestions for the overload control method have been presented [77][78] (not a complete list). Before applying these prevention mechanisms (i.e. preventing throughput collapse), the IMS network should know when and which proxies are under overload. Besides, it is very clear that these overload control solutions are complex and demand computing resources for implementing the overload control mechanisms. More significantly, when deploying the dynamic assignment approach, more efforts should be devoted for the overload situation and throughput collapse as these factors affect the scalability.

Overload control algorithms, for example in [77], demand rejecting messages during the demand burst situation. However, the cost of rejecting a session is nearly equivalent to the cost of serving a session [78]. In a nutshell, turning computing resources used for rejection and overload control mechanisms (e.g. [77]) is into serving the calls

Complexity Level

The pre-defined approach is straightforward and complexity is only associated with properly defining the SIP routing rules. On the other hand, complexity is related with the load balancer and load balancing algorithm in the dynamic assignment approach. At the same time, overload control mechanisms also add complexity in developing the overall system.

Needed Computing Resources

This is associated with the number of users connected to the network and the number of intermediate entities between callee and caller. In IMS, minimum three nodes (P-CSCF1 and S-CSCF1, B2BUA2, and P-CSCF2 and S-CSCF2) are between callee and caller (see Figure 2.6). If deployed the dynamic assignment approach, the load balancer is also added into the overall system. During the low to moderate call rate, the dynamic assignment approach effectively uses computing resources when compared with the pre-defined approach.

Session Setup Latency

As mentioned in the previous sub-section, session setup latency is also associated with traversing three nodes (See Figure 2.6). In the dynamic assignment approach, the load balancer also contributes to the latency.

2.4.5 Indirect Work

The present researcher describes a few recent studies that are more relevant to scalability. The possible ideas - SERVaruka presented in [59] and a self-organization concept to IMS - intend to address the scalability problem. Anyhow, these solutions are not matured. Then, this thesis presents a discussion on

the different approaches that augment the SIP server performance. Nonetheless, these approaches are complementary to the present research.

SERVaruka

The authors [59] propose an algorithm to SIP servers that determines optimal fraction of the SIP requests to be handled statefully in a SIP server in order to increase the call throughput. The amount of state locally maintained is the main idea behind this algorithm. The algorithm dynamically shares the states amongst servers. The experiment results show that scalability is not linear; there is a 15 percent increase in it. The state distribution algorithm for any configuration of servers achieves optimal call throughput in the cost of distributing states dynamically. SERVaruka does not globally address the problem of load scalability while injecting complexity (due to algorithm implementation) in the deployment.

Self-Organization Concept

Self-organization concept promotes to give the control to the network; so that it re-organizes itself quickly when load (number of calls) increases in the network. Generally, designing the right algorithm for such tasks involves a high intellectual cost.

IMS is an operator-controlled IP-based signaling infrastructure that has many functional components and is associated with many interactions with those functional components. The authors propose architecture based on self-organization concepts in order to reduce the operational cost [79]. I consider that this work also fits into the domain of scalability because the proposal has the bottom line of network loads, number of users, and available system resources. The main idea of this proposal is to merge and split functional components among nodes as network load increases [79]. Compared with the proposal presented in this thesis, this solution involves high cost in developing an algorithm, lacks in effectiveness and adds more complexity into the overall IMS system (adding new entity in the message path and reconfiguration cost with S-CSCF and P-CSCF).

More importantly, this proposal does not indicate the relationship between the number of concurrent sessions and needed system resources.

SIP Server Performance

Individual SIP proxy performance is an important factor for increasing the scalability. The fact is that throughput of a stateful SIP proxy is equivalent to half of the throughput of a stateless SIP proxy server. Therefore, SERVaruka proposes a state distribution algorithm in order to increase the overall throughput at a time. However, there are a few techniques to increase the throughput/scalability performance of a single proxy server that is hosted in different environments (e.g. multi-core processor) or designed and implemented in different ways (e.g. SIP parser, concurrency mechanisms - thread and process architecture). They are:

I. SIP parsing: This takes a significant percentage (20% to 40%) of CPU time [80] since SIP is CPU-bound. One method is to separate (or offload)

the SIP parser from SIP stack, so that throughput can be increased. Here, the complete list of the SIP parser is not included, but more information is available in [81].

II. Making SIP server as a bare PC application [82]: A BarePC approach intends to reduce the overhead in the OS. Generally, BarePC applications run without support of an operating system and/or kernel. Here, SIP proxy should handle SIP functions and messages processing, efficient CPU tasking and direct Ethernet-level data manipulation. Therefore, SIP proxy performance scalability can be increased compared to conventional deployment. Additionally, this approach brings much complexity into development.

III. Multi-process programming model [83]: SIP proxy server, running on a multi-core processor, should consider two factors in order to determine performance scalability. They are :

    1. In side the operating system, overhead using the coarse-grained locks in the UDP socket layer.
    2. Multi-process programming model.
        1. overhead from passing socket descriptors among processes.
        2. overhead with sharing transaction objects among processes.

        Additionally, the authors have proposed several incremental optimizations for these issues.

This thesis benefits from these performance optimization approaches and considers these mechanisms when designing the underlying service infrastructure.

2.5 Conclusion

Three perspectives of the IMS architecture are reviewed and the weakness of the existing solutions have been pointed out in this chapter. Initially, the basic idea of the SIP signaling protocol, IMS architecture and session establishment and termination in the IMS architecture are presented in Section 2.1.

Then, the existing mechanisms for providing openness and flexibility in IMS are presented in Section 2.2. This thesis ascertains that the idea of controlling the signaling infrastructure is a major challenge in providing openness to end users. The existing research work focuses on the abstraction of the SIP protocol interface. Significantly, the blue print of the signaling infrastructure - one-size-fits-all-model - hinders the flexibility in the signaling layer. IMS is not feasible to extend or modify in order to provide more openness and flexibility because of its own complexity.

Even though basic voice and video call is enabled in IMS, IMS does not natively support the development of network-based session-based services. This aspect is explained with help of two use cases: user mobility and partial session transfer and retrieval, in Section 2.3. Coordinating properly forking proxy and B2BUA is a problem for user mobility use case in IMS. For the PSTR use case, deploying the 3PPC mechanism in each end point and SIP AS is needed. All the

same, the solution for PSTR in IMS does not support seamless partial session transfer and retrieval at the callee and caller sides.

Finally, this chapter discusses of the scalability aspect with a signaling layer in Section 2.4. This section starts with different definitions for scalability and defines a new evaluation framework for scalability associated with a signaling layer. This evaluation framework has four parameters - how scalable, complexity level, needed computing resources and session setup latency. Based on this framework, it is possible to evaluate any signaling architecture. Following this evaluation framework, Sub-Section 2.4.2 presents a possible architecture that should be considered for the scalability analysis. Existing solution - dynamic assignment in Sub-Section 2.4.3 - in order to support scalability still needs a proper load balancer and algorithm. This new evaluation framework is used to evaluate the IMS signaling layer as presented in Sub-Section 2.4.4. Based on this analysis, a proper capacity plan for each functional node in IMS or over provisioning computing resources can serve the purpose of supporting the high scalability in the IMS architecture.

All the existing solutions for the listed research questions are patch-ups or extensions to the IMS architecture which is itself quite complex. Considering the drawbacks of the IMS architecture, it is important to propose a new communication system that needs to be simple and supports the end user innovation.

Chapter 3

Other Communication Systems

Chapter 2 is devoted to discuss different solutions for the research problems of this thesis. Similarly, this chapter discusses the related work based on P2P SIP, XMPP/Jingle, and Web service Initiation Protocol.

3.1 P2P SIP

P2P overlay is a network of nodes where there is no central entity/server. Besides lack of a single point of failure, P2P overlay is scalable and reliable. In P2P overlay, each user is a peer that helps each other in order to achieve certain objectives such as locating files and users.

The P2P overlay architecture can be classified into structured or unstructured [84] [85]. In unstructured P2P overlay, there is no structure in organizing nodes or content in nodes. In contrast, structured overlays such as Chord [86], Content Addressable Network [87] and Pastry [88] are organized in a way to optimize the lookup latency and join or leave maintenance. In Chord, each node is connected like a ring and each node contains at most Log(N) entries in its finger table to point to other peers. Lookup is done in O(Log N).

The main goal of P2P SIP is to enable communication services over a distributed P2P overlay where the signaling protocol is SIP [9]. A P2P SIP system can thus avoid being dependent upon central server components, as is the case with IMS networks. A P2P SIP system can be either a structured or an unstructured overlay.

Researchers are interested in deploying a P2P SIP system in a mobile environment [85] where unstructured overlay is leveraged to enable sessions. In this case, nodes are in mobility. Therefore, churn rate will not influence the maintenance overhead.

Compared to unstructured P2P overlay, structured P2P SIP systems has less lookup latency, thus P2P SIP remains a structured overlay where mobility of nodes is not an important aspect.

This thesis considers structured P2P SIP systems for the literature review. The research work reported in [89] has shown how Chord based overlay is used

for a P2P SIP system. Besides basic voice/video communication services, services such as offline messages and multi-party conference are proposed in the P2P SIP infrastructure.

Concerning the listed research problems in Section 1.2,

1. A service provider who deploys P2P SIP takes care of the signaling aspect. Each user downloads needed software and does not have any control over the signaling protocol. Instead, it is an operator choice.

2. No relevant work has been reported for the user mobility use case and partial session transfer and retrieval use case.

3. Regarding the scalability aspect, I review the P2P SIP literature and highlight its weakness when used for real-time communication services in Section 3.2.

3.2 Scalability

Chord based P2P SIP infrastructure is proposed in [89] for Internet telephony. There are other kinds of overlays that can be used for P2P SIP infrastructure. For the comparison here, I choose the Chord based P2P SIP that is general enough.

To enable a session between two peers, by employing the Chord based P2P SIP infrastructure, message overhead per node is depending on four parameters as listed below [89]:

1. keep-alive and finger table refresh rate

2. call arrival distribution

3. user registration refresh interval

4. node join, leave, failure rates

Accordingly, message overhead is :

$$M = r_s + r_f(log(N))^2 + c * log(N) + (k/t)log(N) + \lambda(log(N))^2/N$$

Assume that the overlay has N super-nodes. Here k is the number of keys stored per node, rs is REGISTER refresh rate to successor and predecessor to keep the Chord ring correct, rf is refresh rate for finger table entry, call arrival is Poisson distributed with mean c per node, user registration is uniformly distributed with mean interval t per user, and node joining and leaving are Poisson distributed with mean λ.

The above mentioned formula shows how the message overhead is associated with call message routing. Hereafter, scalability analysis is provided according to four parameters based on P2P SIP.

How Scalable

Theoretically, P2P overlay is scalable. Therefore, P2P SIP architecture is also scalable as long as many nodes join the overlay. Any how there are two basic limitations - the first limitation is the capacity (CPU, memory and bandwidth)

of the super-node. And the second limitation is that P2P SIP overlay has much message overhead from peer nodes during registration, refresh, session establishment and node joining and leaving. Dimensioning the right set of super-nodes by taking the message overhead into account enables scalability during the high call rate. This is more linked with the latency aspect, as well (see below).

During high call rate, the failure of some super nodes compels an outage of the overall system. For example, Skype underwent 24 hours of worldwide outage in December, 2010. The cause for this downtime is the failure of supernodes. Initially, supernodes responsible for offline messages were overloaded. Then, over the time, the other supernodes were also overloaded due to message transmissions [90]. Therefore, reliability of P2P SIP based infrastructure is a question.

Complexity

P2P SIP infrastructure does not need a new functional entity (like a load balancer in IMS) in order to bear the high call rate. However, organizing the P2P overlay will be cumbersome when a large number of peers are connected (e.g. big distributed hash table). This factor will add complexity in to the overall system.

Like in Skype, proposing supernodes can reduce the complexity associated with big hash table. Typically, supernodes perform connecting ordinary nodes together and build a distributed database. In fact, these systems do not provide network-based, session-based services.

Needed Computing Resources

A P2P SIP signaling architecture has a huge message overhead as shown in the equation in Section 3.2. This message processing requires huge computing resources. It is not easy to quantify this parameter, but it is obvious that computing resources are needed more in P2P SIP than in MOCSP based system. The reason is that at least signaling messages should traverse log(N) hops in P2P SIP. Traversing many super nodes demands extra processing power. Globally, the message overhead associated with the overlay maintenance is high. This implies that a larger network has a higher message overhead. The churn rate (node joining/leaving) also demands computing resources.

Session Setup Latency

Session setup delay is high;it is always log(N). N is the average number of super nodes. When the number of users is high, there should be many super nodes. This means high latency. In addition, the call setup latency is associated with node dynamics (joining/leaving) [84] in P2P SIP.

3.3 XMPP/Jingle

The Extensible Messaging and Presence Protocol (XMPP), a protocol for streaming Extensible Markup Language (XML) elements between any two network endpoints is defined in IETF RFC 3920 and IETF RFC 3921 [91]. This is not a

real-time protocol, but close to a real-time protocol. XMPP was initially used for instant messaging, present events, offline messaging and voice mailing.

A typical architecture for deploying the XMPP system is a client-server model. A client gets a XMPP access to connect to the server. Servers are connected each other based on TCP. The path length between caller and callee is at least two XMPP servers. To and From headers of XML message are used to establish a session between caller and callee via two XMPP servers.

Jingle is a set of extensions to XMPP for enabling voice/video over IP multimedia sessions and the basic one is XEP-166: Jingle [10]. The initial objective of Jingle design is to provide real-time communication services for XMPP users. The most popular service, Google Talk, is implemented based on XMPP/Jingle.

The XMPP architecture emphasizes more on end points like in SIP. This means that end-to-end message transfer for a session is considered while traversing intermediate entities.

Concerning the research problems of this thesis, a few works have been reported.

1. Every provider will employ their own XMPP servers for providing real time services. In order to be interoperable, everyone should implement the same protocol. Changes are not easy, as well. Providing openness at the protocol level is not reached based on the XMPP/Jingle approach. Likewise, achieving flexibility in the signaling protocol is very cumbersome.

2. I could not find any research work on network-based session services. To be precise, enduser-initiated session transfer (transferring from one person to another) is discussed in the draft [92]. This service does not fall within interest of this thesis.

3. Scalability should be considered in two ways : single provider having many users and many providers having many users. In single provider case, there should be a federation of many XMPP servers. In this architecture, complexity is two servers or hops (in-bound and out-bound). Session setup delay and needed computing resources are proportional to these two hops.

This thesis did not consider configuration overhead that is associated with a federation of XMPP servers.

Addressing scheme used in XMPP systems is user@domain1.com. Therefore, any messages towards a user should be forwarded to domain1.com regardless of the user name. To be horizontally scalable, there should be at least one load balancer between users and XMPP servers.

3.4 Web Service Initiation Protocol (WIP) based Service Oriented Communication

Industry and academia had shown much attention on Web services [93] in order to integrate applications across enterprises. The open standards of Universal Description, Discovery and Integration (UDDI), Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP) are the key pillars of Web services technology. The developers of the application integration benefit from software modularity, reuse, language and platform independence.

Two separate worlds, Web services and IMS/SIP services satisfy different user needs. Hence, some efforts are devoted to integrate both domains; it allows re-using the existing services to provide new converged web and telephony services for enriching user experience.

Due the different nature of these two worlds (e.g. request-response model in Web Services and session based in SIP), the authors propose a signaling protocol and possible architecture for multimedia and voice communication over IP based on Web services standards [11]; as a result, it will naturally enable the composition of Web services.

The name of the signaling protocol is Web Service Initiation Protocol, relaying on WS-Session, replacing the SIP [11]. Signaling can be performed in a end-to-end fashion or via a B2B (back-to-back) broker mode. In any manner, Web Service Initiation Protocol did not evolve further from basic communication services. More importantly, this work did not consider the research problems that this thesis has looked into.

3.5 Conclusion

Three different communication systems such as P2P SIP, XMPP/Jingle, and Web Service Initiation Protocol based Service oriented communication are reviewed in depth. In fact, these three systems did not address the problem directly. The above analysis illustrates that providing extensions or patches will not solve the research problems of this thesis.

Chapter 4

Descriptive Model of IP Media Services

This chapter reviews the research work on compositional media control in [62]. Initially, a protocol has been defined for facilitating composition/media control between two users who are connected through application servers. This new protocol minimizes the programme complexity, programme state and latency compared to SIP.

This chapter includes the architecture-independent descriptive model, and protocol definition for IP media services. Because, the contributions of this thesis depend on the descriptive model, programming primitives and protocol. Note that [62] does not address the research problems of this thesis.

4.1 Introduction : IP Media Services

Telephony service (i.e. voice/video call) is one of the IP Media Services that have signaling and media separation in practice.

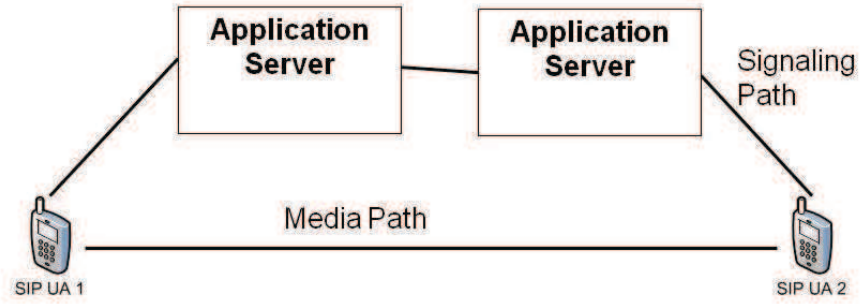


Figure 4.1: General practice of implementing IP media services

A signaling path consists of many nodes (such as application servers) and a media path is direct-to-direct between caller and callee. Nodes in the signaling path do not know each other, executing specific functions based on generality. This triggers to have a cooperation mechanism across nodes between caller and callee. This requirement drives the research work as reported in [62].

4.2 A Descriptive Model of Media Aspect of a Service

4.2.1 Signaling Path

The architecture-independent descriptive model is shown in Figure 4.2.

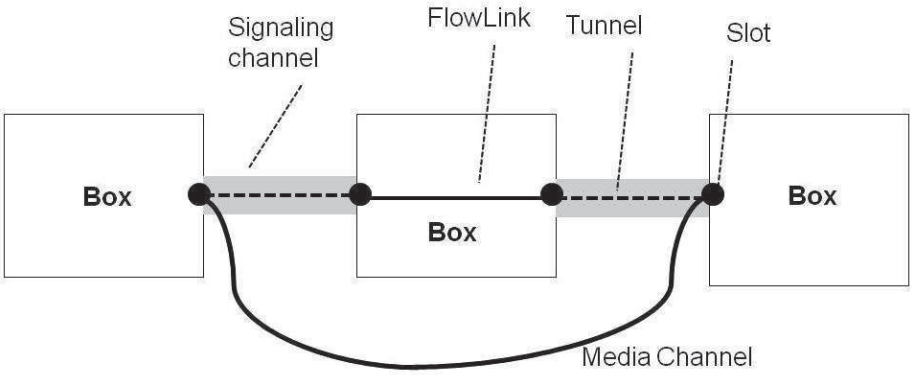


Figure 4.2: Programming abstractions in a signaling path

The descriptions of the key abstractions are as follows:

1. An endpoint involved in media control is indicated as a box.
2. A signaling channel, connecting boxes is two-way, First-In-First-Out (FIFO) and reliable. Therefore, TCP is ideal for signaling messages between boxes. Signaling channels carry tunnel signals and meta-signals that set up and tear down signaling channels.
3. A tunnel is proposed to control a media channel. There are many tunnels (based on different media) within a signaling channel. The slot is a point of intersection between a tunnel and a box. Each tunnel executes signaling protocol as defined in Section 4.3.
4. A flowlink is an object connecting two slots in a box. The flowlink can read and write signal messages at the slots. A chain of flowlinks and tunnels set up a signaling path between boxes.

No one has defined how to implement this descriptive model in IMS or SIP, so far. In my proposal, HTTP message transactions and setting up a Web Socket connection are considered as meta-signals. All the tunnel signals traverse over

the established Web Socket connections. When a caller clicks on the hyperlinks, the Web server responds with a HTTP response that consists of information of availability of the callee and the demanding media services. Refer to Section 5.2, for further details. The vocabulary in the descriptive model specifies the media control and is general enough to leverage in this thesis.

4.2.2 Media Channels

When a signaling path is established between two endpoints, there are media channels between these two endpoints. The endpoint slot is associated with IP address and port number that are used for sending and receiving the media. These attributes are static (not necessarily). Others are dynamic, depending on users desires. The dynamic attributes are described in a finite-state machine as shown in Figure 4.3.

A channel can be either closed, opened, opening or flowing state. In order to change into a different state, an endpoint sends/receives different events (such as open, accept, reject, and modify as shown in Figure 4.3) to the other endpoint.

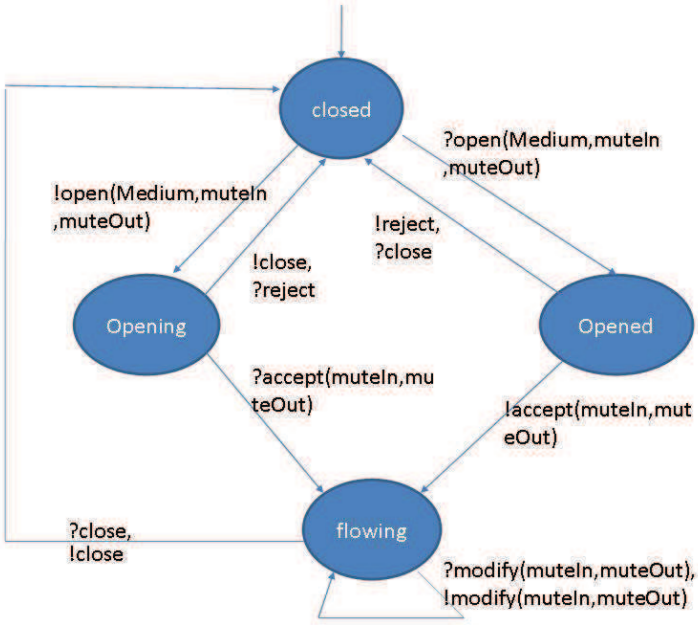


Figure 4.3: The user interface at one end of a media channel shows message transmission between two ends. The events preceded by exclamation marks are chosen by the user, while the events preceded by question marks are chosen by the other end of the channel. Commas separate unrelated transition labels with the same source and sink states.

Due to the different media channel behavior, each slot consists of a medium and state attributes (four different states in Figure 4.3) in the abstract model.

4.3 Signaling Protocol

4.3.1 Protocol Requirements

The major requirement of the signaling protocol is to establish a media flow between two endpoints. This requirement implies that each endpoint should know which IP address and port number are used by the other endpoint for streaming media. A codec for medium streaming in each direction should be agreed by both parties.

Today, Web browsers have started supporting different codecs such as VP8 video codec and iSAC/iLBC voice codec [94]. It is possible to have different codecs for the two directions of a media channel or a single codec for both directions. It is useful that each endpoint knows which codec is used for media streaming. Thus, endpoints can allocate different resources for interpreting the codec.

4.3.2 Protocol Definition

A protocol endpoint i.e., a slot is associated with a particular medium. Therefore, the scope of the protocol definition is one tunnel in one signaling channel. The finite-state machine specification of the protocol is shown in Figure 4.4.

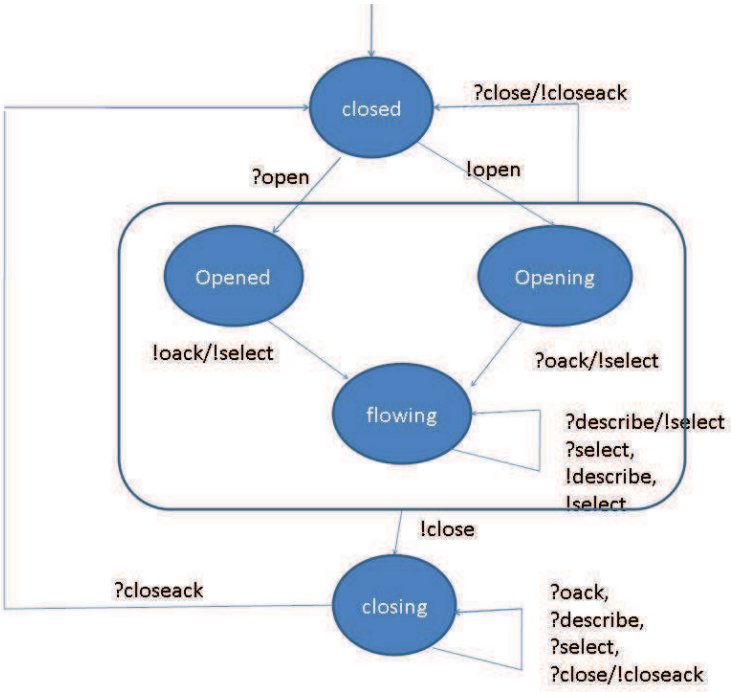


Figure 4.4: Specification of a protocol endpoint. ? means received, ! means sent. ?oack / !select means send select if and when oack is received. !oack / !select means send the two signals in sequence. Commas separate unrelated transition labels within the same source and sink states

The protocol is used to open, modify and close a media channel between

two media endpoints. How protocol is used between two endpoints is shown in Figure 4.5.

An *(*open) signal is used to open a media channel by either end of a tunnel. The other end can respond affirmatively with *(*oack) or negatively with *(*close). Either end can close the media channel at any time by sending *(*close), which must be acknowledged by the other end with a *(*closeack). The role of both close and reject is satisfied by a single abstraction, *close*.

The *open* signal contains information of the medium and a *descriptor*.

A *descriptor* describes an endpoint that is the receiver of media; it consists of an IP address, port number, and a priority-ordered list of codecs that it can handle.

If muteIn, a boolean variable, is true, i.e., the endpoint does not want to accept media, then the only offered codec is *noMedia*.

If an endpoint accepts the *open* signal, the endpoint should send the *oack* signal that carries a descriptor of acceptor.

In response to the descriptor message, an endpoint informs the other endpoint by a *selector*, agreed to send to the endpoint as specified (i.e. IP address of the sender and the port number of the sender) in the descriptor message. This selector message has information of the codec (a single codec) it will be using. A single codec is selected from the list in the descriptor. The selection process may vary, but mostly the codec that has a highest priority in the descriptor is selected. The selector message may have *noMedia*, if the selecting endpoint does not want to send media. This means that *muteOut* of the selecting endpoint is true.

Generally, the selector is a response to the descriptor. If an endpoint receives an *open* signal, it may send an *oack* signal. The endpoint sends a selector message, followed by an oack signal. Here the selector is a response to the descriptor in the open signal. Both endpoints can send the descriptors and selectors many times in a session. Therefore, there should be a mechanism indicating the relationship between descriptor and selector, for example numbering. As soon as an endpoint sends the selector message with a real codec, it starts sending media. Similarly, an endpoint becomes ready for receiving media once it receives the selector message.

This protocol allows to send select signals many times after sending the first selector in response to a descriptor. This means that selector can select the new codec from the list in the descriptor at any time. This case is explained with *select(sel'2)* in Figure 4.5.

Similarly, an endpoint can send a new descriptor after sending or receiving *oack*. The new descriptor may have a new address and (or) a new codec list. Then, the endpoint should respond with a new selector with a *select* signal. This interaction is shown in Figure 4.5 with *descriptor3* and *selector3*.

Preventing a race condition is taken into account when designing this protocol. One possibility for a race condition is that both endpoints send an *open* signal simultaneously within a tunnel. In this case, a simple rule for giving priority to the initiator of the signaling channel is defined. Therefore, the less priority open message will be ignored.

4.3.3 Properties of the Protocol

The user interface at each end of a signaling path supports to reflect the interest of user (Figure 4.4) and to convert the interest of user into implemented protocol (Figure 4.5) and vice versa.

Accept events trigger oack signals at the endpoint user interface. Describe and select signals are needed during the modify events. Descriptors and selectors can carry values of mute variables.

One advantage of this protocol is that both endpoints can send a describe signal and it is selected at the same time. This will not trigger a race condition and will not add complexity.

To make the programme state simpler, the protocol considers not to enforce pairing of describe/select signals that is relevant to one direction of the media transmission. This means that describe can be sent without awaiting a select response for the previous describe signal. In the same, a select can be sent at any time after sending the first select message.

4.4 Protocol Comparison

This section discusses the way the new protocol differs from the SIP. The new protocol differs in three aspects.

1. Transactional nature of SIP vs Idempotent
2. Codec choices
3. Media bundling in SIP vs tunnel within a signaling path

More detailed information is available in [62].

4.5 Conclusion

This chapter reviews the important building blocks for IP media services such as descriptive model and signaling protocol. These building blocks (for example, signaling protocol) are re-used in proposed solutions of this thesis (where appropriate).

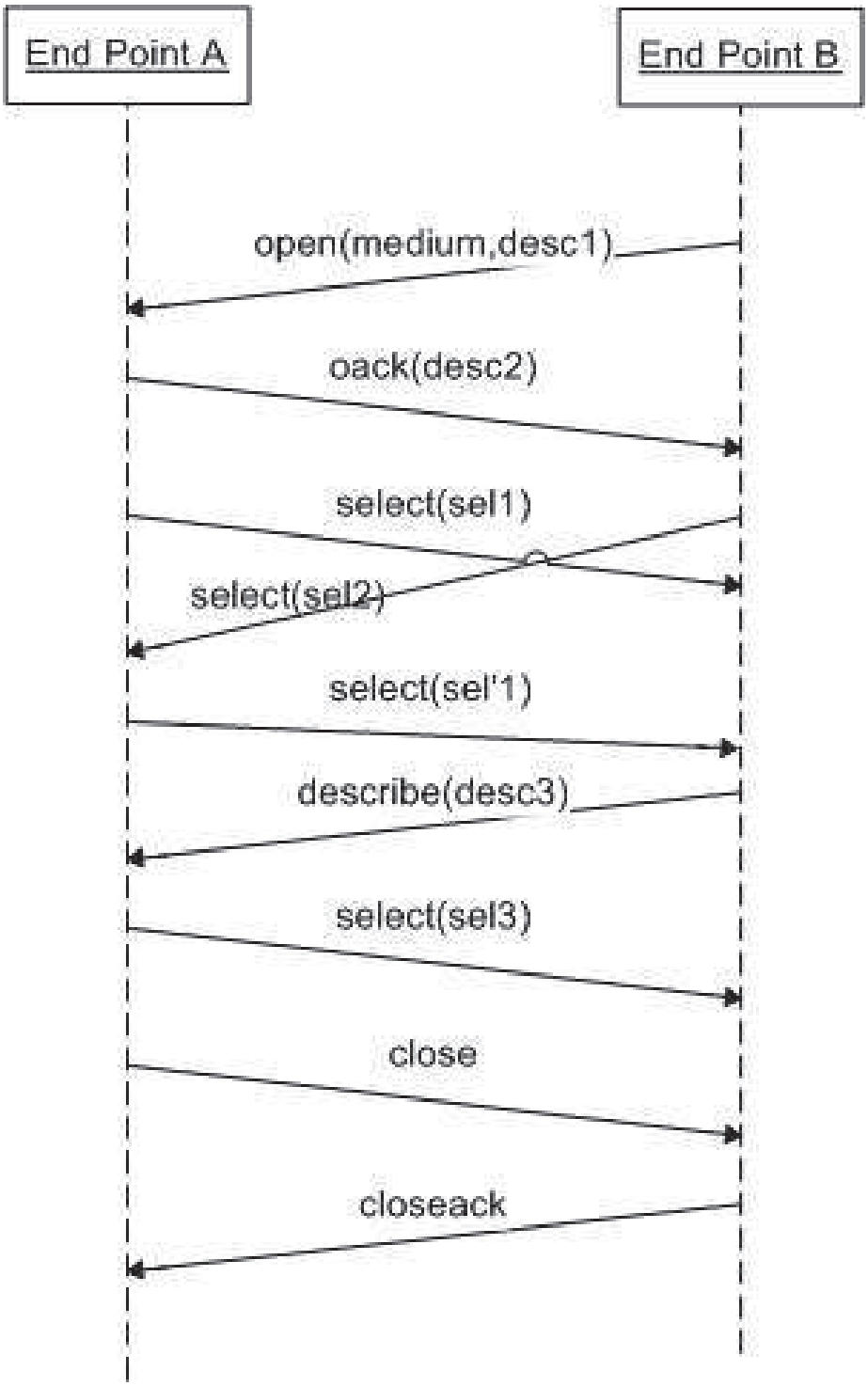


Figure 4.5: Protocol usage between two endpoints

Part II

Contribution

Chapter 5

My Own Communication Service Provider (MOCSP)

From the literature review, it is learnt that the signaling layer middleware is shared by all and is controlled by an operator. In any case, solutions for the research problems of this thesis are complex and demand a high cost for supporting interoperability (in IMS). Hence, this thesis intends to take a feasible new approach for addressing the research problems to design from scratch. This approach is called My Own Communication Service Provider. First this chapter explains the MOCSP concept and MOCSP system. Next, a demonstration of how to make a communication session in the MOCSP system is provided. At last, there is an analytical briefing on openness and flexibility.

5.1 From Concept to System

This section presents the MOCSP concept and system in detail.

5.1.1 Concept

MOCSP is a concept which allows end-users to create their own communication platforms themselves, for communication services, and own them. This concept intends to give all the basic elements of the communication services for end users. For the communication services, end users need control on a signaling layer and media layer. This thesis presents how end users can get complete control in the signaling layer in order to design their innovate needs.

5.1.2 System

I propose a top-down and Web-based approach for realizing the MOCSP concept.

For the top-down approach, the goal is to enable end-users to concentrate on defining what the service should do first, and then dealing with every single detail of how to implement it. It means that end-users are facilitated to define the services rather than defining or putting more focus on control and media flow.

The alignment of the MOCSP system with the Web platform is strongly motivated by three factors:

- Predication of blending of the future of the Web and the future of the human society [4]. It means that end users will accomplish their needs in the Web in the future, as well.
- Openness and flexibility of the Web platform. In the Web platform, users own and control all the functionalities they need in order to develop services. In other words, users do not depend on anyone. Moreover, users get more freedom to modify.
- Simplicity of the Web architecture based on HTTP, URL and HTML. A general perception is that simplicity always wins. In reality, Web has been successful beyond anyone's wild dream. Technical ingredients - HTTP, URL and HTML are easy to understand and simpler in operation.

In the remaining part of this section, the present researcher provides the two basic building blocks that are used in the MOCSP system. One is the formalization of communication services at the high level, and other one is the mapping of these services into a Web concept (e.g. URI) since our realization is based on the Web platform. Then, the architecture of the MOCSP system is provided. Conclusively, this thesis shows that the MOCSP system is a realization of an I-centric communication system for communication services.

5.1.3 Definition of Communication Services

The present researcher takes a top-down approach to define the communication services instead of designing the control protocol. Thus this thesis formalizes the communication services at a high level and in the social context. It means that users want to define how they can be reached by others in different situations [13]. The formalization as shown in Figure 5.1 gives importance to the user session (not like a session in SIP which is composed of transactions and dialog) and becomes the basis for deriving meaningful abstractions on communication services.

A user session typically means person-to-person communication (personal communication) and should have one callee, one or more callers and one or two media. Each user session hides the control session that defines and controls the media sessions between end points; hence, it is viewed at a very high level where caller, callee and medium properties (audio/video/text) are important parameters.

Unlike other communication platforms (e.g. IMS), the present research considers the following factors – identity for users (e.g. SIP address), authentication, control protocol and media protocol in the context of communication services/ session – as supplementary details. Therefore, an implementer (e.g. users) can choose the relevant features based on their context. This approach is contrary to the existing bottom-up approaches.

5.1.4 Communication Hyperlink (CH)

The World Wide Web (WWW, or simply Web) is an information space in which the items of interest, referred to as resources, are identified by global identi-

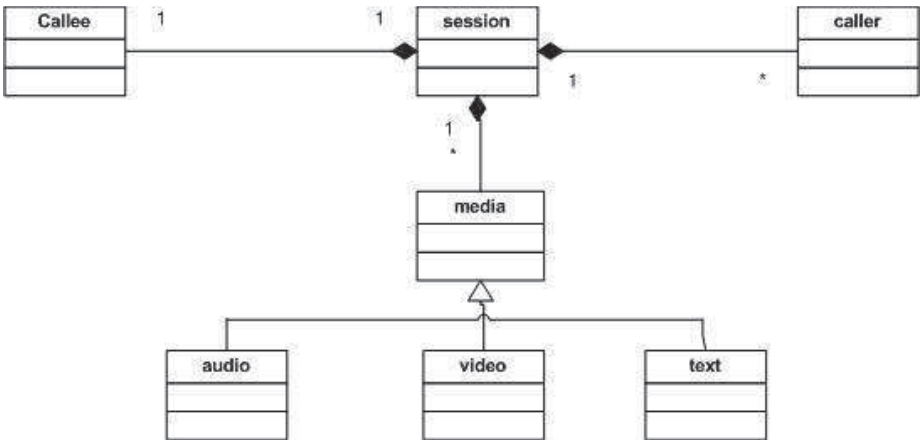


Figure 5.1: A formal definition of communication services

fiers called Uniform Resource Identifiers (URI) [5]. Accordingly, the present researcher considers communication services (e.g. user session) as resources, in which callee, caller and corresponding medium (or user session) are considered as a resource and identified by a global identifier (URIs), called communication hyperlink.

Communication Hyperlink is initially defined in [13] for providing client applications to users (i.e. callers) on demand but this thesis formally defines a resource and communication hyperlink as explained in the previous paragraph.

In WIMS 2.0 [36], an IMS session is modeled as a resource, consisting of signaling protocol information; it enables easy access of session based IMS capability using the HTTP protocol. For example, the format of the URI is like:

OpenAPIsRoot/IMPU/Service/SessionID.

Here, IMPU stands for IMS Public Identities and service is one of the IMS services.

This kind of fine-grained partition is useful in exposing the IMS session, but the present researcher asserts that there is no technical necessity to define all the information within a control session (i.e. fine-grained partition) as a resource for the research proposal of this thesis. Based on CH model[13], once caller clicks on CH, the client that is downloaded immediately will perform necessary actions. Thus, this approach is completely orthogonal to WIMS 2.0.

5.1.5 Architecture of the MOCSP System

The MOCSP architecture follows the principal of signaling and media separation in order to be a flexible architecture. Based on this advantageous practice, the present researcher proposes a high-level architecture of the MOCSP system as shown in Figure 5.2, consisting of control plane and media plane.

The MOCSP control plane is instantiated as a Web application and deployed in a Web server, enriched by communication hyperlinks. Caller and Callee are in two Web browsers connected to the MOCSP Web server. Signaling messages

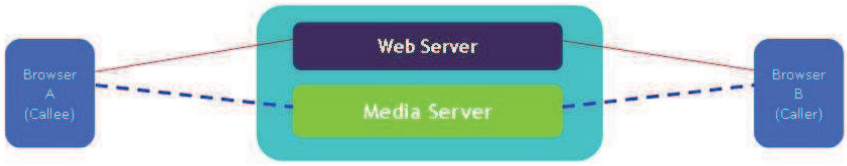


Figure 5.2: Simplified architecture of MOCSP system; thick line shows the control path and dotted line shows the media path

traverse between caller and callee via the Web server.

For media, the present researcher proposes the direct-to-direct media streaming between caller and callee. This means that video and audio are transported separately by Real-Time Transport Protocol (RTP). The codecs for video and audio are not in the scope of this thesis. In addition, multiplexing audio and video is being discussed in the new work group (RTC-WEB) in IETF. This work will improve the media transport. Furthermore, media server (optional entity) is proposed for the transcoding purpose if both caller and callee do not have same codec.

MOCSP Web application (in the control plane) executes three functionalities for callee:

1. Support to create different communication hyperlinks. When a callee creates hyperlinks, service logic for control session is developed for MOCSP Web server, callee side and caller side and placed in the MOCSP Web server. See Chapters 6 and 7 that place a specific service logic in the Web server.

2. Provide a relevant user agent to callee and caller. At the run-time, MOCSP provides dynamically relevant user agents (kind of a client software to manage a session) to caller and callee. Thanks to this simplicity for users, caller is not concerned about user agent, configuration and installation, but depends on the Web browser.

3. Create and manage control session when a caller clicks on communication hyperlinks, given by callee. MOCSP is a key element in session management, for the callee and caller clients in that it does not follow a pure end-to-end paradigm. More details on control session are given in section 5.2.

Naming of the MOCSP System

MOCSP enables each user (typically callee) to host relevant applications in the Web server that has a unique hostname (for example, http://www.siva.mocsp.com). The caller is given a URL. This means that caller is given a URL like http://www.siva.mocsp.com/friends The specific logic in the Web server (here www.siva.mocsp.com) interconnects the caller and the callee.

Each callee has a unique hostname in the mocsp.com domain. It is not mandatory, as users can have a unique hostname in the other domains such as .fr, .edu, etc. This thesis assumes that all users have a unique name in the mocsp.com domain.

5.1.6 MOCSP vs I-Centric Communications

I-centric communication, putting the individual user ("I") in the center of service provisioning rather than offering inflexible services that are unaware of actual customer needs or situations, is proposed in [95][96]. This paradigm emphasizes on giving more importance to individual communication needs, but not to specific technologies. In the I-centric vision, I-centric services adapt to end-users according to the ambient awareness, personalization and adaptation.

Compared with this generic model of I-centric communication in [96], MOCSP uniquely instantiates the I-centric vision for communication services. At this point, I-centric services (i.e. communication services) in the MOCSP system are not concerned with ambient awareness and adaptation, though they will be integrated later. Callee designs I-centric services (i.e. communication services) in the MOCSP system, based on the context (who can be reached and how (media type)) and encapsulates the session control within it. However, MOCSP and I-centric communication take the top-down approach for realizing the services.

5.2 Control Session in the MOCSP System

The main purpose of the control session is to control the media flow, including codec negotiation between caller and callee. However, MOCSP does not define a default signaling protocol because users define their own protocol and extensions based on their unique needs. This section intends to show how a communication session is made possible between two browsers. This use case generates two requirements – what is the signaling protocol; and how to enable asynchronous communication between Web browser and Web server.

The present researcher chooses the protocol for a control session from [62] (re-produced in Chapter 4) compared with the existing protocols such as SIP and XMPP, because:

1. Compared with SIP, the proposal [62] brings many benefits; it is a unilateral protocol, based on TCP reliability; it adopts a simple negotiation mechanism for codec choices; control session can manage and change the media flow in both directions separately and independently.

2. Based on the proposal [62], MOCSP Web Server gains intelligence on the media behaviour of callee and caller. This intelligence is very useful to make decisions in forwarding messages; it can also help to reduce the race conditions in the control session. In other words, two sides (caller and callee) are merged based on the piecewise approach protocol (proposal [62]) in the MOCSP Web Server.

HTTP is a request-response protocol where a Web browser sends a HTTP request to the Web server that replies with a HTTP response [97]. This is called synchronous communication where the Web server cannot 'natively' initiate a HTTP message. Signaling messages in the communication services need to send messages between Web server and Web browser asynchronously. Even though a few mechanisms such as Asynchronous JavaScript and XML (AJAX), polling, long polling and Flash exist, these techniques are not efficient and true bidirectional web communication.

To address this problem, IETF [98] and W3C [99] (through Web Hypertext Application Technology Working Group (WHATWG)) have jointly defined a new technology, WebSocket. WebSocket is "TCP for the Web", allowing bidirectional communications between Web browser and Web server. This technology has two parts: WebSocket protocol and WebSocket API. The WebSocket protocol specification at IETF becomes more stable and mature. An initial handshake and subsequently any messages over TCP is the basic idea behind the WebSocket protocol. The WebSocket API defined by W3C and protocol is available in major browsers such Chrome and Mozilla and major Web servers such as jetty. JavaScript applications create a persistent connection with a server and receive messages via an onmessage callback.

As a result, for asynchronous communication between Web browser and Web server, the MOCSP system depends on the WebSocket technology; it will increase the programming efficiency and reduce the latency for calls set up in the Web environment. The present researcher employs WebSocket for connecting callee and the MOCSP Web Server; and caller and the MOCSP Web Server. The MOCSP Web Server identifies the callee and caller based on the established WebSocket connections.

The call flow for a single user session/control session is proposed in Figure 5.3. The main aim of the call flow is to reduce latency in the call setup. The details of the all steps are given below.

Initially, callee logs in to the MOCSP Web Server (MWS); it is shown as REGISTER message in Figure 5.3. The REGISTER message may have many request and response messages, but for the sake of simplicity, it is not shown. In this phase, Callee Web Browser (Callee WB) creates a Web Socket and waits for receiving a call. Once caller clicks on the communication hyperlink (provided by callee), Caller Web Browser (Caller WB) sends a HTTP POST message to the MWS; it is shown as CLICK message in Figure 5.3.

Once MWS receives the HTTP POST message, it sends the open (medium, nomedia) message to the Callee WB over the established Web Socket. It means that caller is willing to send the medium (e.g. video or audio), but no description of the media flow is provided at this point of time. Callee WB indicates the call arrival and once callee accepts the call, Callee WB sends the oack(desc1) message to the MOCSP Web Server, indicating that callee is willing to receive the media on a particular address, port number and a priority-ordered list of codecs that it can handle.

After the open (medium, nomedia) message to Callee WB, MWS sends the response message to caller (shown as ACCEPT message in Figure 5.3). Next, Caller WB creates a Web Socket with MWS and sends the open(medium, desc2) message. Then, if oack(desc1) message is received by MWS, and open(medium,desc2) message is also received by MWS, oack(desc1) message will be sent to the Caller WB. At the same time, MWS sends the describe (desc2) message to the Callee WB. After that, both callee and caller WBs send the select messages (select(sel2), select(sel1)); it indicates which codec they use to send the media. When both callee and caller WBs send select messages, they start the media transmission. When Caller WB receives select(sel2), it is able to receive the media from the other end. Similarly, when Callee WB receives the select(sel1), it is able to receive the media from other end. In any case, if caller, callee, or MOCSP Web Server wants to stop the session; they can issue a close message and stop it.

There are many ways to establish a session, even based on 3PCC call flow (RFC 3725 flow I) in the MOCSP system, but it will increase the latency of call setup compared to the proposed call flow in Figure 5.3. Since 3PCC works on transactional mode, the Caller WB responds to session once it receives the offer of Callee WB. In Figure 5.3, while MWS receives the descriptor from the Callee WB, Caller WB sends descriptor to the MWS. Importantly, this thesis assumes that media component (media codec and media transport protocol) is part of Web browser and will not be downloaded from the Web server when a session starts. The major Web browser vendors are working on this direction.

This section discusses clearly relevant signaling messages and its transport protocol. In fact, the call flow in Figure 5.3 has a few HTTP messages (for example, CLICK and ACCEPT messages) and some messages (e.g. open, and describe) over Web sockets. However, one important aspect not mentioned so far is the Domain Name System (DNS) which the Web relies on. It means that Web browser has to find the IP address for the Fully Qualified Domain Name (FQDN) that is given in the address bar of the Web browser. DNS behaves as a registry of hostnames and IP addresses and performs a hostname-to-IP address resolution for Web browsers. For example, caller Web browser should make a DNS resolution before sending a HTTP message (indicated as the CLICK message in Figure) to the Web server. Additionally, Chapter 8.2 presents a complete DNS resolution mechanism and impact of DNS.

5.3 Impact on End Users

The MOCSP system can be modeled as shown in Figure 5.4. Since at least two sides involve in a session, callee gets more control in the MOCSP system. This model offers an individual platform (merging the service layer and session control layer in IMS). Initially, the present researcher analyzes the MOCSP system from the perspective of openness and flexibility. Later, there is a discussion on the users' privacy that is ensured in the MOCSP system.

5.3.1 Openness

With communication services, if a user wants to design new services, he needs to have control over the signaling protocol. In the MOCSP system, the end-users gain complete openness or ownership in the signaling protocol. For particular services, users design a proper signaling protocol without depending on any provider.

SIP defines routing between two user agents and semantic of the message in order to enable a session between two user agents. MOCSP system leverages established TCP connections for identifying users and users can define their own semantic of the message for a session. This arrangement does not add complexity to the architecture.

Abstractions on routing and message sequences help end users for developing service compositions. However, users need the basic knowledge of control session and Web programming, on account of simple and individualized platform. Based on this simplification, it is possible to state that MOCSP system will help users to define new services for their needs. In this thesis, standard primitives/abstractions that can help users in reducing programming complex-

ity with control session have not been defined. Moreover, users are left with the freedom of designing the signaling protocol. In Section 5.2, the present researcher presents a general model for the control session which can be used for all the user sessions, yet it is not mandatory to use one control session model for all the user sessions. This call flow ensures that communication services can be made between two Web browsers.

5.3.2 Flexibility

The existing system and protocol (e.g. IMS/SIP) do not quickly adopt changes excepting important features due to security, complexity and interoperability problems. The change process for SIP is explained in RFC 5727 [43]. However, changes can be made quickly to the message sequences of the signaling protocol in the MOCSP system. The present researcher presents the design considerations that enhance the flexibility.

When a service is developed, all the service logic resides in the Web server. The clients are stored in the Web server and are delivered to callers and callees on demand. This means that a callee gets his client on his registration and callers download their clients when they click on a hyperlink that is provided by the callee. If changes are made, they should be importantly reflected in the client side. This can be performed easily in one place. Since the MOCSP platform is developed for an individual, complexity associated with changes is less. When callee creates hyperlinks, only he should decide how it has to behave.

Intelligence of the services is placed in caller's side, callee's side and MOCSP Web server. Conceptually putting more intelligence in the network (MOCSP Web Server) is the main design consideration. This consideration enables to reduce the work at the Web browser and to prevent the race conditions in the signaling protocols. Importantly, MOCSP Web server is not exactly a back-to-back user agent in the signaling path (as in SIP), but it acts as a master element for session control.

The proposal for flexibility can be further explained as follows. The MOCSP system can implement different semantics for the control session for different communication hyperlinks (or user sessions). It means that all the communication sessions are private and only understandable by MOCSP web server and user agents (callee, and caller) or semantics of a session need not to be globally understandable. Therefore, control session (semantic of the session) for each user session can be different. These changes are easily implemented and do not require much analysis of the protocol. This flexibility in the MOCSP system supports end-users to experience personalized communication services.

5.3.3 Privacy

We are living in the Web 2.0 era today, where users are inspired by Web 2.0 applications and are seriously concerned about their privacy on the Internet [100]. In fact, many Web 2.0 applications do not concern users privacy, thus leaking user privacy information easily or Web 2.0 applications do profiling, analyzing and exposing of the personal information. In the same way, telecom service providers adopted many similar methods associated with user privacy: applying the data mining technique to extract the patterns from call detail data and inspecting and filtering the packets in the communication session to gather

both historical and real-time information [101] [102]. All these situations depict privacy as a serious problem with communication services. Technically, MOCSP system addresses this issue for callee. It means that the MOCSP system can only profile and share call detail data (i.e. session details) by virtue of overall control to end-users, thus overcoming the major privacy concern.

5.4 Conclusion

The new architecture for the voice/video over IP communication system is proposed in this chapter. The primary goal of giving openness and flexibility can be achieved in the MOCSP system. In MOCSP, each user designs and deploys own communication system, therefore, openness in the signaling layer is given to end users. MOCSP system is not a one-size-fits-all model, but an individual system for each user. In addition, each session can design different semantics for the signaling protocol as per the users' wish. This gives enough flexibility in the semantic of the protocol.

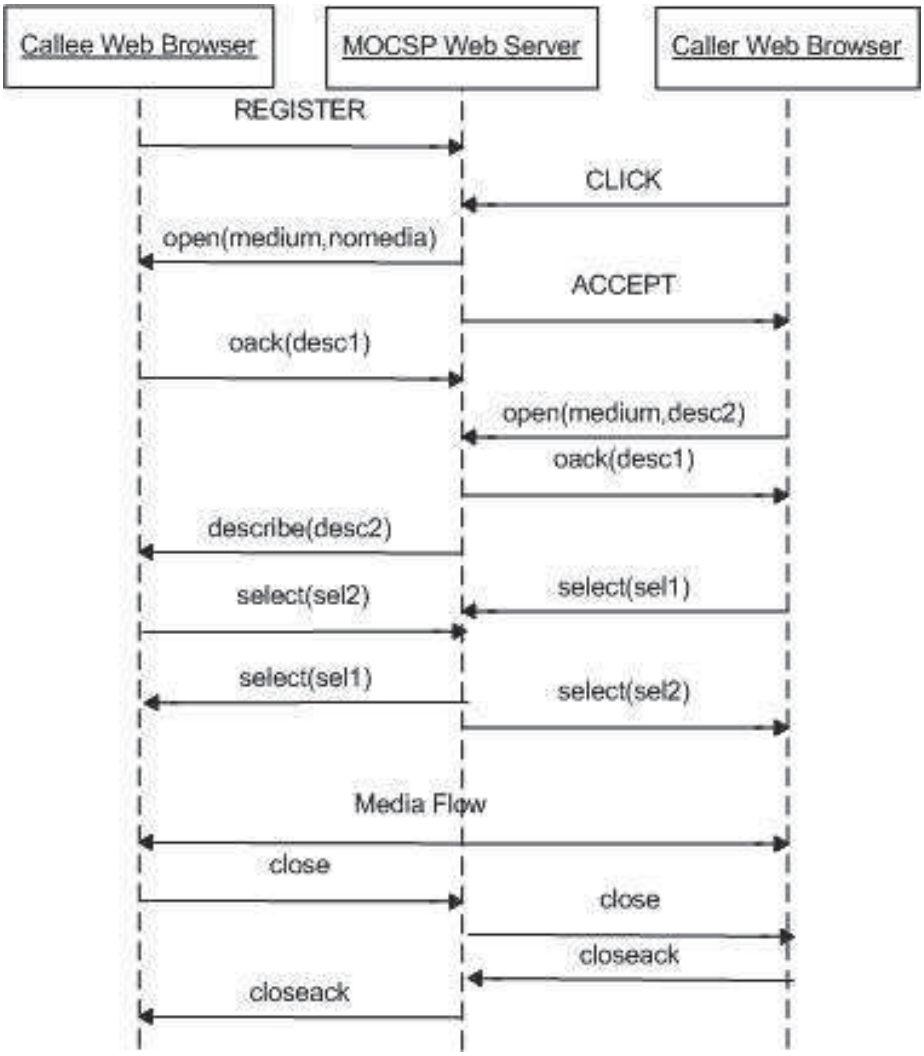


Figure 5.3: A call flow diagram for a session in the MOCSP; A caller initiates the session and a callee terminates the session.



Figure 5.4: High level view of the MOCSP system. Callee and callers are Web browser based clients.

Chapter 6

User Mobility

Chapter 5 presented the MOCSP concept and architecture that strives to provide more openness and flexibility to end users. These openness and flexibility allow to develop new innovative services. One such innovative service is user mobility. This chapter presents a solution for the user mobility use case. The complete description of the use case is presented in Section 2.3.1. The solution includes a descriptive architecture model and a call flow. Afterwards, the present researcher presents a new innovative use case, a missed call situation and a solution that is leveraged based on the proposed descriptive architecture for the user mobility.

6.1 Descriptive Model

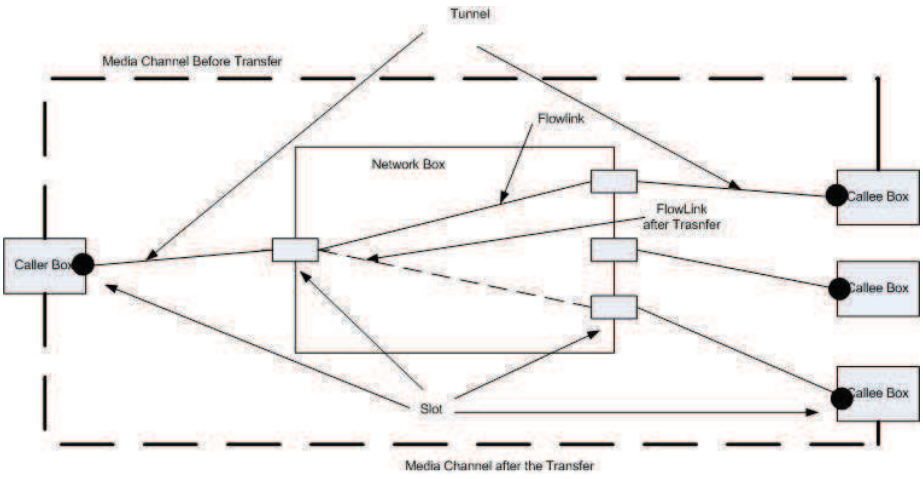


Figure 6.1: Diagram for descriptive model for user mobility

The main idea is to propose a descriptive model that should be architecture-independent. For this reason, engineers have freedom to implement in a particular platform. In this case, a descriptive model is to consider a single entity that will be responsible for managing the media within a session for the user mobility

use case. A general descriptive model for the user mobility is shown in Figure 6.1, leveraging the descriptive model proposed in [62]. The proposal in [62] is initially developed for identifying the correct media behavior by application servers.

The proposed descriptive model consists of three boxes: Caller Box, Callee Box and Network Box. A callee box and a caller box are connected to a network box by a signaling channel that is two way, First-In First-Out (FIFO) and reliable. This model has many callee boxes and only one caller box because of multiple callee registrations. The caller box is terminated once a call is finished, but the callee boxes are in active stage until the callee closes the signaling channel. Figure 6.1 clearly shows media channel and signaling path before and after the session transfer.

Though this model is intuitive, it is important to indicate how it manages a session. For the user mobility use case, session management is principally delegated to the network box which is an intermediary between caller and callee. In an equivalent SIP terminology, the network box is made to act as a register, a B2BUA and a forking proxy. The network box based approach reduces unwanted message processing in each session and complexity in the functional coordination.

6.1.1 Role of Network Box

As mentioned before, a network box can perform either as a B2BUA, or a forking proxy or both. A B2BUA can create, terminate and modify the SIP dialogs. Typically, the B2BUA concept is used to develop arbitrarily complex services [103]. This means that a B2BUA application can keep two or more dialogs and interact with other applications.

However, B2BUA is not initially defined in the SIP specification. Therefore, some research efforts have been devoted to give deep understanding of the B2BUA concept. At a high level, there are two different operational modes identified: Transparent Mode and Handle Mode [104]. These two modes support the interworking of call control by endpoints and network applications. The details of these two modes are given below:

1. Transparent Mode: In this mode, the B2BUA application propagates requests and responses between dialogs that connect two end points. In this case, the dialog identifier, made up of the Call-ID and tags in the From and To headers, should be considered carefully while propagating. The dialog identifier is not preserved across dialogs. For example, the B2BUA applications, behaving in the transparent mode, relay NOTIFY requests and responses related to a REFER request. Assume that the REFER method passes through the B2BUA application.

2. Handle Mode: Handle mode performs necessary actions upon call control requests from endpoints instead of relaying as in the transparent mode. When acting on the handle mode, B2BUA should comply with endpoints. For example, B2BUA sends response messages to the REFER method and creates new dialog (i.e. INVITE message) with the target destination (mentioned in the REFER method).

As these two modes are widely discussed in [104], choosing the right mode totally depends on the specific deployments. However, the basic idea of two modes can be used in the network box. The developer will decide based on the usages. In user mobility and PSTR solutions, the network box is programmed to work in both modes. Information regarding the PSTR solution is available in Chapter 7.

6.1.2 Usage of Network Box in the MOCSP system for the user mobility use case

This section describes how the network box is designed in order to support user mobility. For managing the session, the network box keeps information of callee/caller boxes such as IP address and IP port number or TCP connection and dynamic attributes (such as closed, opening, opened and flowing) of media channel of each end point. Based on this information, logic in the network box performs session management at session initiation or during a session. In fact, the network box sets and updates the flow link with different callee boxes.

The network box connects two tunnels via a flow link. A tunnel is used to control a media channel and is established within a signaling channel. In other words, a signaling path is a combination of two tunnels and one flow link. Flow link is a software entity in the network box, representing a connection of two tunnels.

End points of a signaling path establish the media channel. One end of the media channel can be in four different states: closed, opening, opened and flowing, as modeled in [62]. Caller box and callee box are software entities similar to user agents in SIP, but running in the Web browser. In Chapter 5, caller box and callee box are shown as caller Web browser and callee Web browser, respectively. Network box is a responsible entity in the MOCSP Web server. This descriptive model is easy to deploy in the MOCSP system without facing a lot of engineering problems.

6.2 Call Flow for User Mobility

Before presenting the corresponding call flow, the functionality of the network box is listed for the user mobility use case.

The network box makes decisions about which media channels should exist. In order to make the decision, the network box depends on a context enabler that will satisfy the requirement as listed in Section 2.3.1. The specification of the context enabler is out of scope for this thesis, but a network box needs two functional interfaces such as:

1. Providing the location of the end user (IP address) or code name that helps to identify the terminal.

2. Providing the location of an end user during a session and if the user is in motion.

A lot of information related to context (enabler) can be found in the C-CAST project website in which the context enabler provides user action, user situation and physical environment information [105] [106]. The C-CAST project intends

to address the content creation, adaptation and delivery for multicast and broadcast services based on context information. The session management enabler (SME), a B2BUA application, performs session management for multicast and broadcast services [107].

Separating the context enabler from the communication system serves to reduce the complexity. In [55], M.E. Barachi et al propose a gateway for IMS and Wireless Sensor Network (WSN) internetworking. Data from the WSN provides context information about the user, but the solution proposed in this thesis is more focused on the communication service side.

For the session transfer, this thesis decides to adopt a soft handoff for decreasing the disruption time. Soft handoff means creating a session before breaking the former one. Thus, it will increase the user experience in real time communication services because there is no disruption delay. Soft handoff techniques are widely used in mobile networks. Another aspect that enriches the user experience is that the user does not issue any command for transferring a session. The system follows the user. This feature can be seen in the call flow below (6.2).

I describe the call flow in the next two paragraphs. Callee X signs in to the Web server from the two communication end points identified by callee@diningroom and callee@bedroom in Figure 6.2. It means that both end points open a Web Socket connection with the Web server. The Web Socket connection opening and closing are not shown in Figure 6.2. Once the caller establishes a call (shown as CLICK message in Figure 6.2), the network box checks the context of the callee with the help of the context enabler. The network box forwards the call to the right communication end point if the particular callee signs in from that end point. In Figure 6.2, network box sends the open message to Callee@DiningRoom. In Figure 6.2, CLICK and ACCEPT are two HTTP messages (request and response).

After the ACCEPT message, caller establishes a Web socket connection that is not shown in Figure 6.2. Other messages such as open, select, describe are defined in [62] and are routed via a Web socket connection. Session establishment is shown in Fig 6.2.

Figure 6.2 also shows the call flow for the session mobility. In this case, network box sends the open message to Callee@Bedroom when it receives the information of changed context. If callee accepts the session at the bedroom, network box sends a close message to the old location (dining room) and media description (describe (desc3)) to the caller. Then, the session continues. More importantly, session setup time in this call flow during the movement is reduced because the network box is able to memorize the session description of caller and to send it to the new callee box. See the open (medium,desc2) message sent to callee@bedroom. Desc2 is received by the network box when caller initially starts the session. Some messages (describe and select) are unilateral in Figure 6.2.

6.3 Phenomenon of Missed Call

The previous two sections present the solution for user mobility. However, the user mobility use case considers an ideal situation. One different situation is that the user is not near to any end point, when a new call arrives. Therefore,

this call will be missed because the user is not reachable.

In this case, if callee has activated the call forwarding service to voice mail for this kind of unreachable situation, the caller or incoming call should be forwarded to the video/voice message service (i.e. voice mail service). During the voice/video recording, if the user becomes reachable, he will be notified that one caller is forwarded to the messaging services and wants to communicate. The sample notification message is "Alice is leaving you a message, do you want to receive the call?". In this situation, if the callee is willing to communicate, the call will be established. This use case is referred to as a session-based service composition. It also illustrates the need for a single orchestrator that manages a session across different end points.

This new use case is realized based on the previous descriptive model because voice mail server is an end point like callee, and network box can execute a specific logic that is establishing a session between caller and voice mail, and between caller and callee. This missed call scenario is well explained in Figure 6.3.

Based on this idea, the present researcher develops the call flow for this new use case as shown in Figure 6.4. It is assumed that Figure 6.4 has two kinds of call flow where the thick lines show the session with voice mail server and the dotted lines show the update of the session for the voice communication between caller and callee. In order to update the session, callee should accept the notification from the network box. If the callee is not reached until caller leaves the voice mail server, messages that are shown by the dotted lines will not be executed. The voice mail server sends/receives audio to/from users. The signaling part of the voice mail needs an implementation of "open","modify", and "close" messages that come from the network box. Also, it is not necessary to have a Web Socket connection between network box and voice mail.

6.4 Conclusion

This chapter presents the solution, including abstract architecture and call flows. This solution can be used in any situation where logic needs to be executed during the call establishment and mid-call. Based on this solution, user mobility use case is well explained in this chapter. This solution is simple to develop and deploy in the MOCSP system.

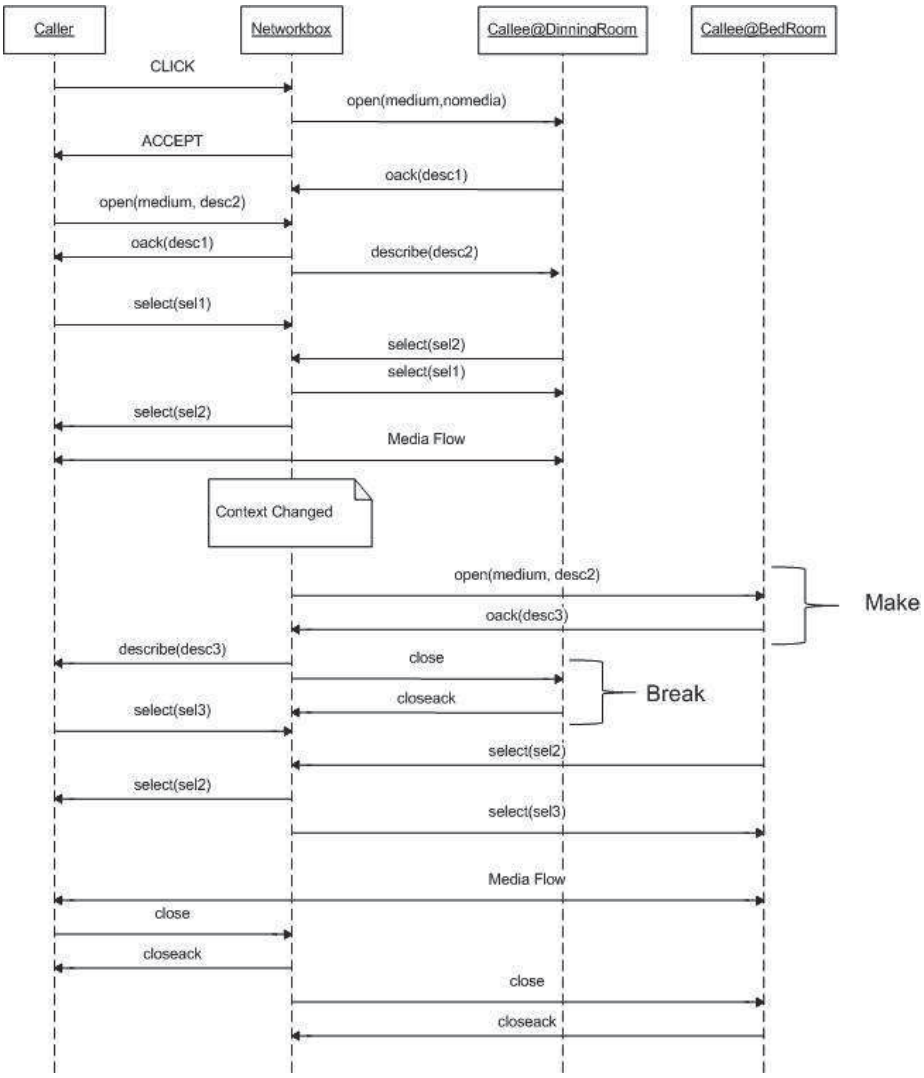


Figure 6.2: Call flow during the session mobility

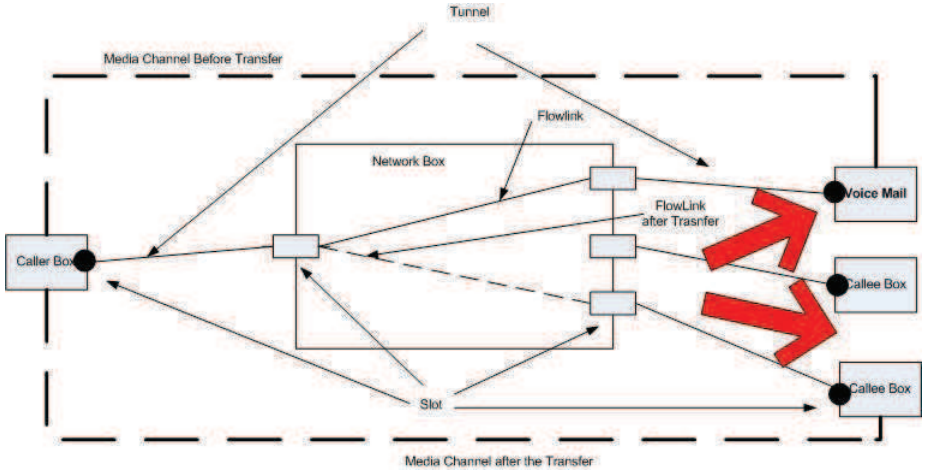


Figure 6.3: Diagram for descriptive model for missed call situation

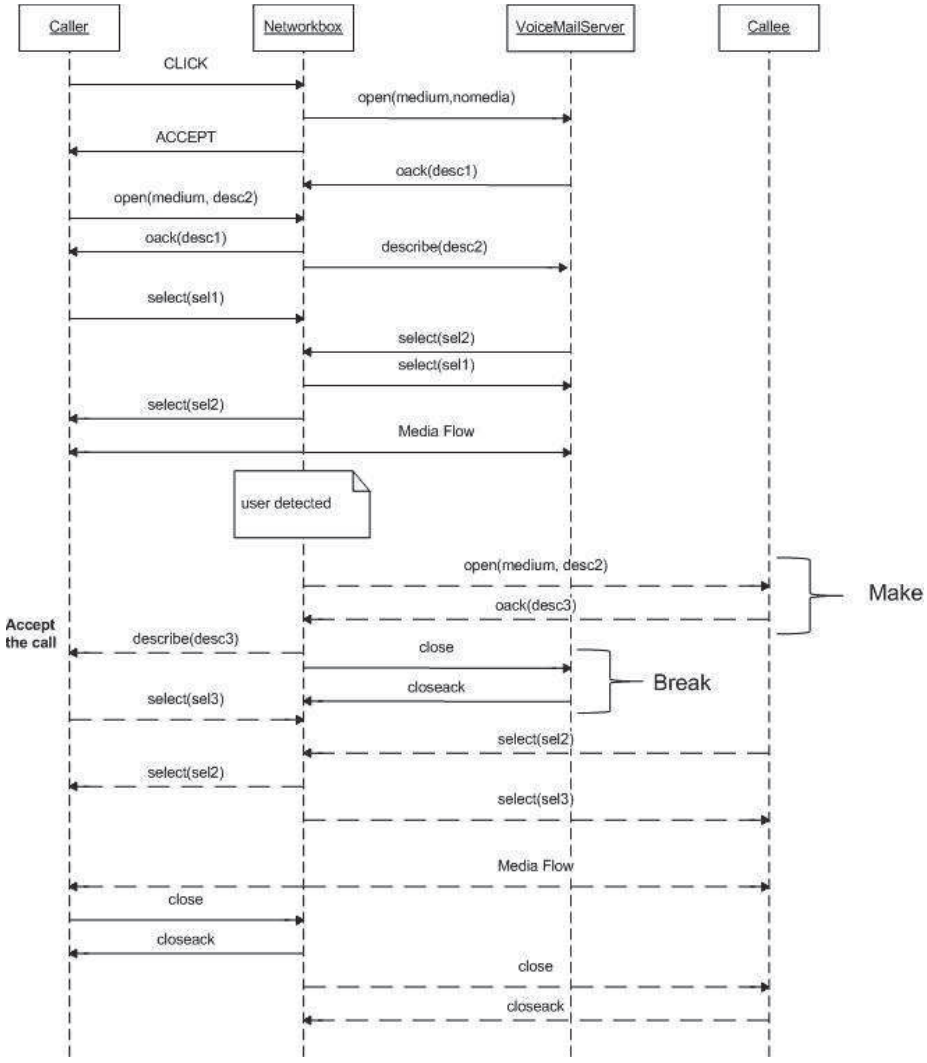


Figure 6.4: Call flow during the voice mail and call

Chapter 7

Partial Session Transfer and Retrieval

Chapter 6 shows the benefits of gaining openness and flexibility in the communication system. As a result, the solution of the user mobility is easy to develop since all the service logic is able to merge into one single place. As a consequence, difficulty in coordinating the intermediate entities is reduced. To the contrary, there is another case, where the cooperation of intermediate entities and end points is needed. The partial session transfer and retrieval is an example for the cooperation between intermediate entity and end point.

This chapter presents a solution for partial session transfer and retrieval. This proposed solution consists of architecture, control protocol and call flow diagrams for network-initiated and user-initiated partial session transfer/retrieval.

7.1 Architecture

The proposed architecture has a network box, caller box and callee box. A caller/callee box is an end-point for callers and callees. Medium devices can send and/or receive the medium (audio/video) and are available and near to a callee and caller during a session. These medium devices can be divided into medium sources or sinks. In Figure 7.1, the medium device is not shown, but the medium sink and source are shown. A network box coordinates the media flow across the caller/callee box and medium devices (and medium sinks and sources) according to user requests or its understanding of callee and caller context. In this sub-section, I describe each entity shown in Figure 7.1 and its functionality.

7.1.1 Network Box

In Chapter 6, a network box, a signaling layer entity, is proposed to accomplish complete session mobility. In this chapter, the network box is leveraged for PSTR. While the network box manages the session across the medium devices based on medium classification (see the next sub-section), it interacts with the caller and the callee via caller/callee box for PSTR.

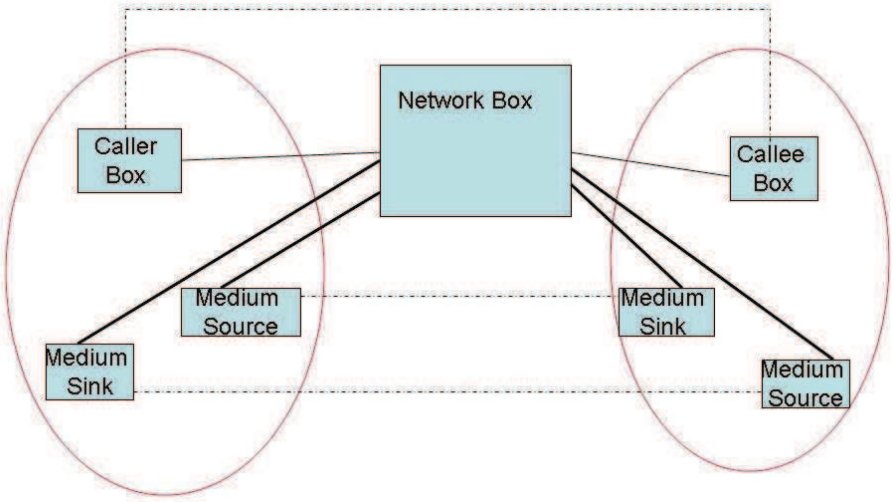


Figure 7.1: Architecture of the partial session transfer and retrieval. Thick lines show the connectivity established by the network box to medium sources and sinks. Dotted lines are possible paths for media flow

7.1.2 Description of a Medium Device, Medium Sink and Medium Source

Generally, medium devices are capable of sending/receiving the media. Each medium device can play different roles such as behaving as a sink, source or both for a medium (audio/video). A medium sink is a device that is able to receive media from any destination (in IP network). Similarly, a medium source is a device that is able to send media. If both capabilities are placed in one device, it is called as a medium device. These different roles are shown in the tree structure in Figure 7.2.

In addition, medium devices have a control interface, therefore, the network box can instruct medium devices for both sending and receiving of media. The present researcher proposes naming the media device based on the uniform resource identifier (URI) format and to access the control interface via Web Socket connections. For example, a control interface of a medium device is ws://example.com/service or ws://ip:port/service/resource. Web socket APIs facilitate a simple integration between medium devices and the network box via an asynchronous manner.

A detailed description of medium devices can help to differentiate the devices that can be either source, sink or both - fine-grained description. This will allow the medium device to be instructed based on the medium or the medium source/sink. In SIP, manipulation of Session Description Protocol (SDP) is needed to understand which medium the device supports. Here, the devices are explicitly defined based on their medium capabilities. The proposed naming convention based on URI is used to describe and to control services. The list of URIs for audio and video is shown below:

1. ws://ip:port/service/medium/audio

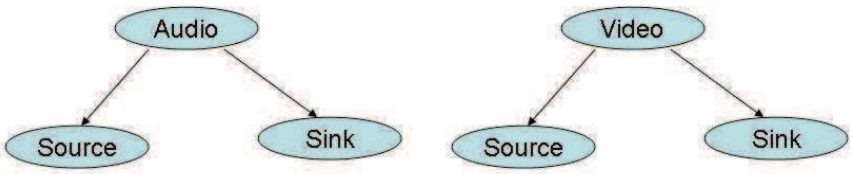


Figure 7.2: The classification of medium

2. ws://ip:port/service/medium/audio/source
3. ws://ip:port/service/medium/audio/sink
4. ws://ip:port/service/medium/video
5. ws://ip:port/service/medium/video/source
6. ws://ip:port/service/medium/video/sink

Each device should implement a three-message logic – 'open', 'modify' and 'close'. The URI provides two details: the address of the device and which medium it supports.

Each media device has a user agent that can manage signaling and media. The present researcher proposes the protocol stack shown in Figure 7.3, by replacing a complete UPnP protocol stack [68] that deals with addressing, discovering, description, controlling, eventing and presentation. This protocol stack is light-weight in terms of processing compared to the UPnP stack that depends on Simple Object Access Protocol (SOAP).



Figure 7.3: Protocol stack for medium devices

7.1.3 Caller Box/Callee Box

A caller box and a callee box represent a caller and a callee who establish a session using their Web browser. A caller box or a callee box is shown as a Web widget - in fact, each medium device can be represented as a Web widget as shown in Figure 7.4, which shows a sample user interface with the caller/callee widget, medium source widget, and medium sink widget. Therefore, users can

perform a simple drag-and-drop action for PSTR. It is paramount to mention that the caller and callee box are always part of a session. This means that although all media components are transferred to nearby devices, the main contact points for users are caller and callee boxes.

Later on, there is a discussion on how the caller and callee boxes manage the protocol for PSTR in which the user is informed and in control during the network-initiated session transfer and retrieval, in section 7.3.



Figure 7.4: Sample user interface for partial session transfer and retrieval.

7.2 Protocol

Before diving into call flows, I describe the approach for developing the control protocol that will support PSTR. As this approach attempts to separate the concerns, the two protocols are identified, namely, media control [62] and my protocol which is also known as auxiliary protocol. These two protocols are dedicated to two different concerns; therefore, the overall protocol development complexity will be reduced. The separation of concerns is one of the key principles of software engineering.

With the auxiliary protocol, abstractions facilitate carrying PSTR information between the network box and caller/callee boxes. This auxiliary protocol functions based on a request and response model, relying on TCP. All the abstractions in the auxiliary protocol are listed in Table 7.1.

Abstractions needed to initiate PSTR by a caller or callee	**Abstractions needed to initiate PSTR by network box**
Split (URI)	IsSplit (URI)
Splitted (URI)	YesSplit(URI)
NoSplit(URI)	NoSplit(URI)
Retrieve (URI)	IsRetrieve(URI)
Retrieved (URI)	YesRetrieve (URI)
NoRetrieval (URI)	NoRetrieve (URI)

Table 7.1: List of abstractions for the auxiliary protocol

The abstractions above carry out the goals of transfer and retrieval. For example, if a caller/callee wants to transfer a partial session, they send a Split message to the network box. If the network accepts the Split message, it will send the Splitted message. Otherwise, the network box sends the NoSplit message. Each abstraction has a single argument in the form of a URI (as mentioned

in Section 7.1.2). The next section explains how the media control protocol and the auxiliary protocol work together.

7.3 Call Flow

This section presents two different scenarios: network-initiated and user-initiated PSTR. These call flows are composed of media control protocol and auxiliary protocol. The present researcher has made a change in the caller/callee box during the transfer/retrieval of a particular medium compared with [62]. During the session transfer, the caller box or callee box considers that muteIn, and muteOut are true [62]. If a session is retrieved, muteIn and(or) muteOut become false.

Since transfer or retrieval can be initiated by users or the network, users are always privileged. All the activities initiated by the network side should be approved by the user. This means that the user keeps complete control.

7.3.1 Network-Initiated Partial Session Transfer/Retrieval

This sub section illustrates partial session transfer/retrieval at callee and caller sides by the network box. Since the user always has the control, the network box requests user approval for PSTR. For this purpose, this thesis proposes to use two different messages such as IsSplit and IsRetrieve for PSTR. These two messages inform the goals to user for approval. The complete call flow is divided into three parts:

I. Session establishment between caller and callee and transferring the session partially at the caller side.

   Figure 7.5 shows details of a session established between caller and callee. Once the media channel is established between callee and caller, network box asks caller for Partial Session Transfer (PST) by sending the IsSplit message. If caller accepts, the session will be transferred as indicated in the IsSplit message. To transfer a session partially, network box sends the open message to the new device and modifies media parameters of the caller box by sending the describe message.

II. Transfer the session partially at the callee side

   There is another transfer on the callee side as shown in Figure 7.6. In this case, network box asks permission of the callee box by sending the IsSplit message. When callee accepts the request, session is transferred partially at the callee side.

III. Retrieve the partial session back to the caller side and terminate the session

   As in the previous situation, network box requests caller to retrieve the session by sending the IsRetrieve message. When caller accepts the request, the session is retrieved. For retrieval, the network box closes the existing connection and updates media parameters with the caller box. Then, if a user (caller or callee) wants to stop the session, he/she can send a close request to the network box that ensures disconnections with all the devices involved-including caller and callee boxes. This is explained in Figure 7.7.

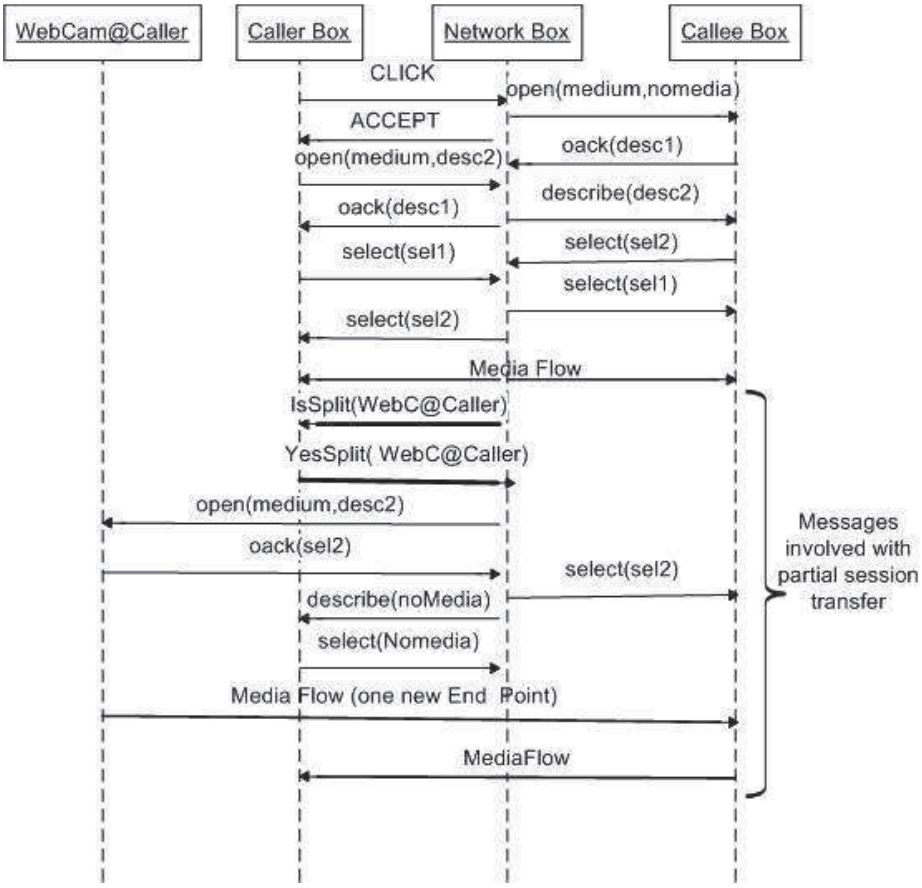


Figure 7.5: A call flow for session establishment between caller and callee and for partial session transfer at the caller side.

Three different figures show the important snapshots during PSTR by the network box. Any how, these following points are not defined in this thesis:

1. How does the network box know the devices near to caller/callee during the session?

2. How does the network box make a decision that the splitted session should be retreived?

To serve this purpose, context information of users and medium devices (location and capability of the device) is useful. Some research efforts look at this problem within a network (e.g. Bluetooth and a UPnP within a local area network). These efforts do not completely address the problem. Therefore, this complementary work will be performed in the future. This future work enables to perform automatic partial session transfer and retrieval by the network box.

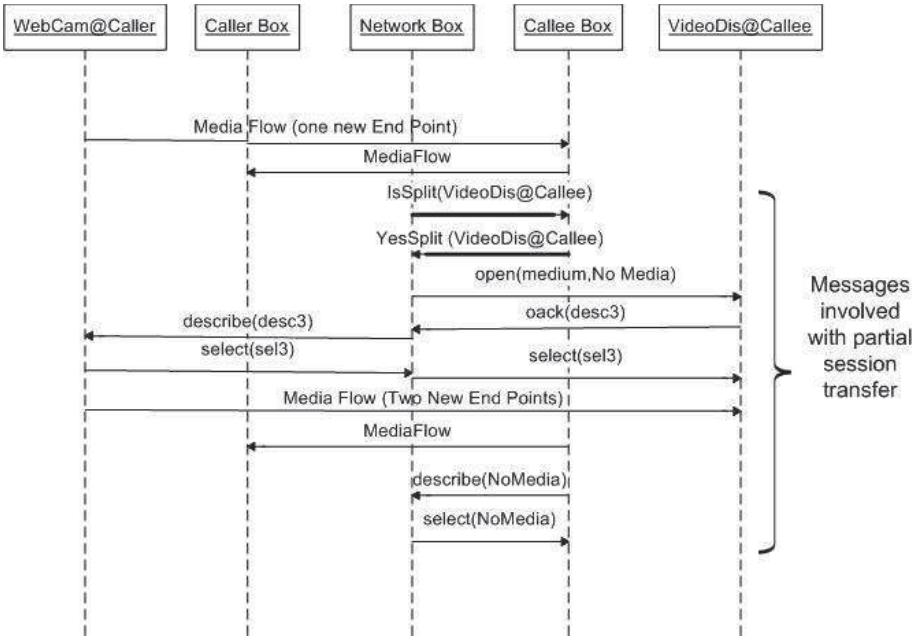


Figure 7.6: A call flow for partial session transfer at the callee side.

7.3.2 User-Initiated Partial Session Transfer/Retrieval

For this scenario, a user issues two commands such as Split(URI) and Retrieve(URI) from the caller/callee box to network box. Depending on the URIs, callee/caller box and network box perform media control.

If a URI refers to a medium source in a transfer request or a Split message, the network box sends an open request to the URI mentioned (i.e. medium source) and a description message to caller/callee box. Then, medium source accepts the request; and network box sends a describe message with the nomedia description to callee/caller box. As a result, callee/caller box does not send the media. Later, the network box sends the Splitted message to callee/callee box as a confirmation. This example call flow is shown in Figure 7.8 and includes WebCam@caller, caller box and network box entities.

If a URI refers to a medium sink, the network box opens a connection with the medium sink. If the medium sink accepts, the network box sends the Splitted message back. If the caller/callee box receives the Splitted message, the caller/callee box sends the describe message with nomedia. This interaction is included in Figure 7.8 that spans between VideoDis@callee, callee box and network box.

If a URI in a Split message refers to a medium, the network box sends an open message to the medium device. When the medium device accepts the request, the network box sends back the Splitted message. After sending the Splitted message by the network box or receiving the Splitted message by the caller/callee box, either end sends a describe message in order to make MuteOut and MuteIn true. The corresponding call flow is not included in Figure 7.8.

At last, when the caller/callee box demands a retrieval from the medium

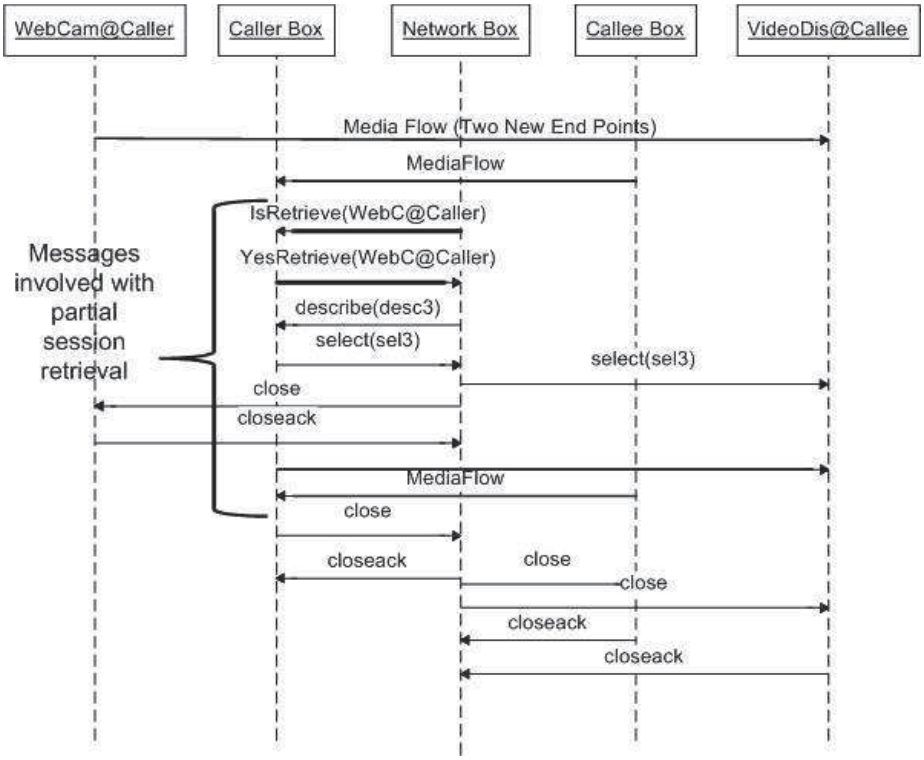


Figure 7.7: A call flow for partial session retrieval at the caller side and session termination.

source, the network box closes the existing media session with medium source and updates a media session with a new configuration. This is shown in Figure 7.8 where caller box issues a retrieve command. Moreover, if any user wants to close the session, the network box will take care of closing all the media session.

7.3.3 High Mobility Situation

In Section 7.3.1 and 7.3.2, call flow diagrams with details of two different scenarios have been presented. However, these diagrams do not show four important scenarios that are classified as high mobility situations in this thesis.

I. Caller and callee send the split message at the same time. In this case, the central entity manages the session for all participants. When network box gets the split message from one end, it re-produces the two messages : open and describe messages. In addition to this, network box properly manages the description parameters across all the entities during a session. This case will not add any constraints like SIP. In a SIP, if both sides send an INVITE message, there is a race condition.

II. Caller and callee send the retrieve message at the same time. Like in the previous case, network box retrieves the partial session back at both sides when both caller and callee are requested. This situation will also not add any constraint to the solution presented in this thesis.

III. Caller and callee send a split and retrieve message, respectively at the same time. The network box manages opening up a new media device and closing down another device while consistently managing the session description parameters.

IV. caller/callee sends the split message while network box sends the IsSplit message. But, there is a possibility that users and the network box can simultaneously send the message for the same purpose. This means that when the user sends a split message, the network box also sends IsSplit message as shown in Figure 7.9. This may lead to an undesirable situation and must be taken into account.

    To handle this situation, the solution presented in this thesis depends on a medium classification based on the tree structure shown in Figure 7.2. If Split and IsSplit messages have the same URI, then the IsSplit message will be rejected automatically by the caller/callee box. If the Split and IsSplit message have exclusively different URIs, the IsSplit message is shown in the user interface for the user approval. Similarly, medium classification also helps to reduce any conflict raised by a Retrieve and IsRetrieve request.

7.3.4 Limitations

This sub-section provides a list of requirements that are not considered when designing the solution for the PSTR in MOCSP.

The first issue is that changing the primary contact point of caller/callee during the session is not considered. Caller/callee box is the primary contact for caller/callee. Even if the media part of the session is transferred to near by devices, caller/call box will be active. In addition, terminating the session should be decided by caller/callee box. However, users sometimes wish to change the primary contact point to one of the devices that is part of a partial session. This aspect is also useful, but not considered in the current design and implementation. For this need, it is possible to extend the solution of the user mobility.

The following two requirements are mostly relevant with user-initiated partial session transfer and retrieval. When a user informs about the transfer, the user only receives the results - success or failure. However, it may be helpful if the user is provided with the referral progress indication. Since message loss is not possible in this solution, implementation of this feature will not be difficult and important. The last requirement is that the user should be able to inform the media devices to keep the session after issuing the transfer request. However, by default, the session will not be closed unless the respective user issues a close request.

7.4 Conclusion

This chapter demonstrates the solution for the PSTR use case, leveraging the MOCSP system. In the solution, the network box, behaving in a handle mode, supports the needs of the user and network-initiated transfer and retrieval. It is possible to have a partial session transfer and retrieval at the caller and callee

sides, simultaneously. A single orchestrator and separation of concern in the signaling protocol are the main design considerations for the PSTR solution.

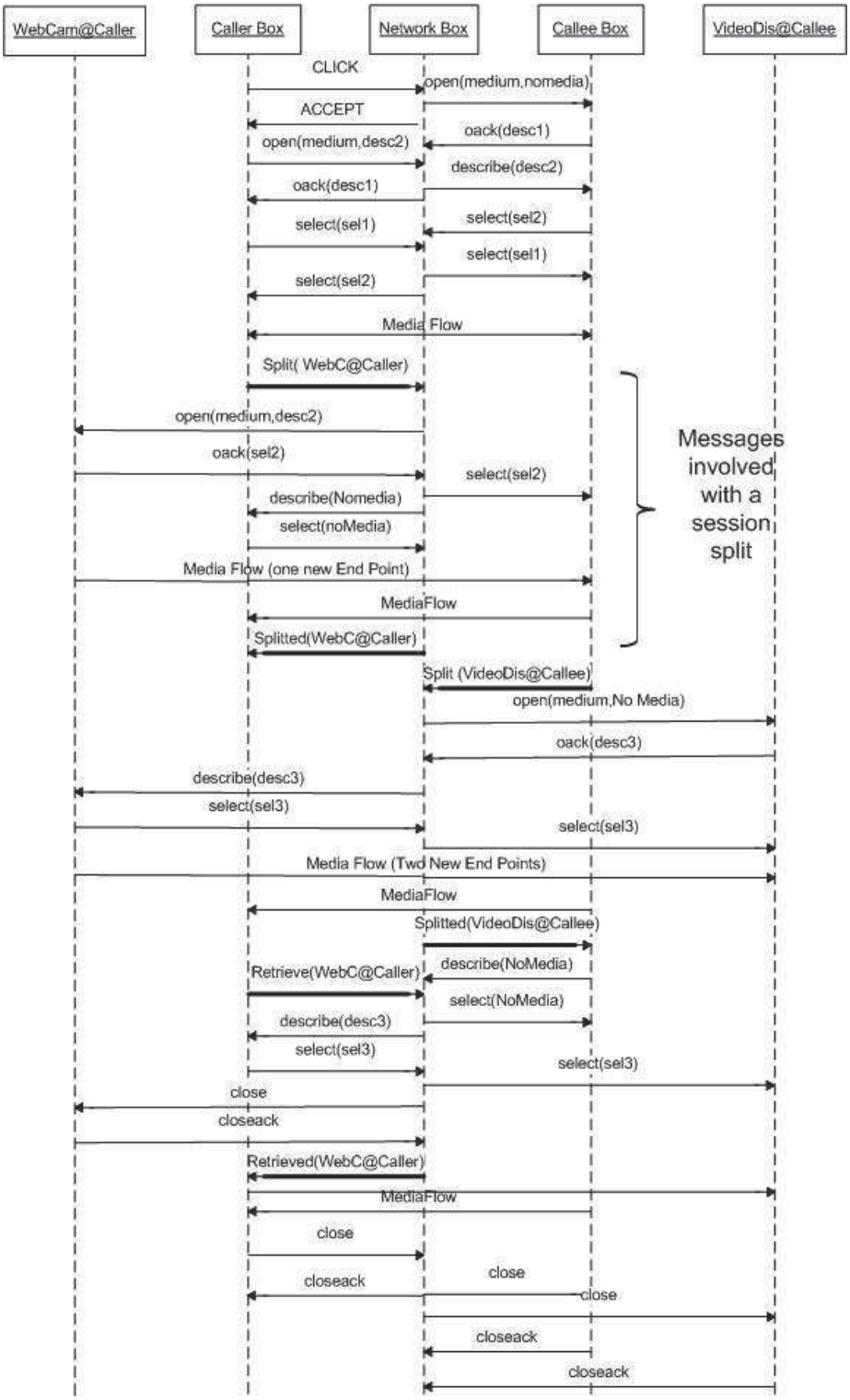


Figure 7.8: A call flow for user-initiated partial session transfer and retrieval

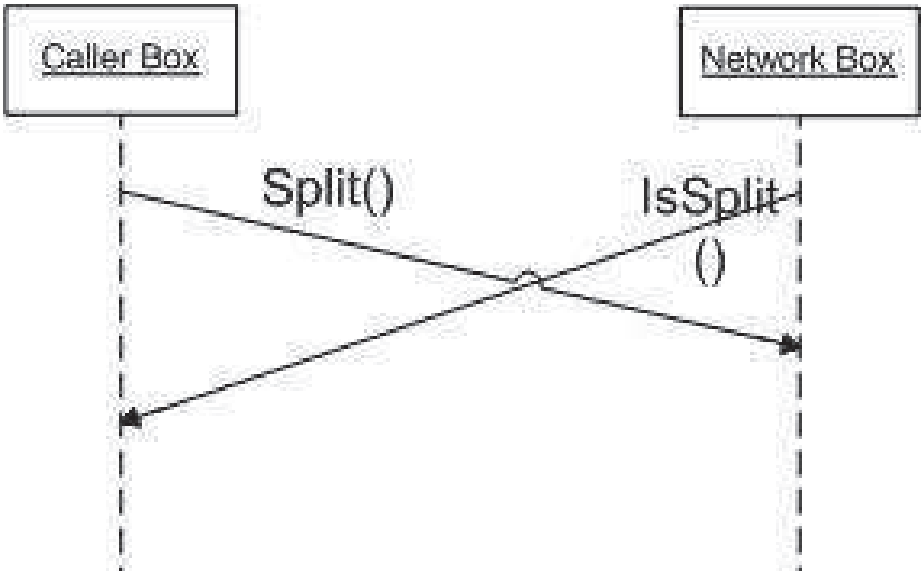


Figure 7.9: A sample scenario showing both ends sending a message for the same purpose.

Chapter 8

Scalability

Chapter 6 and 7 emphasize the need of having a central entity to manage the sessions. However, what will happen when many users intend to use these services at the same time? This chapter proposes a solution in order to support a higher call rate based on MOCSP, followed by a scalability analysis based on four parameters: scalability limit, complexity level, needed computing resources, and session setup latency. Impact of DNS is also included into the scalability analysis since MOCSP is realized over Web. This chapter includes four sections : 1) MOCSP based global architecture, 2) impact of DNS, 3) calculation of needed computing resources, and 3) scalability analysis.

8.1 MOCSP based Signaling Architecture

As explained in Chapter 5, each user should have a unique MOCSP instance in the Web platform, identified by a global Web identifier. In addition, callers and callees use their web browser for their sessions. A MOCSP instance provides relevant logic (e.g. logic implemented in JavaScript) for callers and the callee and deploys a network box in the web server. The network box (proposed in Section 6.1) manages a session between caller and callee.

The simple message flow path diagram between caller and callee is shown in Figure 5.3. More details can be found in Chapters 6, & 7. To make it clear, it is not necessary to use a specific signaling protocol in the MOCSP system, but the user can choose the one that satisfies his needs. The present researcher assumes that media streaming between caller and callee is established in a direct peer-to-peer manner. Therefore, scalability aspect with the media layer (e.g. media server) will not be an issue.

Overall architecture can be diagrammatically shown as in Figure 8.1, where users do not depend on each other for locating other users. Quite simply, this approach is individually running the MOCSP systems. Here, the number of circles is equivalent to the number of users who are using the MOCSP system. Seen in this way, there should be 6 billion MOCSP instances for providing services to 6 billion people. Each dot in Figure 8.1 is expanded as in Figure 5.3. The following paragraphs discuss three aspects: impact on Domain Name System (DNS) [108], roles of a service provider, and failure of a single physical node.

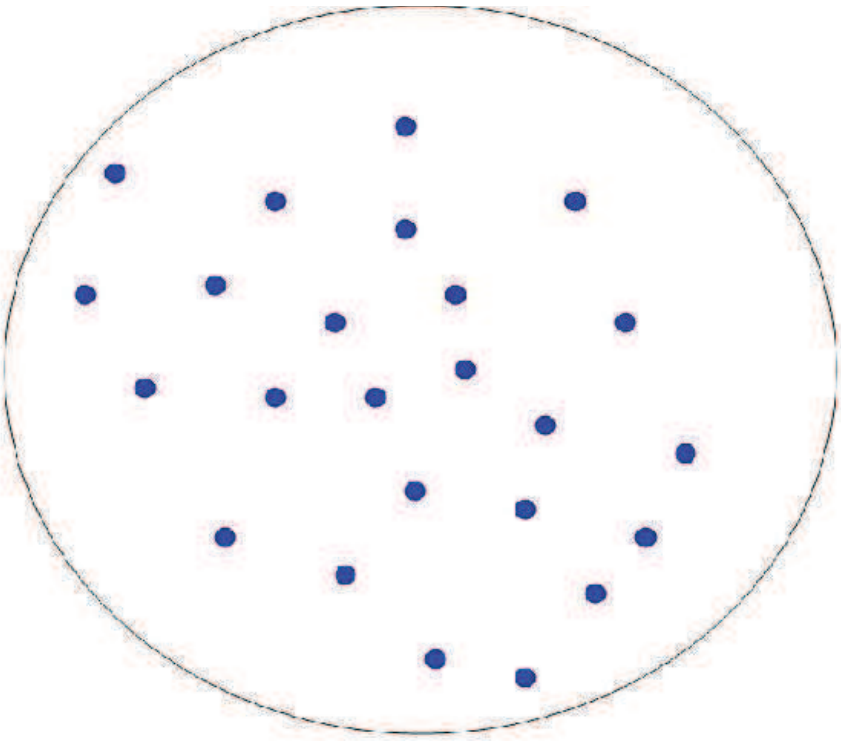


Figure 8.1: A scalable architecture for communication services. Each small circle is represented by each user/callee.

In the Web platform, the entity that is not visible to users is DNS. Impact on DNS should be considered when designing the MOCSP system. This thesis assumes that the number of DNS records should be equivalent to the number of users using the MOCSP system, as each user has an unique address. These entries are static, meaning that the IP address of the MOCSP instance will not change over the time (i.e., there is no need to change the IP address of the network box). As an example of a dynamic assignment, first the SIP UA1 connects one time to P-CSCF1, and later on it connects to another P-CSCF2 for the reason that P-CSCF1 would be under overload.

Whenever a callee logs into the MOCSP system, the callee depends on the DNS to resolve the address as it happens in the Web browsing. In this case, local cache of DNS resolver helps at the callee side. Similarly, callers also depend on DNS resolving. The dependency of the local cache of the DNS resolvers is associated with the frequency of the calls a caller makes with a particular callee. The MOCSP system does not assign any new functions to DNS, but only depends on it for address resolution. This resolution is similar to SIP where proxies and SIP ASs need to resolve SIP URI (which can be identifiers for users) to an IP address.

Another important aspect of the MOCSP system is how users deploy their MOCSP system in the Web platform. The user has to do two things: the first is to pick his unique Web address (e.g. URL) and the second is to host his

MOCSP system on a physical web server. For these two needs, users may need to depend on a service provider who can allocate a unique Web address and physical space on the Web server.

I discuss what will happen if one physical server fails. In this case, until the physical server is recovered, users who deploy their MOCSP systems will not able to receive calls. But they can still make calls through communication hyperlinks [13]. If other physical servers do not fail, those physical servers will perform as usual and will not receive much traffic.

During high call rate periods, the failure of some super nodes compels an outage of the overall system [90], because other super nodes receive more traffic including many retransmission messages. Therefore, P2P SIP-based infrastructure is less reliable compared with this thesis proposal, in which the network boxes are separated individually.

As suggested in [71], three factors compromise load scalability. They are: 1) scheduling of a shared resource, 2) scheduling of class of resources, and 3) insufficient exploitation of parallelism. In the MOCSP case, two factors out of three mentioned above are important. First exploitation of parallelism is natural in the MOCSP system. Every instance is unique and separated. Load scalability will be improved. Second sharing of resources (like in proxy server in IMS) is not an issue because of unique instances for each user. However, underlying infrastructure (Web server for MOCSP) and its scalability are not discussed here. This means that scheduling of shared resources (e.g. CPU, or memory) should be considered in the future.

8.2 Impact of DNS

The MOCSP solution depends on DNS because the solution develops on Web. In MOCSP, caller and callee need to depend on the DNS resolution to connect to the Web server (ref to Section 5.2). However, an analysis of the scalability of signaling layer based on MOCSP should consider the scalability of DNS. This section is devoted to discuss the latency of DNS lookup and scalability of DNS in detail. Initially, the basic concepts defined in DNS are presented. Next, the caching mechanism used in DNS to reduce DNS lookup delay and to improve the DNS scalability is discussed. Finally, a conclusion that is important to the MOCSP solution is presented.

8.2.1 Basic concepts in DNS

It is important to present the basic concepts and important terminology of DNS system. Therefore, it allows readers to understand easily latency, and scalability issues in the DNS system. These two issues make an impact on the proposed MOCSP solution.

The basic function of DNS is to map between human understandable hostnames (such as siva.mocsp.com) and IP addresses (both IPv4 and IPv6). Web clients (or resolvers) query DNS name servers for resolving hostnames. Typically, DNS is a globally distributed, scalable, hierarchical, and dynamic database. Apart from resolving hostnames to IP addresses, DNS can be used for a number of other purposes such as finding a host that handles the mail service for

a domain. All the different types of DNS queries are specified in the DNS specification. See the pointers in [109] to get relevant RFCs.

Even though DNS can be used for different applications (e.g. mail exchange), this thesis considers only Web applications. In this case, two kinds of mapping are needed. A mapping in DNS is determined by a resource record. The two common types of resource records are address records (A records) and name server records (NS records) [110]. An A record specifies the IP address of a hostname; an NS record specifies the name of a DNS server that is authoritative for a zone. Thus, NS records are used to handle delegation paths.

In DNS, the namespace is hierarchically organized as shown in Figure 8.2 (all the sub-domains from the root are not included). The root of the hierarchy is the central server called root server. Sub-domains (such as .com) are delegated to other servers that are called as authoritative servers. These authoritative servers are responsible for the portions of the namespace. The domains such as .com and .org are called top-level domains. The dedicated "generic top level domain" (gTLD) servers are responsible for each generic top level domain.

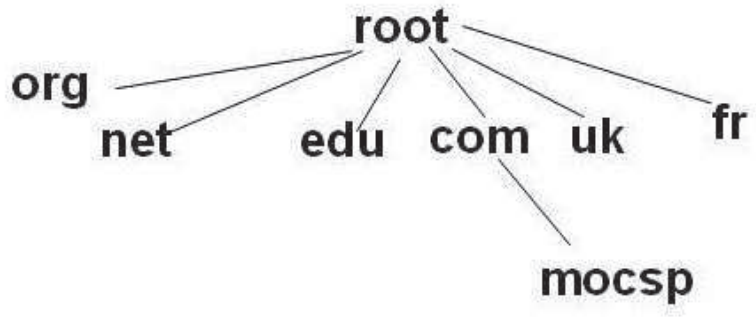


Figure 8.2: Hierarchical Namespace in DNS.

8.2.2 DNS lookup and Caching

Any website, for example http://www.siva.mocsp.com, should be reachable from a Web browser. To initiate the transaction with a host that is identified as http://www.siva.mocsp.com, the Web browser should find the IP address for the hostname. Then, the Web browser makes a TCP connection towards Web server.

In a typical Web browsing, the Web browser uses a stub resolver that looks at a local nearby DNS server (e.g., recursive DNS nameserver) for a hostname (e.g., http://siva.mocsp.com) into IP address resolution. If the recursive DNS nameserver does not have a response, it sends the request to the root DNS server.

The root server returns a referral response that is a collection of name server records, if delegated responsibility for a particular name is saved in the root server. By choosing one of the name server records, the local server repeats the

Step No	Description
1	Query for address of siva.mocsp.com
2	Referral to com nameserver
3	Query for address of siva.mocsp.com
4	Referral to mocsp.com
5	Query for address of siva.mocsp.com
6	Address of siva.mocsp.com

Table 8.1: Description of each step in Figure 8.3

question until it finds the answer. The graphical description of the DNS lookup is shown in Figure 8.3.

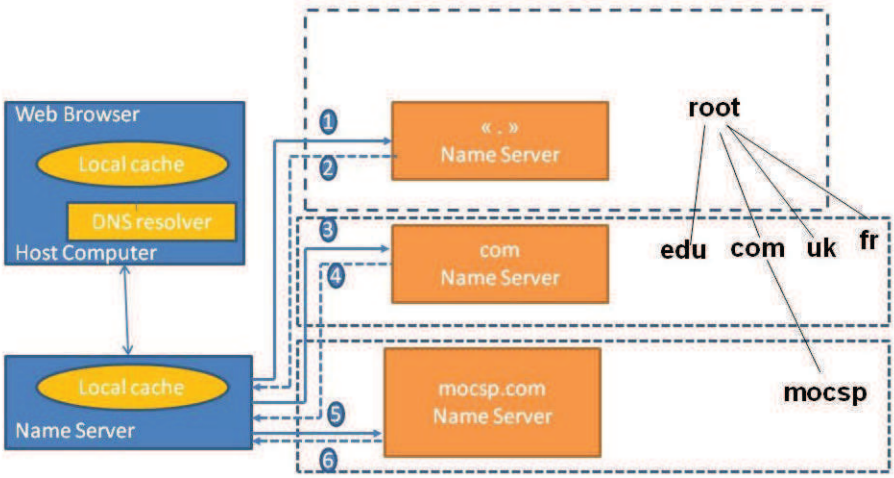


Figure 8.3: Resolution of http://siva.mocsp.com in the Internet.

As shown in Figure 8.3, the hostname-to-IP address resolution has to traverse many nodes and each resolution request must reach the root server. Therefore, DNS lookup latency is high and root server may become overloaded. To overcome these two problems, a DNS caching technique is proposed. More information about DNS cache is available in [111]. DNS cache means that the resource records from authoritative name servers are stored in non-authoritative name servers. These non-authoritative name servers (for example, local cache server in Figure 8.3) respond to the user requests quickly.

It is important to make sure that copies of resource records in non-authoritative servers are consistent with original resource records over the time. Therefore, it is imperative to control DNS cached records. Here, the basic idea is to control the cache of each resource record using expiration time that is referred as Time To Live (TTL) in the DNS specification. Each resource record is expired in a DNS cache server according to the value set in the TTL parameter. By adjusting the TTL value, the administrator of a domain can control the valid resource

record in the DNS server. After the expiration of the resource record, the DNS cache server fetches data afresh from the authoritative origin server once a new query reaches the DNS cache server.

The scalability of DNS is widely believed to be high because of two factors. One is hierarchical design around administratively delegated name spaces and aggressive use of caching. DNS performance and Effectiveness of DNS caching is discussed in [111], which gives more importance to NS records than A records. This argues that caching NS records substantially reduces the DNS lookup latency even though it may involve some referrals to complete the lookup. Cached NS records are especially beneficial as they greatly reduce the load on the root server. Consequently, the cacheability of NS-records that efficiently partition the name space is a major reason for the scalability of DNS.

For example in Figure 8.3, the response obtained in step 2 and 4 are stored in the resolver cache during the particular TTL value. If the NS record of the mocsp.com domain is cached in the local cache server for one day (assume that TTL value for the NS record is 24 hours), the same query requests within 24 hours will be resolved with two hops (local cache server and authoritative name server). It means that the local cache server directly finds the authoritative server and gets the response back to the client. There is another possibility that the local cache server can keep the A- records for specified TTL value. This will reduce the latency, as well. However, the effect of caching A-record is not considered for the scalability analysis. Finally, without cache, it takes 4 hops for the query to resolve as shown in Figure 8.3.

In a nutshell, the caching aspect in DNS reduces latency and increase the scalability by eliminating the DNS requests to the root server and the corresponding top level domain servers.

DNS cache at client side

Additionally, DNS cache is available in the operating systems to avoid the DNS lookup delay for the regularly visiting websites.

In this case, DNS A-records are kept in the OS for a period as specified in Time to Live (TTL). If TTL becomes zero, the resolved domain name in the cache is removed and a new request to that domain is forwarded to the nameserver. Each DNS record has also a unique TTL value. Therefore, the latency with DNS name resolution (for the previously resolved name) would become zero, performing better than caching A-records and NS-records in the local cache server.

The mechanisms at the client side vary across different OSs. Detailed information about DNS cache in windows OS can be found in [112], including configurations, activation and deactivation services.

8.2.3 Conclusion: DNS lookup delay and Scalability of DNS

The four different configurations are possible to perform the DNS lookup. The first one is typically following the DNS hierarchy path. The second and third one are caching A-records and NS-records in the local cache server. And the last one is caching in client side. Out of these four different configurations, this thesis depends on caching NS-records in the local cache server for analyzing

the DNS lookup delay and scalability aspect. Therefore, the average lookup delay is 2 hops (local name server and authoritative server). This assumption is consistent with the possible DNS lookup delay for Web browsing [113].

In MOCSP, each session incurs 2- hop delay (on an average). This means that the caller needs to find the callee's Web address before sending signaling messages to make a session. It is possible to avoid the 2-hop delay if a caller regularly makes many sessions with the the same callee and have a caching mechanism at the client side.

While considering scalability, the authoritative server for the mocsp.com domain should have sufficient computing resources to handle the anticipated DNS query requests (equivalent to the number of sub-domain in the mocsp.com). In the worst case scenario, no caching mechanism is used and the other three DNS servers (root server, top level domain name server and name server of mocsp.com) should have similar capacity to serve the anticipated DNS query requests.

8.3 Calculation of Number of Servers

This section presents the estimation of the Web servers when providing six billion MOCSP systems.

8.3.1 Performance of the Web server

System Performance Evaluation Cooperative (SPEC) is one of the most successful performance standardization bodies. SPEC benchmarks high-end work stations and servers such as SIP server and Web server. WEB2005 and WEB 2009 are available benchmarks for measuring the performance of the Web servers; they consider three different work load – banking, e-commerce and support [114]. Each workload calibrates the maximum number of simultaneous user sessions that a web server is able to support while still meeting the specific throughput and error rate requirements.

Apache Web server can manage around 50000 concurrent connections in a server [115] & [116]. [116] indicates that all the concurrent connections support to download big data content as specified in SPEC web 2005 [114]. This benchmarked Web server helps to estimate computing resources for the solution stated in this thesis.

8.3.2 Number of Network boxes in a Web server

In MOCSP, a network box is hosted in the Web server to manage the session between callee and caller. Therefore, two TCP connections are needed for a session. As shown in the sample deployment scenario in Figure 8.4, if a Web server hosts five network boxes, then 5 TCP connections are used for callee's connections only. It means that the Web server should have capacity to open the new 5 TCP connections for the caller's side at any time. Therefore, a Web server hosts a number of network boxes that should be half the number of concurrent TCP connections that Web server can handle at a time. The single Web server [116] hosts 25 000 network boxes or serves 25 000 users.



Figure 8.4: A sample deployment scenario.

Based on this calculation, I estimate that around 240 000 similar Web servers [116] (25 000 into 240 000 is equivalent to 6 billion) are enough to provide communication services for 6 billion people.

8.3.3 Discussion on the Calculation

It is important to consider the behavior of TCP based servers, because the behavior impacts on the calculation. As said in [73], "maintaining TCP connections affects neither the performance of establishing new TCP connections nor of exchanging user data". The main bottleneck is with CPU cycles and kernel memory when each connection is sending data simultaneously. It means that the sustainable request rate and the transaction response time are important parameters for performance. The same authors indicate that the bottleneck of sustainable request rate is the thread queue, where the number and lifetime of threads cause queuing delay of threads in the thread-pool model. I believe that any of the findings in [73] will not affect the calculation in this thesis, because the size of the signaling message is less than the size stated in [115] [116]. In this MOCSP model, each TCP connection may carry maximum of 5 into 500 bytes (open/oack/describe/select) during the session setup.

8.3.4 Virtual Web Hosting

Virtual web hosting is a technique for running multiple virtual web servers on a single physical host computer. This technique is used by commercial web hosting service providers because of manageability, efficiency and scalability of the service infrastructure. This technique totally depends on virtual DNS resolution that can be achieved by adopting two different approaches: name-based and IP-based.

In the name-based approach, each web site in the single machine shares a single public IP address using a unique name. The Web server forwards the HTTP requests to the right virtual site based on the information supplied in the form of host field in an HTTP request. The host header field is supported by almost all browsers today. RFC 2616 defines the Host request-header field that specifies the Internet host and port number of the resource being requested.

The IP-based method gives an unique IP address to each virtual web site. The HTTP requests are resolved based on the IP address, but not the name. The name-based approach is preferred compared with the IP-based approach on account of IP address limitation and operating system limitation. On the other hand, a few technical limitations (e.g. secure socket layer) prevent using name-based virtual hosting.

This thesis argues that each MOCSP instance is a web site, so the name-based approach should be used for the virtual DNS resolution when deploying many MOCSP instances in a single physical machine. It is also worth noting that the IP-based method does not bear any difficulties and it should be easy to get an IP address for each MOCSP instance when IPv6 is ubiquitously rolled out.

8.4 Scalability Analysis

Based on the proposed scalability framework in Section 2.4.1, this section provides a scalability analysis for the MOCSP architecture. This scalability analysis considers two components: MOCSP and DNS. MOCSP is responsible for session management and executing services (See user mobility and PSTR) and DNS helps Web browser to perform the hostname to IP resolution.

8.4.1 How Scalable

By default, each MOCSP instance is separated and individually needed resources will increase along with a number of network boxes and a number of calls received. Therefore, it is easy to adopt an approach of vertical scalability. This means that the number of MOCSP instances that can be added is limited by the capacity of a single physical server or resources have to scale along with the number of network boxes deployed and the number of calls received. Based on this approach, the overall system is scalable.

Relying on the distributed arrangement and cacheability of NS-records and A-records, DNS is scalable - that is able to handle many DNS requests.

8.4.2 Complexity

The MOCSP architecture is straightforward and simple. The users who implement the MOCSP systems do not need to implement load balancing techniques and need not worry about the routing aspect defined especially in SIP and P2P overlay. A session is established between two users, based on TCP connections. The messages needed for a session are sent back and forth between two web browsers via a network box. At last, the service provider should consider a proper calculation of deploying of network boxes in a single server.

It is worth mentioning about the difference between MOCSP and existing communication systems. The key issue in the existing communication system has a server federation. This means that in typical scenarios, if two SIP servers (Assume single session passes through those two servers) are operated by different entities, signaling protocol should be agreed upon, either by standardization or by other means of agreement. This means that both should talk using the same protocol or one should provide a gateway function. Furthermore, if there

are many operators, routing and configuration should be defined in advance to provide services across the operators.

This kind of federation contributes much to complexity of the existing communication systems. For example, the XMPP federation configuration is a difficult task even for a system administrator, as reported in [117]. The MOCSP approach completely eliminates the complexity associated with the federation of nodes or servers. In MOCSP, signaling messages of the session originator will be forwarded to the proxy (i.e. network box) of the receiver thanks to the Web and DNS system.

The relevant DNS mechanism is presented in Section 8.2. Initial configuration and operation of DNS is relatively easy and does not cause complexity.

8.4.3 Needed Computing Resources

As analytically proved in section 8.3, the complete system needs approximately 240, 000 Web servers for supporting 6 billion people who use MOCSP systems. In this case, any 3 billion people can call the remaining 3 billion people. In this calculation, computing resources needed for the media layer are not considered. But computing resources required for DNS operations should be included into the evaluation. At least, the name servers for the mocsp.com domain should have enough computing capacity to handle the higher number of requests per second (e.g. three billion calls per second). Additionally, required computing resources at the local (cache) name server and root name server are also taken into account.

Even though it is possible to calculate needed computing resources for the MOCSP systems, it is not possible to quantitatively compare the MOCSP system with the existing other communication systems. Analytically, I argue that MOCSP needs minimum resources. For a session, a network box that consists of two TCP connections should handle the session and perform state management (i.e. stateful mode) for caller and callee. In the MOCSP system, a network box needs computing resources only for state management and forwarding and there is a resource demand for a DNS lookup. In contrast, two nodes (S-CSCF2 and B2BUA2 in Figure 2.6) at least should perform the state management and messaging forwarding in IMS. When the number of calls increases, there should be a load balancing entity (sometimes more) in the dynamic assignment (See Section 2.4.3). This overhead will increase based on different configurations. In P2P SIP, message overhead is proportional to log(N) and will be higher when the number of nodes is very high (say billions)

To be more precise, the role of the service provider is clarified again here. There might be a single provider who can install all the needed Web servers (roughly 240, 000 physical servers). However, obtaining an unique name and Web space is the role of the end user who has to design the signaling protocol based on his needs and deploy it in the Web server.

When compared with the pre-defined assignment approach in IMS, the number of physical servers is less in the MOCSP approach because the only entity here is between caller and callee.

Essentially, computing resources needed for the DNS name servers are also included in overall calculation of needed computing resources. These DNS name servers should serve for the highest number of DNS query per second.

8.4.4 Session Setup Latency

The session setup latency incurs at the network box and DNS. The initial lookup will be performed by each caller via DNS. After that the caller sends and receives messages (HTTP messages and messages over Web socket) to and from the network box in order to establish a session.

In Figure 5.3, the time between sending the CLICK message and receiving the select(sel2) message is referred to as session setup latency. The CLICK message is an HTTP request that needs to resolve the hostname into an IP address at the beginning. The average DNS lookup delay is two-hop. Based on the discussion before, session setup delay is proportional to three-hop delay.

Additionally, relying on the capacity planning (presented in Section 8.3), session setup latency can be independent of the number of calls made in the MOCSP system or session setup time will not be dependent on the load in the server.

8.5 Conclusion

The scalability analysis based on the MOCSP system is presented in this chapter. The analysis framework is based on the following four parameters - scalability level, complexity level, required computing resources and session set up latency. MOCSP is a scalable method when increasing computing resources (for MOCSP Web server, and the mocsp.com nameserver). Additionally, this method does not introduce complexity.

Chapter 9

Discussion

The previous four chapters, i.e., Chapter 5, 6, 7 and 8 offered solutions for the research problems of this thesis. The basic solution goes for a new design from scratch, not for extensions to existing communication systems. It is important to verify that the new solution is able to support the already existing services. In this light, this thesis finds two services and evaluates whether these services can be implemented in this new system. These two services do not show any similarity to the established research problems.

Initially, this chapter presents a use case of transfer call control that involves three parties, followed by a solution in SIP and MOCSP. Later, there is a discussion on IP mobility and its impact. Finally, the groundwork for implementing the solutions is presented.

9.1 Transfer Call Control

RFC 5589 listed a number of transfer services including blind transfer, consultative transfer and attended transfer [118]. This kind of call transfer is very useful in people's day-to-day life, for example, a customer calls to an agent who in turn transfers the call to the expert after consulting with the expert. These services are well-addressed in the SIP standard.

As explained in RFC 5589, fundamentally, this kind of transfer is multiparty call control and needs a minimum three parties. These call transfers take place after the initial media establishment. Call transfer is achieved using the REFER method [27]. This kind of situation is not discussed in the MOCSP system. In the user mobility use case, there are two parties and one is moving across different terminals. In PSTR, only two parties are key entities and the network box coordinates the media flow across devices near to the parties. Therefore, looking for a solution for transfer call control will shed some light on the new design.

This section discusses the transfer with consultation hold. Initially, transfer call control protecting the transfer target ([118] Section 7.2) is presented, including the SIP based solution. Later, the possible difficulties while providing this service in the MOCSP system is examined .

9.1.1 SIP based Solution

First of all, as mentioned earlier, three parties are involved in a call transfer, the description of each party is as follows:

I. Transferee: the party that is being forwarded to the Transfer Target.

II. Transferor: the party that triggers the transfer.

III. Transfer Target: the new party that is being introduced into a call with the Transferee.

Like a customer's call to an agent (as explained previously), transfer with consultation hold involves a session between the Transferor and the Transfer Target before the transfer actually takes place. This is implemented with SIP Hold using INVITE and Transfer using REFER.

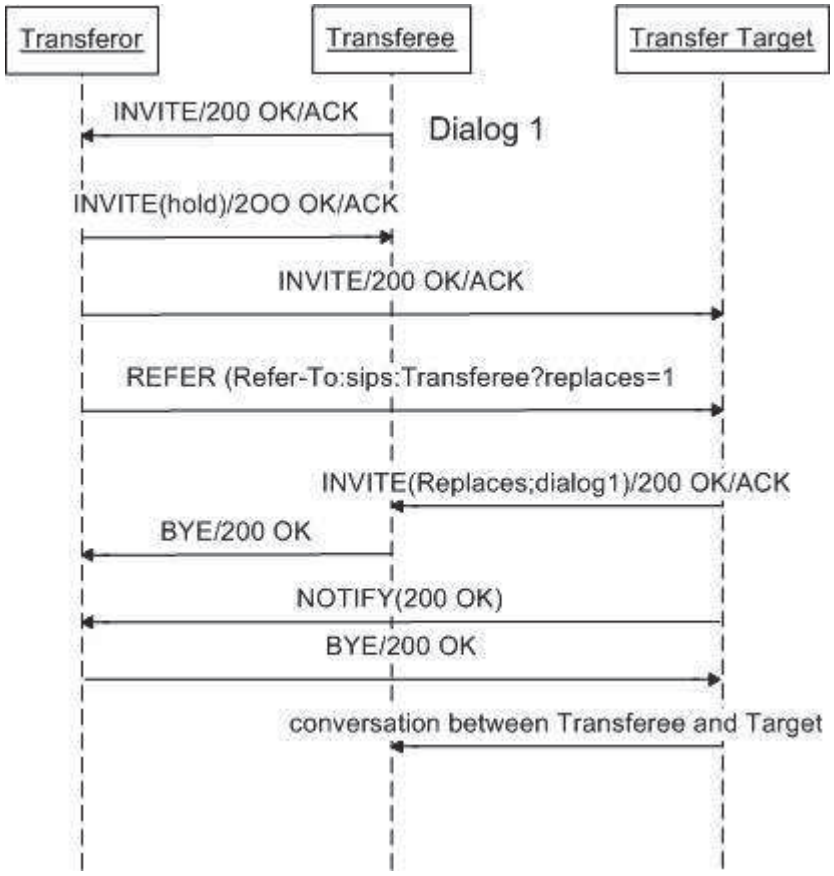


Figure 9.1: SIP call flow for call transfer with consultation hold, protecting the transfer target

In this case, Transfer Target is informed by receiving the REFER method that has the address of Transferee. Here, Transfer Target is protected, because Transferor does not provide the identity of Transfer Target to Transferee. The

relevant call flow is shown in Figure 9.1 based on [118](Ref. Section 7.2). Here, one assumption is made that the Transferee's agent has a reliable mechanism that associates a new call with the call it already has with the Transferor. This means that there is no new call on the appearance.

9.1.2 MOCSP and Transfer Call Control

Figure 9.2 shows the entities needed for transfer with consultation hold. Here, Transfer Target is protected. This means Transferee does not know Transfer Target. Initially, caller box (Transferee) initiates a call with callee box (Transferor). The relevant call flow for this interaction is available in Section 5.2. Now, callee box (Transferor) thinks of consulting with an expert. So, callee box (Transferor) makes a call to callee box (Transfer Target). Here, callee box (Transferor) puts caller box (Transferee) on hold. Obviously, desc(nomedia) is used for putting the call on hold. These two different calls need to depend on the corresponding network boxes. In fact, there is no change in the signaling protocol.

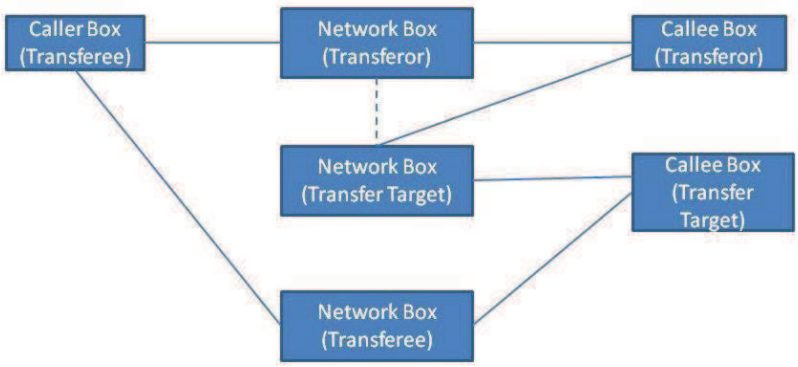


Figure 9.2: Entities involved in transfer with consultation hold (protecting Transfer Target) call in MOCSP

While discussion continues, callee box (Transferor) decides to make a connection between caller box (Transferee) and callee box (Transfer Target) and moves out of call. For this purpose, one message (like Split message in the PSTR solution) should be defined. However, the problem is with the identity of caller box (Transferee) that is not global, but only known to the network box (Transferor), in the MOCSP system. Caller makes a call based on the communication hyperlink provided by callee.

To solve this problem, there are two solutions. The first one is that caller box (Transferee) explicitly mentions its network box to callee box (Transferor). Then, it will pass to callee box (Transfer Target) that makes a new session. This is a simple solution, but requires an input from the user. The second solution is based on the coordination of network box (Transferor) and network box (Transfer Target). The relevant network boxes are connected by a dotted line in Figure 9.2. This thesis leaves the solution to this coordination of network

boxes, but it is typically harder than the first solution, because of the federation across servers/network boxes.

9.2 IP Mobility

A communication session is forced to terminate when a device changes its IP address due to mobility or connectivity to different access networks. Change of IP address of mobile terminal can not be avoided in many situations such as connectivity from 3G to 3G Femto Access Point (not always changing the IP address) or 3G to WLAN, etc. When this IP mobility takes place, it is important to give user experience by continuing the session without disturbing or expecting inputs from users.

Mobility has been a widely discussed topic and this thesis considers an application layer solution rather than other layer solutions, for example on an IP layer (e.g. Mobile IP). Explicitly, the reason for this choice is that the application layer solution is easy to implement. In a typical situation, both terminals (caller and callee sides) may undergo IP mobility during the session. This is the worst case situation, as well. The following subsection explains the best solution based on the SIP. Then, a possible solution based on the MOCSP system is presented.

9.2.1 SIP based Solution

SIP protocol has a simple solution for IP mobility. This means that the terminal only needs the RE-INVITE method in order to support the IP mobility. The terminal device that has undergone IP mobility, sends a RE-INVITE message to the other end of the session. The SIP specifies the RE-INVITE method in order to update the session parameters such as IP address of endpoint, codec, etc. The RE-INVITE message can be sent only after the initial INVITE has been completed.

In fact, the solution has two parts - off-call and on-call. As the name implies, off-call means users are not in a call. During the IP mobility,"off-call" mobility management consists of the Registration process. The "on-call" handover is performed using the RE-INVITE messages. After the IP mobility, the SIP RE-INVITE message is sent to the opponent's side. In the communication services, both sides can undergo IP mobility at the same time. This situation triggers both sides to send the RE-INVITE messages. Thus, it will create a race condition. Consequently, the delay to re-setup the session due to the race condition is higher than of 2s [64].

During the IP mobility, the SIP based solution does not require user attention. Both endpoints (callee and caller) update automatically the session descriptors. Here, any intermediate entity (e.g. B2BUA) is not part of the session establishment.

9.2.2 MOCSP and IP Mobility

In HTTP, messages are sent between Web server and Web browser, in an established TCP connection. A TCP connection is identified uniquely by a set of four parameters: source IP address, source port number, destination IP address,

and destination port number. During the single IP mobility in the end-device, the established TCP connection is terminated and a new TCP connection has to be established by the client, therefore, message loss is possible. In this case, application layer has to take care of message loss. In the Web browsing context, application layer does not consider the message loss, instead it makes a new HTTP request.

TCP Migrate

TCP Migrate is a mechanism to migrate a new TCP connection that uses another IP address/TCP port pair in an application-transparent manner [119]. Therefore, applications do not consider message failures and recoveries. Also, a migrate compliant TCP stack should be deployed in the client side and server side.

MOCSP based Solution

The solution consists of two parts - off-call and on-call - like a SIP. During the off-call situation, callee only needs to connect to the Web server, not caller. This will take place before callee gets an open message (Ref Figure 9.3). The on-call situation is shown in Figure 9.3 marked with numbers 2 and 5. If IP mobility takes place, this device should send a new description message. This means that with two unilateral messages (desc/selec), the session can be continued.

If callee changes the IP address, callee sends the media immediately. But callee receives the media once it receives the select message from caller. This situation is the same when a caller alone changes the IP address. However, if both change the IP address, both will send the describe message unilaterally.

However, much care should be given to places marked with the numbers 1, 3 and 4 in Figure 9.3. Here, reliability and message loss are taken into account when proposing a solution. The similar situation is not considered in SIP, where RE-INVITE can not be sent before an INVITE transaction is finished. In order to address this problem, TCP Migrate is considered for the MOCSP based solution for IP mobility. TCP reliability is maintained, whereas message loss will be eliminated. In those regions, message loss is simply recovered, thanks to TCP migrate. However, application logic should verify whether a new description is used for the media transport.

9.3 Implementation Choices

In the middle of the thesis, two different implementations are being carried out by two different players - WebRTC.org [94] and Ericsson Labs [120]. In fact, the MOCSP system has two layers - signaling and media layer, but on the implementation, components can be classified at the Web browser and Web server sides. While Web browser deals with the media transport (sending and receiving) and signaling part, Web server deals with the signaling messages. The following section describes the recent implementation in Web browser and Web server.

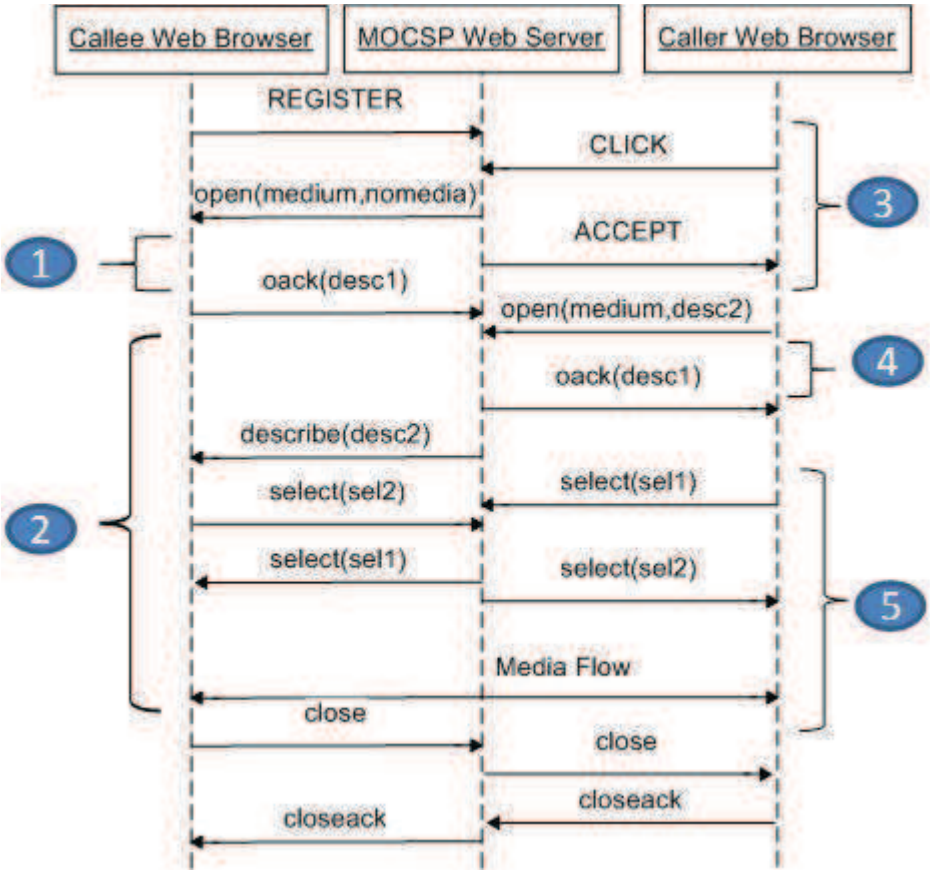


Figure 9.3: A typical call flow in the MOCSP system with important points that need to be considered during IP mobility

9.3.1 Web Browser

The existing Web browsers need a plug-in module in order to natively support the real-time communication services. To avoid plug-in dependency, W3C through its Web Real-Time Communication Working group coordinates in defining and specifying all the required features in the Web browser. The scope of the W3C is to define:

I. API functions to explore device capabilities, e.g. camera, microphone and speakers (currently in scope for the Device APIs & Policy Working Group)

II. API functions to capture media from the local devices (camera and microphone) (currently in scope for the Device APIs & Policy Working Group)

III. API functions for encoding and other processing of the media streams,

IV. API functions for establishing direct peer-to-peer connections, including firewall/NAT traversal

V. API functions for decoding and processing (including echo canceling, stream synchronization and a number of other functions) of those streams at the incoming end,

VI. Delivery to the user of the media streams via local screens and audio output devices (partially covered with HTML5)

Even though W3C does not produce the stable standards, it is worth mentioning about some of the important APIs. The latest draft of W3C [1]-WebRTC 1.0: Real-time Communication Between Browsers-included the specification of Stream API, Peer-to-Peer Connections, The data stream, Garbage collection and Event definition. Details of Stream API, Peer-to-Peer Connections, and Event Definition are presented as follows:

I. Stream API: This group has seven important interfaces – MediaStream, LocalMediaStream, MediaStreamTrack, AudioMediaStreamTrack, MediaStreamTrackList and MediaStreamrecorder. The key need of sending and receiving of different media (audio or video) is considered in the specification.

MediaStream represents different streams of media regardless of local and remote location. The MediaStream class is presented as follows:

[Constructor (MediaStreamTrackList? audioTracks, MediaStreamTrackList? videoTracks)]
interface MediaStream {
readonly attribute DOMString label;
readonly attribute MediaStreamTrackList audioTracks;
readonly attribute MediaStreamTrackList videoTracks;
MediaStreamRecorder record ();
attribute boolean ended;
attribute Function? onended;
};

II. Peer-to-Peer Connections: PeerConnection is an important interface defined for the dealing of media transport between two Web Browsers. It is a coarse-grained interface, taking care of NAT/Firewall, SDP creation and processing and media transport. A complete information of the PeerConnection interface is given below:

[Constructor (DOMString configuration, SignalingCallback signalingCallback)]
interface PeerConnection {
void processSignalingMessage (DOMString message);
const unsigned short NEW = 0;
const unsigned short NEGOTIATING = 1;
const unsigned short ACTIVE = 2;
const unsigned short CLOSING = 4;
const unsigned short CLOSED = 3;
readonly attribute unsigned short readyState;
const unsigned short ICE_GATHERING = 0x100;
const unsigned short ICE_WAITING = 0x200;

```
const unsigned short ICE_CHECKING = 0x300;
const unsigned short ICE_CONNECTED = 0x400;
const unsigned short ICE_COMPLETED = 0x500;
const unsigned short ICE_FAILED = 0x600;
const unsigned short ICE_CLOSED = 0x700;
readonly attribute unsigned short iceState;
const unsigned short SDP_IDLE = 0x1000;
const unsigned short SDP_WAITING = 0x2000;
const unsigned short SDP_GLARE = 0x3000;
readonly attribute unsigned short sdpState;
void addStream (MediaStream stream, MediaStreamHints hints);
void removeStream (MediaStream stream);
readonly attribute MediaStream[] localStreams;
readonly attribute MediaStream[] remoteStreams;
void close ();
attribute Function? onconnecting;
attribute Function? onopen;
attribute Function? onstatechange;
attribute Function? onaddstream;
attribute Function? onremovestream;
};
```

A detailed description of each element is given in Section 4.1 of [1].

III. Event: There are many events associated with all the interfaces defined above. The summary of defined events (See Section 8 of [1]) is as follows:

Interface Name	The list of events
MediaStream	ended
MediaStreamTrack	muted, unmuted, ended
MediaStreamTrackList	addtrack, removetrack
PeerConnection	Connecting, open, message, addstream, removestream

Table 9.1: List of Events defined in [1]

Based on these defined APIs, this thesis includes two different implementations carried out by WebRTC.org and Ericsson Labs.

1. WebRTC.org

   The overall architecture proposed by WebRTC.org [94] is modified as shown in Figure 9.4 in order to include the MOCSP application. This architecture intends to show two different groups of users – web browser developer and Web app developer. MOCSP is developed by a Web app developer, interfacing the Web browser via Web API edited by W3C (See the starting of this section).

   WebRTC provides voice engine, video engine and transport components, giving flexibility to Web browser developers to develop custom modules for audio capture/display, video capture, network I/O and Peer Connection API. More importantly, the session management is still under the control of WebRTC.org, who implements the signaling at least at the abstract level. This

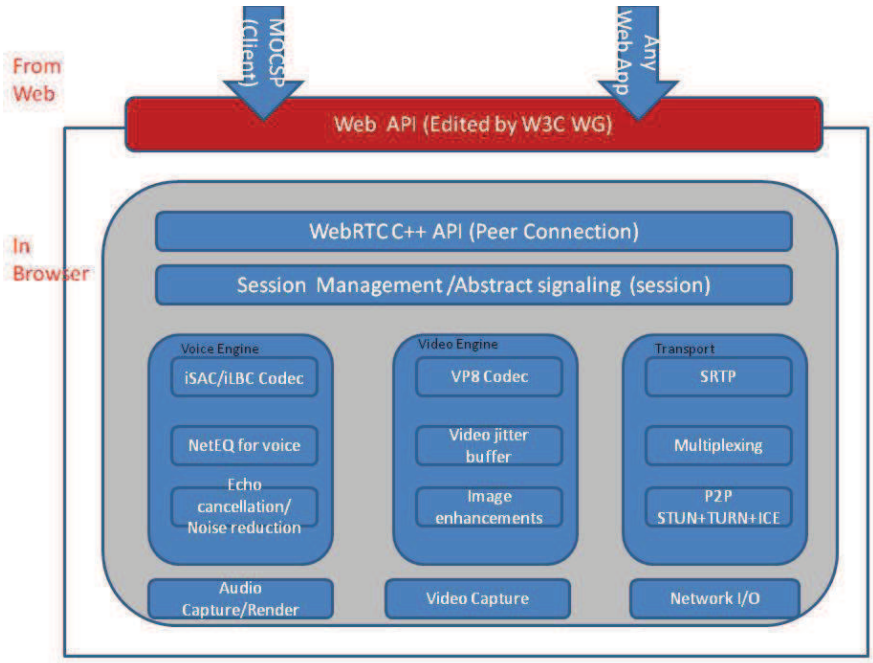


Figure 9.4: Architecture of WebRTC including MOCSP client side.

component should be studied further while the implementation gets mature, because this thesis proposes to give complete control of the signaling to the Web developer. For the usage, MOCSP can benefit from the current WebRTC.org solution, but it is not mandatory.

The details of some components in Figure 9.4 are provided here:

1. Transport/Session: Some components of libjingle are re-used to develop the session components, without requiring the XMPP/Jingle protocol.

   RTP Stack - Real time protocol is used for media transport.

   STUN/ICE - This component allows calling over a network that has STUN and ICE mechanisms.

   Session Management: This component intends to give a high level view of signaling protocol, leaving the protocol implementation decision to application developers.

2. Voice Engine: This component takes care of audio media chain, from sound card to network interface. iSAC, iLBC, NetEQ for voice, acoustic echo canceller (AEC), noise reduction (NR) are important sub-components within this component. iSAC is a wideband and super wideband audio codec for VoIP and streaming audio, using 16 KHz or 32 KHz sampling frequency with an adaptive and variable bitrate of 12 to 52 kbps.

   iLBC is a narrowband speech codec, defined by IETF RFC 3951 and 3952. It uses 8 kHz sampling frequency with a bitrate of 15.2 kbps for 20ms frames and 13.32 kbps for 30ms frames.

NetEQ: This component keeps latency as low as possible while maintaining the highest voice quality. This means that reducing the effects of network jitter and packet loss is made possible using a dynamic jitter buffer and error concealment algorithm.

Removing the acoustic echo (resulting from voice being played out and coming into the active microphone) is performed by a software based signal processing component, AEC.

Another component that removes certain types of background noise usually associated with VoIP call is called NR.

3. Video Engine: Like voice engine, video engine considers the video part, from camera to network and network to screen. The video codec is VP8 from the WebM project. VP8 is designed for low latency. In addition, the other two sub-components are video jitter buffer and image enhancements. Like NetEQ, video jitter buffer conceals the effects of jitter and packet loss. Image enhancement removes video noise from the image capture by the webcam.

The PeerConnection interface is well explained in [94] in terms of how it should be developed. There are two classes - PeerConnection and PeerConnectoinObserver - defined by WebRTC.org to implement the PeerConnection interface. These details are very important for Web App developers including MOCSP.

PeerConnectionObserver class is an abstract interface for a user defined observer. Registering an observer class can be performed using RegisterObserver(). The details of PeerConnectionObserver class are as follows:

```
class PeerConnectionObserver {
public:
  virtual void OnError();
  virtual void OnSignalingMessage(const std::string& msg);
  virtual void OnAddStream(const std::string& stream_id,
                           int channel_id,
                           bool video);
  virtual void OnRemoveStream(const std::string& stream_id,
                              int channel_id,
                              bool video);
};
```

Information regarding the functions and their signatures can be found in [94]. Similarly, the detail of PeerConnection is as follows:

```
class PeerConnection {
public:
  explicit PeerConnection(const std::string& config);
  bool Initialize();
  void RegisterObserver(PeerConnectionObserver* observer);
  bool SignalingMessage(const std::string& msg);
  bool AddStream(const std::string& stream_id, bool video);
```

```
  bool RemoveStream(const std::string& stream_id);
  bool Connect();
  void Close();
  bool SetAudioDevice(const std::string& wave_in_device,
                      const std::string& wave_out_device);
  bool SetLocalVideoRenderer(cricket::VideoRenderer* renderer);
  bool SetVideoRenderer(const std::string& stream_id,
                        cricket::VideoRenderer* renderer);
  bool SetVideoCapture(const std::string& cam_device);
};
```

The OnSignalingMessage function in the Peerconnection class and SignalingMessage function in PeerConnection class are used to pass SDP messages between peers. Now this implementation arrangement supports only the offer/answer mechanism.

2. Ericsson Labs Solution: The solution is almost similar to WebRTC.org, but, it depends on open source codecs such as for Voice, Speex and for video, Theora, both are available in Ubuntu. The Ericsson modified WebKit library allows to experiment the PeerConnection and MediaStream APIs [120]. The relevant source code is available now at [121].

9.3.2 Web Server

In MOCSP, the rendez-vous function is performed in the Web server that may host many MOCSP instances, so that it may expect many concurrent requests. In order to handle concurrent requests, it is possible to develop web server in two different ways, based on threads or events. In a threaded Web server (e.g. Apache Web server), each request is handled by a unique thread. In contrast, evented Web server has one event loop to manage concurrent multiple requests. Therefore, evented Web server avoids CPU context switching and massive execution stacks in memory. The list of terminology used in the event and thread based systems is given in Table 9.2.

Events	Threads
Event handlers	Monitors
Events accepted by a handler	Functions exported by a module
SendMessage / AwaitReply	Procedure call, or fork/join
SendReply	Return from procedure
Waiting for messages	Waiting on condition variables

Table 9.2: Terminology list that is originally available as presented in [2]

The unique benefit of the method (i.e. thread or event) can be validated by considering the load profile of a server, for example, how many threads can be spawned before performance degrades. However, quantitative and qualitative measure are influencing the decision. MOCSP can be developed based on an evented architecture because the work being done is fundamentally evented: upon receiving an input data from the client, the server processes and relays the data. In contrast, the business logic implemented by web applications is

more naturally described in a synchronous style. The callbacks required by an evented architecture become unwieldy in a complex application code.

Another consideration is memory coordination and consistency. Evented servers executing in a single event loop do not need to worry about the correctness and performance implications of maintaining consistently shared memory, but this may be a problem for threaded servers. Threaded servers therefore attempt to minimize the memory shared among threads. MOCSP gains benefits from event based model since each client session does not share among them. But fundamentally stateful servers like caches, and SIP proxy servers cannot avoid this problem.

Thread-based Web server needs to depend on the underlying platform. [2] argues that thread implementation is a major obstacle for the performance issue, rather than threading paradigm. In this case, if one thread makes blocking calls to external resources, other threads are prevented from using CPU. This situation is in favour of considering evented architecture as relatively more appealing.

Based on the above mentioned arguments, event based architecture is the right choice for the MOCSP system and I choose nodejs.org [122] for building evented servers. Therefore, it is able to get optimal performance. Note that the debate on the right server whether event based or thread based does not conclude concretely.

Node.js

Here, this thesis explains about Node.js, from its internal architecture to applications developed using a Web server, a Websocket and a multi-processor node modules. These applications are needed for the MOCSP deployment.

By design, Node.js - server side JavaScript environment- is a single threaded platform, built on V8 JavaScript engine, for easily building fast, scalable network application [122]. To allow a single thread to handle many concurrent requests, Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. This makes it an excellent fit for the MOCSP system. As of today, big industry players such as Microsoft, ebay, LinkedIn and Yahoo! have included node.js into their applications because of its scalability and performance.

The high level architecture of Node.js with all the components is shown in Figure 9.5 [123]. Node.js depends on the libev event loop library [124], using epoll or kqueue for the scalable event notification mechanism. To avoid CPU time loss usually made by waiting for an input or an output response (database, file system, web service, etc.), Node.js uses the full-featured asynchronous I/O libeio library [125].

Conceptually, Ruby's Event Machine or Python's Twisted is similar to Node.js. However, Node.js is considered as superior because node.js presents the event loop as a language construct instead of as a library. This means that Node.js does not use a blocking call to start the event-loop, for example start the event loop through blocking call EventMachine::run(). In fact, Node.js simply enters the event loop after executing the input script and makes the event loop exist when there are no more callbacks to perform. This behaviour is like browser javascript - the event loop is hidden from the user.

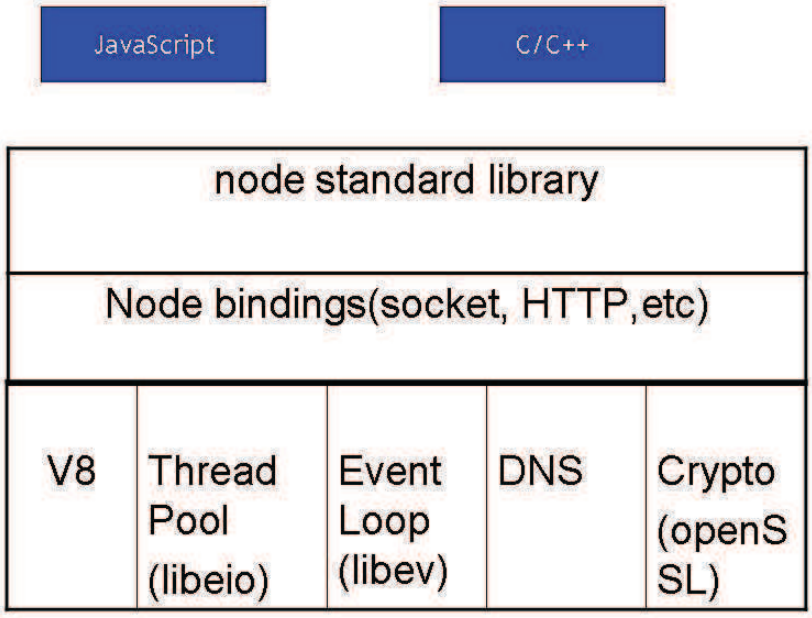


Figure 9.5: The high level architecture of Node.JS

There are three important achievements based on Node.js that are relevant to the present research work:1) HTTP server, 2) Web socket implementation and 3) multi processor concurrency.

I. HTTP server is well written in Node.js. It is an event infrastructure with a HTTP parser from [126].

II. Web Socket implementation. There are a few web socket implementations on node.js. The present researcher intends to use the websocket.io from [127] for the MOCSP deployment. The web socket protocol is standardized as RFC 6455, but websocket.io supports protocol version 13.

III. Multi processor concurrency is an important aspect in today's computing. In this case, what will happen to this single threaded Node.js? Will it scale across a multi-processor? To be scalable, a new process can be created via child_process.fork() and all these processes will be scheduled in parallel. For load balancing the incoming connections across multiple processes, cluster module is developed, available in [128].

    In parallel, Yahoo investigates multi-core HTTP server with Node.js [129], but focusing exists on scalability with computational complexity using Web worker APIs.

9.3.3 Sample Code for the MOCSP system

To validate the concepts proposed in this thesis, the last two sub-sections (Section 9.3.1 and 9.3.2) present possible directions for implementation. In this sub section, the present researcher explains the development with the help of source code. Initially, the developed source code for the server side is provided. In this case, the source code is written in JavaScript and runs in the Node.js environment. Later, code for the browser side is provided. Here it is a Hypertext Markup Language (HTML) page.

The main JavaScript file, Index.js, executed in the server is shown below. First, it is important to create a Web server and a Web socket server in order to understand the HTTP and WebSocket protocol requests from clients. Interestingly, it is possible to leverage the third party library for developing the MOCSP source code. Therefore, no effort is given to develop the Web server and Web socket from scratch. Two libraries, http and socket.io [127] (see the source code focusing on require() function), are used for the MOCSP main server code.

The developed MOCSP server is listening on port 8081, able to distinguish the HTTP and Web socket connection. Initially, callee is logged into the MOCSP server when making a request of index.html file. Similarly, caller can log into the MOCSP server by making a request of callerindex.html. These requests are severed in an asynchronous fashion. The CPU is not blocked during the file reading, thanks to Node.js development. In order to understand this remark, look at the source code starting with the fs.readfile() function.

This fs.readfile() function has two parameters, file name and callback function. The callback function performs the job for either success or failure.

Listing 9.1: Source of Index.js

```
var app = require('http').createServer(handler)
  , io = require('socket.io').listen(app)
  , fs = require('fs');

var Url = require("url");

var hat = require("hat"); // generate socketID

var count = 0;
var sockets = [];     //number of sockets.
var calleeid, callerid


app.listen(8081);

function handler (req, res) {


var pathname = Url.parse(req.url).pathname;

console.log("Path Name"+pathname);
```

```
        var filePath = '.'+pathname;
        console.log("current path"+filePath);
        if (filePath==="./indexcaller.html")
        {
              filePath = "./indexcaller.html";
                          console.log("Current Path"+filePath);
        }
         else
        {
            filePath="./index.html";
            console.log("Current Path"+filePath);
        }

  fs.readFile(filePath,
//__dirname + '/index.html',
  function (err, data) {
    if (err) {
      res.writeHead(500);
      return res.end('Error loading index.html');
    }

    res.writeHead(200);
    res.end(data);
       });
}


io.sockets.on('connection', function (socket) {

  socket.emit('connected', { msg: 'Welcome from server' });

  socket.on("getid", function() {
        var uuid = hat();
        sockets[uuid] = { id: uuid, socket: socket};

                  if(count==0)
                  {
                      calleeid=uuid;
                      console.log("calleeid is set>>" + calleeid);
                      count++;
                   } else
                   {

                     callerid=uuid;
                     console.log("callerid is set>>" + callerid);
                     count++;
                   }
                 socket.emit('getid', { id: uuid });
        console.log("Socket with id: " + uuid + " saved.");
    });
```

```
    socket.on("sendSigMsg", function(data) {
          console.log("Signaling message received.");
          console.log("Signaling message received: " + data.msg);

          console.log("Sending to Socket " + data.id)
                  console.log("Sockets length..."+ sockets.lenght);

       if (calleeid==data.id)
              //if (sockets[data.id]){
    sockets[callerid].socket.emit("recvSigMsg", { id: data.id, msgBody: data.msg });
       else if (callerid==data.id)
    sockets[calleeid].socket.emit("recvSigMsg", { id: data.id, msgBody: data.msg });

      });

    socket.on('disconnect', function(){
          console.log("User disconnected.");


             });

});
```

The most important aspect is how a Web socket connection behaves during the different phases - connection, sendSigMsg, recvSigMsg, and disconnect. In the present researcher's case, when each client gets connected, the server provides each client an unique ID. A sendSigMsg event is called when a client sends data to the server via the established Web socket connection. When data is ready in the Web socket server side to client, recvSigMsg event is called to send the data to client. These events and the related call back functions are written in an asynchronous pattern. For example, look at the code starting with socket.on("sendSigMsg", function(data)); function. Knowingly, the algorithm of this thesis demands the server to send the data to client that does not itself send those data. For instance, in Figure 5.3, MOCSP receives the oack(desc1) message from the callee and sends another oack(desc1) to the caller.

In the prototyping work, there are two different html files, Index.html and Indexcaller.html, for callee and caller. To experience the real time communication, these files should run in Ericsson made web browser that is available in [120]. The source code of index.html is provided below.

Index.html is the main interface for a caller who hosts his/her MOCSP instance. This html page performs signaling and media functionality. This means that signaling channel via Web Server and peer-to-peer connection for media should be set up between caller and callee.

Listing 9.2: Source of Index.html

```
<html>
<script src="/socket.io/socket.io.js"></script>
<script>
```

```
 var socket = io.connect('http://172.25.71.214:8081');
 var signalingId,
    Peer= null;


socket.on('connected', function (data) {

    chatHistory.value += "From server: " + data.msg + "\n";
    socket.emit('getid', {});

            });




    socket.on('getid', function(data){
        chatHistory.value += "Received id: " + data.id + "\n";
        signalingId=data.id;
        chatHistory.value+="Assigned id" + signalingId;
                                                  });

    socket.on('recvSigMsg', function(data){

        if(data.msgBody==="OPEN")
            {
               var  r=confirm("Will you ready to accept the session?")
               if(r==true)
               {
                    var msgoack="OACK";
                    socket.emit('sendSigMsg', {id: signalingId, msg: msgoack

                     Peer = new webkitPeerConnection("NONE", sendSigMsg);
                     console.log("OACK send!");

                     Peer.onaddstream= receiveVideoStream;


               }else
                  {
                   var msgnotopen="NOTOPEN";
                   socket.emit('sendSigMsg', {id: signalingId, msg: msgnotop
                   }
              }else
                  {
                     console.log("Peer processing remote message!");
                     console.log("Message body here<<<");
                     console.log(data.msgBody);

     //chatHistory.value += "msg from server: " + data.id +"interconnection
```

```
                if(Peer!=null)
                        Peer.processSignalingMessage(data.msgBody);
                 }
        });


function sendSigMsg (msg) {
        //console.log("Signaling.sendSigMsg>>>: " + msg);
        socket.emit('sendSigMsg', {id: signalingId, msg: msg});
    }



function startChat() {
   //alert("Euraka..");
   chatHistory.value += "ME: " + chatMessage.value + "\n";
   chatHistory.value+="ID checking...." + signalingId;
   socket.emit('sendSigMsg', {id: signalingId, msg: chatMessage.value});
   chatMessage.value = "";

   }

window.onload = function() {
    //content = document.getElementById('content');

    //chatHistory  = document.getElementById('chatHistory');
    //chatMessage  = document.getElementById('chatMessage');

    selfView     = document.getElementById('selfView');
    peerView     = document.getElementById('peerView');
    addButton    = document.getElementById('addButton');
    removeButton = document.getElementById('removeButton');

    removeButton.disabled = true;
}

/* ------------------------------------------------------------------------
 * Voice & video
 * ------------------------------------------------------------------------ */

var selfView;
var peerView;

var addButton;

function requestSelfVideo() {
                console.log("requestSelfVideo");
    navigator.webkitGetUserMedia('audio,video', function(stream) {
                console.log("got self stream");
```

```
        selfView.src = webkitURL.createObjectURL(stream);
        if(Peer!=null)
         {
              console.log("ready");
              Peer.addStream(stream);
         }

    });
    addButton.disabled = true;
    removeButton.disabled = false;
}

function receiveVideoStream(e) {
    console.log("onaddstream event: receiveVideoStream");
    peerView.src = webkitURL.createObjectURL(e.stream);
}

function stopSelfVideo() {
                selfView.src = 0;
                addButton.disabled = false;
                removeButton.disabled = true;
                try {
                        Peer.removeStream();
                } catch (err) { console.log(err); }
}




</script>

<body>
<div id="chat">
    <form id="chatForm" action="javascript:startChat();">
      Text chat
      <input id="chatMessage" type="text" size="40">
      <input type="submit" value="submit">
    </form>
    <textarea id="chatHistory" readonly cols="80" rows="50"></textarea>
  </div>

  <form id="videoForm">
    <div id="videos"></div>
      <table align=center>
        <tr><td>Self View</td><td>Peer View</td></tr>
        <tr height="240">
          <td width="320">

            <video id="selfView"
```

```
                width="320" height="240" autoplay controls>
            </video>

          </td>
          <td width="320">

            <video id="peerView"
                width="320" height="240" autoplay controls>
            </video>

          </td>
        </tr>
        <tr><td>
            <input id="addButton"
                type="button"
                value="My Camera"
                onclick="requestSelfVideo();">

                <input id="removeButton"
                type="button"
                value="Remote Camera"
                onclick="stopSelfVideo();">
          </td>
          <td>
          </td>
        </tr>
      </table>
  </form>


</body>
</html>
```

To explain the Index.html source, the source code is divided into two parts: HTML body part and JavaScript part. In the HTML body part, there is a new tag called video as follows.

```
<video id="selfView" width="320" height="240" autoplay controls>
</video>
```

There are two video tags representing self video view and remote peer video view. This video tag should be supported by the browser vendors such as Mozilla and Google Chrome. There are two buttons with onclick functions – one is to show self view and the other is to stop the display of the remote view. Other tags are understandable to HTML programmers.

A more important logic is embedded in the form of JavaScript. This logic has three parts: Web socket connection, signaling message processing (including sending and receiving) and media connection.

To enable Web Socket connections at the browser, web socket library from [127] is included into the source code as shown below:

```
<script src="/socket.io/socket.io.js"></script>
```

Once web socket connection is established, the three events - connected, getid, and recvSigMsg are triggered. Generally, connected and getid events are triggered once, but recvSigMsg is triggered many times within a single cycle. The callback function for the recvSigMsg event processes all incoming messages (typically signaling messages) from the Web server.

According to call flow in the MOCSP system (see Figure 5.3), callee receives the OPEN message once caller clicks on the hyperlink provided by callee. If callee accepts the new session, callee will send the OACK message. When callee sends the OACK message, Peer object is created with the new webkitPeerConnection ("NONE" , sendSigMsg) function . sendSigMsg is a call back function to process the SDP messages from the other peer. Otherwise, callee will send the NOTOPEN message. Socket.emit() function is used to send the signaling messages from the browser to Web server.

For the media display and transport, three functions are important. They are requestSelfVideo(), receiveVideoStream() and stopSelfVideo(). When the requestSelfVideo() function is called, browser will start the media capture and display at own video display. At the same time, video data will be linked to the Peer object in order to send them to the remote peer via the Peer.Stream(stream) function. Two functions - navigator.webkitGetUserMedia() and webkitURL.createObjectURL (stream) should be supported by the browser vendors.

To display the remote peer video, the receiveVideoStream(e) function will be triggered. Once triggered, browser receives the media and displays at the remote peer display.

The source code of indexcaller.html is almost similar to Index.html except for a few modifications, reflecting a unique behaviour at the caller side.

The prototype needs to rely on the Web browser, modified version of the WebKit library on Ubuntu system [121] and does not consider the NAT/Firewall issue because it is performed within the local area network. At this point, SDP messages are passed to the Peer objects without any modification at the web server. Codecs that are available in Ubuntu are used in this experimentation. For voice, Speex is used. And, Theora is used for video.

9.3.4 Summary

The prototype work is being carried out now based on Node.js and Ericsson developed Webkit browser. In addition, this thesis keeps a close watch on the recent development of WebRTC at W3C and IETF. However, there are two different ways to describe the media endpoints. The next section presents them in detail.

9.4 SDP Offer/Answer vs Media Description

In communication services, description of media endpoint and negotiation on parameters between endpoints is the core part of the communication services. Importantly, negotiation generally needs to agree on codec between endpoints before those endpoints send/receive media. This aspect is vehemently discussed by the SIP community. However, there is another way to present the media endpoint as presented in [62], called media based negotiation. The early implementation of WebRTC at the Web Browser side considers only SIP negotiation

mechanism. This thesis intends to argue for the implementation of both these features, so end users (Web App developer or MOCSP app developer) gain benefits from their appropriate usages.

This section has three subsections, namely Session Description Protocol (SDP), offer/answer mechanism and Media based Negotiation. SDP is a format, accepted by the SIP community, for describing media endpoints that are involved in a session. The offer/answer mechanism is used for the negotiation between media endpoints. Alternatively, Media based Negotiation can be used for the negotiation.

9.4.1 Session Description Protocol

Caller and callee in communication services negotiate parameters during the session initiation and mid-call. This aspect globally gives emphasis on the need for interoperability. Therefore, the SIP community intends to specify a common format for describing the multimedia communication sessions. This format is known as Session Description Protocol (SDP). The current specification of SDP is available as RFC 4566 [15] that obsoletes the initial version of SDP published in April,1998.

SDP is used in the negotiation especially in the offer/answer mechanism that is included in the following subsection. Before explaining the mechanism, important information regarding the SDP is provided here. SDP has a set of properties and parameters for each media types and format and is designed to be extensible to support the new media types and format. For example, VP8 video is a new media type and IETF now considers proposing a new RTP Payload Format for VP8 Video as discussed in [130].

Each end point in a session with the help of SDP is represented by a series of fields, one per line. The form of each field is as follows.

<character>=<value>

Where <character> is a single case-significant character and value is the structured text whose format depends upon attribute type.

An SDP message has three main sections, detailing the session, timing, and media descriptions. The important information are either the type of media (video, audio, etc.), the transport protocol (RTP/UDP/IP, H.320, etc.), or the format of the media (H.261 video, MPEG video, etc.). Each message may contain multiple timing and media descriptions. All names are unique within the session, time, or media. =* indicates values as optional and each field must appear in the order shown below.

```
Session description (name and purpose)
       v=  (protocol version)
       o=  (originator and session identifier)
       s=  (session name)
       i=* (session information)
       u=* (URI of description)
       e=* (email address)
       p=* (phone number)
       c=* (connection information—not required if included in all media)
```

```
   b=* (zero or more bandwidth information lines)
   One or more time descriptions ("t=" and "r=" lines; see below)
   z=* (time zone adjustments)
   k=* (encryption key)
   a=* (zero or more session attribute lines)
   Zero or more media descriptions


Time description
   t=  (time the session is active)
   r=* (zero or more repeat times)



Media description (information needed to receive those media (addresse
ports, formats, etc.)
 ), if present
   m=  (media name and transport address)
   i=* (media title)
   c=* (connection information—optional if included at
        session level)
   b=* (zero or more bandwidth information lines)
   k=* (encryption key)
   a=* (zero or more media attribute lines)
```

SDP can be used along with different applications (e.g. multicast application), but this thesis is interested to consider it along with SIP. All the available parameters are presented here, but more details can be found in [15]. However, to be concrete, a sample message shown in [15] is included below to explain how it is used in the implementation.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

IETF proposed changes in order to accommodate VP8 video media as follows:

```
m=video 49170 RTP/AVPF 98
a=rtpmap:98 VP8/90000
a=fmtp:98 version=0
```

The address and port, to which the data is sent, is decided by the media and transport protocol. Media attribute (a=* parameter) is also important, having "session-level" attributes, "media-level" attributes, or both. Session-level attributes convey information relevant globally and media-level attributes are information specific to individual media.

Two different forms of attribute fields are as follows:

a=<attribute>

a=<attribute>:<value>

For example, a=recvonly, a=rtpmap:96 L8/8000, or a=orient:landscape.

The following four attributes are important to understand the difference between SDP offer/answer and media description:

1 a=recvonly: This information can be either session-level or media-level, but it informs that it behaves in the receive mode.

2 a=sendrecv: The endpoint behaves in the send and receive mode. "sendrecv" SHOULD be assumed as the default for communication sessions unless they ("sendonly", "recvonly", "inactive") are mentioned explicitly.

3 a=send-only: If the endpoint decides to behave like in send only mode, send only attribute is included in to the SDP message. It is logically equal where two media descriptions may be used – one sendonly and one recvonly.

4 a=inactive: This attribute can be either session level or media level, but puts the users on hold- no media is transported.

Another information in the SDP message is media description: m=<media> <port> <proto> <fmt> ...

Since each session has a number of media sessions, each media session is indicated with an "m" field.

<media> is the media type – "audio","video", "text", "application", and "message", although this list can be extended in the future, this work only considers "audio", "video", and "text".

<port> is the transport port to which the media stream is sent. <proto> is the transport protocol. The following transport protocols are defined in the standard.

1. UDP: denotes an unspecified protocol running over UDP.

2. RTP/AVP: denotes RTP used under the RTP Profile for Audio and Video Conferences with Minimal Control running over UDP.

3. RTP/SAVP: denotes the Secure Real-time Transport Protocol running over UDP.

<fmt> is a media format description.

9.4.2 Offer/Answer Mechanism

RFC 3264 - The Session Description Protocol (SDP) offer/answer presents a mechanism for two endpoints to reach a common view of a multimedia session between participants [131]. This simple mechanism proposes that one participant offers the other participant a description of the desired session from their perspective, and the other participant answers with the desired session from their perspective. SIP uses the offer/answer mechanism that can also be used with other protocols.

In a simple offer/answer model based on SDP, one participant (either caller or callee), called the offerer, in the session generates an SDP message that constitutes the offer - the set of media streams and codecs the offerer wishes to use, along with the IP addresses and ports the offerer would like to use to receive the media. In return, the other participant, called the answerer, generates an answer, which is an SDP message that responds to the offer provided by the offerer. The answer has a matching media stream for each stream in the offer, indicating whether the stream is accepted or not, along with the codecs that will be used and the IP addresses and ports that the answerer wants to use to receive media. After the initial offer/answer exchange, negotiations can take place in order to update the session.

The offer and answer are sent via SIP methods such as SIP INVITE, UPDATE, 200 OK, etc. More importantly, an SDP message used in the offer/answer model must contain exactly one session description and the offer/answer exchange is atomic. The session returns to the state prior to the offer if the answer is rejected.

Direction attribute is also important because it is linked with SDP and negotiations. For example, if the offerer wants to only send media on a stream to its peer, it must mark the stream as sendonly with the "a=sendonly" attribute. In this case, the port number and IP address in the answer indicate where the answerer would like to receive the media stream. Similarly, three other direction attributes ("a=inactive" , "a=sendrecv", "a=recvonly") are used to send the preferences of the participants.

SDP and offer/answer mechanism consider the port number and IP address of RTP endpoints. RTP is end-to-end real time media transport protocol. Relying on UDP, RTP adds timestamps (for synchronization), sequence numbers (for packet loss and reordering detection) and payload format headers in each packet, in addition to encoded media data in the payload.

There is another protocol called RTP Control Protocol (RTCP)[14]. RTCP monitors RTP transmission statistics and quality of service (QoS) and helps synchronization of multiple streams. RTCP works together with RTP. The port number assignment at the sender and receiver side is straightforward. RTP uses even port number at both sides and the associated RTCP communication chooses the next higher odd port number. In addition, usage of RTCP does not depend on the direction attributes. In this analysis, the need of RTCP is overlooked, hence, the impact of RTCP on SDP and offer/answer mechanism is not provided. These details are available in RFC 3264 [131].

Besides the port number and IP address, media formats for each media stream are important information that is conveyed in SDP. All media descriptions should include "a=rtpmap" mappings from RTP payload types to encodings for each type of RTP streams. Essentially, the preference order is used to

organize the different media formats - the highest goes first. Receiver of the offer should select the format with the highest preference that is acceptable to the receiver.

The offerer is prepared to receive media immediately after the offerer has sent the offer (assume that offer has recvonly streams). In the case of RTP, offerer may receive media before the answer arrives. In the case of sendonly streams in the offer, media will not be sent until the answer (consists of needed address and port) arrives.

The caller and callee need a complete view of the session that is made by the agreement on parameters between them. After the initial offer and answer mechanism, it is possible to modify characteristics of the session (nearly all aspects) at any point by either participant. In this case, new offer may be similar to previous offer or different. If there is a new offer with the new SDP, some constraints are placed on construction. Modifying the session can be either adding a new session, removing a media stream, modifying a media stream, or putting media stream on hold. Here the description of each part that is responsible for modifying a session is provided.

1. Adding a Media Stream: There are two ways to inform the answerer that the offerer will add new media streams. The first one is to add new media descriptions below the existing one. And the second is to disable the media stream port to zero. In response, answerer can reject or accept by replacing an appropriately structured media description in the answer. Similarly, answerer can also add a media stream.

2. Removing a Media Stream: General rule to remove existing media streams is to create a new SDP with the port numbers of those streams set to zero. The all attributes present previously may be omitted in the new SDP that may list just a single media format. In response, answerer marks the port of the media stream set to zero. Similarly, answerer follows the same procedure like offerer (omit all attributes and a single media format).

   Removing a media stream implies that media is no longer sent and will be discarded if received. In fact, the resources associated with these media streams are released.

3. Modifying a Media Stream: It is possible to change all parameters of the session. Therefore, details of the modification can be classified into four groups as follows:

   I. Modifying Port, Address, or Transport: To modify the port number of a media stream, offerer creates a new media description, with the port number in the m line different from the corresponding stream in the previous SDP. In this case, only port number is changed and the others remain unchanged. On the implementation, offerer is prepared to receive media on the old and new port until answer reaches. This kind of arrangement is made to avoid loss of media during this transition. It is critical that answerer should send the media to the new port if answerer accepts the request. If rejected, offerer should be ready to receive it on the old port. Similarly, answerer is also able to change its port number.

Like port changes, IP address and transport changes also follow the same procedure, except that connection line is updated for the IP address modification.

II. Changing the Set of Media Formats: A set of media formats with the "m=" line is changed (new format introduced or the format removed) during the session. Like in the port changes, as soon as answerer sends its answer, the answerer starts sending the media using any formats in the offer that were also present in the answer and should use the most preferred format in the offer that was also listed in the answer. In contrast to changes in the set of media formats, changes in ports may require changes in resource reservation or rekeying of security protocols. The drawback is that the answerer can not change the media format on their own unless the offerer starts to do so.

III. Changing Media Types: As opposed to adding a new stream, the media type (audio, video, etc.) for a stream may be changed. It means that the same logical data is being conveyed, but just in a different media format. Similar to the other changes (e.g. Port), the answerer should start sending the new media type and formats as soon as he/she receives the offer. This means that the offerer will be prepared to receive media with both the old and new types until the answer is received, and media with the new type is received and reaches the top of the playout buffer.

IV. Changing Attributes: This allows to change any other attributes in a media description using an offer/answer method.

4. Putting a Unicast Media Stream on Hold: It allows one party to request the other party not to send media stream. This job can be explained easily with the help of directionality attributes. If the current stream is sendrecv, it is placed on hold by marking it as sendonly. Similarly, if the current stream is recvonly, it is placed on hold by marking it inactive.

Demonstration with Offer/Answer Exchange

In this sub-section 9.4.2, the negotiation mechanism is explained with the basic elements. To make understandable, an example is provided here. Assume that Alice, caller, provides initially an offer to Bob, callee. In the offer, Alice wishes to communicate using a bidirectional audio stream and two bidirectional video streams, using H.261 (payload type 31) and MPEG (payload type 32). The offer looks like this:

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=
c=IN IP4 host.anywhere.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVP 31
a=rtpmap:31 H261/90000
m=video 53000 RTP/AVP 32
```

```
a=rtpmap:32 MPV/90000
```

Important details of the above offer are connection IP address of host.anywhere.com, port for receiving audio 49170, and port for receiving video at 51372 and 53000.

Bob wishes to receive or send the second video stream (i.e. m=video 53000 RTP/AVP 32),not in the first video. Therefore, Bob sends an answer like below:

```
v=0
o=bob 2890844730 2890844730 IN IP4 host.example.com
s=
c=IN IP4 host.example.com
t=0 0
m=audio 49920 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 0 RTP/AVP 31
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

More important details are the connection IP address of Bob, host.example.com, audio port of Bob, 49920 and video port of Bob, 53000. Essentially, both sides should use the same codec for sending and receiving the media. This feature is technically not necessary, but it places more constrains on the implementation. Based on this reasoning, the following sub-section includes one alternative for the offer/answer mechanism.

9.4.3 Media based Negotiation

Chapter 4 includes a descriptive model of IP Media service, detailing different aspects such as abstract architecture, and protocol description. The main idea is on new signaling protocol based on different media (e.g. video or audio). The complete detail of this signaling protocol is provided in Chapter 4. However, the significance of the protocol is provided here in order to support within the PeerConnection class (See Section 9.3.1) in the Web Browser. Currently, PeerConnection implementation intends to support the SDP based negotiation.

In media based negotiation, media channels are opened, closed, and modified only by their end points. Based on this basic idea, media based negotiation differs in three points: 1) basic synchronization, 2) codec choice, and 3) tunnel within a signaling path.

Basic synchronization roots from HTTP that is based on the request and response transaction model. It means that a client sends a request and then only server sends back a response. HTTP design influences the design of SIP. This synchronization is useful when a transport protocol is unreliable, for example UDP. If the SIP messages travel over UDP, the transactional model is used to recover the lost messages.

At the core of SIP, the INVITE/OK/ACK transaction is used to open and modify the media channel. Based on INVITE from an endpoint, the other endpoint accepts the request with the success message, OK method. In response, the initiator acknowledges with the ACK method. This INVITE/OK/ACK transaction carries offer and answer messages between endpoints. Once an INVITE message is initiated by one endpoint, until it finishes with the ACK message, the other endpoint can not initiate an INVITE method. This means that

an overlap is not possible in the same signaling path. If the overlap takes place, which is called a race condition, both INVITE methods will fail. Then, both should wait for a randomly chosen time and retry.

This bottleneck (developing a protocol from the basic INVITE/OK/ACK transaction) demands to design an idempotent protocol from the transaction nature. The media based negotiation can send new describe or select signals at any time (See Section 4.2, 4.3 and [62]) once the media channel is opened. This media based negotiation gains the benefit of reliability from TCP. As mentioned earlier, modification of media channel in one direction will be independent of modification of other direction media flow. In addition, the overall protocol is not constrained in terms of programming complexity and performance, because, describe and select signals have update information without changing the fundamental state compared to the INVITE/OK/ACK transaction. Therefore, it can be described as idempotent.

Codec choice in the media based negotiation is different from SIP. In a SIP, an INVITE method has a set of codecs that the offerer can handle. The answerer chooses one codec. The SIP based negotiation takes a toll on performance. The media based negotiation decouples codec choice in the two directions. In each direction, one end point sends unilaterally a set of possible choices in a descriptor and the other end chooses one in a selector. It is guaranteed that once an end point has received a selector, it knows exactly which codec it is expected to interpret. In this unilateral nature, both endpoints send descriptors and selectors concurrently.

The last difference is media bundling. In the media based negotiation, every tunnel within a signaling path controls one media channel, to be completely independent of every other tunnel. This is not the case in SIP in which SIP signal for controlling media considers all media channels. This media bundling in SIP makes it more difficult to program intermediary entities. Additionally, media bundling increases the probability of race conditions, causing significant performance penalty. This means that a transaction to control a video channel contends with a transaction to control an audio channel on the same signaling path.

This thesis intends to defend that this aspect is also included into the PeerConnection class in the Web Browsers, because media based negotiation is significantly better than the SIP. In any manner, the SIP is a dominant protocol, widely used in operator and service provider networks. Therefore, these two - offer and answer negotiation mechanism based on SDP and media based negotiation - should be considered to be included into the Web browsers. Web application developers choose the right method based on needs. Another aspect could be Interoperability. It explains what will happen if both approaches are separately supported in two endpoints? This kind of situation is not considered at this point. However, it is possible to propose a gateway solution in order to address this problem.

9.5 Conclusion

This chapter considers two different use cases in order to validate the new system - MOCSP. The solution for Call Transfer with consultation hold does not gain the same result or is slightly complex than SIP solution. Essentially, MOCSP

does not naturally consider the interactions between the network boxes.

Moreover, IP mobility will not be used for comparing the difference between the SIP and newly proposed MOCSP system, because session continuity can be achieved by sending new session description information. If TCP migrate complaint Web server and Web browser are deployed in their respective places, session continuity can be enabled at any stage. However, TCP migrate deployment is not mandatory for session continuity. The idea used in SIP can be exploited for IP mobility and session continuity in the MOCSP system.

In fact, MOCSP is a good candidate for the communication system, supporting the existing services. However, MOCSP requires a mechanism for interactions between network boxes. This will enable to propose new services based on MOCSP.

Additionally, in order to perform the prototype work, the recent developments at standard bodies and open source contributions are discussed. Finally, this thesis views two different negotiation mechanisms - offer and answer negotiation mechanism based on SDP and media based negotiation, because both should be included in to the Web browsers in order to support real time communication services natively.

Part III

Conclusion

Chapter 10

Conclusion

Shifting from the paradigm of a single creator to many creators with communication services is an important goal. This paradigm shift has taken place in the Web platform by releasing the creative power of the people. However, this goal is not addressed with communication services by the research community.

The goal of empowering the users with communication services has many technical challenges. I present three aspects:

I. providing openness and flexibility of the communication infrastructure to end users.

II. reducing the complexity behind the development of network-based session services.

III. improving the scalability of the signaling layer in the communication infrastructure.

The main contribution of this thesis is a proposal of the MOCSP concept and system for inter-personal communications. In MOCSP, users (callees) get control over the signaling protocol and can design services based on their needs. Callees create communication hyperlinks that are shared with callers for establishing the sessions. Clients for the callers are downloaded when callers click on the given communication hyperlinks. Therefore, it is very easy to change the semantic of the signaling protocol in order to meet the changing needs. This freedom on communication services (i.e. without waiting for any functions from any body) allows the users to innovate with communication services. The MOCSP concept supports diversity while managing the complexity of the communication services.

Two different services are considered to explain the importance of network-based, session-based services and how system development is simplified for them. The two use-cases represent two different requirements: service logic executed during the call establishment and mid-call, and cooperation of network call control and endpoints.

The solutions for the network-based, session-based services depend on a single entity called network box. The main logic is placed in the network box based on the sessions. This approach is validated by two different use cases : user mobility and partial session transfer and retrieval. In the user mobility use case that

Properties	IMS (pre-defined)	P2P SIP	MOCSP
Scalability	Well (need proper capacity planning)	Well	Well
Complexity	SIP routing configuration	Arrangement of the overlay.	Only Web hosting of the MOCSP instance
Needed computing resources	Higher than MOCSP	Allocating resources for message overhead (proportion to O(log(N)))	Resources for hosting network boxes and resources used in DNS
Session setup latency	Higher than MOCSP	Olog(N)	Average three hops (network box and two hops in DNS)

Table 10.1: Comparison of different signaling architectures based on the proposed scalability evaluation framework. Here, N is the number of super-nodes in a P2P overlay

needs network-initiated session transfer, the network box performs as a forking proxy, a B2BUA and a registration server. This means that the network box behaves as a media orchestrator. This model can be used for another use case called missed call situation. Compared with the SIP based solution, MOCSP based user mobility solution can save the computing processing power at least by removing six SIP message parsing and processing overhead per session in a SIP proxy.

For PSTR, this single media orchestrator - the network box - is responsible for partial session transfer and retrieval across multiple devices. This architecture facilitates network-initiated and user-initiated partial session transfer and retrieval at the caller and callee sides. It means that any number of transfer and retrieval can be performed by a network box and (or) users within a single session. In addition, the complexity for developing this solution is reduced in two ways: 1) via a single media orchestrator at the network box and by 2) a signaling protocol design. I separate the signaling protocol into media control and auxiliary protocol based on the software engineering approach - 'separation of concern'. Since the MOCSP solution is based on the Web, an end user (callee/caller) can transfer and retrieve the partial session by the drag-and-drop of widgets in their Web browser. This widget-based approach will also increase the user's experience.

This thesis analyzes the scalability of a signaling architecture for interpersonal communication services based on four criteria: the level of scalability, level of complexity, the amount of computing resources needed and the session setup latency. Table 10.1 lists the comparison of three different signaling architectures based on four aspects. This preliminary analysis shows clearly that the MOCSP based architecture can outperform the other two existing architectures, IMS and P2P SIP. By taking advantage of parallelism, the MOCSP based architecture scales without adding complexity. More importantly, the number of calls does not influence the session setup time, relying on the right capacity planning. In a nutshell, this approach can manage any number of calls (supporting state-ful network based session services (e.g. user mobility)) while

increasing resources without compromising the average response time.

Chapter 11

Future Work

This chapter discusses future work into four sections: 1) prototype and further validation, 2) methodology for scalability validation, 3) contribution to standard bodies and 4) possible extensions.

11.1 Prototype and Further Validation

The conceptual framework for answering the listed research problems has been established in this thesis with analytical results. However, it is possible to extend by prototyping the solutions listed in the contribution part. This section presents all the relevant prototyping work except scalability validation to be done in future. The scalability validation is given separately in Section 11.2

Prototyping the MOCSP system as proposed in Chapter 5 is a major step. This has two impacts on validation. The first is to validate the communication services between two browsers where signaling protocol, media components and asynchronous communication between Web browser and Web server work together without inducing problems. The second impact is to encourage users to use the MOCSP system. Over time, it is possible to check how users benefit from this model by analyzing it with questions like: 1) How many new session-based services have been developed based on their needs? 2) Has openness and flexibility been useful to end users? This will be very effective validation.

The second validation will be prototyping the user mobility solution. This prototyping helps to verify the integration of context information, forking proxy and network-initiated session transfer. Moreover, latency during the soft handoff will be measured for user acceptance.

The third prototyping is to validate three aspects with PSTR: 1) Validation of the PSTR at the callee and caller sides regardless of the originator - network-initiated and user-initiated. 2) Validation by measuring latency during the transfer and retrieval in different settings (e.g. Web browsers behind firewalls/proxies). 3) Validation to formalize the behaviour of the network box, caller and callee boxes when they send messages (e.g. IsSplit() and Split()) with the same intention. All these three validations can be performed on the developed prototype. In parallel, an evaluation of protocol design (separation of media control protocol and auxiliary protocol) can be performed with the help of formal verification - model checking languages (e.g. Promela language,

Alloy [132]) and model checkers (e.g. SPIN [133], Alloy Analyzer). It is possible to have an automated method for the verification of temporal safety properties of concurrent and distributed systems in model checking. This model checking evaluation indicates possible consistencies/inconsistencies with constraints and goals of the proposed design.

11.2 Scalability Validation

This section details a methodology to validate the scalability aspect of MOCSP against the other communication systems based on the established framework. The proposed scalability evaluation framework consists of four parameters: scalability limit, complexity level, needed computing resources, and session setup latency.

Scalability depends on dimensioning of the system in advance with an anticipated performance. This turns out that anticipation of expected performance should be satisfied. In the MOCSP case, dimensioning of a system has to take into account two factors: performance of Web server (including hardware architecture and software systems) and frequency of call requests.

In MOCSP, each MOCSP instance is unique and does not depend on other MOCSP instances. However, allocating the number of MOCSP instances to a particular computing system (hardware, OS and Web server) depends on the anticipated performance. It means that if a particular computing system has x MOCSP instances, x calls should be made concurrently. This core aspect (allocating the number of MOCSP instances and the number of simultaneous calls, and session setup latency) should be validated through an experimentation.

[134] evaluates the performance of SIP proxy in different configurations (such as state-less vs. stateful proxying, using TCP rather than UDP, or including MD5-based authentication). Throughput (i.e. calls per second) and response time are measured against the different configurations in IBM BladeCenter with Red Hat Enterprise Linux and Gigabit Ethernet connectivity.

To provide a similar environment for both experimentations (IMS/SIP or MOCSP), this thesis will choose the same hardware (e.g. two Intel Clovertown quad-core chip) and OS (prefer Ubuntu) for both experimentations. Initially, the experimentation indicated in [134] will be carried out with two configurations (stateful UDP No auth and stateful TCP No Auth). In these two configurations, proxy behaves as stateful and no authentication support, but it will support two different transport protocols (UDP and TCP).

Later, on the same hardware and OS, an experiment based on MOCSP will be performed. To be informative, I provide the relevant lab setup and configurations both at the server side and client side. Then, there is a discussion on what parameters should be measured. Based on these parameters, the question how analysis can be performed in order to make a concrete conclusion is explained.

11.2.1 Experimentation Setup

- Figure 11.1 shows the experimentation setup that will be used to evaluate the scalability.

- Server side hardware and OS:

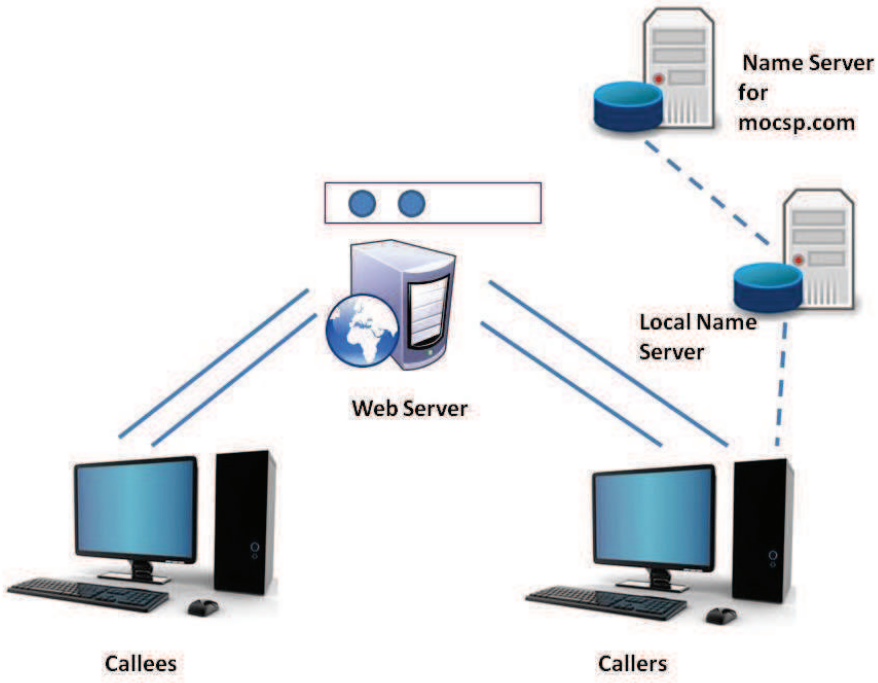


Figure 11.1: Lab experimentation setup. Each thick line shows the TCP connection to be used for sending (as well as receiving) the signaling messages. Each connection indicates each client connection to the MOCSP instance.

- Hardware and OS configuration for server: each hardware and OS performs differently. This thesis prefers to use two Intel Clovertown quad-core chips, and each core is with 2.2 GHz and exploits instruction-level parallelism techniques. This hardware setup is used in [83] to measure the throughput (calls per second) on the UDP protocol stack and multi-process (MP) programming model.
- This experimentation will use the Ubuntu operating system with a Linux 2.6.18 kernel on Intel Clovertown.
- It is possible to modify the Linux kernel default sysctl settings [135] and TCP options. Initially, default values will be used. If needed, relevant changes will be made as performed in [136].
- A target Web server will be developed using Node.js. Each MOCSP instance is uniquely deployed in Node.js based Web server and is named as siva.mocsp.com, paul.mocsp.com, noel.mocsp.com, etc. See Section 9.3.2 for more detail of Web server development.
- In the multi-core platform, there are two different configurations possible for Node.js. First, a single instance of node (a single thread) will be running in the two Intel Clovertown quad-core chips. Another arrangement is to add the cluster module [128] into Node.js in order to take advantage of multi-core systems. The cluster module launches a cluster of node processes to handle the load (number of

requests). The two different setups mentioned above are considered in this experimentation.

  - Regarding the domain name setting, this thesis should register for the mocsp.com domain and deploy the name server for the mocsp.com domain. This name server responds to all DNS queries from clients.
  - The virtual DNS resolution is activated in the Web server. It means that a single physical machine hosts many MOCSP instances. Requests to each MOCSP instance are forwarded based on host information given in the HTTP header.
  - To obtain OS statistics and functional profiling results, oprofile [137] is used in the server side

- Client Side:

  - This thesis will develop a new load generator called MOCSPp that is similar to SIPp [138]. The load generator is deployed in both sides – One is at the callee side and the other one is at the caller side. The caller side load generator produces (and receives) messages as shown in Figure 5.3. Likewise, the callee side load generator produces (and receives) as shown in Figure 5.3. This load generator creates many client instances in a single machine. Therefore, it is easy to make the experiment. MOCSPp will initially support the basic session establishment (as shown in Figure 5.3) and will only support one transport protocol (TCP).
  - Two load generators (one for callee and the other for caller) will be deployed in two different machines running on Ubuntu OS.
  - The message flow is the same as shown in Figure 5.3. Here the basic session setup is considered. In terms of features, this session setup is equivalent to INVITE-200-OK in SIP. The effect arising from registration (indicated as a REGISTER message in the call flow in Figure 5.3) is not considered.

- DNS arrangement and network configuration

  - In order to reflect a real situation, two DNS servers are deployed in the experimentation setup. The caller sends a DNS query to the local name server (shown in Figure 11.1) for the DNS name resolution. Once it received a response from the name server for the mocsp.com domain, the local name server responds to callers. Similarly, each callee also performs the DNS operation in order to reach the MOCSP web server.
  - All the servers are deployed within the experimentation network and connected by an Ethernet switch.

- Measurements

  - Based on the implementation setup, two parameters, number of calls per second (throughput) and session setup time, are measured during the experimentation. Additionally, when the number of calls per second keeps increasing, what will happen to the session setup latency?

The throughput is measured when CPU is reaching 90 percent of its level.

- Throughput is referred to the number of completed calls per second.
- Session setup latency is defined as time between when the caller clicks on the communication hyperlink (CLICK as shown in Figure 5.3) and when a select(sel2) message is received. This delay is perceived by the user who is initiating a call. There are some less interesting parameters such as call duration and termination (i.e., close).
- The impact of DNS will be analyzed for the session setup latency and scalability.
- For each metric (throughput and latency), 5 runs are performed to find the value. Each run lasts for 120 seconds after a 5 second warm-up time. Finally, the cumulative distribution function (CDF) of response times for various load levels will be drawn to illustrate how response time varies with load, particularly at 95th and higher percentiles.
- In this experimentation, a threshold value for throughput will be fixed when session setup latency starts increasing dramatically. Therefore, the anticipated performance will be satisfied.

- Analysis:
  - The first comparison is between the IMS and MOCSP to find out that which setup supports (peak throughput) a higher number of calls per second?.
  - Based on the results (number of MOCSP instances deployed in a single Web server), it is possible to extrapolate the needed computing resources when the number of MOCSP instances is given. This way, it is easy to find the number of Web servers (including hardware) to be deployed that is equivalent to the given number of MOCSP instances divided by proven number of MOCSP instances in that hardware and software setup.
  - The final conclusion on the scalability aspect will be made by considering and analyzing the two different experimentations.
  - In fact, performance can be increased when some possible changes are made in OS, Web server and transport protocols. For example, applications are executed directly in a hardware box without using an operating system in bare computing [82]. Therefore, it is possible to eliminate overhead in the OS. There are other possible changes in OS as shown in [136]. These changes are to modify the Linux kernel default sysctl settings [7] and TCP options. Similar to [134], SIP proxy server is benchmarked in a multi-core platform [83]. The relevant experimentations based on the above mentioned different configurations will be made in the future.

11.3 Contribution to Standard Bodies

This thesis can feed its contributions into two standard bodies: 1) Real-Time communication in the Web Browser platform at IETF and W3C and 2) signaling and media transfer mechanism among networked devices in homes, offices, and elsewhere by UPnP.

From the concept, architecture and use cases, I argue that MOCSP promotes Web and communication services convergence. This paradigm adds communication related resources (i.e. user session identified by URI) into a remarkable information space of the Web that grows across languages, cultures, and media [5]. Recently, IETF has created one working group called Real-Time Communication in WEB (RTC-Web). The main desire is to standardize the basis for voice/video communication so that multimedia session can be established between any two compatible browsers. In the mean time, at W3C , the Web Real-Time Communications Working Group, part of the Ubiquitous Web Applications Activity, is to define client-side APIs to enable Real-Time Communications in Web browsers.

Even though these activities are in the early stage, this thesis provides a list of activities that is in the scope of MOCSP requirements. At IETF, RTC-Web defines the multimedia capabilities for web-browsers that require necessary protocols such as RTP, STUN, ICE, and an understanding of SDP (RFC 4566). Besides, the much debated topic is whether signaling protocol should be defined as part of RTC-Web or not. MOCSP defends that designing a proper signaling protocol is the role of developer (in the present research, each end user). Hence, this thesis argues against defining the signaling protocol as part of RTC-Web.

At last, MOCSP implementation can provide some requirements or can benefit from the established standards. The vision of IETF and W3C is to enable innovation on top of the basic components. MOCSP will be a pioneer based on this vision and will be complemented by IETF and W3C activities.

UPnP Forum standardizes network devices in a home and office environment. It establishes standard Device Control Protocols (DCPs) across multi-vendors. In fact, the present researcher proposes a naming scheme and the protocol stack for a media device (media source and sink) for supporting the PSTR in Chapter 7. For ubiquitous usage, it is important to standardize the naming scheme and the protocol stack at the UPnP Forum. This activity is planned to be performed in future.

11.4 Possible Extensions

A few works are to be performed in order to obtain a complete communication system. However, I identify three activities that are essential extensions for the MOCSP system.

First of all, a Web server receives different amounts of calls at a time and each call has different holding time. This call rate variation brings about many challenges that need to be solved in order to use computing

resources effectively. Since the call rate varies, there is a higher possibility that computing resources are under-utilized. The two parameters - scaling increment and scaling efficiency in [72] - are the two important properties of an underlying computing infrastructure. This means that these two parameters are not associated with communication systems especially in the MOCSP system, but provided by the underlying computing infrastructure.

To address the under-utilized problem, this thesis proposes to exploit the elasticity and cloud computing concept. Initially, the idea is to estimate the right size of computing resources/network boxes in a single Web server that can sustain a pre-defined number of calls per second (i.e. call rate). Then, based on the variation of call rate, excessive computing resources (i.e. unused computing resources) will be given to resource-hungry-cloud applications. For achieving this goal, there are two candidate solutions using hypervisor and container based operating system [139].

Hypervisor provides high isolation whereas container based operating system supports high efficiency. However, solutions mentioned in the previous paragraphs do not answer the question of scaling increment and scaling efficiency. Essentially, this proposed work should be performed with dynamic resource allocation across virtual machines or processes. This extension is primarily important for the MOCSP system, in order to solve the under-utilization problem. The work on the understanding of overhead and benefits of introducing virtualization is planned to be performed in the future.

Next, MOCSP will help the (Web-) developer users to build their own applications. It is important to know how to reach all kinds of end users who can develop their own communication services easily. One technique might be defining the protocol agnostic primitives. Another aspect is to improve the ability of end-user programmers who develop error free software. Hence, this work will need a further exploration of End user software engineering/End user programming [140]. This research may provide tools and techniques for end users to become programmers.

Finally, defining a security framework for the MOCSP system is an essential activity. In the existing systems, e.g. IMS, identification of a caller and a callee is not at the same level as in the MOCSP. This means that the identification of a callee is important and identification of a caller is not important based on the communication hyperlink concept. This situation may change as callee wishes to know who the caller is? The existing security mechanisms for the Web, for example OpenID [141] and OAUTH [142] can be extended to support the MOCSP system. Moreover, security problems at the infrastructure level - at the Web browser and at the Web server should be considered in parallel. For an instance, the security threat with the WebSockets is widely discussed by the research community [143]. The possible solutions for this kind of threat will effect the current implementation of the MOCSP system. The other possible security threats may be associated with DNS rebinding and cross-origin resource binding. This security work will be performed in the future.

Bibliography

[1] "Webrtc 1.0: Real-time communication between browsers." http://dev.w3.org/2011/webrtc/editor/webrtc.html.

[2] R. von Behren, J. Condit, and E. Brewer, "Why events are a bad idea (for high-concurrency servers)," in *Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9*, (Berkeley, CA, USA), pp. 4–4, USENIX Association, 2003.

[3] T. O'Reilly and J. Battelle, ""web squared: Web 2.0 five years on." http://www.web2summit.com/web2009/public/schedule/detail/10194.

[4] R. Guha, "Toward the intelligent web systems," in *Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on*, pp. 459 –463, july 2009.

[5] "Architecture of the world wide web." http://www.w3.org/TR/2004/REC-webarch-20041215/.

[6] Alcatel-Lucent., "Immersive communications." http://www.alcatel-lucent.com/immersive-communications/.

[7] 3GPP, "Ip multimedia subsystem (ims);stage 2 (release 10)," in *TS 23.228 V10.3.1 (2011-01)*.

[8] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Sip: Session initiation protocol." IETF RFC 3261, June 2002.

[9] P. SIP, "http://www.p2psip.org/,"

[10] S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan, and J. Hildebrand, "Xep-0166: Jingle." http://xmpp.org/extensions/xep-0166.html.

[11] W. Chou, L. Li, and F. Liu, "Wip: Web service initiation protocol for multimedia and voice communication over ip," in *Web Services, 2006. ICWS '06. International Conference on*, pp. 515 –522, 2006.

[12] M. J. Arif, S. Karunasekera, and S. Kulkarni, "Sovoip: middleware for universal voip connectivity," in *Proceedings of the 2007 ACM/IFIP/USENIX international conference on Middleware companion*, MC '07, (New York, NY, USA), pp. 23:1–23:6, ACM, 2007.

[13] M. Boussard, P. Jabaud, O. Le Berre, F. Poussiere, and P. Labrogere, "Communication hyperlinks: Call me my way," in *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, pp. 1 –5, 2009.

[14] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "Rtp: A transport protocol for real-time applications." IETF RFC 3550.

[15] M. Handley, V. Jacobson, and C. Perkins, "Sdp: Session description protocol." IETF RFC 4566, July 2006.

[16] "Programmableweb." http://www.programmableweb.com/telephony, 2011.

[17] "Parlay x 3.0 specifications." http://docbox.etsi.org//TISPAN/Open/OSA/ParlayX30.html

[18] J. Rosenberg, J. Peterson, H. Schulzrinne, and G. Camarillo, "Best current practices for third party call control (3pcc) in the session initiation protocol (sip),." IETF RFC 3725, April 2004.

[19] R. M. Arlein, D. R. Dams, R. B. Hull, J. Letourneau, and K. S. Namjoshi, "Telco meets the web: Programming shared-experience services," *Bell Labs Technical Journal*, vol. 14, no. 3, pp. 167–185, 2009.

[20] E. Cheung and P. Zave, "Principles, systems and applications of ip telecommunications. services and security for next generation networks," ch. Generalized Third-Party Call Control in SIP Networks, pp. 45–68, Berlin, Heidelberg: Springer-Verlag, 2008.

[21] H. Jiang, A. Iyengar, E. Nahum, W. Segmuller, A. Tantawi, and C. Wright, "Load balancing for sip server clusters," in *INFOCOM 2009, IEEE*, pp. 2286 –2294, april 2009.

[22] V. Hilt and I. Widjaja, "Controlling overload in networks of sip servers," in *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, pp. 83 –93, oct. 2008.

[23] J. Rosenberg, "The session initiation protocol (sip) update method." IETF RFC 3311, September 2002.

[24] J. Rosenberg and H. Schulzrinne, "Reliability of provisional responses in the session initiation protocol (sip)." IETF RFC 3262, June 2002.

[25] A. B. Roach, "Session initiation protocol (sip)-specific event notification." IETF RFC 3265, June 2002.

[26] A. Niemi, "Session initiation protocol (sip) extension for event state publication." IETF RFC 3903, October 2002.

[27] R. Sparks, "The session initiation protocol (sip) refer method." IETF RFC 3515, April 2003.

[28] J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, "Session initiation protocol (sip) extension for instant messaging." IETF RFC 3428.

[29] C. Holmberg, E. Burger, and H. Kaplan, "Session initiation protocol (sip) info method and package framework." IETF RFC 6086, January 2011.

[30] 3GPP, "Overview of 3gpp release 5 - summary of all release 5 features," tech. rep., 3GPP - ETSI Mobile Competence Centre, 2003.

[31] 3GPP, "Ts 24.228: Signaling flows for the ip multimedia call control based on sip and sdp." http://www.3gpp.org/ftp/tsg_sa/wg3_security/TSGS3_19_London/Docs/PDF/S3-010340.pdf, July 2001.

[32] 3GPP, "Ts 24.229 ip multimedia call control protocol based on session initiation protocol (sip) and session description protocol (sdp); stage 3 (release 9)." http://www.quintillion.co.jp/3GPP/Specs/24229-940.pdf, June 2010.

[33]

[34] N. Banerjee and K. Dasgupta, "Telecom mashups: enabling web 2.0 for telecom services," in *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, ICUIMC '08, (New York, NY, USA), pp. 146–150, ACM, 2008.

[35] "Jsr 289: Sip servlet v1.1." http://www.jcp.org/en/jsr/detail?id=289.

[36] D. Lozano, L. Galindo, and L. Garcia, "Wims 2.0: Converging ims and web 2.0. designing rest apis for the exposure of session-based ims capabilities," in *Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST '08. The Second International Conference on*, pp. 18 –24, sept. 2008.

[37] J. Rosenberg, J. Lennox, and H. Schulzrinne, "Programming internet telephony services," *Network, IEEE*, vol. 13, pp. 42 –49, may/jun 1999.

[38] L. Burgy, C. Consel, F. Latry, J. Lawall, N. Palix, and L. Reveillere, "Language technology for internet-telephony service creation," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 4, pp. 1795 –1800, june 2006.

[39] X. Wu and H. Schulzrinne, "Programmable end system services using sip," in *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 2, pp. 789 – 793 vol.2, may 2003.

[40] W. Jouve, N. Palix, C. Consel, and P. Kadionik, "Principles, systems and applications of ip telecommunications. services and security for next generation networks," ch. A SIP-Based Programming Framework for Advanced Telephony Applications, pp. 1–20, Berlin, Heidelberg: Springer-Verlag, 2008.

[41] F. Latry, J. Mercadal, and C. Consel, "Staging telephony service creation: a language approach," in *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications*, IPTComm '07, (New York, NY, USA), pp. 99–110, ACM, 2007.

[42] P. Zave, "Principles, systems and applications of ip telecommunications. services and security for next generation networks," ch. Understanding SIP through Model-Checking, pp. 256–279, Berlin, Heidelberg: Springer-Verlag, 2008.

[43] J. Peterson, C. Jennings, and S. R., "Change process for the session initiation protocol (sip) and the real-time applications and infrastructure area." IETF RFC 5727, March 2010.

[44] P. Zave, "Modularity in distributed feature composition," in *Software Requirements and Design: The Work of*, 2009.

[45] M. Jackson and P. Zave, "Distributed feature composition: a virtual architecture for telecommunications services," *Software Engineering, IEEE Transactions on*, vol. 24, pp. 831 –847, oct 1998.

[46] M. Kolberg and E. H. Magill, "Managing feature interactions between distributed sip call control services," *Comput. Netw.*, vol. 51, pp. 536–557, February 2007.

[47] X. Wu, J. Buford, K. Dhara, V. Krishnaswamy, and M. Kolberg, "Feature interactions between internet services and telecommunication services," in *Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications*, IPTComm '09, (New York, NY, USA), pp. 10:1–10:12, ACM, 2009.

[48] P. Zave, "Mid-call, multi-party, and multi-device telecommunication features and their interactions," tech. rep., Principles, Systems and Applications of IP Telecommunications, August 2011.

[49] H. Schulzrinne, "Personal mobility for multimedia services in the internet," in *Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Services*, (London, UK), pp. 143–161, Springer-Verlag, 1996.

[50] J. A. Tuijn and D. Bijwaard, "Spanning a multimedia session across multiple devices," vol. 12, (New York, NY, USA), pp. 179–193, John Wiley & Sons, Inc., February 2008.

[51] T.-P. Wang and K. Chiu, "An efficient scheme for supporting personal mobility in sip-based voip services," *IEICE Transactions*, vol. 89-B, no. 10, pp. 2706–2714, 2006.

[52] T.-P. Wang and H.-Y. Lee, "User location management for personal mobility in sip-based voip services," in *Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on*, pp. 910 –914, aug. 2008.

[53] P.-h. Chang and T.-P. Wang, "Supporting personal mobility with integrated rfid in voip systems," in *Proceedings of the 2009 International Conference on New Trends in Information and Service Science*, (Washington, DC, USA), pp. 1353–1359, IEEE Computer Society, 2009.

[54] B. Moltchanov, M. Knappmeyer, C. Licciardi, and N. Baker, "Context-aware content sharing and casting," in *12th ICIN*, (Bordeaux, France), October 2008.

[55] M. Barachi, A. Kadiwal, R. Glitho, F. Khendek, and R. Dssouli, "The design and implementation of architectural components for the integration of the ip multimedia subsystem and wireless sensor networks," *Communications Magazine, IEEE*, vol. 48, pp. 42 –50, april 2010.

[56] C. Jacob, H. Pfeffer, D. Linner, S. Steglich, L. Yan, and M. Qifeng, "Automatic routing of semantic sip messages in ims," in *Internet Multimedia Services Architecture and Applications, 2008. IMSAA 2008. 2nd International Conference on*, pp. 1 –6, dec. 2008.

[57] T. Tang, Z. Mi, and R. Peng, "Adaptive service provisioning through context-aware sip proxy," *Networking and Services, International conference on*, vol. 0, pp. 277–281, 2008.

[58] J. Rosenberg, "Obtaining and using globally routable user agent uris (gruus) in the session initiation protocol (sip)." IETF RFC 5627, October 2009.

[59] V. A. Balasubramaniyan, A. Acharya, M. Ahamad, M. Srivatsa, I. Dacosta, and C. P. Wright, "Servartuka: Dynamic distribution of state to improve sip server scalability," *Distributed Computing Systems, International Conference on*, vol. 0, pp. 562–572, 2008.

[60] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer, "Session initiation protocol (sip) session mobility." IETF RFC 5631, October 2009.

[61] S. Shanmugalingam, N. Crespi, and P. Labrogere, "My own communication service provider," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on*, pp. 260 –266, oct. 2010.

[62] P. Zave and E. Cheung, "Compositional control of ip media," *Proceedings of the 2006 ACM CoNEXT conference*, pp. 18:1–18:12, 2006.

[63] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer, "The virtual device: Expanding wireless communication services through service discovery and session mobility," in *In Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking, and Communications (Montreal, Aug. 22–24). IEEE Communications Society*, pp. 73–81, 2005.

[64] M. Hasebe, J. Koshiko, Y. Suzuki, T. Yoshikawa, and P. Kyzivat, "Example call flows of race conditions in the session initiation protocol (sip)." IETF RFC 5407, December 2008.

[65] M.-X. Chen, C.-J. Peng, and R.-H. Hwang, "Ssip: Split a sip session over multiple devices," vol. 29, (Amsterdam, The Netherlands, The Netherlands), pp. 531–545, Elsevier Science Publishers B. V., July 2007.

[66] M.-X. Chen and F.-J. Wang, "Session integration service over multiple devices," *International Journal of Communication Systems*, vol. 23, no. 5, pp. 673–690, 2010.

[67] R. Sparks, "The session initiation protocol (sip) referred-by mechanism." IETF RFC 3892, September 2004.

[68] "Upnp." www.upnp.org.

[69] A. Vilei, G. Convertino, and F. Crudo, "A new upnp architecture for distributed video voice over ip," in *Proceedings of the 5th international conference on Mobile and ubiquitous multimedia*, MUM '06, (New York, NY, USA), ACM, 2006.

[70] J. Dean, "Challenges in building large-scale information retrieval systems: invited talk," in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, (New York, NY, USA), pp. 1–1, ACM, 2009.

[71] A. B. Bondi, "Characteristics of scalability and their impact on performance," in *Proceedings of the 2nd international workshop on Software and performance*, WOSP '00, (New York, NY, USA), pp. 195–203, ACM, 2000.

[72] J. Y. Kim, G. W. Bond, E. Cheung, T. M. Smith, and H. Schulzrinne, "An evaluation framework for highly available and scalable sip server clusters," in *Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications*, IPTcomm '11, (New York, NY, USA), pp. 1:1–1:10, ACM, 2011.

[73] K. Ono and H. Schulzrinne, "Principles, systems and applications of ip telecommunications. services and security for next generation networks," ch. One Server Per City: Using TCP for Very Large SIP Servers, pp. 133–148, Berlin, Heidelberg: Springer-Verlag, 2008.

[74] "Fowler/noll/vo (fnv) hash." http://isthe.com/chongo/tech/comp/fnv/.

[75] "Moving average." http://en.wikipedia.org/wiki/Moving_average.

[76] G. Kambourakis, D. Geneiatakis, T. Dagiuklas, C. Lambrinoudakis, and S. Gritzalis, "Towards effective sip load balancing: the snocer approach," in *3rd Annual VoIP Security Workshop*, (Berlin, Germany), June 2006.

[77] Y. Hong, C. Huang, and J. Yan, "Modeling and simulation of sip tandem server with finite buffer," vol. 21, (New York, NY, USA), pp. 11:1–11:27, ACM, February 2011.

[78] C. Shen and H. Schulzrinne, "On tcp-based sip server overload control," in *Principles, Systems and Applications of IP Telecommunications*, IPTComm '10, (New York, NY, USA), pp. 71–83, ACM, 2010.

[79] A. Dutta, C. Makaya, S. Das, D. Chee, J. Lin, S. Komorita, T. Chiba, H. Yokot, and H. Schulzrinne, "Self organizing ip multimedia subsystem," in *Proceedings of the 3rd IEEE international conference on Internet multimedia services architecture and applications*, IMSAA'09, (Piscataway, NJ, USA), pp. 118–123, IEEE Press, 2009.

[80] J. Zou, W. Xue, Z. Liang, Y. Zhao, B. Yang, and L. Shao, "Sip parsing offload: Design and performance," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pp. 2774 –2779, nov. 2007.

[81] J. Janak, "Sip proxy server effectiveness," master thesis, Department of Computer Science, Czech Technical University, Prague, Czech, May 2003.

[82] A. Alexander, A. Wijesinha, and R. Karne, "A study of bare pc sip server performance," *Systems and Networks Communications (ICSNC), 2010 Fifth International Conference on*, pp. 392 –397, aug. 2010.

[83] Z. Jia, Z. Liang, and Y. Dai, "Scalability evaluation and optimization of multi-core sip proxy server," *Parallel Processing, 2008. ICPP '08. 37th International Conference on*, pp. 43 –50, sept. 2008.

[84] J. Maenpaa and G. Camarillo, "Analysis of delays in a peer-to-peer session initiation protocol overlay network," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pp. 1 –6, jan. 2010.

[85] C.-M. Cheng, S.-L. Tsao, and J.-C. Chou, "Unstructured peer-to-peer session initiation protocol for mobile environment," in *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pp. 1 –5, sept. 2007.

[86] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, (New York, NY, USA), pp. 149–160, ACM, 2001.

[87] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, (New York, NY, USA), pp. 161–172, ACM, 2001.

[88] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, Nov. 2001.

[89] K. Singh and H. Schulzrinn, "Peer-to-peer internet telephony using sip," Tech. Rep. CUCS-044-04, Department of Computer Science, Columbia University, New York, NY, Oct 2004.

[90] "Cio update: Post-mortem on the skype outage." http://blogs.skype.com/en/2010/12/cio_update.html.

[91] P. Saint-Andre, "Streaming xml with jabber/xmpp," vol. 9, pp. 82 – 89, sept.-oct. 2005.

[92] M. Podgoreanu, P. Chitescu, and P. Saint-Andre, "Xep-0251: Jingle session transfer," 10 2009.

[93] W3C, "http://www.w3.org/2002/ws/."

[94] "Webrtc." http://www.webrtc.org/reference.

[95] S. Arbanowski, S. van der Meer, S. Steglich, and R. Popescu-Zeletin, "The human communication space: Towards i-centric communications," *Personal Ubiquitous Comput.*, vol. 5, pp. 34–37, January 2001.

[96] S. Arbanowski, P. Ballon, K. David, O. Droegehorn, H. Eertink, W. Kellerer, H. van Kranenburg, K. Raatikainen, and R. Popescu-Zeletin, "I-centric communications: personalization, ambient awareness, and adaptability for future mobile services," *Communications Magazine, IEEE*, vol. 42, pp. 63 – 69, sept. 2004.

[97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – http/1.1." RFC 2616, June 1999.

[98] I. Fette and A. Melnikov, "The web socket protocol." http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-11, June 2011.

[99] "The web socket api." http://dev.w3.org/html5/websockets/.

[100] "Why web 2.0 will end your privacy." http://www.bit-tech.net/columns/2006/06/03/web_2_privacy/.

[101] "A privacy manifesto for the web 2.0 era." http://gigaom.com/2008/01/08/a-privacy-manifesto-for-the-web-20-era/.

[102] R. Copeland, "Network intelligence - facilitate operators win in mobile broadband era," in *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, pp. 1 –6, oct. 2009.

[103] G. W. Bond, T. M. Smith, E. Cheung, and P. Zave, "Specification and evaluation of transparent behavior for sip back-to-back user agents," in *Principles, Systems and Applications of IP Telecommunications*, IPTComm '10, (New York, NY, USA), pp. 48–58, ACM, 2010.

[104] E. Cheung and T. Smith, "Getting sip endpoints and network call control to work well together," *Principles, Systems and Applications of IP Telecommunications*, August 2011.

[105] "Context casting (c-cast)." http://www.ict-ccast.eu/.

[106] C. A. L. Boris Moltchano, Michael Knappmeye and N. Baker, "Context-aware content sharing and casting," in *In: ICIN 2008,*, (Bordeaux, France), 2008.

[107] A. Al-Hezmi, T. Magedanz, J. A Jaen Pallares, and C. A Riede, "Evolving the convergence of telecommunication and tv services over ngn," in *International Journal of Digital Multimedia Broadcasting*, p. 11, 2008.

[108] P. Mockapetris, "Domain names - concepts and facilities." IETF RFC 1034, November 1987.

[109] "Dns resource records." http://www.zytrax.com/books/dns/ch8/.

[110] P. Mockapetris, "Domain names—implementation and specification." IETF RFC 1035, Nov 1987.

[111] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "Dns performance and the effectiveness of caching," *IEEE/ACM Trans. Netw.*, vol. 10, pp. 589–603, Oct. 2002.

[112] "How to disable client-side dns caching in windows xp and windows server 2003." http://support.microsoft.com/kb/318803.

[113] R. Cox, A. Muthitacharoen, and R. Morris, "Serving dns using a peer-to-peer lookup service," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, (London, UK, UK), pp. 155–165, Springer-Verlag, 2002.

[114] "Specweb2005." http://www.spec.org/web2005/.

[115] C. MacCarthaigh, "Scaling httpd 2.x to 50,000 concurrent downloads." In ApacheCOn EU„ June 2006.

[116] RedHat, "Specweb2005 benchmark using red hat enterprise linux 5.2 on a hp proliant dl580 g5, v1.1," February 2009.

[117] S. Foley and W. MacAdams, "Trust management of xmpp federation," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pp. 1192 –1195, may 2011.

[118] R. Sparks, A. Johnston, and D. Petrie, "Session initiation protocol (sip) call control - transfer." IETF RFC5589, June 2009.

[119] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, (New York, NY, USA), pp. 155–166, ACM, 2000.

[120] "Ericsson labs." https://labs.ericsson.com/apis/web-real-time-communication/.

[121] "Update on webkit contribution." https://labs.ericsson.com/developer-community/blog/update-webkit-contributions.

[122] "Nodejs." http://nodejs.org/.

[123] http://blog.zenika.com/index.php?post/2011/04/10/NodeJS.

[124] "Libev." http://software.schmorp.de/pkg/libev.html.

[125] "Libeio." http://software.schmorp.de/pkg/libeio.html.

[126] "Http paraser." https://github.com/joyent/http-parser/tree/master.

[127] "Websocket.io." https://github.com/LearnBoost/websocket.io.

[128] "Cluster." http://learnboost.github.com/cluster/.

[129] "Multi-core http server with nodejs." http://developer.yahoo.com/blogs/ydn/posts/2010/07/multicore_http_server_with_nodejs/.

[130] P. Westin, H. Lundin, M. Glover, J. Uberti, and F. Galligan, "Proposal for the ietf on "rtp payload format for vp8 video" draft-westin-payload-vp8-02." http://tools.ietf.org/html/draft-westin-payload-vp8-02.

[131] J. Rosenberg and H. Schulzrinne, "An offer/answer model with the session description protocol (sdp)." IETF RFC 3264, June 2002.

[132] "Alloy modelling language." http://alloy.mit.edu/community/.

[133] "Spin." http://spinroot.com/spin/whatispin.html.

[134] E. M. Nahum, J. Tracey, and C. P. Wright, "Evaluating sip server performance," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, pp. 349–350, June 2007.

[135] "Linux kernel. linux ip sysctl documentation.." http://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt.

[136] B. Veal and A. Foong, "Performance scalability of a multi-core web server," in *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, ANCS '07, (New York, NY, USA), pp. 57–66, ACM, 2007.

[137] "Oprofile." http://oprofile.sourceforge.net/news/.

[138] "Sipp." http://sipp.sourceforge.net/.

[139] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors," in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, (New York, NY, USA), pp. 275–287, ACM, 2007.

[140] M. Burnett, C. Cook, and G. Rothermel, "End-user software engineering," *Commun. ACM*, vol. 47, pp. 53–58, September 2004.

[141] "Openid." http://openid.net/.

[142] "Openauth." http://oauth.net/.

[143] L.-S. Huang, E. Chen, A. Barth, E. Rescorla, and C. Jackson, "Talking to yourself for fun and profit," *In H. J. Wang, editor, Proceedings of W2SP*, May 2011.

Appendix A

French Summary

A.1 Motivation

Le Web a changé la façon dont les gens exécutent les activités quotidiennes et a évolué rapidement (par exemple le Web 2.0, HTML5) depuis l'invention du Web en 1989, par Sir Tim Berners Lee. Une des raisons de ce succès est que "Le Web adopte des technologies relativement simples avec une évolutivité suffisante, l'efficacité et l'utilité "; deux autres facteurs qui contribuent également à la réussite du Web sont les suivants: son ouverture et sa flexibilité. L'ensemble de ces facteurs servent comme un moyen clé pour la haute participation des usagers à la création d'applications. En conséquence, le Web ouvre la voie à des applications innovantes dans l'ère d'Internet où les utilisateurs finaux deviennent producteurs et consommateurs de contenus et de services.

Différents services de communication de la distribution du courrier de lettres à la téléphonie, voix / vidéo sur IP (Internet Protocol), e-mail, forums de discussion Internet, des conférences vidéo / audio, et immersive communication ont évolué au fil du temps. Comme les autres services de communication, la voix / vidéo sur IP est un besoin essentiel des personnes. Par exemple, autour du premier trimestre de 2011, le nombre d'utilisateurs de Skype connectés a atteint 30 millions - 30 millions de personnes sont en ligne en même temps. En outre, il existe un besoin urgent - la fourniture de services de communication pour les personnes dans la planète, car les abonnements de téléphonie mobile et le nombre de personnes connectées à l'Internet chez eux ne cessent d'augmenter de jours en jours. Environ deux milliards de personnes se sont connectées à l'Internet dans le premier trimestre de 2011. Comme nous sommes de plus en plus connectés à Internet, nous faisons tous partie d'un monde en réseau.

La Voix / vidéo sur système de communication IP est basé sur la célèbre architecture en couches constituée d'une couche de signalisation et une couche de support. Le protocole de signalisation est utilisé pour créer, modifier et terminer des sessions médias entre les participants. Les systèmes de communication existants peuvent être construits soit par IP Mul-

timedia Subsystem (IMS) basée sur Session Initiation Protocol (SIP) , Peer-to-Peer (P2P) SIP, XMPP / Jingle (comme GTalk) , Web Service Initiation Protocol (WIP) based Service oriented communication (SOC), Service oriented VoIP (SOVoIP) , or Web personal communication systems.

En règle générale, les systèmes de communication adoptent une approche bottom-up pour fournir des services de communication de masse pour les utilisateurs sans beaucoup de personnalisation, servant ainsi d'intermédiaires entre appelés et appelants. L'appelé est le récepteur des appels et l'appelant est à l'origine des appels.

Toutefois, deux systèmes de communication largement utilisés sont IMS et SIP P2P, où SIP est utilisé comme protocole de signalisation. L'architecture IMS divise la couche de signalisation en deux couches: une couche de contrôle de service et une couche de service. La couche de contrôle de service a beaucoup d'entités fonctionnelles entre l'appelant et l'appelé. Sauf les agents utilisateurs, toutes les autres entités fonctionnelles sont contrôlées par l'opérateur. Cela signifie que l'opérateur doit déployer les nœuds fonctionnels requis. Les nœuds effectuent en majorité des fonctions de rendez-vous pour des messages de signalisation d'une session. De même, les services tels que la présence sont déployés dans la couche de service, en s'appuyant sur la couche de contrôle de service commun. Les objectifs importants de spécifications IMS sont de réduire CAPEX et OPEX du fournisseur, de favoriser la convergence telecom et Internet et de raccourcir le temps de mise sur le marché de nouveaux services. Plus important encore, la pensée de l'innovation de l'utilisateur final est sous-estimée dans la conception initiale des systèmes de communication IMS.

Beaucoup de recherches ont été consacrées au développement de nœuds fonctionnels et des protocoles (SIP, Real-Time Transport Protocol (RTP), Session Description Protocol (SDP)) pour l'architecture IMS. Un travail prolongé telles que la fourniture de la fonction clic-to-dial pour les utilisateurs Web est effectuée dans le but d'attirer les contributions des utilisateurs finaux. Cependant, ces approches ne guident pas l'innovation de l'utilisateur final. Par exemple, selon " ProgrammableWeb ", il ya environ 200 mashups qui sont des services de communication liés à 5100 mashups. Un mashups est un service en combinant les services existants. Le principal facteur derrière ce peu de mashups de services de communication, c'est que ces systèmes de communication ne pouvaient pas répondre à divers besoins des utilisateurs finaux. Cela déclenche une autre question que l'infrastructure contrôlée du prestataire de services devrait changer de middleware pour les chercheurs et les geeks à une réalité au jour le jour pour des milliards de personnes.

A.2 Enoncé du problème

La question de recherche globale de cette thèse tente de répondre est la suivante:

"Comment faire pour permettre aux utilisateurs de contribuer / développer différents services de communications basée sur les mashups? Ou Quand les utilisateurs pourront bénéficier de la liberté (ou sans la permission) d'innover dans des services de communication? "

Cette thèse considère que les utilisateurs ne sont pas naïfs sur les informations et les technologies de la communication. Afin d'être en mesure de répondre à la problématique de la recherche, j'ai défini un ensemble de questions de recherche importantes qui traitent le problème en détail.

I. L'infrastructure de communication existante, IMS, ne possède pas assez d'ouverture et de flexibilité pour attirer des tiers ou des utilisateurs finaux pour développer des applications basées sur les services de communication.

   Je définis l'ouverture et la flexibilité comme suit: 1) l'ouverture signifie que des utilisateurs développent leurs applications ou mettre en œuvre leurs nouvelles idées (pour les nouveaux services) avec moins de fiabilité. 2) la flexibilité signifie que les applications développées peuvent être facilement modifiées.

   Les principales caractéristiques des services de communication sont la session / le contrôle des appels, le routage entre l'appelant et l'appelé, et la gestion d'état. Tout en définissant le protocole de contrôler les fonctionnalités, le fournisseur de services (SP) d'abord offre un contrôle d'appel Application Programming Interface (API) afin d' autoriser aux développeurs tiers.

   Les fonctionnalités liées aux communication de services qui sont maintenant définies par SP sont données aux utilisateurs finaux par les API. Ces fonctionnalités / API sont sous la forme de SIP API ou API HTTP ParlayX défini. L'interface IMS Service Control (ISC) est SIP API, standardisé pour les développeurs ou tiers. Par conséquent, des services basés sur Third Party Call Control (3PCC) peuvent être mis au point. Cependant, le développement des services est lourd pour les développeurs en raison de la complexité du protocole SIP et l'architecture. Beaucoup de recherches ont été menées pour cacher la complexité de l'infrastructure de l'opérateur contrôlé en fournissant des abstractions ou des API. Un haut niveau d'abstraction est défini par le groupe Parlay (par exemple Réseau IMS) pour les développeurs Web. Par exemple, les applications click-to-cadran est basée sur 3PCC. Basé sur une application clic-to-dial, un utilisateur dans le site Web peut lancer et terminer une session. Cela signifie que ces API fournissent des fonctionnalités limitées. Ces approches ne rencontrent pas des besoins de l'utilisateur final (c.-à-pas beaucoup différents services composés sur la base de cette API).

   Essentiellement, la plupart de ces API (par exemple les types de données de session) sont au niveau de la recherche et ne sont pas déployés dans le réseau pour une utilisation utilisateur final parce qu'ils sont l'injection de la complexité dans la plate-forme. Le niveau de complexité est également associée à la granularité de API. En

outre, le troisième les utilisateurs du parti ou à la fin dépendent fortement de la prestataire de service particulier.

En outre, le modèle de conception du système de communication est semblable au one-size-fits-all modèle où chaque session doit se comporter de façon similaire. Le noyau les aspects du protocole de signalisation sont définies par les normes. Les changements (Pour de nouvelles fonctionnalités) dans le protocole de contrôle doivent être analysés avec plus de soins. Jusqu'à changements ajouter dans les lieux potentiels, ces nouvelles fonctionnalités ne seront pas activées. Par exemple, si un client n'a pas la fonction et établit une session, cette nouvelle fonctionnalité ne sera pas activée. En règle générale, les changements pour les fonctions prennent beaucoup de temps à sa dérouler pendant la normalisation et le déploiement.

En fournissant l'ouverture et la flexibilité suffisante pour les utilisateurs finaux, ces utilisateurs obtiendront le contrôle sur la couche de signalisation. Cette liberté encourage les utilisateurs finaux à composer de nouveaux services basés sur leurs besoins.

II. Basé sur le réseau, la session des services sont très difficiles à développer dans le l'infrastructure existante. Le réseau basé, basée sur la session d'interfonctionnement des services avec une session ou sont exécutées dans une session. Par exemple dans l'appel Renvoi sur occupation, si l'appelé est occupé, l'appel sera transmis à la prédéfinie destination par l'appelé. Ce service est exécuté uniquement lors des établissements d'appel. Beaucoup d'efforts ont été consacrés à composer ces services. Ce genre de problème est dénommé métrages d'interactions. Cependant, il ya un cas particulier où le service doit être exécuté au cours des établissements d'appels et/ou au milieu des appels. Une telle situation n'est pas considérée par la communauté des chercheurs.

   En outre, l'élaboration d'une solution basée sur le protocole SIP et l'architecture IMS apporte une plus grande complexité et nécessite une connaissance profonde des interdépendance des noeuds fonctionnels dans le réseau IMS. En outre, basés sur le réseau liée à la session services nécessitent la coopération de contrôle des appels du réseau et les points de terminaison, par exemple, le transfert partiel session et de recherche (PSTR). En PSTR, la logique de service devrait être déployée dans des points de l'Application Server SIP (AS) et de fin de support network initiés et initiée par l'utilisateur PSTR. Dans le SIP ou les spécifications IMS, ce type dont la coopération est négligée. Au lieu de cela, le fait de placer la logique indépendamment dans le contrôle des appels du réseau et les points de terminaison est encouragée. Ce déploiement n'est pas efficace dans l'architecture IMS, car indépendamment déployé la logique des services dans les points intermédiaires et de fin ne peut pas être facilement coordonnée.

   En outre, cette PSTR crée une complexité de déploiement. L'utilisateur et le réseau des approches latérales ont besoin de déployer le mécanisme 3PCC, respectivement. Cela signifie que pour PSTR au côté de l'utilisateur, deux instances de 3PCC doivent être exécutées. Pour la PSTR par le côté réseau, une instance de 3PCC doit être exécutés

dans le SIP AS. Par conséquent, cette séparation de déploiement ajoute à la complexité dans le développement d'une solution.

Au-delà, Eric et al mettant l'accent sur la composition de multiples basées sur SIP 3PCC contrôleurs. Si plusieurs contrôleurs basés sur SIP sont dans une voie de signalisation, comment le système se comporter? Leur recherche est dans le sens d'avoir de nombreux contrôleurs d'appels dans la voie de signalisation.Cet aspect n'est pas considéré comme dans cette thèse.

PSTR permet aux utilisateurs mobiles de transférer et de récupérer en partie à des médias des dispositifs qui sont situés dans leur voisinage. De même, les services qui peuvent être déployé dans le réseau enrichissent l'expérience utilisateur.

III. L'infrastructure de signalisation tels que IMS et P2PSIP devient complexe quand un taux d'appel élevé est connu.

La couche de signalisation dans IMS a de nombreuses entités fonctionnelles telles que Proxy-Call Session Control Function (P-CSCF), service (S-) CSCF et AS. Deux aspects énumérés ci-dessous devraient être pris en compte lorsqu'un taux d'appel élevé est connu. Premièrement, la capacité d'un nœud qui mettent en œuvre les fonctions, PC-SCF, S-CSCF, etc est finie, par conséquent, il devient saturé à un moment donné. Deuxièmement, le protocole de signalisation est un protocole basé sur l'état contrairement à Hyper Text Transfer Protocol (HTTP). Ces états sont conservés dans ces entités fonctionnelles. Par conséquent, les messages entre l'appelant et l'appelé doivent être compatibles. Cela signifie dans une session tous les messages doivent suivre le même chemin. Les deux aspects ci-dessus énumérés entravent une solution efficace pour l'évolutivité du problème. Une solution possible consiste à accueillir un équilibreur de charge au sein de la couche de signalisation comme une entité fonctionnelle supplémentaire.

H.Jiang et al proposent un équilibreur de charge plus fine qui a une connaissance de la SIP tout comme différentes opérations dans le coût SIP et le traitement des transactions différentes. Cette solution permet de plus en plus de l'évolutivité du service de la couche de contrôle en transmettant le trafic de procurations sur la base de la capacité. Toutefois, cette équilibreur de charge tombe de manière récursive sur le problème de évolutivité. En outre, aucun travail n'a été signalé sur la façon de résoudre le problème de l'évolutivité quand un grand nombre d'utilisateurs sont connectés à des services (Logique de service, par exemple dans SIP AS) dans la couche de service.

Lorsque vous utilisez load-balancers, le prestataire de services devrait être conscients de la congestion. Cela signifie que chaque nœud ne doit pas être surchargé pendant le taux d'appel élevé. En cas de surcharge dans un ou plusieurs nœuds physiques, il devrait y avoir un effondrement de débit dans le réseau IMS / SIP. En outre, la surcharge du réseau ne sera pas récupérer facilement. Contrôle de surcharge dans un réseau des serveurs SIP est un sujet largement débattu dans la recherche et la norme des organisations (par exemple l'IETF). Les solutions existantes pour éviter la congestion sont

inefficaces et complexes, des ressources informatiques exigeants pour la congestion algorithme d'évitement.

Ce niveau plus élevé de complexité qui est injecté par l'équilibreur de charge et le mécanisme d'évitement de la congestion exige des ressources informatiques supplémentaires et ajoute de la latence à la configuration de la session. Cette utilisation inefficace des ressources de l'informatique sape également les efforts écologique d'informatique.

A.3 La solution proposée

Basé sur la revue de la littérature, il est établi que la couche de signalisation middleware est partagée par tous les utilisateurs et est commandée par un opérateur. Toutefois, des solutions pour les problèmes de recherche de cette thèse sont complexes et demandent des coûts élevés pour garantir l'interopérabilité (dans l'IMS). Par conséquent, cette thèse se propose de prendre une nouvelle approche pour résoudre les problèmes de recherche à travers une nouvelle conception. La nouvelle approche est nommée " My Own Communication Service Provider" (MOCSP) et est traduit par un fournisseur personnelle de communication. La partie principale de la solution réside dans le concept MOCSP et la spécification du système MOCSP dans la plate-forme Web.

Tout d'abord, on présente les concepts de base et le système de MOCSP. Ensuite, il est démontré comment établir une session de communication dans le système MOCSP. Après, une analyse sur la l'ouverture de la flexibilité du système est exposé. Le système MOCSP permet le développement de services orientés réseau ou orientés session. Effectivement, une solution pour les deux cas d'utilisation différents - la mobilité d'utilisateur et le transfert de la session partielle et récupération (PSTR) est présentée. Enfin, l'aspect de montée en charge pour la couche signalisation est défini et discuté sur la base du système MOCSP.

A.3.1 MOCSP: concept et système

MOCSP est un concept qui permet aux utilisateurs finaux de créer (et disposer) par eux-mêmes leurs plates-formes pour les services de communication. Ce concept vise à fournir tous les éléments de base des services de communication pour les utilisateurs finaux. Pour les services de communication, les utilisateurs finaux ont besoin d'un contrôle sur les couches de signalisation et de média. Cette thèse présente les mécanismes qui permettent aux utilisateurs finaux de contrôler complètement la couche de signalisation afin de concevoir les services innovants dont ils ont besoin.

Je propose une approche top-down (de haut en bas) basée sur le Web pour la réalisation du concept MOCSP. L'architecture MOCSP suit le principe de la séparation de la couche de signalisation et celle des média afin de proposer une architecture flexible. Sur la base de ce principe, je propose une architecture de haut niveau du système MOCSP comme le montre la figure 5.2, constitué de plan de contrôle et le plan média.

Le plan de contrôle est instancié MOCSP comme une application Web et est déployée dans un serveur Web, enrichi par des hyperliens de communication. L'appelant et l'appelé sont dans deux navigateurs Web connecté au serveur Web MOCSP. Les messages de signalisation circulent entre l'appelant et l'appelé via le serveur Web.

Pour les flux médias, je propose un streaming point-à-point entre l'appelant et l'appelé. Cela signifie que la vidéo et l'audio est transporté séparément par Real-Time Transport Protocol (RTP). Les codecs vidéo et audio ne rentrent pas dans le champ d'application de cette thèse. En outre, le serveur média est proposé (entité en option) dans le but de transcodage si les deux appelant et l'appelé ont des codecs différents.

A.3.2 Le flux d'appels

Le but principal de la session de contrôle est de contrôler le flux des médias, y compris la négociation de codec entre l'appelant et l'appelé. Toutefois, MOCSP ne définit pas un protocole de signalisation par défaut parce que les utilisateurs définissent leur propre protocole et extensions en fonction de leurs propres besoins. Cette section se propose de montrer comment la session de communication est rendue possible entre les deux navigateurs. Ce cas d'utilisation génère deux exigences. L'un est la définition du protocole de signalisation et l'autre est de savoir comment permettre les communications asynchrones entre le navigateur Web et le serveur Web.

Pour la communication asynchrone entre le navigateur Web et serveur Web, le système MOCSP dépend de la technologie WebSocket ; il va augmenter l'efficacité de programmation et réduire le temps de latence pour les appels établis dans l'environnement Web. J'emploie WebSocket pour connecter l'appelé et le serveur Web MOCSP, et l'appelant et le serveur Web MOCSP. Le serveur Web MOCSP identifie l'appelé et l'appelant sur la base des connexions WebSocket établies.

Le flux d'appels pour une session utilisateur/une session de contrôle est proposé dans la figure 5.3. Le principal but du flux d'appel est de réduire la latence pour l'établissement de l'appel. Le protocole de signalisation décrit le contrôle du point de terminaison médias en utilisant les messages open/oack/desc/sel/close/ackclose.

A.3.3 L'ouverture et flexibilité

Le système MOCSP peut être modélisé comme le montre la figure 5.4. Comme au moins deux intervenants sont impliqués dans une session, l'appelé reçoit un plus grand contrôle dans le système MOCSP. Cette modèle offre une plate-forme individuelle (la fusion de la couche de service et la couche de contrôle de session dans IMS).

Avec les services de communication, si un utilisateur veut concevoir un nouveau service, il a besoin du contrôle du protocole de signalisation. Dans le système MOCSP, les utilisateurs finaux ont une accessibilité complète dans protocole de signalisation. Pour certains services, les utilisateurs conçoivent un protocole de signalisation adéquat, indépendamment

de tour fournisseur. Le système et le protocole existants (par exemple IMS / SIP) ne supportent pas des changements aisément sauf pour des caractéristiques importantes en raison de problèmes de sécurité, de complexité et d'interopérabilité. Le processus de changement pour le protocole SIP est expliqué dans la RFC 5727. Cependant, les changements peuvent être faits rapidement pour les séquences de messages du protocole de signalisation dans le système MOCSP. La flexibilité obtenue peut être expliquée plus en détail comme suit: Quand un service est développé, toute la logique de service réside dans le serveur Web. Les clients sont stockés dans le serveur Web et sont livrés aux appelants et aux appelés à la demande. Cela signifie qu'un appelé obtient son client sur son inscription et les appelants téléchargent leurs clients en cliquant sur un lien hypertexte qui est fourni par l'appelé. Si des modifications sont apportées, qui devrait avoir un impact important dans le côté du client. Cela peut être effectué facilement en un seul endroit. L'intelligence des services est placée dans le côté de l'appelant, côté de l'appelé et le serveur Web MOCSP. Conceptuellement mettre plus d'intelligence dans le réseau (MOCSP Web Server) est la considération principale de la conception.

A.3.4 La mobilité utilisateur

La solution pour le cas d'utilisation considérant la mobilité des utilisateurs est développée sur la base du système MOCSP. Cette solution simplifie le développement, parce que notre solution permet de réduire les nœuds d'interdépendance entre l'appelant et l'appelé en un seul nœud qui est appelé "Network box" (la boîte de réseau). Fondamentalement, cette thèse agrège deux fonctionnalités existantes (qui sont complémentaires) en une seule fonction.

La solution peut être décrite sur la base d'un modèle abstrait. L'idée principale est de proposer un modèle descriptif qui devrait être indépendant de l'architecture. Par conséquent, les ingénieurs ont la liberté de mettre en œuvre une plate-forme particulière. Dans ce cas, un modèle descriptif est de considérer une seule entité qui sera responsable de la gestion des médias au sein d'une session pour le cas d'utilisation la mobilité des utilisateurs. Un modèle descriptif général pour la mobilité des utilisateurs est illustré dans la figure 6.1, en s'appuyant sur le modèle descriptif utilisé pour identifier le comportement correcte des médias par les serveurs d'applications. Le modèle descriptif proposé se compose de trois composants: Caller Box, Callee Box and Network Box (resp. composant appelant, composant de l'appelé et la boite réseau).

Bien que ce modèle soit intuitif, il est important d'indiquer comment notre modèle gère une session. Pour le cas d'utilisation la mobilité des utilisateurs, la gestion de session est principalement déléguée à la "Network box" qui est un intermédiaire entre l'appelant et l'appelé. Dans une équivalente avec la terminologie SIP, la "Network box" est amenée à agir comme "Register", "B2BUA" et "forking proxy". L'approche basée sur la "Network box" réduit le traitement des messages indésirables dans chaque session et la complexité de la coordination fonctionnelle.

La "Network box" est une entité responsable dans le serveur Web MOCSP. Ce modèle descriptif est facile à déployer dans le système MOCSP sans faire face à beaucoup de problèmes d'ingénierie.

A.3.5 Partial session transfer and retrieval

La solution pour le cas d'utilisation du transfert de session partielle et de récupération est exposée en se basant sur le système MOCSP. Le nœud central peut accomplir les objectifs des utilisateurs (l'appelant et l'appelé) telles que le transfert et la récupération des médias partiels. Cette solution permet n'importe quel nombre de transfert de session et la récupération des côtés de l'appelé et de l'appelant de manière efficace quel que soit les initiateurs.

La solution consiste en l'architecture et le protocole de contrôle, et les diagrammes de flux d'appels pour le transfert/récupération de sessions initiées à partir du réseau ou par l'utilisateur. Dans ce résumé, je vous présente l'architecture de ce cas d'utilisation.

L'architecture proposée, montrée dans la figure 7.1, dispose d'une "network box" et "Caller Box", "Callee Box" (boîte du réseau, boite appelant/appelé). "Caller Box" et "Callee Box" sont des points finaux pour les appelants et les appelés. Les "medium device" peuvent envoyer et/ou recevoir le média (audio / vidéo) et sont disponibles à proximité de l'appelé et l'appelant lors d'une session. Ces "medium device" peuvent être divisés en "medium source" et "medium sink". La figure 7.1 ne montre pas le "medium device", mais montrent le "medium source" et le "medium sink". Une "network box" coordonne les flux médias à travers les "Caller Box" et "Callee Box", et "medium device" selon les demandes des utilisateurs ou de sa compréhension du contexte de l'appelé et l'appelant.

Sur la base de cette architecture, je conçois un protocole adéquat qui peut être utilisé pour le transfert/récupération de sessions initiées à partir du réseau ou par l'utilisateur.

Dans la solution, la "network box", se comportant en mode gestion, prend en charge les besoins de transfert/récupération de sessions initiées à partir du réseau ou par l'utilisateur. Le principal avantage de cette solution est sa simplicité en tirant profit de la souplesse et l'accessibilité du système MOCSP (l'unique " orchestrateur ") et de la séparation des problématiques dans le protocole de signalisation.

A.3.6 La montée en charge

L'évaluation du système de MOCSP par rapport à la montée en charge se fait en fonction de quatre paramètres : les limites de la montée en charge, le niveau de complexité, les ressources de calcul nécessaires et la latence de l'établissement de la session.

I. Une montée en charge linéaire : Par défaut, chaque instance MOCSP est séparée et les besoin individuels de ressources augmentent linéairement avec le nombre des "network box" et le nombre d'appels reçue.

Par conséquent, il est facile d'adopter une approche de montée en charge verticale. Ceci signifie que le nombre d'instances MOCSP qui peuvent être ajoutés est limité par la capacité d'un serveur physique. Sur la base de cette approche, le système global est évolutive (monte en charge). De plus, les ressources de calcul nécessaires pour une session de communication ne varieront pas lorsque le taux d'appel varie. Cela signifie que les ressources doivent monter en charge linéairement avec le nombre de "network box" déployé et le nombre de appels reçus.

II. Complexité: L'architecture MOCSP est claire et simple. Les utilisateurs qui mettent en œuvre les systèmes MOCSP n'ont besoin ni de mettre en œuvre des techniques d'équilibrage de charge et ni de se préoccuper de l'aspect de routage définies dans le protocole SIP et les réseaux overlay P2P. Une session est établie entre deux utilisateurs, basés sur des connexions TCP. Les messages nécessaire pour une session sont envoyés dans les deux sens entre deux navigateurs par l'intermédiaire de la "network box". Enfin, le fournisseur de services devrait envisager correctement le déploiement des "network box" dans un serveur unique.

III. Ressources de calcul nécessaires: Selon nos calculs, le système complet a besoin d'environ 240, 000 serveurs Web afin de supporter 6 milliards de personnes qui utilisent des systèmes MOCSP. Dans ce cas, 3 milliards de personnes peuvent appeler l'autre 3 milliards de personnes. Dans ce calcul, les ressources informatiques nécessaires pour la couche médias ne sont pas prises en considération. Même si il est possible de calculer les ressources informatiques nécessaires pour les systèmes MOCSP, il n'est pas possible de comparer quantitativement avec un autre système de communication existant. Analytiquement, je démontre que MOCSP requière le minimum de ressources. Pour une session, une "network box" se compose de deux connexions TCP. De plus, la "network box" doit gérer la session et effectuer la gestion d'état pour l'appelant et l'appelé. Dans le système MOCSP, une seule entité est comprise entre appelant et l'appelé; une "network box" a besoin de ressources informatiques que pour la gestion d'état. Cela signifie que notre approche élimine des services de recherche inutiles qui se trouvent dans de nombreux nœuds intermédiaires. Par conséquent, le besoin de ressources informatiques est moindre que IMS et SIP P2P.

IV. La latence de l'établissement de la session: De toute évidence, le temps de latence de l'établissement de la session peut être rendue indépendant du nombre des appels effectués dans le système MOCSP. Ceci est possible grâce à la montée en charge linéaire et l'unicité de l'entité (c.-à-d O(1)) entre l'appelant et l'appelé. Enfin, cette évaluation montre que le système basé sur MOCSP est une solution simple pour un taux d'appel élevé par rapport aux systèmes de communication existants.

A.4 Conclusion

Passer du paradigme d'un seul créateur à plusieurs créateurs avec des services de communication est un but important. Ce changement de paradigme a été rendu possible avec une plate-forme Web en aidant le pouvoir et la puissance créative des gens/utilisateurs. Cependant, ce but n'est pas adressé avec des services de communication par la communauté de recherche.

La contribution principale de cette thèse est une proposition du concept MOCSP ainsi qu'un système pour communications interpersonnelles. Dans MOCSP, les utilisateurs/les appelants peuvent contrôler leur appel à travers un protocole de signalisation et peuvent concevoir leurs besoins. Le concept MOCSP supporte la diversité en gérant la complexité des services de communication.

Deux services différents permettent d'expliquer l'importance de la session de service de la base de réseau (network-based) et comment le développement de système est simplifié pour deux cas d'utilisation. Les deux scénarios représentent deux exigences différentes : le service logique exécutée pendant l'établissement d'appel et le milieu d'appel et la coopération du contrôle d'appel réseau et du fin d'appel.

Les solutions pour les services de session à base de réseau dépendent d'une seule entité appelée 'Network box'. La logique principale est placée dans cette network box basée sur les sessions. Cette approche est validée par deux scénarios d'utilisation : mobilité utilisateur et partial session transfer/retrieval (PSTR).

Comparé à la solution SIP, la solution utilisateur basé sur MOCSP peut économiser la puissance de traitement de calcul au moins de six messages dans un contexte de session en SIP proxy. L'architecture PSTR facilite le transfert de session partiel amorcé par le réseau et amorcé l'utilisateur ainsi que la récupération coté interlocuteur et appelant. Cela signifie que n'importe quel numéro de transfert peut être exécutée par la network box et (ou) des utilisateurs dans une seule session.

La complexité pour développer cette solution est réduite de deux façons : 1) Via un orchestrateur médiatique à la 'Network box' et par 2) un protocole de signalisation. Je sépare le protocole de signalisation dans le contrôle médiatique et le protocole auxiliaire basé sur l'approche de génie logiciel - ' séparation de préoccupation" (separation of concern)..

Cette thèse a analysé l'évolutivité d'une architecture de signalisation pour des services de communications interpersonnelles basées sur quatre critères - le niveau d'évolutivité,le niveau de complexité, le besoin des ressources informatiques nécessaires de calcul et l'installation de session de latence (session setup latency). L'architecture basée sur MOCSP surpasse deux autres architectures existantes IMS et P2P SIP.

En profitant du parallélisme, l'architecture MOCSP échelles linéairement sans ajout de complexité. Puisque le niveau de complexité de MOCSP est O (1), les ressources nécessaire de calcul et la latence d'installation de session sont moins importantes que les besoins de signalisation de l'architecture dans l'IMS et de SIP P2P.

Plus important encore, le nombre d'appels n'a pas d'influence sur le temps de configuration de session. En un mot, cette approche permet de gérerle nombre d'appels (soutenir l'état-Ful services réseau basés sur des sessions (la mobilité utilisateur, par exemple).), tandis que l'augmentation des ressources de façon

linéaire, sans mettre en péril le temps de réponse moyen.