HAL
open science

Service composition in converged service environment

Cuiting Huang

To cite this version:

Cuiting Huang. Service composition in converged service environment. Other [cs.OH]. Institut National des Télécommunications, 2012. English. NNT : 2012TELE0009 . tel-00762644

HAL Id: tel-00762644

https://theses.hal.science/tel-00762644v1

Submitted on 7 Dec 2012

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 TMSP-UPMC

Spécialité : Informatique et Télécommunications

Par Cuiting HUANG

Service Composition in Converged Service Environment

Soutenue le 2 Mai 2012 devant le jury composé de :

Ken Chen	**Professeur à l'Université de Paris 13**	Rapporteur
Stéphane Frénot	**Professeur à l'INSA Lyon**	Rapporteur
Hassnaa Moustafa	**Docteur à Orange Labs**	Examinateur
Woo Seop Rhee	**Professeur à l'Université Nationale Hanbat**	Examinateur
Nassim Laga	**Docteur à Orange Labs**	Examinateur
Bénédicte Le-Grand	**Professeur à l'Université de Paris 6**	Examinateur
Noël Crespi	**Professeur à Institut Télécom Sudparis**	Directeur de Thèse

Thèse n°2012TELE0009

Acknowledgments

I first would like to thank my supervisor, Professor Noël Crespi, who gave me the opportunity to do this thesis, and has been providing the help, suggestions and encouragement in all the time during this thesis.

I warmly thank Professor Ken Chen, Professor Stéphane Frénot, Mrs. Hassnaa Moustafa, Professor Woo Seop Rhee, Mr. Nassim Laga, and Professor Bénédicte Le-Grand for agreeing to participate in my thesis committee.

I would like to thank Professor Djamal ZEGHLACHE, the director of RS2M department and Mrs. Valérie Mateus, the assistant of RS2M department, for all of their kindness and help in these years.

I would also like to express my gratitude to Professor Xiaodi Huang, and all the current and previous colleagues in our team: Mehdi, Ali, Xiao Han, Zhenzhen, Nassim, Hongguang, Khali, Siva, Angel, Gyu Moung, Mrs. Rebecca, Siddhartha, etc. as well as my friends in ITS. Their comments, suggestions and kindness also helped me for completing this work.

Finally, I acknowledge my dear family who always supports me. Especially, I would like to give my special thanks to my husband Hongwu for all of his understanding, supports and encouragements.

Publications

I. C. Huang, G.M. Lee, and N. Crespi, "A Semantic Enhanced Service Exposure Model for Converged Service Environment", *IEEE Communications Magazine*, pp. 32-40, vol. 50, Mar., 2012

II. C. Huang, N. Crespi, and A. Maaradji, "Service Composition in IMS Environment: An Evolved SCIM Based Approach", *Forthcoming*, in *Proc. International Conference on Digital Information and Communication Technology and its Applications (DICTAP2012)*, Bangkok, Thailand, May. 16-18, 2012

III. C. Huang and N. Crespi, "A Service Composition Model for Dynamic Service Creation and Update in IMS/Web 2.0 Converged Environment", in *Proc. Sixth Asian Internet Engineering Conference (AINTEC'10)*, pp. 95-102, Bangkok, Thailand, Nov. 17-19, 2010

IV. C. Huang and N. Crespi, "An Automatic Service Composition Model in IMS/Web Converged Environment", *Poster presented in Principles, Systems and Applications of IP Telecommunications (IPTComm'10)*, Munich, Germany, Aug. 2-3, 2010

V. C. Huang and N. Crespi, "Extended Policies for Automatic Service Composition in IMS", in *Proc. IEEE Asia-Pacific Services Computing Conference (APSCC'09)*, pp. 254-259, Singapore, Dec. 7-11, 2009

VI. C. Huang and N. Crespi, "Enriched SCIM for Service Composition Within IMS", in *Proc. International Conference on Engineering Management and Service Sciences (MASS'09)*, pp. 1-4, Beijing, China, Sept. 20-22, 2009

VII. H. Zhang, Z. Zhao, S. Shanmugalingam, C. Huang, and N. Crespi, "Quality-assured and Sociality-enriched Multimedia Mobile Mashup in NGWN", *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, Jan. 2010

VIII. A. Maaradji, C. Huang and N. Crespi, “Towards Personalized Service Composition on IMS: A Basic Approach”, in *Proc. International Conference on Advanced Infocomm Technology (ICAIT’09)*, Xi’an, China, Jul. 7-9, 2009

Abstract

Service Oriented Architecture (SOA) is now acknowledged as a central paradigm for service delivery, acquisition and consumption. Under SOA, services are loosely coupled and thus greatly facilitating the interactions among service providers, service consumers and repositories. Over last decade, prodigious research and technology developments have been achieved for enhancing this paradigm both in IT and Telecom worlds, resulting supporting technologies, such as the IMS architecture for NGN, the mashup technologies for web services. However, there are still some substantial challenges hindering network and service evolutions. From the service point of view, these challenges mainly relate to certain essential functionalities in the SOA paradigm such as service description, discovery, access and composition management.

The goal of this thesis is to provide upcoming networks, mainly targeting Next Generation Network (NGN) and Future Internet (FI), with enhanced mechanisms to deploy revenue-generating and competitive services in a rapid and cost-effective manner. To achieve this objective, our research work concentrates on two topics in this field: i) Service composition management that facilitates the service creation and maintenance; ii) Service exposure that provides underlying supports for service composition management.

After a deep investigation of the related SoTA for service composition both in IT and Telecom domains, we propose an automatic service composition model relying on an IMS/Web converged environment. This proposed service composition model is intended to be one in which even non-professional users can easily reuse existing services, capabilities and resources to create new services by enforcing the automatic service selection, composition and monitoring. To further improve the automatic service composition feature, three strategies including passive update, active update and hybrid update are proposed and analyzed. Thereby the automatic service composition paradigm is enabled to go beyond design time to dynamically compose services at runtime. These enhancements allow users to profit from a highly personalized, meaningful communication and interaction experience.

Considering the significantly important role of service exposure in the paradigm of service composition, we then introduce three different service exposure strategies for catering to the different domains' service requirements. We first propose a centralized service exposure framework for a variety of services, including Telecom/Web/Device/user-generated services. This framework mainly aims at enhancing the user-centric feature and the convergence feature, as well as providing the unified access to diverse services, thereby enabling a seamless interaction among heterogeneous services and facilitating the reuse of these existing services by both professional and non-professional users. Subsequently, perceiving

the challenges generally encountered by the centralized solutions such as single access, high maintenance cost, and easy to lead to performance bottleneck, we analyze the feasibility of using decentralized P2P to support the distributed service exposure in large scale. Then two P2P based service information sharing models are designed to complement the centralized service exposure model: i) A hierarchical P2P based model, which reuses the structured P2P overlay (i.e. Chord) for guaranteeing the service discovery efficiency, meanwhile adopts the concept of abstract service publication and discovery for enabling the ambiguous service searching which is indigent in current structured P2P systems. ii) A triplex P2P overlay based model, which mainly targets the devices offered services. In this model, we use the gateways to delegate the devices residing in them for the global service exposure, and use a triplex overlay based architecture, which includes an underlying unstructured P2P layer, a Semantic Overlay Network (SON) based overlay and a service dependency overlay, for the service information sharing and discovery. These introduced architectures and mechanisms for service exposure are analyzed and evaluated through a set of implementations and simulations.

Key words:

Service composition, service exposure, service discovery, service publication, convergence, NGN, Web, IMS

Table of Contents

Acknowledgments ..... **i**

Publications **1**

Abstract **3**

Table of Contents **5**

List of Figures **9**

List of Tables **13**

Chapter 1 Introduction **15**

1.1 Service delivery in next generation service environment 17

1.2 Thesis motivations and objectives 18

1.3 Thesis contributions 20

1.3.1 Service composition in NGN environment 21

1.3.2 Service exposure in converged environment 21

1.4 Thesis outline 23

Chapter 2 State of the art **25**

2.1 Next Generation Networking 25

2.1.1 Next Generation Network (NGN) 25

2.1.2 Next Generation Internet (NGI) 29

2.1.3 Convergence of NGN and NGI to Future Networks 30

2.2 Next generation services 31

2.3 Next generation service provisioning 32

2.3.1 Service composition 34

2.3.1.1 Service Oriented Architecture (SOA) 34

2.3.1.2 Service composition for Web services 35

2.3.1.2.1 Web services 35

2.3.1.2.2 Web service composition using SOA 37

2.3.1.3 Service composition for Telecom services 43

2.3.1.3.1 Original service delivery mechanism in IMS 43

2.3.1.3.2 OMA Service Environment (OSE) based solution 45
2.3.1.3.3 Service Capability Interaction Management (SCIM) based solution....46
2.3.1.3.4 Other service composition solutions in Telecom 47
2.3.2 Service exposure 50
2.3.2.1 Parlay/OSA gateway 52
2.3.2.2 ParlayREST 54
2.3.2.3 OneAPI 55
2.3.2.4 Wholesale Application Community (WAC) 55
2.3.2.5 Other service exposure solutions 56
2.4 Summary 57
Chapter 3 Service composition in NGN environment 59
3.1 Service composition challenges statement.... 59
3.1.1 Convergence 59
3.1.2 User-centricity 60
3.1.3 Automaticity 61
3.2 An automatic service composition model in converged environment.... 61
3.2.1 Global overview for the service composition model 61
3.2.2 Functional architecture overview....62
3.2.3 Service profiles 63
3.2.4 Main components....66
3.2.5 Basic composite service execution process 66
3.2.6 Basic service creation and service exposition processes....67
3.2.7 Basic service discovery process....70
3.2.8 Enhanced service profiles for automatic service composition 71
3.2.8.1 Extended service profiles: abstract service profile and concrete service profile....71
3.2.9 Automatic service creation process 75
3.2.10 Automatic service update process....76
3.2.10.1 Passive update process 76
3.2.10.2 Active automatic update process....78
3.2.10.3 Comparison and discussion for passive and active solutions....79
3.3 Proof of concept prototype.... 81
3.4 Summary 85

Chapter 4 Service exposure in converged environment 87

4.1 Service exposure challenges statement 87

4.2 A semantic enriched service exposure model in converged environment 92

4.2.1 Overall architecture 92

4.2.2 Service exposure processes.....94

4.2.2.1 Service exposure for Telecom services.....94

4.2.2.1.1 Telecom service access and invocation 95

4.2.2.1.2 Telecom service publication and discovery 97

4.2.2.2 Web service exposure 97

4.2.2.3 Device service exposure.....99

4.2.2.4 User generated service exposure.....103

4.2.3 Usage scenarios 106

4.2.3.1 Converged service creation.....106

4.2.3.2 Converged service execution 109

4.2.4 Discussion.....112

4.3 A P2P based distributed service exposure model 114

4.3.1 Some backgrounds and related work for P2P 114

4.3.1.1 Traditional unstructured P2P based solutions 114

4.3.1.2 Traditional structured P2P based solutions 115

4.3.1.3 Other P2P based solutions.....116

4.3.2 Overall architecture 118

4.3.3 Adaptation of P2P overlay to service publication and discovery.....119

4.3.4 Node ID and resource key assignment.....122

4.3.5 Node joining and departure processes 128

4.3.5.1 Super node's join and departure processes 128

4.3.5.1.1 New super node joins to the network.....128

4.3.5.1.2 A super node leaves the network 132

4.3.5.2 Ordinary node's join and departure processes 133

4.3.6 Service publication process 134

4.3.6.1 Publish or remove an abstract service from the network 134

4.3.6.2 Publish and remove a concrete service from the network.....138

4.3.7 Service discovery process.....142

4.3.8 Performance analysis 145
4.3.8.1 Performance estimation equations 146
4.3.8.2 Example analysis 149
4.3.9 Discussion 155
4.4 A P2P based service exposure model in IoT environment 157
4.4.1 Device service exposure 159
4.4.1.1 Local device gateway 160
4.4.1.2 Local device exposure 162
4.4.1.3 Global device exposure 163
4.4.2 P2P based service information sharing 163
4.4.2.1 Triplex overlay architecture overview 164
4.4.2.2 SON layer generation 165
4.4.2.3 Service discovery based on triplex overlay 169
4.4.3 Performance analysis and discussion 173
4.5 Summary 180
Chapter 5 Conclusion and Future Research Directions 182
5.1 Contributions 182
5.1.1 Automatic service composition model in IMS/Web converged environments 182
5.1.2 Service exposure in converged environment 183
5.2 Perspectives 185
Chapter 6 Résumé Français 187
I. Introduction 187
II. Contexte 188
III. Composition de services dans un environnement NGN 190
IV. Exposition de services dans un environnement convergent 196
V. Conclusion 203
References 205
Abbreviations 215
Appendix 220

List of Figures

Figure 2-1: ITU-T NGN reference model 26
Figure 2-2: 3GPP IMS architecture overview 28
Figure 2-3: SOA basic architecture..... 34
Figure 2-4: SPICE ACE architecture 38
Figure 2-5: BPEL designer screenshot 40
Figure 2-6: Yahoo Pipe screenShot 41
Figure 2-7: Ezweb - creating new service by chaining widgets screenshot 42
Figure 2-8: Widget linker screenshot..... 42
Figure 2-9: Simplified IMS architecture..... 44
Figure 2-10: IMS user profile 44
Figure 2-11: OSE architecture 45
Figure 2-12: SCIM within IMS environment 46
Figure 2-13: SPICE functional architecture 48
Figure 2-14: Composition framework components within overall SERVERY architecture 49
Figure 2-15: Ericsson's Application Router (AR) for service chaining 49
Figure 2-16: FOKUS's Telco-enabled Web mashup architecture 50
Figure 2-17: Parlay/OSA architecture 54
Figure 2-18: ParlayREST architecture in OMA 54
Figure 2-19: OneAPI interfaces for accessing to Telecom resources 55
Figure 2-20: WAC: service deployment across devices, platforms and networks 56
Figure 3-1: Unified service composition model relying on IMS/Web converged environment 62
Figure 3-2: Service Composition Engine (SCE) in IMS/Web converged environment... 63
Figure 3-3: An example of a simplified user service profile 64
Figure 3-4: An example of a simplified atomic service profile..... 64

Figure 3-5: An example of a simplified composite service profile 65
Figure 3-6: A usage example for the atomic service advertisement 69
Figure 3-7: An example of Service Comparator 70
Figure 3-8: Functionalities combination 72
Figure 3-9: Abstract composite service profile 72
Figure 3-10: Abstract atomic service profile 73
Figure 3-11: Concrete atomic service profile 73
Figure 3-12: Concrete composite service profile 74
Figure 3-13: Simplified service creation example 76
Figure 3-14: Service profile for passive update 77
Figure 3-15: Service policy for active update 79
Figure 3-16: Service profile for hybrid solution 80
Figure 3-17: MSN robot like service creation interface 82
Figure 3-18: Interpretation for natural language based request 83
Figure 3-19: Simplified call flow for service creation and execution through SCE..... 84
Figure 4-1: Relationships among three service exposure solutions 91
Figure 4-2: Overall architecture for converged service exposure 93
Figure 4-3: Telecom service exposure 95
Figure 4-4: Web service exposure 98
Figure 4-5: Service exposure for devices 100
Figure 4-6: Publish a concrete device as a widget 101
Figure 4-7: Service composition scenario for device offered services 102
Figure 4-8: Call flow for the execution of a device related composite service 103
Figure 4-9: User generated service publication and discovery 104
Figure 4-10: A UI example for the service publication front-end 104
Figure 4-11: A UI example for the service description file generation front-end 105
Figure 4-12: A simplified service composition logic 106
Figure 4-13: Call flow for converged service creation 109

Figure 4-14: Call flow for converged service execution 112
Figure 4-15: Overall architecture for the hierarchical P2P service information sharing system 118
Figure 4-16: Simplified service classification method 120
Figure 4-17: Service category tagging for super nodes in Global Service Overlay 121
Figure 4-18: Subset service overlays 121
Figure 4-19: An example of abstract service tagging for super nodes in Subset Service Overlay..... 122
Figure 4-20: Internal finger table in a super node..... 126
Figure 4-21: A new super node is added to service overlay 131
Figure 4-22: Relationship between super node and ordinary nodes 134
Figure 4-23: Abstract service publication process..... 136
Figure 4-24: Concrete service publication process for $K_{ch} = K_{sh}$ 140
Figure 4-25: Concrete service publication process for $K_{ch} \neq K_{sh}$ 140
Figure 4-26: Adding a new concrete service to ordinary node 141
Figure 4-27: Deterministic service discovery process 144
Figure 4-28: Ambiguous service discovery process 145
Figure 4-29: Relationship between the average hops and the number of nodes..... 150
Figure 4-30: Relationship between the maximum hops and the number of nodes..... 151
Figure 4-31: Relationship between index size and the number of nodes 151
Figure 4-32: Relationship between the proportion of super nodes and the average hops 152
Figure 4-33: Relationship between the maximum hops and the proportion of super nodes 153
Figure 4-34: Relationship between the index size and the proportion of super nodes ... 154
Figure 4-35: An example of the local device gateway 160
Figure 4-36: Triplex overlay for P2P based service discovery..... 164

Figure 4-37: An example for Local Abstract Service Table 166
Figure 4-38: An example of a simplified Ordinary Neighbor Node Table..... 167
Figure 4-39: An example for SON Linkage Table 168
Figure 4-40: Flowchart for SON Linkage Table generation..... 169
Figure 4-41: Comparison for the abstract services' dependency..... 171
Figure 4-42: Flowchart for service discovery process in triplex layers based service exposure model 172
Figure 4-43: Comparison for the success rates among three types of P2P systems (p=0.1%, R=10%, q=10%, k=50%)..... 176
Figure 4-44: Comparison of the success rates among three types of P2P systems (p=0.1%, R=5%, q=20%, k=50%)..... 176
Figure 4-45: Comparison for the success rates among three types of P2P systems (p=1%, R=10%, q=10%, k=50%)..... 177
Figure 4-46: Comparison for the success rates among three types of P2P systems (p=1%, R=5%, q=20%, k=50%)..... 177
Figure 4-47: Comparison of the success rates (S) by modifying the dependency probability (K) 178
Figure 4-48: Average messages needed for achieving the success rates of 70%, 80% and 90% 179

List of Tables

Table 2-1: Telecom service exposure platforms and their exposed services 57

Table 3-1: Functionally equivalent atomic service profiles 74

Table 3-2: Comparison of three service update strategies 81

Table 4-1: Objectives and contributions of three service exposure solutions 91

Table 4-2: Comparison among centralized solution, Chord based solution and hybrid Chord based solution 119

Table 4-3: Summarization of the keys' structures 124

Table 4-4: External finger table in a super node 126

Table 4-5: An example of Local Nodes Index 133

Table 4-6: Update for local node index table 138

Table 4-7: Local concrete service index table 141

Table 4-8: Relevant parameters for centralized solution, Chord based solution and hybrid Chord based solution 146

Table 4-9: Simulation parameters of the networks 175

Chapter 1 Introduction

The last decade witnesses the tremendous evolution in both Telecom and Internet worlds. The advancement of mobile communication and broadband Internet access technologies enables us to be connected *any time, any where* through diverse means (e.g. voice/video call, Short Message Service (SMS), Multimedia Messaging Service (MMS), Instant Message (IM), email, social network, etc.). They also greatly ease and enrich both our professional and personal lives. Companies can do their business with each other by exchanging information via Internet, and perform the online transactions regardless of the geographical limitation. Users can enjoy themselves with various online entertainments (e.g. shopping, gaming, audio, video, social interaction, etc.). These rapid evolutions make it almost impossible to image what our life would be without computer, mobile phone, and Internet. On the other hand, users' expectations on the services they can get from networks are becoming higher, especially after the emergence of Next Generation Network (NGN) and Next Generation Internet (NGI). NGN and NGI are both considered as the foundations for next generation networking and proclaim to provide substantial support for the next generation service delivery. Therefore, how to effectively create revenue generating and competitive services in a fast-paced environment, how to reduce both their creation and maintenance cost, and their time to market, as well as how to optimize their lifecycles, are the very relevant, challenging questions for network operators and services providers.

NGN [1] was introduced for enabling the Telecommunication operators to better take advantage of their core competency and to prepare for the emerging telecommunications environment, and thus achieves the goals of reducing the deployment cost, optimizing the infrastructure investments, facilitating the new service generation, and consequently retaining and/or enforcing their roles in the more and more competitive service market. The general idea behind NGN is to reuse the Internet technologies such as Internet Protocol (IP) for confronting the competition from Internet domain, thus it is labeled as "all-IP" by comparing with the previous generation of Telecommunication networks, which are mainly based on circuit switching, Signaling System #7 (SS7) and various hierarchies (e.g. Synchronous Digital Hierarchy (SDH)) for transport. Furthermore, NGN makes a clear separation between the network access and services, and moves the network architecture from the traditional vertical approach (i.e. the access, control and services are closely tied) towards a horizontal approach, where the transfer functions, transport and service control functions, and the application functions are separated into three independent layers, and each layer provides reusable elements to other layers [2].

NGI [3], which is also known as Future Generation Internet (FGI) or Future Internet (FI), was born as an approach to deliver new services upon “Internet 2” or “Beyond IP”. It aims at providing a set of improvements over Internet, to name a few, service/technology independence, scalability, simplicity, addressing, security, privacy, mobility, reliability, manageable end-to-end Quality of Service (QoS)/Quality of Experience (QoE), openness, fast deployment of personalized services and business effectiveness, etc. To achieve these goals, NGI revolutionizes the networking and service aspects following either an evolutionary way or a clean slate approach.

Regarding these evolution emphases in IT and Telecom domains, we can notice that the distinction between the Telecom networks and the IT networks is being blurred, especially from the service’s view point. Convergence becomes one of the fundamental characteristics for both IT and Telecom networks. The upcoming service environment is generally considered as a converged service environment in which the different kinds of services can interact each other seamlessly, regardless of the underlying heterogeneities.

Over the last decade, prodigious progress has been made in converging various wired, mobile and wireless networks for providing ubiquitous services to end users, thus has achieved *any time, any where* service providing promise. Currently, the convergence is taking place between Telecom and IT worlds. One of the most prominent examples is the alliance between IP Multimedia Subsystem (IMS) and Web 2.0 for benefiting form bridging Telecom potentials of reliability and trustworthiness with Web’s openness and flexibility.

The concept of “Web 2.0” began with a conference brainstorming session between O’Reilly and MediaLive International in 2004, and gained its popularity in a very short time. The advent of Web 2.0 brings a significant impact on the service usability and provisioning by encouraging the involvement from end-user for innovative services and/or content creation, that is, an end-user is not only considered as a service/content consumer, but also encouraged to be a service/content contributor. Web 2.0 revolutionized both software engineering methods and users’ interaction manner with software features. One crucial characteristic of Web 2.0 is that: software features are no longer packaged as an independent application; instead, they are split into generic Web services and published to a common registry for promoting cross-network and cross-organizations sharing, collaboration, reusability, and integration. Some examples of Web 2.0 include social networking sites, blogs, wikis, video sharing sites, hosted services, web applications, mashups and folksonomies.

IMS was introduced by 3rd Generation Partnership Project (3GPP) in 2002 Release 5, and has been widely recognized by the Telecom world as the reference architecture for NGN. IMS enforces the service control over IP based infrastructures, enables the access independent feature, and guarantees a secure,

ubiquitous service experience for user. It has gone towards making service development more flexible, such as the overlay architecture that enables the convergence of voice, data and multimedia traffic over an IP based infrastructure, the unified service control layer based on Session Initiation Protocol (SIP), the existing of the service capabilities layer and the Initial Filter Criteria (iFC) based service invocation mechanism.

Regarding the characteristics and evolutions trends of Web 2.0 and IMS, it is explicit that both of them emphasize the improvement of service delivery for fulfilling the more and more complex requirements from end-users. Both of them adopt the concept of service composition as a basic service delivery mechanism that allows the creation and execution of complex services by integrating the distributed autonomous service building blocks. Related research work is being carried out for years, and various service composition approaches have been proposed by different standardization organizations and companies catering to different network requirements. Nevertheless, some attributes, such as convergence, automaticity and user-centricity are still the open issues, which are being pursued by both industry and academia. This thesis aims to address these issues for providing an agile service composition model which satisfies the best practices in the converged service environment.

1.1 Service delivery in next generation service environment

Services are considered to be the most important impact factor for the success in the competitive customer-driven market. Both service providers and operators are struggling to find out innovative ways to brand and bundle new services, to achieve operational cost reductions, and to strategically position themselves in the service market. Service Oriented Architecture (SOA), which enables the loose coupling and dynamic binding between services, is widely accepted as the main principles and methodologies for next generation service delivery. It distinctly characterizes two roles for the services: service provider and service consumer. Service providers need to create services, make them accessible for third parties and publish details of their services. Service consumers discover services and then invoke them when needed. This service creation principle impacts the service delivery both in IT and Telecom domain. In the IT domain, instead of only using the silo method in which the software features are packaged as an independent application, the service functions are split into autonomous service building blocks in order to enable the cross network and cross organization sharing, collaboration, reusability and integration. In the Telecom domain, a great deal of efforts also have been made for catering to this trend, especially after the introduction of the next generation networks such as IMS, in which the service layer is separated from the control layer and underlying transport layer, and thus gives more evolution freedom for the services

themselves. For instance, in IMS service architecture, a service capability layer, which contains a set of application-independent building blocks for offering generic functionalities to support various multimedia applications, is separated from the control layer. Such building blocks are called Service Capabilities by 3GPP, and called Enablers by Open Mobile Alliance (OMA). Generally, a service capability provides a predefined functionality by itself, but a new and more revenue generating service is created by combining multiple service capabilities. Nowadays, creating value-added services by reusing the existing service components, which is well known as Service Composition, has become a key aspect for service delivery in the SOA landscape both for IT and Telecom worlds.

For being reused for other applications, a service first needs to be known and accessible by users or other applications. In the IT domain, services are generally exposed through open Application Programming Interfaces (APIs), and the Object Access Protocol (SOAP) and Representational State Transfer (REST) technologies are widely used for facilitating the invocation of services. Moreover, a set of the service description and publication mechanisms such as Web Service Description Language (WSDL), Universal Description, Discovery and Integration (UDDI) as well as the later semantic annotation mechanisms are adopted for facilitating the service publication and discovery. Meanwhile, being menaced by the IT competitors, Telecom operators are forced to open themselves to both professional and non-professional users for the purpose of retaining or enlarging their service market. Parlay/OSA Gateway, Next Generation Service Interfaces (NGSI), and OneAPI were specified by the standardizations for enabling the access to the Telecom services/capabilities by outer world applications through unified access interfaces. The emergence of these technologies enables the Telecom operators to expose their services to developers and users, meanwhile the third party services such as map and social networks are also enabled to be used by Telecom applications. This thus enables a highly synergy between two domains and achieves the goal of convergence from the service layer.

1.2 Thesis motivations and objectives

The research for service composition in the converged environment has been carried out for years. Many solutions have been proposed for catering to the different network requirements. However, most of current service composition mechanisms within the IT and Telecom worlds are evolving independently from each other based on the different underlying infrastructures. For example, Telco networks nowadays are trying to enhance their openness to other services and thus enable the service creation through the service composition paradigm. Recently, a paradigm named as “Telco 2.0” [4] was forged by Telecom operators for addressing the larger range of markets, which is not only targeting the local developers. In this

paradigm, Telecom operators expose the functionalities, such as SMS, MMS, Localization, and Voice over IP (VoIP), to third party developers. Meanwhile, the third party services such as Maps and social networks are reused by the Telecom applications. However, this openness generally addresses the needs of certain business markets and is currently limited to those services that are standardized. Moreover, there is limited room for users, or even the third party service providers to extend and build upon those services. On the other hand, for Internet services, such as social networking services, they are open in the sense that anyone with internet access can subscribe to and use the published services, and third parties also can offer additional service functionalities through open APIs. However, it encounters certain challenges such as security, QoS, privacy, etc. which contrarily are the essential functions provided by Telecom services. Some other models, such as Over-The-Top (OTT), have emerged for competing with the Telco model for enabling the consummation of communication services by non Telecom services (e.g. search for Google, communication control for Skype, or social network for Facebook) [5]. Being different from the Telco model, which enables the internetworking among different operators relying on the strong standardization and the peering agreement, OTT based ecosystems currently are independent and non-interoperable neither at the control layer nor at the application level. For instance, an iPhone app will never run on Android, even where operators attempted with Parlay or Java APIs for Integrated Networks (JAIN) to provide common APIs to their network capabilities. That is due to the fact that the openness of OTT greatly depends on OTT providers' power (competitive advantages) and strategies, and most of them adopt the walled-garden paradigm for protecting their revenues from being shared by others. As the openness and the seamless interoperation among different kind services are considered as necessary for the converged service environment, they need to be further improved for enabling the agile converged service provisioning.

Moreover, even some improved service composition management mechanisms, such as Service Capability Interaction Manager (SCIM) and OMA Service Environment (OSE), have been specified by the Telecom standardization organizations. However, these solutions are generally based on Telecom technologies (e.g. using SIP for the service management). They thus mainly target the Telecom savvy developers. Such mechanisms for cross application domain brokerage and service description are still missing. As the convergence is an inevitable trend for Telecom and IT worlds, how to provide a unified and effective service composition mechanism for enabling the service creation regardless of the underlying network architectures and the service types (e.g. IMS, Web, Devices, etc) is questionable for future networks.

Furthermore, in the new philosophy of services, a user is no longer limited to the role of a service consumer, but also a service contributor, who contributes her/his intelligence for the new services, such as

the successful experience of Web 2.0 shows to us. Currently, such kind of service composition mainly targets the professional users or the advanced users who have at least certain service development skills. Even some so-called user centric tools for the service creation such as Yahoo Pipes, they are still too complicated for the ordinary users or cannot support complex tasks. Concerning the possibility to extend the service composition fashion to a larger scale, including the ordinary users who do not have any development skill, the more user centric and practical mechanisms are considered as necessary. These user centric mechanisms should not be limited to the improvement of the service creation process as most of current solutions do, but also should include the auxiliary processes, such as the service publication, service discovery, service adaptation, etc.

Facing the service composition challenges, this thesis aims to improve the service composition paradigm in the converged service environment by addressing the issues such as convergence, automaticity, and user-centricity. To achieve this goal, an agile service composition model is proposed for empowering end-users to access IMS and Web services seamlessly through different types of terminals, and for enabling the service creation and interoperability, regardless of the different types of services (IMS services, Web services, or device offered services). Meanwhile, the user-centricity feature is enhanced by hiding the backend complexity and simplifying the service creation interface and process, as well as by providing the automatic service creation and update mechanisms.

As service exposure is considered as a prerequisite for the service composition, we introduce new service exposure mechanisms, including service publication, service discovery and service information sharing in both centralized and distributed service environment, for catering to the different service composition contexts. These enhanced service exposure mechanisms can act as a complement for the service composition model.

1.3 Thesis contributions

In this thesis, we consider the IMS/Web converged environment as a concrete service environment for the next generation service delivery. Within this scope, this thesis first concentrates on the service composition management based on such converged environment. Considering the important role played by service exposure for the service composition paradigm, we then investigate the existing service exposure mechanisms, and propose the solutions for satisfying the different service environment requirements.

1.3.1 Service composition in NGN environment

Relying on an IMS/Web 2.0 converged environment, we propose an extensible service composition model which uses the concept of service profile for encapsulating the service composition information. Based on this model, service profiles are further divided into different types: abstract atomic service profile, concrete atomic service profile, abstract composite service profile and concrete composite service profile. That is for realizing the automatic service composition, which mainly relates to the service update and service creation processes. Moreover, an active approach, a passive approach, and a hybrid approach are proposed for balancing the latency and real-time characteristics. With these enhancements, both user's experience and involvement are greatly improved. The service lifecycle is optimized, and the cost for service creation and maintenance is decreased.

1.3.2 Service exposure in converged environment

Considering that service exposure plays a significant role in the next generation service delivery, we propose three exposure models that act as a complement for the service composition model: a centralized semantic enriched service exposure model, for the converged service environment; a hierarchical P2P based service exposure model, for the distributed service environment; and a triplex service exposure model, for the Internet of Things (IoT) environment. These service exposure models separate the service discovery and selection functionalities from the service composition environment. Thus they not only release the service composition environment from the burdensome service discovery task, but also enable the reuse of larger-scale services, regardless of their underlying heterogeneities. Therefore, the sharing of the different kinds of services among different platforms can be further improved.

The first proposed service exposure model relies on a centralized platform for improving the cross domain service interoperability and the user-centricity feature. It enables the services derived from different domains (i.e. Telecom/Web/Device/User) to be integrated into a composite service regardless of their underlying heterogeneities. Meanwhile, the user-centric feature is enhanced by applying the semantic annotation both to the service description and to the user request, as well as by empowering a user to publish and share his/her created service in a user-friendly way. Moreover, a variety of interfaces, including APIs, widgets, and natural language based tools, are provided for satisfying the requirements of users at the different levels. Thereby the users profit from a highly personalized, meaningful communication and interaction experience.

The central solutions are generally considered as easy to implement and to enforce the monitoring on the services. However, they easily attract the malicious attacks, introduce a single failure point, require a

high maintenance cost, and are not scalable and extensible enough to deal with the dynamic service environment. Moreover, centralized systems limit the interoperability among different platforms, which leads to many isolated service islands, and thus incurs the inevitable resources redundancy. Overcoming these limitations of the centralized solutions, the decentralized Peer-to-Peer (P2P) can be considered as another choice to support distributed service exposure on a large scale. Thus we propose an alternative P2P based service exposure model. This model uses the structured P2P overlay as the distributed service repository network. Adopting the Chord protocol as the base, it thus ensures the efficiency of query routing in the large-scale P2P networks. Meanwhile, the model introduces the abstract service of publication and discovery, which incorporates with the unified semantics and reasoning engines for the ambiguous service searching. The hierarchical network and resource organization in the model enables a system to quickly identify the peers that most likely contain the services satisfying user's requirements, and service discovery efficiency is thus greatly improved.

Along with the evolutions of service composition and Internet, nowadays, not only the Telecom and Web services are involved in the service composition, but also the device-offered services, especially after the popularization of IoT in these years, are more and more involved in service composition. Some intrinsic features of these small or embedded devices make their exposure to be more complicated, and the classical solutions for Telecom and Web service exposure are not currently feasible to be used directly for the device offered services. Thus we propose another P2P based distributed service exposure model, which mainly targets the device-offered services. This service exposure model enables both the powerful devices that can access to the IP network directly, and the weak devices that need a gateway to connect them to the external parties, to expose themselves in a global network. This model enables the service information sharing among different users, and different service platforms, regardless of the underlying heterogeneities and complexities. Moreover, to avoid the single failure point for the centralized UDDI likes solution and the possible appearance of the bottleneck while the number of devices increases to a certain level, a triplex layers based P2P service discovery mechanism, including an unstructured P2P layer, a Semantic Overlay Networks (SON) layer and a service dependency layer, is proposed for improving the service discovery efficiency.

The contributions of the original publications are summarized as follows: Paper I describes centralized service exposure mechanism that relates to Section 4.2; Paper II and III address the service composition management including automatic service creation and automatic service adaptation, they mainly relate to Section 3.2 and Section 3.3; Paper IV, V and VII deal with the service composition issue in the IMS

environment, and they are mainly SIP and SCIM based. This part is not included in the manuscript, but some concepts are reused in Chapter 3.

The above mentioned research results have been contributed to three major projects that the author had been involved: the Exoticus project (Jan. 2008 – Dec. 2009) is a national project under the SYSTEM@TIC cluster, which aims to address service modelling and service composition in NGN; the SERVERY project (Nov. 2008 – Oct. 2011) is a European project under the EUREKA ICT cluster, and its goal is to enable a Service Marketplace that bridges the Internet and Telco world by merging the virtues of these two worlds; and the DiYSE (Sept. 2009 – Mar. 2012) is a European ITEA-2 project, it aims to enable people to direct their everyday environment into a highly personalized meaningful communication/interaction experience, and contribute substantially to the open Internet-of-Things world and the transition to Web 3.0.

1.4 Thesis outline

This thesis is organized as follows: Chapter 2 provides background information on the next generation networking, service composition, and service exposure. The main contributions of the thesis are illustrated in Chapter 3 and Chapter 4. In particular, Chapter 3 describes the proposed automatic service composition model, as well as the automatic service creation and update processes based on this model. A MSN robot likes natural language based composer is also introduced as the proof-of-concept for this service composition model. Chapter 4 depicts the strategies for the service exposure in the converged service environment. Three disparate solutions are presented for targeting the different kinds of services. Finally, this thesis is concluded by summarizing the contributions, the advantages of these proposed solutions, as well as presenting some future work in Chapter 5.

Chapter 2 State of the art

In this chapter, we investigate the research and the standardization work on the networking and service evolutions, as well as the related supporting technologies over the last decade. This state of the art (SoAT) mainly focuses on NGN in Telecom domain and NGI in IT domain, which both can be considered as the prophase for Future Networks (FN), the service delivery mechanisms based on them, and the relevant service exposure solutions, which complement the service composition in the next generation converged service environments. We analyze the advantages and limitations of the existing approaches in order to propose, in the following chapters, the new service delivery architectures and mechanisms that cater to the requirements of future networks.

2.1 Next Generation Networking

Nowadays, both the telecommunication network and internet are developing towards the "future" or "next" generation. NGN and NGI are two most popular technologies in their respective domains for this goal.

2.1.1 Next Generation Network (NGN)

NGN is a broad term for "some future versions of networking" in telecommunication core and access network. It originates from Telecom industry and standardization organizations such as International Telecommunication Union – Telecommunication Standardization Sector (ITU-T), European Telecommunications Standards Institute (ETSI) and 3GPP. For instance, ITU-U defined "*a next-generation network (NGN) is a packet-based network which can provide services including Telecommunication Services and able to make use of multiple broadband, quality of service enabled transport technologies and in which service related functions are independent form underlying transport-related technologies. It offers unstructured access by users to different service providers. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users*"[1]. In the last decade, ITU-T devoted special interest for identifying the requirements of NGN and defined the NGN reference model as illustrated in Figure 2-1[6].

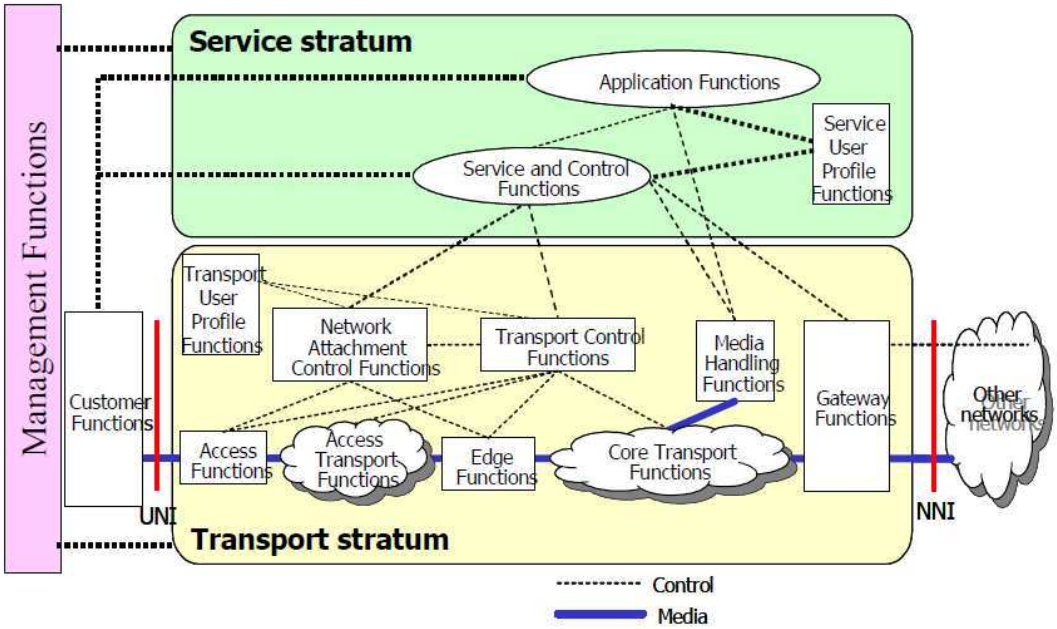


Figure 2-1: ITU-T NGN reference model [Source from ITU-T workshop in 2005 May]

Differing from previous Telecom systems, which rely on SS7 and SDH similar hierarchies for transport, NGN is commonly built upon the Internet Protocol (IP), and thus it is generally labelled with the term “all IP”. It intends to provide a common, unified, and flexible control environment that can support multiple types of services and management applications over different types of transport, thus achieves the *any where*, *any time* service providing promise. It proclaims to allow increasing productivity by creating new usage of data and voice services, enable service provider to offer real time and non real time communication services, and allow them to provide nomadicity and mobility of both users and devices. It also promises to reduce cost for infrastructure maintenance.

To achieve these goals, NGN involves several fundamental changes for the Telecom systems: (1) Migrate the traditional vertical stovepipe network architecture to the IP-based layered architecture; (2) Combine both Internet and Telecom services to offer the innovative services to the users; (3) Emphasize the convergence, including both the fixed-mobile convergence and the media (e.g. data, voice, video via the same network) convergence, for improving the user experience; (4) Separate the service layer and control layer from the transport layer, thus enable them to be evolved independently, meanwhile, facilitate and enforce the control of services; (5) Introduce the simpler and open network architectures for enabling the quicker introduction of new services; and (6) Support a wide range of services and applications by using the mechanisms based on the modular and flexible structure of elementary service building blocks.

After the research in past years, some solutions and mechanisms, which enable network operators to move towards the implementation, integration and deployment of NGN in their platforms, have been proposed by different standardization organizations. Among these solutions, IMS is a key example of an instantiation of NGN.

Originally being approved by 3GPP, IMS is a part of Universal Mobile Telecommunications System (UMTS) Release 5 in 2002 [7]. It aims to provide multimedia services and enable network and services converge into a single service control platform on top of various access networks. In Release 5, IMS was initially specified for providing end-to-end session establishment, user and network security control, policy control between the access network and the application server invocation. Both GSM and GPRS access networks were supported in this release. Then Release 6 added the support for the inter-working with Wireless-LAN (WLAN), and introduced new service capabilities, such as Presence, Messaging, Conferencing and Push-to-talk Over Cellular (PoC); Furthermore, the enhancement for supporting the fixed networks, including Packet Cable access [8] and xDSL, was introduced in Release 7 by working together with Telecoms & Internet Converged Services & Protocols for Advanced Networks (TISPAN) and PacketCable. More recently, Long Term Evolution (LTE) and System Architecture Evolution (SAE) were adopted in Release 8, and enhanced in the following releases such as Release 9 and Release 10. The ongoing work for Release 11 focuses on the advanced IP interconnection of services.

One important feature of IMS is that it introduces a horizontal control layer, which isolates the service layer from the access layer. The control layer performs the multimedia session establishment, modification, control and termination by using SIP.

SIP [9], specified by Internet Engineering Task Force (IETF), is a signaling protocol for establishing, modifying and terminating communication sessions in IP network, such as voice and video calls over IP. It was adopted by 3GPP in IMS to provide mobile and fixed multimedia services over heterogeneous networks. The flexibility of SIP eases the service creation procedure.

Within the IMS control layer, one of the key innovations is the separation of a SIP server into a Proxy-Call Session Control Function (P-CSCF), Serving-Call Session Control Function (S-CSCF), and Interrogating-Call Session Control Function (I-CSCF).

- P-CSCF: the first entry point to the IMS domain and serves as the outbound proxy server for IMS terminals. It can be located either in the home network of the IMS operator or in the visited network where the IMS terminal is currently roaming.

- I-CSCF: the entry point within the IMS operator's network for other operators. It acts as an inbound SIP proxy server for selecting the appropriate S-CSCF to the user by querying to HSS during the IMS registration process.
- S-CSCF: the key component of IMS for session controlling and service invocation. It is always located in home network. It uses Diameter Cx and Dx interfaces to download the user profiles from HSS. It is not only responsible for the registration, but also acts as a redirect server for routing the mobile originated or terminated request. Its iFC-based processing logic enables it to trigger IMS services.

Home Subscriber Server (HSS) is another important component for the IMS core network. HSS is a central database, which is used to store the IMS users' profiles. It contains the subscription-related information and performs the authentication and authorization of the user. It also provides the subscriber's location and IP information.

Figure 2-2 illustrates the functional architecture of IMS. Thanks to this horizontally integrated service architecture, IMS enables the interactions among a variety of communication services. Moreover, this architecture also enables the fast development of services, and supports the share and reuse of common functions such as authentication, presence, and charging across different services.

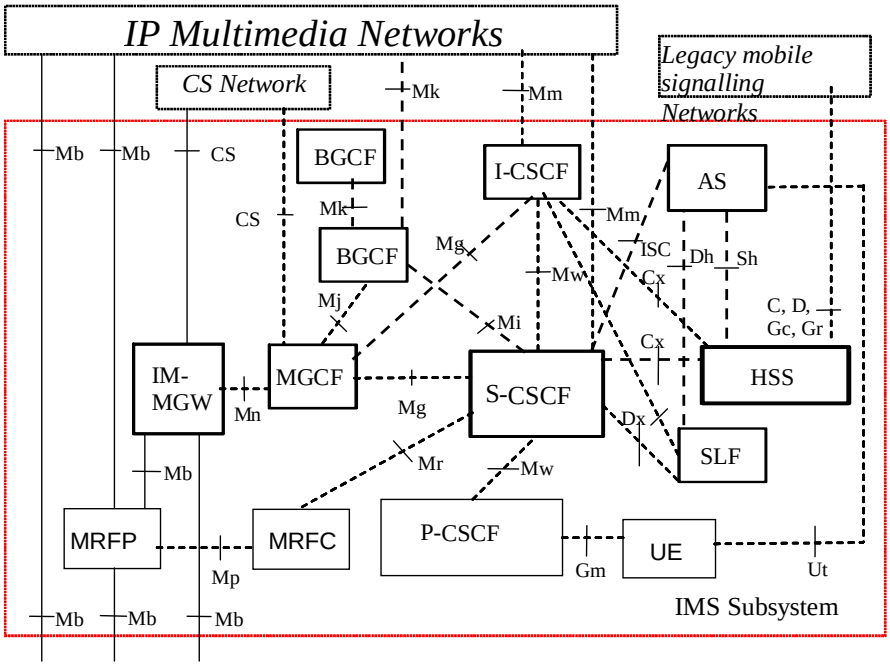


Figure 2-2: 3GPP IMS architecture overview [Source from 3GPP TS 23.228 version 8.6.0 Release 8]

Another great feature within IMS service architecture is that: a capabilities layer is defined, which contains a set of secured and combinable application-independent building blocks that offer generic functionalities to support various multimedia applications. Such building blocks are called Service Capabilities by 3GPP [10], and Enablers by OMA. Some well-known capabilities can be cited as Presence [11], Conferencing [12], Messaging [13], Location, Network Address Book (NAB), etc. The existence of such a layer brings a new perspective for service creation, i.e., a service capability provides a predefined generic functionality, so that a new and more revenue generating service can be created by combining multiple capabilities.

2.1.2 Next Generation Internet (NGI)

In the past 30 years, Internet has gained a tremendous success and greatly revolutionised our social habits and business ways. It has evolved from a network of computers and information into a network of people. Nowadays, there are 1.5 billion people using Internet, and it is predicted that, just in a few years, over 4 billion users and billions of devices and objects will be connected anywhere and anytime. Before this comes to reality, Internet must undergo important changes. Internet now is starting to face some technical limitations such as mobility, scalability, wireless generalization, broadband evolution, service multiplicity, reliability, availability, manageability, security and QoS, as well as other societal, economical and business related aspects. Consequently, the concept of NGI was born to resolve these weaknesses in current IP networks. It is always labelled as “Internet 2” or “Beyond IP”. In these years, a great amount of discussions and research efforts have been done around the world by many initiatives. To name few examples, Global Environment for Network Innovations (GENI) in USA [14], Future Internet Research & Experimentation (FIRE) in Europe [15], Architecture Design Project for New Generation Network (AKARI) in Japan [16] and others [17]-[19]. Currently, most of NGI networks are being designed relying on two main approaches: evolutionary approach and clean slate approach.

- Evolutionary approach: the system is moved from one state to another with incremental patches.
- Clean slate approach: the system is redesigned from scratch to offer improved abstractions and/or performance, while providing similar functionalities based on new core principles.

The basic idea behind clean slate approach is “*temporarily ignoring the strong design constraints imposed by the exiting Internet in order to better understand requirements and explore potential benefits*” [20]. Nowadays, there are several respective approaches proposed by the different organizations. Firstly, in 2005 GENI was announced in US with the objective to provide an infrastructure on which new architectures can be implemented and tested. Meanwhile, the Future InternNet Design (FIND) program was

launched for simulating the architecture proposals to be implemented in GENI. The European Commission is involved in the future internet research from the Seventh Framework Programme (FP7) which began in January 2008. 4WARD [21] is the flagship clean slate project in FP7. Korea also sponsors some future Internet clean slate architecture work under the aegis of a Future Internet Forum (http://fif.kr).

Such clean slate approach does not necessarily mean that all the existing protocols and technologies should be replaced. Contrarily, it can be considered to be complementary to the evolutionary approach for moving towards the Internet to Future Networks.

2.1.3 Convergence of NGN and NGI to Future Networks

Nowadays, one trend for the networking evolution is the convergence of Internet and Telecom for bridging the openness and flexibility of Internet and the fidelity and trustworthiness of Telecom. In fact, along with the rapid evolutions in Telecom and Internet, the new technologies blur the boundaries between Telecom and Internet. Just as what we introduced for NGN, it adopts the IP technologies for evolving the Telecom system to satisfy the next generation service requirements, and integrates the IT services, especially the Web based services, to the intrinsic Telecom services for generating innovative services to their subscribers. For instance, besides the convergence example of IMS, the concept of Telco 2.0 has emerged in these years to describe how the principles of Web 2.0 are being leveraged within Telecom world. On the other hand, internet has expanded rapidly in these years. It is on its way to handling all of our emerging and new service needs, especially after the development of new technologies that enable voice transmission via packet networks (mainly on an IP basis). Even there are still some weaknesses, NGI is proposed for addressing these weaknesses in current Internet.

E. Mikoczy et al. compared NGN and NGI requirements for identifying their common interests and differences in [22]. Some common requirements for these two kinds of networks are listed as the followings:

- Converged distributed architecture
- Addressing and identity
- Security and privacy
- End-to-end QoS and QoE
- Mobility
- Content delivery

- Management aspects

- Protocol stack complexity, future proof protocols

- Openness, interconnection and third party applications

- Innovative and personalized services

Future Network (FN) is a general term for the upcoming networks. For supporting the requirements of future emerging applications and users, FN is anticipated to provide futuristic functionalities beyond the limitations of the current IP based network. ITU-T started a focus group in 2009 for analyzing the requirements, principles, and concepts for FNs [23]. The main concerns for the FN research within ITU-T include: scalability/ubiquity, security/robustness, mobility, heterogeneity, QoS, reconfigurability, context-aware, manageability, data-centric, network virtualization, and economics, etc. Both NGN and NGI are considered as valid candidates for FNs. Regarding the commonalities and differences of these two kinds of networks and their evolution trends, we believe that the convergence of NGN and NGI will change both IT and Telecom worlds, probably achieve the goal of providing the best of both two worlds, and finally evolve towards the FNs. In this thesis, we analyze such a convergence from the view point of services as illustrated in the following sections.

2.2 Next generation services

As shown in above introduced NGN architectures, the separation of services from the network transport and control layers is a key cornerstone of NGN. And this strategy enables the service providers to provide some NGN services which proclaim to help them to differentiate themselves from their competitors, achieve operational cost reductions, and strategically position themselves in the competitive market, and thus retain and/or attract more lucrative customers for guaranteeing their revenues.

Several characteristics of the NGN services were introduced in [24]:

- *Ubiquitous, real-time, multimedia communications*
- *More "personal intelligence" distributed throughout the network*
- *More "network intelligence" distributed throughout the network.*
- *More simplicity for users.*
- *Personal service customization and management.*
- *Intelligent information management.*

NGI is being designed to overcome current limitations and address emerging trends as we illustrated in upper part. With these new characteristics for NGI, some specific trends and characteristics can be identified for the services provided by Future Internets as listed in the followings [25].

- *Nomadic access becomes the norm*
- *From ad-hoc usage to an always-on experience*
- *From static information access to dynamic and time-critical services*
- *From free-of-charge access to value-based transactions*
- *Creating trust*
- *Simplification of the user interface*
- *Increased machine-to-machine (M2M) interaction*

2.3 Next generation service provisioning

Nowadays, service market is customer-driven, and the competition among service providers is becoming more and more intensive. To keep their roles in the service market, service providers and operators have to take advantage of their core competencies, find the innovative ways to provide new services which satisfy customers' more and more fastidious requirements, meanwhile, reduce the operational cost. In this context, service composition, which reuses existing services and network resources for creating a novel service, is recognized as an important approach for next generation provisioning.

Most of service composition solutions are based on Service-Oriented Architecture (SOA), which plays a significant role for the prosperity of Web services. This concept is consistent with NGNs in that the service layer is entirely separated from the transport layer and control layer, thus ensures the autonomy of each plane and enables their independent development and deployment; Further, this separation allows re-implementing the existing services into a new and enhanced service, regardless of the types of services – IMS based services, Legacy services or Web services.

Service composition is related to the positive interaction among services, which enables the share and reuse of service building blocks by various services in order to offer integrated services to the users. These reusable service building blocks, called Service Capabilities or Service Enablers in some NGN networks, are the modular and self contained service components that can be either used independently, or shared by services in order to offer enriched services to users. The inter-working among autonomous services consists

of enabling the composition of different service components, managing the interoperability between these components, and supporting the simple introduction of innovative services.

The interoperability and cooperation between different service components in the service layer allows: (1) to enrich and personalize the application servers based on the user needs, preferences and contexts provided by the service capabilities; (2) to prevent the overlap and redundancy of the same functionalities offered by service capabilities and implemented by different application servers; (3) to reduce the time-to-market of application servers; (4) to support the easy and low-cost integration and deployment of application servers for service providers; and (5) to manage the reuse of user information and preference applied in different application servers. During the service composition process, all kinds of service inter-working should be independent of the underlying control functionalities.

To achieve the aim of service composition, it requires some additional mechanisms for managing the interoperation and cooperation of services. Firstly, it requires an intelligent mechanism to manage the service composition. That is to say, it should control the access of different application severs to the service capabilities and manage the service interactions between the application servers and service capabilities by respecting the confidentiality of user context. Meanwhile, it should be able to efficiently route the message to the most appropriate component without the human intervention. Some non-functional features, such as the QoS, privacy, and charging, are also needed to be taken into account, since the effectiveness of a composite application server is not only highly dependent on the functionalities combination, but also on user's preference and the context (i.e. the network conditions and constraints). Secondly, the composition of services essentially relies on the prerequisite that the service capabilities which are reused in the service composition have been made available for the composite application beforehand. This prerequisite is related to the paradigm of service exposure, whose aim is to make the exiting autonomous services to be understandable and interoperable in the large-scale network without the consideration of the underlying heterogeneities. This prerequisite can be achieved by reusing some open service interfaces (open APIs) which are consistent with one essential attribute of NGN - openness: an open development environment based on APIs will enable service providers, third party application developers, and potentially end-users to create and introduce applications quickly and seamlessly. This will speed up the introduction of new services by giving service providers more control on the service introduction process and allowing the reuse of the existing application components. It will also provide the opportunities for creating and delivering services to the broader audience.

In the following subsections, we present an overview of the relevant work from two aspects: Service Composition, which focuses on the service capabilities interoperation management, and Service Exposure which focus on making available of the underlying autonomous services.

2.3.1 Service composition

Creating value-added services by reusing the existing service components, which is well known as service composition, has become a key aspect for service delivery both in IT and Telecom worlds. In the last decade, a great deal of work has been carried by both industrial and academic organizations, and diverse solutions have been proposed for catering to the different network and service requirements. Most of current proposed solutions are based on SOA.

2.3.1.1 Service Oriented Architecture (SOA)

SOA is a widely adopted architecture paradigm that enables service providers to publish their services and service consumers to discover them. It facilitates the development of applications by combining the loosely coupled services across independent entities (or organizations). The general architecture of SOA is as shown in Figure 2-3.

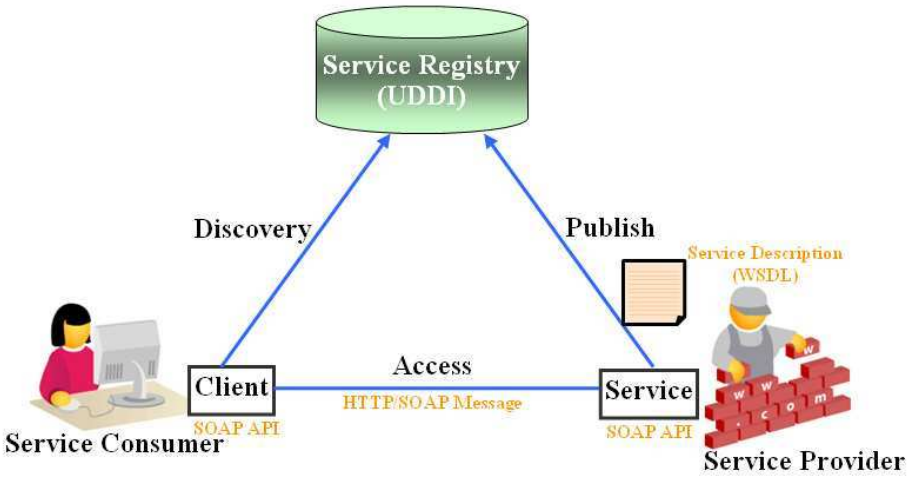


Figure 2-3: SOA basic architecture

As illustrated in Figure 2-3, there are three roles in the SOA paradigm: service consumer, service provider and common registry. The service provider creates a service and provides its functional description to a common registry. The registry is a central entity from service providers and service consumers' points of view, though it might be physically distributed over several platforms. The service

consumers can search in that registry to find their desired services. They can subsequently invoke these services.

Generally, the SOA paradigm is based on a set of principles. For instance, Thomas Erl defined 9 principles for SOA [26]:

(1) Standardized service contract: services adhere to a communications agreement, as defined collectively by one or more service-description documents.

(2) Service loose coupling: services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.

(3) Service abstraction: beyond descriptions in the service contract, services hide logic from the outside world.

(4) Service reusability: logic is divided into services with the intention of promoting reuse.

(5) Service autonomy: services have control over the logic they encapsulate.

(6) Service granularity: a design consideration to provide optimal scope and right granular level of the business functionality in a service operation.

(7) Service statelessness: services minimize resource consumption by deferring the management of state information when necessary.

(8) Service discoverability: services are supplemented with communicative metadata by which they can be effectively discovered and interpreted.

(9) Service composability: services are effective composition participants, regardless of the size and complexity of the composition.

These principles enable the distinct services to interoperate with each other, regardless of their underlying implementation issues, thus provide a new way for service creation - service composition.

2.3.1.2 Service composition for Web services

2.3.1.2.1 Web services

A Web service is defined [W3C, 2004] as "*a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (especially WSDL). Other systems interact with the Web services in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other*

Web-related standards". This definition highlights the main technologies used within Web Service Architecture (WSA) such as WSDL, Simple Object Access Protocol (SOAP), HTTP and Extensible Markup Language (XML). Moreover, it also implies the facilities for publishing and discovering services. UDDI is currently the main technology to provide these facilities through either centralized, P2P, or even hybrid approaches.

To invoke Web services, user can use either Remote Procedure Call (RPC) based approach, SOAP based approach, or REST based approach.

RPC is an early use style for Web services. RPC Web services mainly present a distributed function (or method) call interface, thus they are recognized as "operation oriented". This use style is sometime criticized for not being loosely coupled, since it is generally implemented by mapping services directly to language-specific functions or method calls. Some other approaches with the similar use style of RPC can be cited as Object Management Group's Common Object Requestor Broker Architecture (CORBA), Microsoft's RPC or .Net Remoting and Sun Microsystems' Java/Remote Method Invocation (RMI).

SOAP based Web services, are currently the most widely supported use style in the IT world, due to their loose coupling feature. In the SOAP based architecture, the basic communication unit is a message rather than an operation. This kind of services is thus often referred to be "message oriented". That is to say, the service providers create services, describe these services' interfaces using WSDL, and publish them to a common registry relying on UDDI. Then service developers use a discovery interface to discover services within the UDDI registry, and invoke them by creating a SOAP message. This approach provides a common way to describe, publish, and invoke services, hides the underlying implementation details from developers, thus it greatly facilitates the service invocation and new service creation.

REST stands for Representational State Transfer. It was introduced and defined by Roy Fielding in his Ph.D dissertation [27] as a software architectural style. It was characterized by representing any piece of information that an application can access as a "Resource", and each "Resource" is identified by using a Uniform Resource Identifier (URI) or Uniform Resource Locator (URL). It also leverages the HTTP methods (particularly GET, POST, PUT, DELETE methods) to constraint the interfaces to a set of well-known, standard operations for making the service invocation as simple as making an HTTP request. As the focus of REST is on interacting with stateful resources, rather than with messages or operations, it is labeled as "resource-oriented". An architecture based on REST can also use some WSDL-like description languages to describe its resources. For instances, Sun Microsystems published Web Application Description Language (WADL) for providing the facilities to describe resources, W3C has added resource description support to the WSDL 2.0 specification.

2.3.1.2.2 Web service composition using SOA

Service composition is defined as the process of combining a set of services or capabilities to provide a new application. This process is also known as “Mashup” for Web services. The SOA based service composition can be mainly classified into three main categories: static service composition, automatic service composition, and semi-automatic service composition. In the following subsections, we detail each approach for the Web services composition and analyze the related technologies.

2.3.1.2.2.1 Static service composition for Web services

Static service composition is the most basic approach to create the composite services. It means a combination of two or more services by a developer at the design time using programming language through some Integrated Development Environment (IDE), such as Eclipse IDE. This kind of service composition generally targets the professional developers. The term “static” implies that once a service is created, it cannot be modified by users any more. SOAP and REST based architectures are currently the most used ones that enable the static service composition.

From the technical perspective, this kind of composition mainly relies on using APIs, which are provided by the existing services’ providers for enabling developers to discover and reuse these services. Some examples for the SOA based APIs can be cited as Java SOAP Client, PHP SOAP for the backend APIs, and IBM Dojo toolkit extension and jQuery SOAP client for the front-end APIs.

On the one hand, the static service composition enables services to be deeply integrated with each other. The stability and quality of services can be assured. The services may work fine as long as the Web service environment and service components does not, or only rarely change. Mircosoft Biztalk and BEA WebLogic are some examples of static service composition engines. On the other hand, it implies a long time to market. That is because it needs to collect the user’s need firstly, then discover and choose the component services, and perform the development process by developers manually. Moreover, once the application is created and deployed, users cannot modify it any more. If required any necessary modification, the application needs the intervention of the developers for reengineering, re-deploying and re-testing it.

2.3.1.2.2.2 Automatic service composition for Web services

Automatic service composition is characterized by creating a composite service automatically at runtime according to a user request, which can be based on natural language. It mainly relies on Semantic Web and the natural language interpretation technologies. Semantic Web was originally envisioned to be a system that enables machines to understand and respond to complex human requests based on their meanings. To

achieve this goal, the services must be described semantically. Semantic Markup for Web Services (OWL-S), Web Service Semantics (WSDL-S), and Web Service Modeling Language (WSML) are some well-known semantic languages for service descriptions. One example for automatic service composition is the Automatic Composition Engine (ACE) defined in SPICE project [28]. In the SPICS project, the available services are described in a semantic based language called SPATEL (Spice Advanced Service Description Language for Telecommunication Services). SPATEL allows specifying various services in a platform independent manner, along with semantic annotations for functional and non-functional properties. The automatic service composition process is performed by four basic components (Natural Language Processing, Composition Factory, Property Aggregator and Matcher) residing in ACE as illustrated in Figure 2-4.

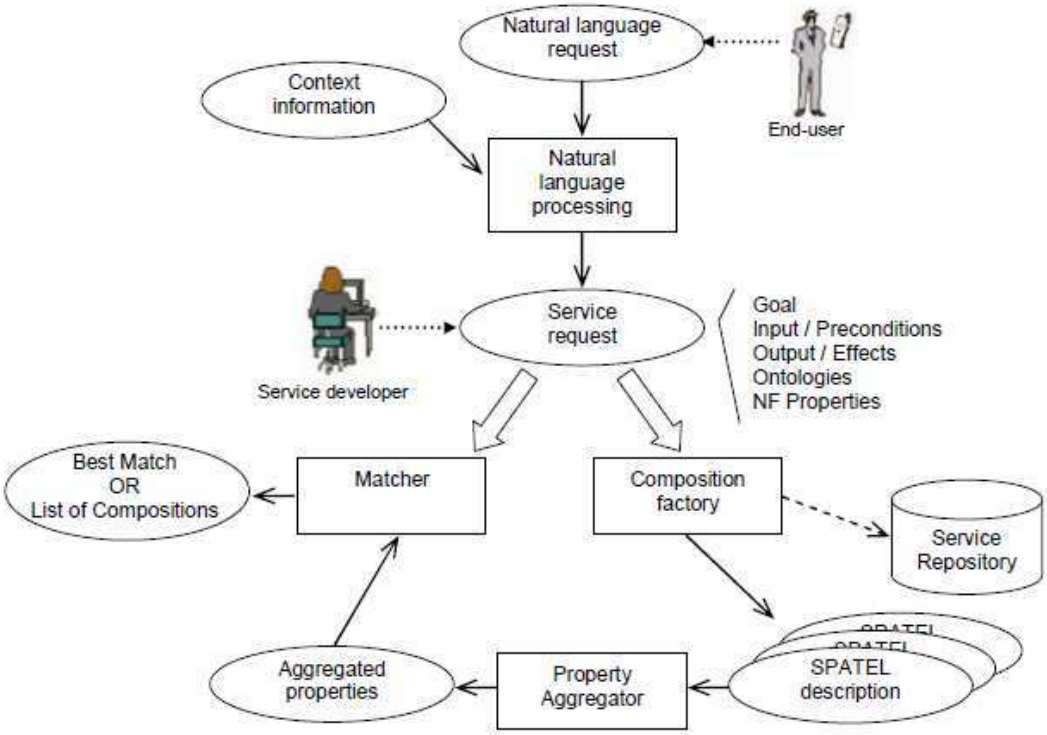


Figure 2-4: SPICE ACE architecture [Source from [28]]

Automatic service composition is definitely an ideal approach for creating services, since it enables users to dynamically obtain a new service responding to their spontaneous needs, which may be expressed by using the natural language. However, automatic service composition still suffers from research issues that prevent it from being a mature solution for providing customized services to users. Some of such issues can be viewed as how to clarify the natural language ambiguity, how to provide a comprehensive dictionary of concepts that enable the description of all existing services, and how to appropriately infer the service

logic according to users' natural language input, etc. All of these issues give rise to the inaccuracy of the automatic service composition.

2.3.1.2.2.3 Semi-automatic service composition for Web services

Semi-automatic service composition includes all the mechanisms that provide the assistance to users during the service creation process at design time or at runtime. It makes the service creation process easier and faster, or even improving service lifecycle by providing the automatic adaptation. This kind of service compositions ranges from incrementally assisting a user at each step of the service composition process (e.g. the Social Composer [29]) to inferring the concrete workflow according to a predefined template or pattern by associating the runtime user and service environment information (e.g. eFlow [30]).

Since the semi-automatic service composition generally needs the intervention from users or developers, consequently graphical service creation environments are widely used for facilitating the interaction between backend service composition engine and users/developers. According to the different kinds of service creation environments, semi-automatic service composition can be further classified into three main sub-categories: IDE based service composition, Web browser based, and widget based.

In order to speed-up the service creation process for developers and advanced users, several graphical frameworks such as JBPM, Eclipse BPEL Designer, and jABC (Java Application Building Center) have emerged. These frameworks are all based on process definition languages such as BPEL (Business Process Execution Language), JPDL (JBPM Process Definition Language) and SLG (Service Logic Graphs). The creation tools are thus based on intuitive graphical interfaces that enable assembling the black boxes, according to the logic of the service needed by user. The logic is defined by wiring the black boxes and mapping the outputs and the inputs as shown in Figure 2-5. Other operations like conditions and loops can be added graphically as well, and the component services that map to the black boxes need to be discovered by developer manually. This category of service compositions are based on IDEs, thus they are generally targeting the professional developers or advanced users.

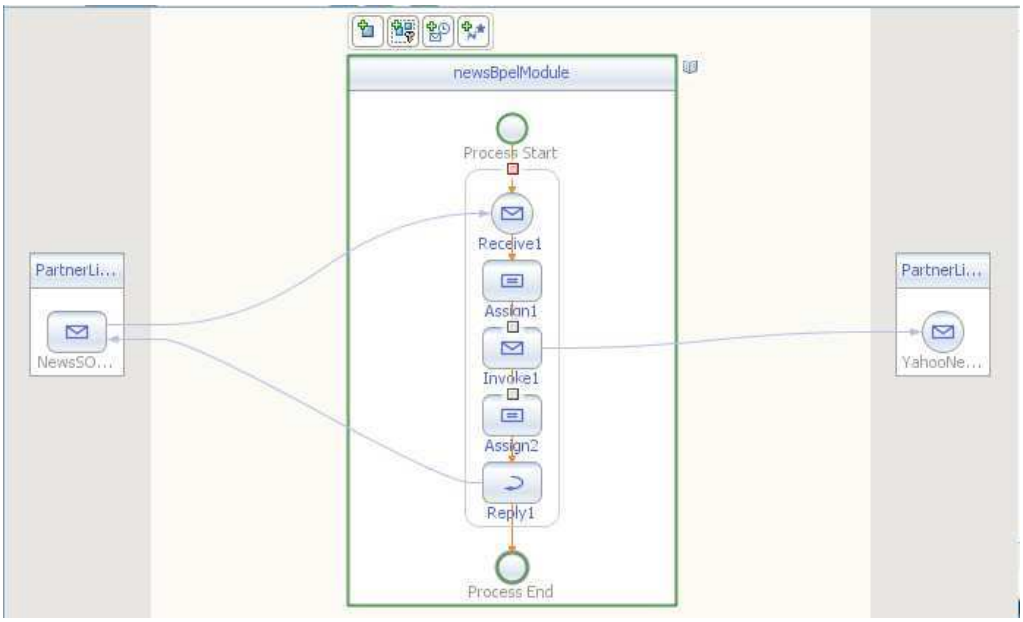


Figure 2-5: BPEL designer screenshot

Other frameworks attempt to get closer to users by implementing the composition capability directly at the Web browser level, either as a web application or as a browser plugin. Examples of this kind of frameworks are Yahoo Pipe, MARMITE and OPENMashup. Compared to the IDE based frameworks, the Web browser based frameworks simplify the services creation process to certain extent. The service creation environment is more user friendly than the IDE based environment. For example, Yahoo Pipe provides a graphical user interface for building data mashups that aggregate web feeds, web pages, and other services, creating Web-based applications from various sources, and publishing those applications, as shown in Figure 2-6. A typical service creation process is as the followings: after a user enters to the Pipe editor, she/he can drag the desired modules from the library and wire these selected modules together. Then the user can edit the modules by adding data, locations, numbers, texts or URLs as the input. After the user validates her/his pipe, and gives the name and a short description of the pipe, a pipe is created and can be published for being shared with others. However, this kind of service compositions is still not user-friendly enough for the ordinary users, and can be considered as the one targeting the advanced users. That is because they are still based on defining a flowchart by using different and probably complex operations, and require users to understand at least URL, input, output, web service invocation, condition, loops, etc.

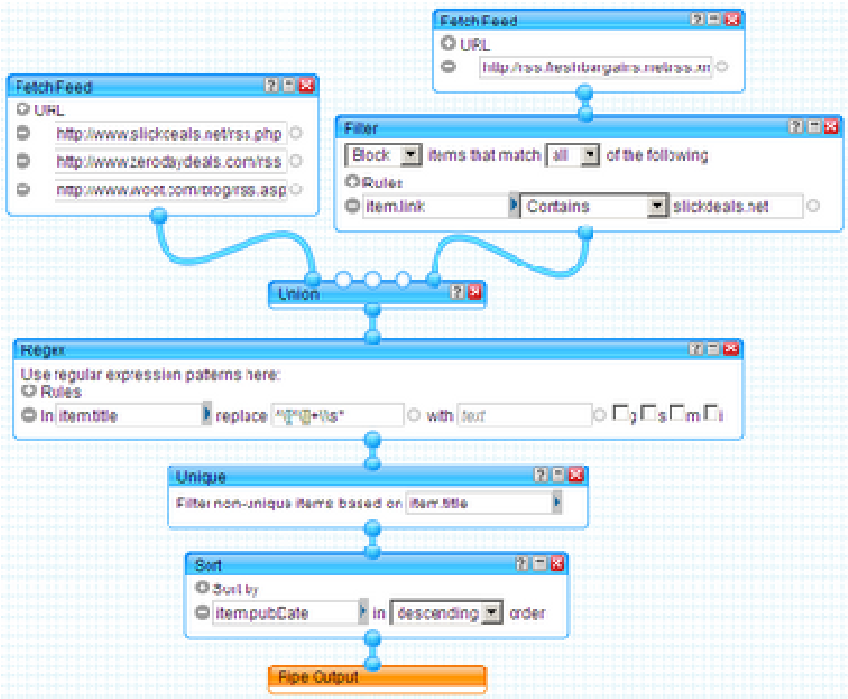


Figure 2-6: Yahoo Pipe screenShot

A widget [31] is a portable chunk of content(s) or interactive functionality such as showing users the latest news, the current weather, and a language translator. It is generally embedded within a page of HTML (e.g. Web page), or runs on the desktop (e.g. Windows or Mac OS), or even on the mobile device. Examples of widgets include: Yahoo widget, Google gadget, Apple dashboard widget, and Facebook widget. Because they are so small and smart, they are widely used by bloggers, social network users, owners of personal web sites, as well as they exist in home page sites such as iGoogle. Nowadays, widgets become more and more interactive. They can not only present the resource or functionality of the backend application or data, but also trend to be able to collaborate with other widgets for forming a new composite service at runtime. Some platforms also empower user to create a new service by chaining individual widgets together, e.g. the EzWeb widget chaining tool [32] and the widget linker [33]. EZWEB is a widget container web application. The particularity of this framework by comparing with other widget aggregators is that it enables the end-user to chain widgets between each others by mapping compatible output of a widget with input of others as shown in Figure 2-7. This enables the end-user to add functionalities of one widget to another.

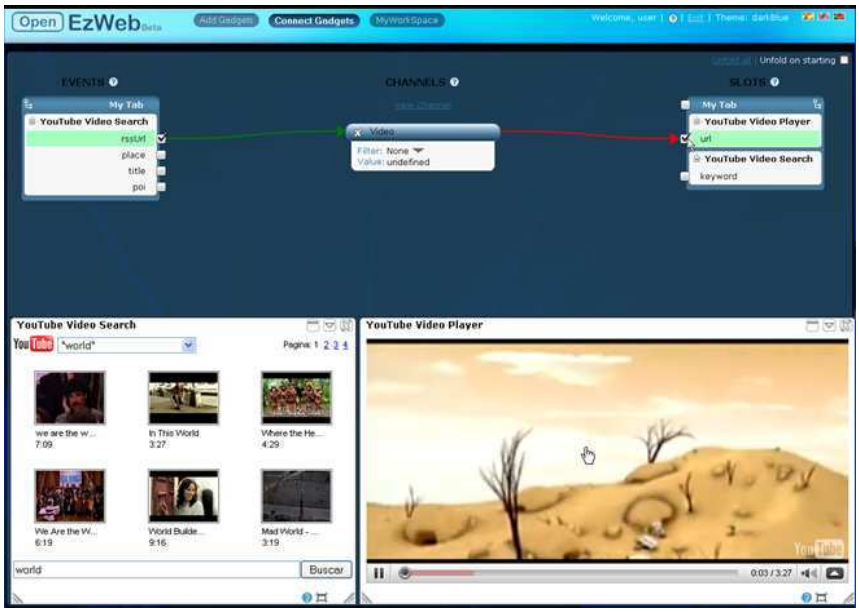


Figure 2-7: Ezweb - creating new service by chaining widgets screenshot

In [34], the authors proposed another more user-centric widget based service exposure and service creation tool - inter-widget communication tool, as shown in Figure 2-8. This inter-widget communication tool can detect the input/output matching, create links between loaded widgets automatically at runtime, and display them to users. Consequently, a composite service can be created just by a simple drag-and-drop method (i.e. user just needs to drag a widget and drop it upon another widget), and the additional functionalities are added to the existing widgets automatically, as long as the end user loads widget into his service environment without any input/output matching.

Figure 2-8: Widget linker screenshot

2.3.1.3 Service composition for Telecom services

In our previous research, for Telecom service composition, we mainly focused on the IMS based solutions. As introduced above, IMS was introduced by 3GPP to provide IP-based multimedia services. It defines an overlay architecture that enables the convergence of voice, video, data, and mobile network technology over an IP-based infrastructure on top of various packet switched and legacy networks.

IMS differs from other IP-based environments by providing a set of secure and combinable application independent building blocks. These building blocks offer generic functionalities to support various multimedia applications. Among these building blocks, commonly known as Service Capabilities or Enablers, we can mention Presence service, Group Management service, Messaging service and Conferencing service. The existence of such a layer brings a new perspective for service creation in Telecom domain, i.e., a service capability provides a predefined functionality by itself, but a new and more revenue generating service is created by combining multiple service capabilities.

2.3.1.3.1 Original service delivery mechanism in IMS

In IMS service architecture, one of the great features is the possibility to chain services on the SIP signalling path, which is well-known as *Service Chaining*. In this concept, for delivering multiple applications within a single session, the session is routed sequentially among the different application servers. When the session passes, each application server performs its role in its turn, which is typically acting as *Back-to-Back User Agent (B2BUA)*, *SIP Proxy*, or *Terminating User Agent*.

For the current IMS architecture, most of the service chaining is done by S-CSCF (Serving-Call Session Control Function) as shown in Figure 2-9, in which all the relevant capabilities are chained into SIP signalling path through the predefined iFC. That's to say, once S-CSCF receives a request from user, a User Profile as illustrated in Figure 2-10 is retrieved from HSS. Such User Profile contains an ordered list of iFC that is a couple of a Boolean expression called Trigger Point, and an Application Sever address associated to this Trigger Point. The iFC indicates in each condition (defined in Trigger Point) which Application Server should be invoked. The conditions in Trigger Point may be made on the SIP method, the SIP headers and their content, the originating or terminating session cases and on the session description in the body part of the SIP message.

S-CSCF then verifies each iFC following the priority order. Once a Trigger Point in iFC is matched, S-CSCF dispatches the SIP message to the corresponding Application Server (AS). After processing the request, AS sends back the response to S-CSCF, and S-CSCF continues to analyze the unevaluated iFCs

and determines whether to route the SIP request to next AS, or terminate the service chaining and send back the response to an end user.

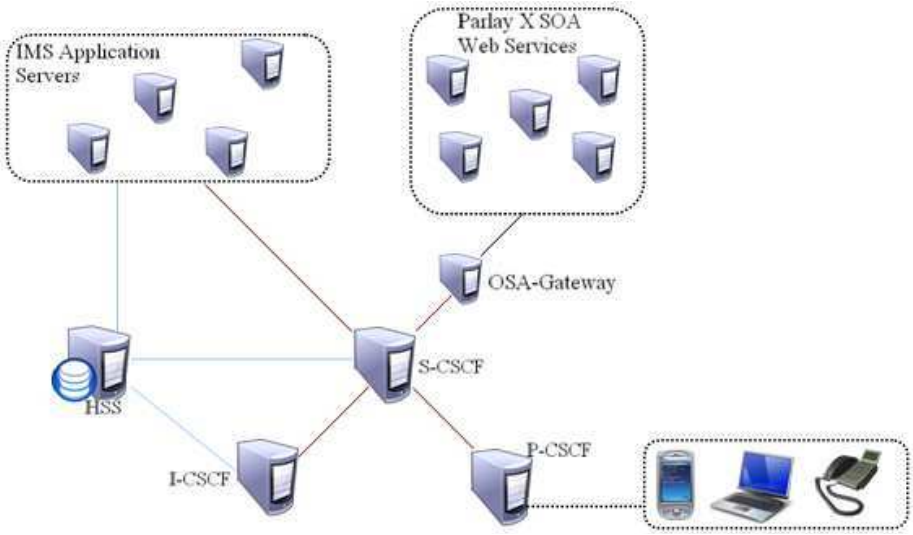


Figure 2-9: Simplified IMS architecture

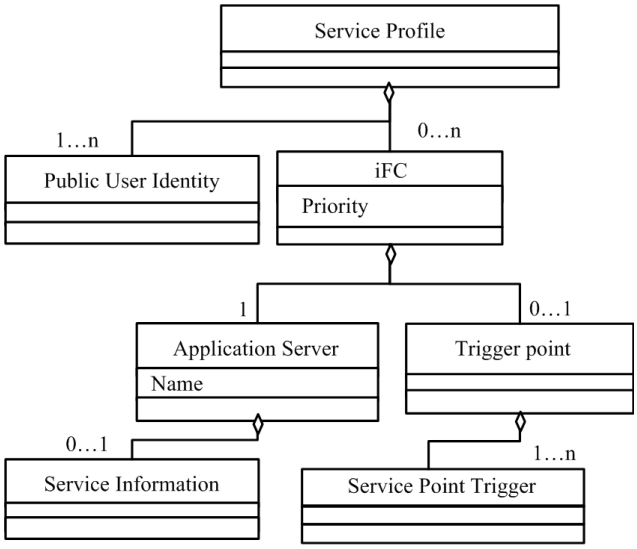


Figure 2-10: IMS user profile

The iFC is a static, powerful but simple mechanism that allows chaining services and service capabilities based on predefined conditions and priorities. However, it is not efficient enough for complex service composition and brings some issues: (1) within this iFC based service chaining, all the services are invoked sequentially, that will incur the efficiency limitation in a case when some capabilities have no interdependence among them and can be executed in parallel. (2) It also brings some issues for S-CSCF itself, e.g. it may cause a heavy processing by S-CSCF in addition to other tasks it already has. (3) Multiple

interactions between S-CSCF and ASs may lead to increased end-to-end latency for service delivery. (4) The limitations on flexibility and extensibility (e.g. convergence with other types of services outside the IMS scope) are some unresolved problems faced by this mechanism.

Addressing the above challenges, a more advanced mechanism for complementing the S-CSCF based solution is considered to be necessary for next generation service delivery. Different standardization organizations (e.g. 3GPP, OMA) or projects (e.g. SPICE, EXOTICUS, SERVERY, etc.) initiated to propose solutions for resolving some of the specific issues for service composition in NGN. According to their proposals, adding a new entity for handling the service composition issues becomes an explicit consensus for the service composition evolution.

2.3.1.3.2 OMA Service Environment (OSE) based solution

The OMA Service Environment (OSE) [35] architecture, as shown in Figure 2-11, was defined by OMA as an open platform providing a common structure and rule set for the OMA specified enablers. It permits efficient and reliable creation, deployment, and management of services in multi-operator and multi-vendor environments.

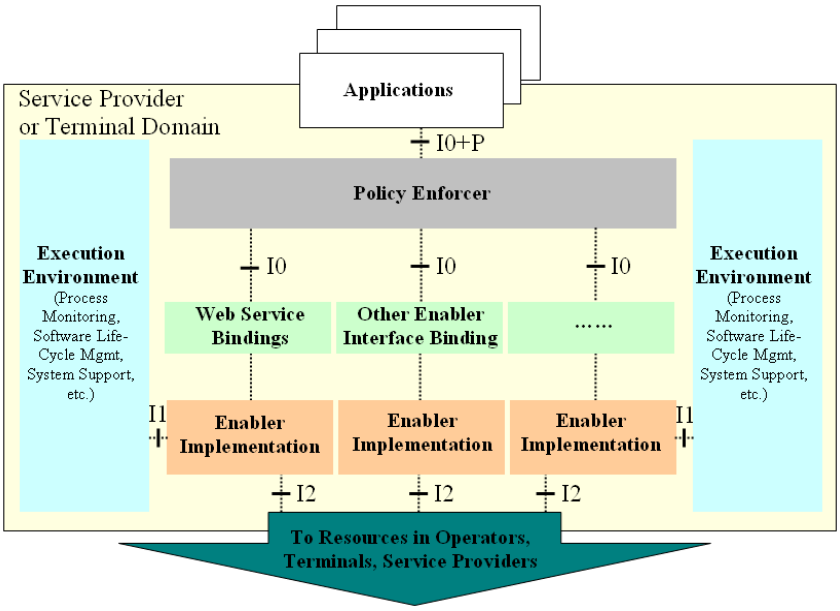


Figure 2-11: OSE architecture

In the OSE architecture, the capabilities as well as the network technologies can be abstracted by the enablers. Applications do not contact directly with capabilities or underlying networks, as a result they will be affected neither by the evolution of the capabilities, nor by the variations of underlying network. Enabler Interface Binding provides the specific formats for accessing enablers. Policy Enforcer provides a policy-

based management mechanism for protecting the service provider's resources from unauthorized requests and for managing the uses of these requests through appropriate charging, logging and enforcement of user privacy or preference. It also provides a generic mechanism for exposing resource to third parties.

However, in this approach the cooperation among services is only limited to OMA enablers. Besides, it doesn't specify how to enable sharing and reusing services capabilities by different application servers. The extensibility of the rapid introduction of new service enablers and applications remains an important issue.

2.3.1.3.3 Service Capability Interaction Management (SCIM) based solution

Derived from the "service interactions management" issue, SCIM was proposed by 3GPP at the beginning of IMS. As shown in Figure 2-12, it was introduced as a standalone entity located between S-CSCF and application servers (ASs) for managing interactions among ASs. However, years later, the specification of SCIM is still limited to few sentences [36].

Using SCIM to compose service capabilities requires an individual access to each capability. Consequently, we need a detailed knowledge of all these services capabilities to be invoked and the relationship between them. It also requires an interface between SCIM and application servers layer to support the identification of each service capability.

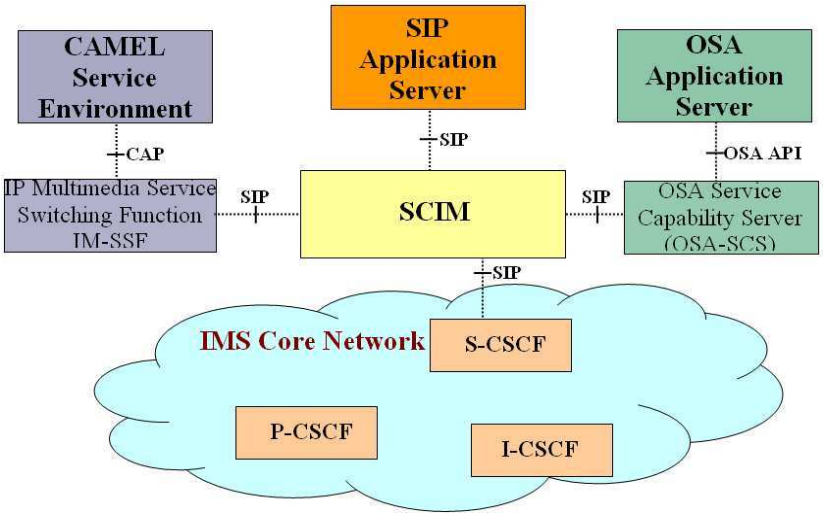


Figure 2-12: SCIM within IMS environment

Since the limitation of the SCIM specifications in IMS standards, and there is no consensus for service composition mechanism within IMS until now, some research work tried to define the SCIM architecture and features (e.g. SCIM defined by Ubiquity [36], WCS SCIM provided by Convergin [37], SCIM provided by Alcatel-Lucent). These SCIMs could be a service interaction manager, or a proxy for service switching, or a protocol translation gateway. In [38], authors proposed an architecture, where they put on

the SCIM the responsibility of managing the interactions between composed services and enablers. To provide the ability to customize services according to end-user's preferences, in [39] authors proposed an evolution of SCIM architecture by integrating it into the S-CSCF entity (IMS core session layer). This enables selection and orchestration of enablers according to user personalized data. In [40], we also proposed a SCIM based service composition model in which SCIM acts as an intermediate between the network control layer and the service layer for enabling the reuse of service capabilities. The advantages of this proposed SCIM based service composition model include: it simplifies the iFC execution logic, releases the extra processing responsibilities from S-CSCF by shifting the capabilities interaction handling task from S-CSCF to a standalone entity; it provides a unified capabilities access interface and completely isolates the application layer from IMS core network; it does not need a great modification to the already defined substantial core network architecture, since SCIM is based on SIP and can be seen by S-CSCF as a normal SIP application server.

2.3.1.3.4 Other service composition solutions in Telecom

There are also some solutions provided by the projects (e.g. SPICE and SERVERY projects) or by the industrial or academic organizations (e.g., Ecrisson's Application Router, Fokus's Open Communication Server, BEA's WebLogic Communication Platform, etc.)

SPICE project aims at designing, developing and providing operation efficient and innovative mobile service creation/execution platforms for networks beyond 3G. It defines a user centric service creation environment in which a set of service enablers, covering network related function (e.g. presence), as well as knowledge based enablers (e.g. profile, context, etc.) are made available to the service creation by both professional and non-professional users, and enables them to create their service easily by providing users friendly service creation tools (e.g. natural language service composer).

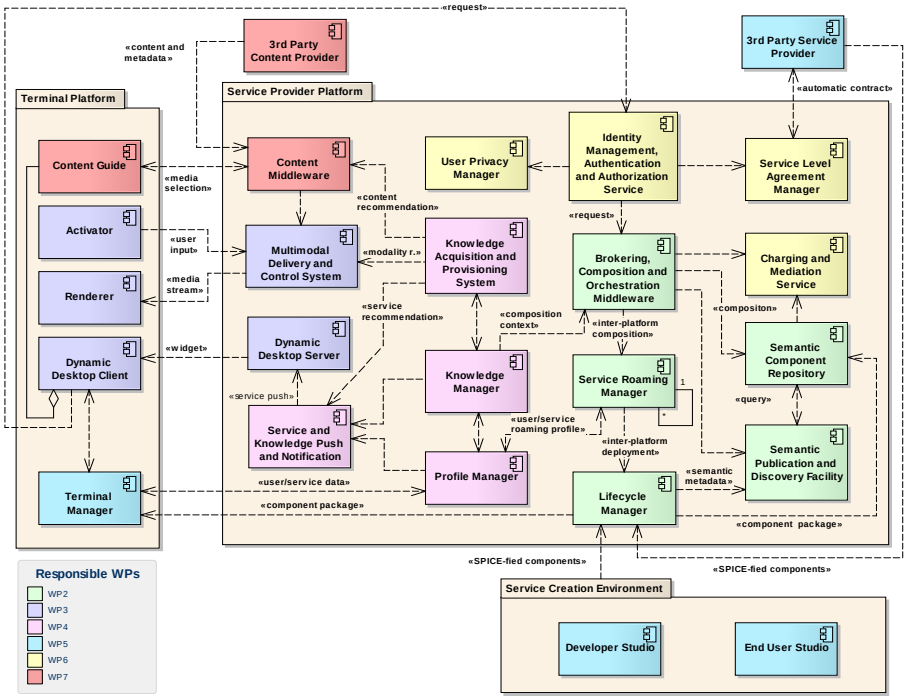


Figure 2-13: SPICE functional architecture [Source from SPICE deliverables]

SERVERY project defines a Service Creation Environment, which enables the Telecom based services to be reused by using Web technologies. Three kinds of service creation tools, including the Eclipse likes IDE for professional developers, the mashup based service creation environment for service creation amateurs, and the natural language composer for the non professional users, are provided. It also provides an IMS/Web converged environment, which is named as Web Multimedia Subsystem (WMS), for providing a common set of control capabilities, media capabilities and service delivery capabilities that are required to allow end-users to access converged Web, Media and IMS services seamlessly through different types of terminals.

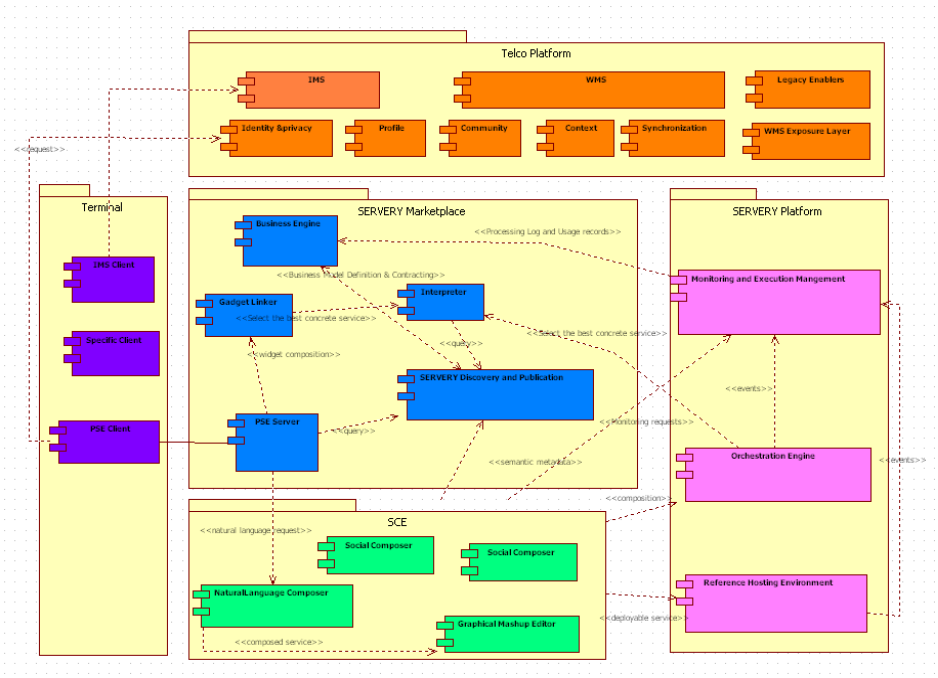


Figure 2-14: Composition framework components within overall SERVERY architecture [Source from SERVERY Deliverable 5.2]

Application Router (AR) is another widely used method for managing the SIP based services' chaining. One example is the service composition engine proposed by Ericsson, which uses an Application Router (AR) [41] for service routing as shown in Figure 2-15. In most of such solutions, AR is responsible for determining which application and in which order the applications should be invoked.

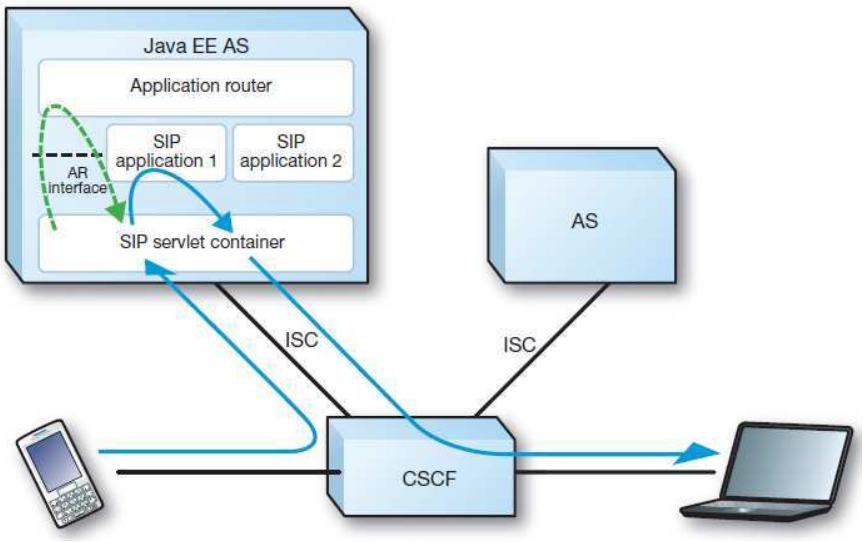


Figure 2-15: Ericsson's Application Router (AR) for service chaining [Source from [41]]

FOKUS defines a policy-based Service Broker which, is used to mediate the 3rd Party applications and service enablers as shown in Figure 2-16. Their solution is mainly based on the OSE specified by OMA for the service mediation, and based on the Parlay/OSA gateway for the service exposure.

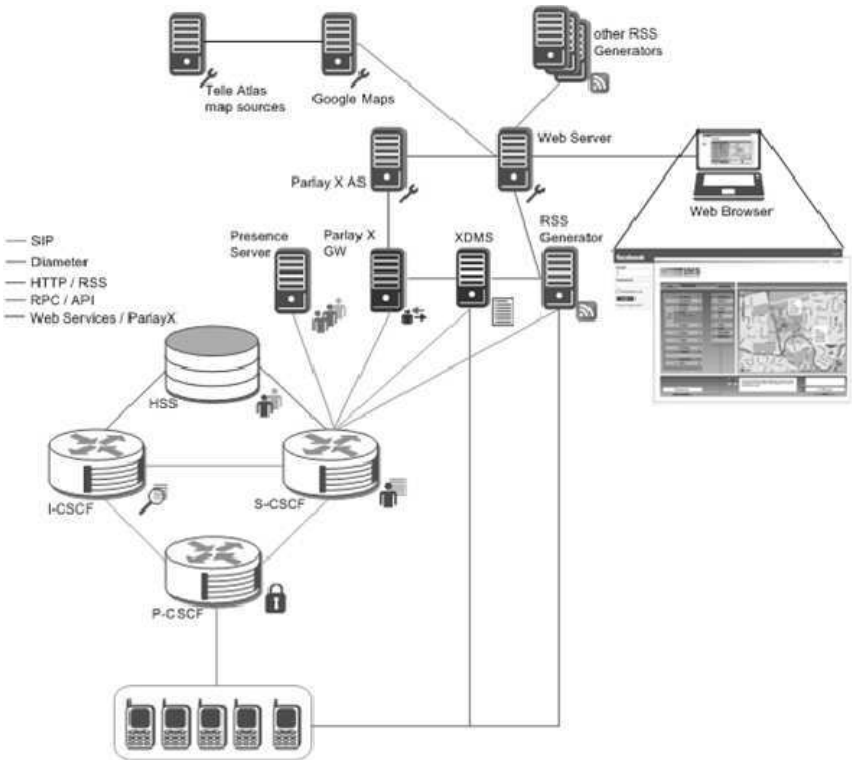


Figure 2-16: FOKUS's Telco-enabled Web mashup architecture [Source from [58]]

Regarding the development of service composition for Telecom, we can note that one trend for the service delivery is the convergence between Telecom and Web. For instance, service capabilities inside an operator's network are accessible for external applications by translating Telecom protocol and capabilities into a set of APIs (e.g. Parlay/OSA Gateway), which is associated with requirement of service exposure. In the following subsection, we investigate the next generation service delivery from another aspect - service exposure, for complementing the current service composition mechanisms.

2.3.2 Service exposure

To enable a service to be reusable, a service needs to be known and is accessible by users and/or developers. This is related to the requirement of service exposure. In the Internet domain, most service composition solutions are based on SOA whose initial objective is to facilitate services' integration across independent entities or organizations by using a set of service publication and discovery facilities. The Web services mainly are exposed through APIs, which are typically defined as a set of Hypertext Transfer

Protocol (HTTP) request messages along with a definition of response message, following SOAP, REST, or Extensible Markup Language – Remote Procedure Call (XML-RPC) principles.

To facilitate service discovery, a set of mechanisms such as WSDL, UDDI, as well as the semantic annotation and ontology technologies (e.g., Web Ontology Language (OWL), WSDL-S, Semantic Annotations for WSDL and XML Schema (SAWSDL), etc.) have been specified for helping the publication component to better describe the services, and the discovery component to easier identify the right content/service/device that matching the user's requests, context and preferences.

WSDL [42] is a widely used language for the Web service description. It is based on XML, therefore it is platform and language independent. Its structure describes four important aspects: available functions, data types used by request and response messages, grounding information, and the service's location (e.g. URL). And a Web service client is generally able to extract all information regarding a service from its WSDL file.

Once a WSDL file is created, it can be published to a common registry for enabling the later discovery by other developers. The most common approach for the service description file publication is based on UDDI. UDDI is an open XML based registry sponsored by OASIS [43]. Currently, the service matching process for the UDDI registries are generally keyword based for providing some approximate results, such as by matching the category, name, location, business, binding and tModel. Some extensions also have been provided based on Information Retrieval (IR) techniques [44] [45].

To improve the service discovery efficiency, semantic annotation is used for the purpose of replacing traditional keyword-based matching. The original approach to add semantic annotations to a software service is to describe it by using natural language and add it in WSDL files in the <documentation> tag, for making it to be understood by third party developers. However, such a description is unstructured and limited to be understood by humans, so that it is only useful for the static service composition. To enable the machines to be able to consume such semantic information, developers need a new approach for defining their services. That is, Semantic Web, which was coined by Tim Berners-Lee in 1998, and was mainly standardized by W3C. According to W3C, "*Semantic Web is about common formats for integration and combination of data drawn from diverse sources, where on the original Web mainly concentrated on the interchange of documents. It is also about language for recording how the data relates to real world objects. That allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wired but by being about the same thing*"[46]. The main aim behind the Semantic Web related technologies is to provide a structured and detailed way to describe service capabilities, grounding aspects, policies and other related information, and to make such

information to be machine understandable for enabling the advanced reasoning mechanisms. Therefore, it facilitates the service discovery and service composition, especially the automatic service composition in which machines are considered as the direct semantic consumer. Over the last decade, a set of semantic related technologies have been standardized such as DARPA Agent Markup Language for Services (DAML-S), OWL-S, WSDL-S, Web Service Modeling Ontology (WSMO), etc. These advanced technologies generally associate with the term of “Ontology”.

In the computer science field, ontology refers to a formal knowledge representation in a specific domain. It is originally used in Artificial Intelligence (AI) to model real objects and enable automatic reasoning on their properties. Currently, it is used for the web services by defining a set of concepts and relationships between them, and thus enables the advanced reasoning for the services. Resource Description Framework (RDF) and OWL are two most widely used languages for modeling ontologies.

Relying on these basic service exposure facilities, a variety of advanced solutions are proposed by academies and industries for improving the service discovery efficiency and user’s experience by associating with auxiliary information, such as context information [47], QoS [48], user intent [49] and service reputation [50].

Meanwhile, in Telecom domain, an application independent capabilities layer is separated from the control layer to allow the reuse of the existing Telecom services, as specified in IMS. Besides the original S-CSCF based service chaining mechanism and the solutions relying on an additional entity for service brokering (e.g., SCIM, OSE) which are purely SIP based, one recent trend is to use Web technologies to access and compose Telecom capabilities. This trend is closely tied to particular service exposure frameworks such as Parlay/OSA Gateway, ParlayREST, NGSI, and OneAPI specified by the standards bodies. These frameworks aim to make service capabilities inside an operator’s network accessible for external applications by translating Telecom protocols and capabilities into a set of APIs. Along with standards bodies, operators, service providers, academic and Telecom communities are trying to propose their solutions for catering to this trend.

2.3.2.1 Parlay/OSA gateway

Parlay/OSA (Open Service Architecture) Gateway was specified by Parlay Group, which worked closely with ETSI and 3GPP. It aims at making service capabilities and network resources inside the operator’s network accessible for external applications by abstracting the network capabilities into a set of APIs.

Some important APIs exposed by Parlay include [51]: call control, SMS, MMS, Charging, Location, Conference, Address book, Presence, and Terminal Capabilities. These APIs are specified by the CORBA Interface Definition Language (IDL) and WSDL.

The Parlay/OSA architecture is illustrated in Figure 2-17. It makes service capabilities inside the operator's network accessible to extra applications via a gateway. This gateway translates various Telecom protocols into a set of APIs and exposes the functions of these protocols in a way that is much simpler to be used by non-Telecom specialists. Meanwhile, it ensures to Telecom network operators that any application using the Parlay API cannot affect the security and integrity of their network.

It mainly includes a Parlay Framework (FWK) and several Service Capabilities Features (SCFs):

- Parlay Framework (FWK): provides an initial contact point to the client applications for discovering the services that can be used by applications. It also considers all required security precautions and defines the service level agreements.
- Service Capability Features (SCF): consist of a large number of individual interfaces that are designed to allow the application providers access to the capabilities within the network.

There are two sets of APIs residing in the Parlay Framework. They allow services to access the core network functionalities transparently:

- Parlay APIs: target to the developers who already have some knowledge of the underlying Telecom network.
- Parlay X APIs: target to the developers who have very limited knowledge of the Telecom underlying environment. It is based on Web service interfaces. In other words, Parlay X targets the IT and Web service developers who do not necessarily know the details of call control and call routing.

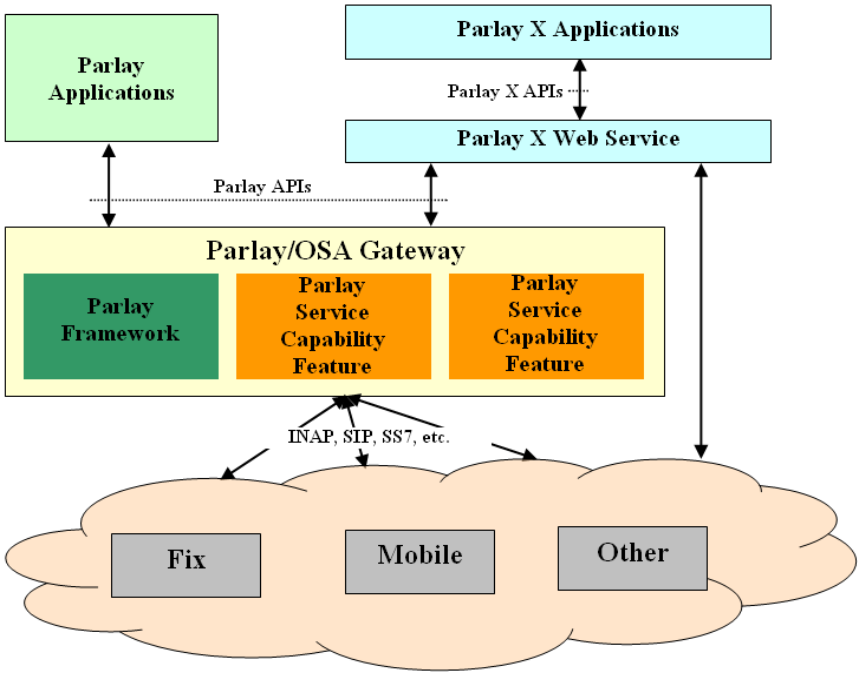


Figure 2-17: Parlay/OSA architecture

In order to adapt the OSA/Parlay architecture to IMS, the Parlay gateway must be used on top of IMS components. This allows IMS services to be implemented by using Parlay APIs instead of using SIP directly. Consequently, the gateway must translate the Parlay API methods to SIP messages.

2.3.2.2 ParlayREST

ParlayREST is specified by OMA for defining the RESTful binding for Parlay X Web services [52]. The REST protocol binding are defined for an abstract API based on existing OMA enablers.

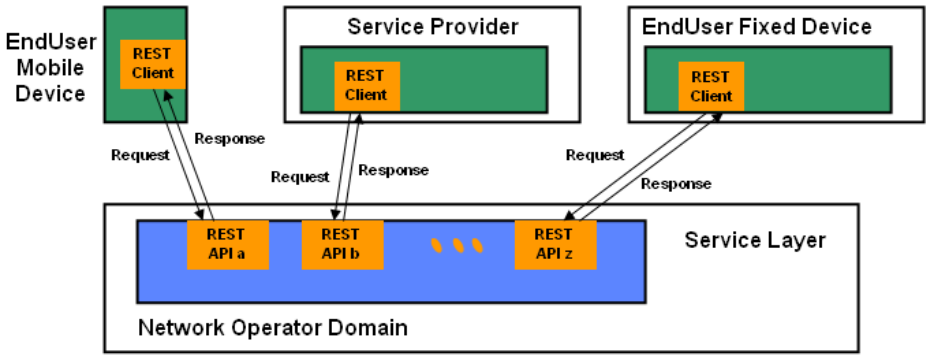


Figure 2-18: ParlayREST architecture in OMA [Source from OMA-ERELD-ParlayREST-V2_0]

In ParlayREST version 1.0, it defines the REST protocol binding for the following Parlay X Web services: Short Messaging, Multimedia Media Messaging, Payment and Terminal Location.

In ParlayREST version 2.0, it defines the RESTful style APIs including: Presence, Address List Management, Call Notification, 3rd Party Call, Audio Call, Terminal Status, and Device Capabilities.

2.3.2.3 OneAPI

OneAPI [53] is the current valid standards specified by GSM Association (GSMA) for exposing network capabilities over HTTP. For the aims of reducing the resource-costing and time-to-market for developing a new service, OneAPI defines a commonly supported, lightweight, and Web friendly APIs to allow Mobile and other network operators to expose useful network information and capabilities to an IT or Web application developers. OneAPI reuses existing standards (e.g. OMA PXPROF), and advises standards bodies what Web developers expect from network operators' APIs, thus such standards can evolve accordingly.

Figure 2-19 illustrates the OneAPI platform which is used to enable the access to the Telecom network resource.

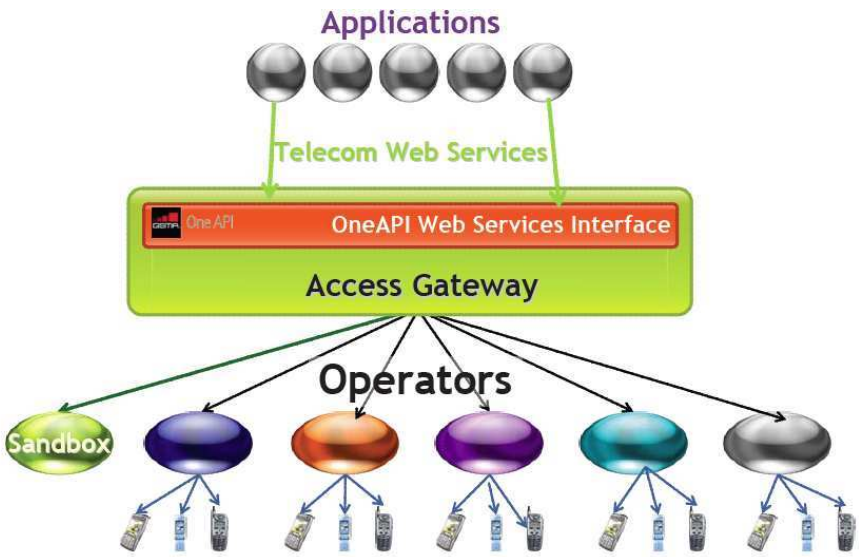


Figure 2-19: OneAPI interfaces for accessing to Telecom resources [Source from [54]]

Currently, OneAPI allows both Parlay X Web service (SOAP) and REST style APIs to access core network capabilities, such as SMS, MMS, Location, Payments, device capability discovery, data connection bearer indication, and call control capabilities.

2.3.2.4 Wholesale Application Community (WAC)

Wholesale Applications Community (WAC) [55] is an organization established in 2010 by 24 Telecom companies. It aims to create a unified open platform that provides the single interface for accessing to the

device functionalities and network resources, thus allows developers to build an application across multiple devices, operating systems and networks, as shown in Figure 2-20.

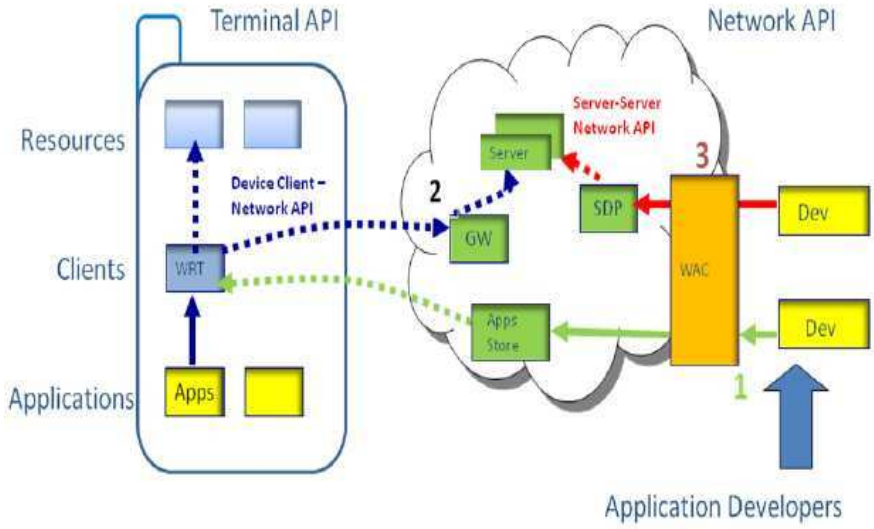


Figure 2-20: WAC: service deployment across devices, platforms and networks

In order to create service across devices, platforms, and operators, WAC promotes the use of mobile web technologies. Its applications are mainly widget based which is known as WAC widget. In order to facilitate the widget creation, WAC provides a set of comprehensive handset APIs (e.g. AccelerometerInfo, AddressBookItem, Applicationtypes, AudioPlayer, Camera, Device, DeviceStateInfo, Exception, etc.), a set of common web services, which provide capabilities hosted by the operators (e.g. billing, account balance lookup), and the widget SDK, which provides emulators, documentations, and other tools required to develop WAC widgets.

2.3.2.5 Other service exposure solutions

Along with standard bodies, operators, service providers, academies and Telecom communities propose their solutions for catering to this trend. For instance, Telefonica proposed a WIMS 2.0 Service Platform [56] to expose Telecom capabilities through RESTful APIs and Portable Service Element; ORACLE deployed a Communications Service Gatekeeper [57] to provide operators a single, centralized service exposure platform with diverse choices of application interfaces, including native Telecom interfaces, SOAP Web services, SOA Web services and RESTful Web services; FOKUS defined a JavaScript based Web 2.0/Telco Service Exposure Gateway [58] relying on Parlay/OSA Gateway and OSE specifications to expose Telecom enablers to Web 2.0 mashup.

Moreover, there are also some commercialized exposure platforms provided by operators such as BT [59], Orange [60], Deutsche Telecom [61], Telefonica [62], and Vodafone [63], in which operators are seeking the possible business models enabling them to benefit from exposing their intrinsic resources and capabilities to others. The services they expose are summarized in Table 1. All of these listed platforms expose the Telecom services through APIs and/or widgets.

Company	Project	Exposed services
BT	Ribbit	• SMS, voice call, conference call, presence, location
Orange	Orange Partner	• Advanced APIs: contact everyone, multimedia conference, device capability enabler, location, SMS internet, click-to-call, M2M, API manager • Instant APIs: location, SMS, MMS, click-to-call, click-to-conference, voicemail, voicemashup, email, mobeasy • Personal APIs: authentication, personal calendar, personal contacts, personal content, personal favorites, personal messages, personal photos, personal profile, personal rich profile, payline
Deutsche Telekom	Developer Garden	• Send SMS, Conference call, Voice call, IP location, voice record lab
Telefonica	BlueVia	• O2 litmus APIs: network connection status, O2 litmus location, manage device, view account status, manage post pay bolt on's, view device compatibility, O2 litmus customer charging • Open moviforum APIs: receiving SMS, receiving video call, SMS send, MMS send, SMS 2.0, copiagenda • Developers movistar APIs: SMS, MMS, WAP-push
Vodafone	Betavine	• SMS, WAP-push, application trigger messages

Table 2-1: Telecom service exposure platforms and their exposed services

Examining these existing service exposure mechanisms, we observe that almost all of these solutions are still designed for a specific group of developers and/or the advanced users. If they want to encourage user involvement instead of exclusivity and really include non-professional users, the user-centric feature still needs to be significantly enhanced.

2.4 Summary

In this chapter, we have investigated the networking evolution in the last decade, mainly focusing on NGN for Telco and NGI for Internet. By analyzing their network and service characteristics, we have learned that convergence is an inevitable trend for both of them, and the upcoming service environment would be a converged service environment, in which services derived from different domains should be able to interoperate with each other seamlessly, regardless of their underlying heterogeneities. In such a converged service environment, service composition is recognized as the principal service delivery mechanism. We then studied the current service composition mechanisms in IMS and Web domain respectively. Our study shows that even a great progress has been made in these years, however, there are still some challenges

hindering the further evolution for the paradigm of service composition, such as convergence, user-centricity, and automaticity. In Chapter 3, we propose an enhanced service composition, which mainly addresses the aforementioned three challenges, by considering both the advantages and limitations of the exiting solutions we investigated in this chapter. Moreover, in this chapter, we also have made a survey on service exposure, including both service discovery and publication mechanisms, which can be considered as an indispensable complement for the paradigm of service composition. The experience we learned from this survey provides a substantial foundation to define an improved service exposure model, which we introduce in Chapter 4.

Chapter 3 Service composition in NGN environment

Nowadays, the service market is more and more customer-driven, and the competition among operators and/or service providers is more and more intensive, both them are trying to optimize their infrastructure and network investment and get the most lucrative position in the service market. Many questions arise while operators and service providers are trying to achieve this goal, such as how to deploy revenue generating and competitive service in a fast-paced environment, how to reduce their deployment and maintenance costs, how to decrease their time to market, and how to optimize their lifecycle. Service composition, which enables creating novel services by reusing the existing service components, is recognized as a promising choice for both IT and Telecom domains. In this chapter, some service composition challenges are outlined. These challenges motivate us to propose an enhanced service composition model in the Web/IMS converged environment.

3.1 Service composition challenges statement

From the state of the art of service composition we introduce in Chapter 2, we note that even service composition has been widely recognized as an important trend for next generation service delivery, and the relevant research work has been carried out for years catering to different requirements, nevertheless, there are still some open issues being pursued by both industry and academia nowadays. In the following subsections, we outline three challenges for current service composition, which we address in our proposed service composition model. These three challenges are: convergence, automaticity and user-centricity.

3.1.1 Convergence

Regarding the evolutions in IT and Telecom worlds, convergence is widely acknowledged as an inevitable trend by both industry and academia. One of the most prominent examples is the alliance between Web 2.0 and IMS. Web 2.0 brings a significant impact on the Internet service provisioning by encouraging the contribution of end-users for innovative services and/or content creation. IMS is widely recognized by the Telecom world as the reference architecture for NGN by enforcing the service control over IP based infrastructures, enabling the access independent feature, and guaranteeing a secure and ubiquitous service

experience for user. Apparently, such convergence predicatively will bring a new perspective for next generation service delivery.

Meanwhile, in both Web 2.0 and IMS environment, service composition is considered as a basic service delivery mechanism that allows the creation and execution of complex services by integrating distributed autonomous service building blocks. However, most of current service composition mechanisms within IT and Telecom worlds are evolving rather independently from each other based on the different underlying infrastructures. For instance, in Telecom, IMS has opted for IP based, especially SIP based call control protocols for voice, data and multimedia real time communications. This consideration promises to easily provide the convergence between Telecom and Internet. The S-CSCF and iFC based service chaining mechanism were specified for IMS based service delivery. SCIM was proposed by 3GPP and OSE was specified by OMA for providing service brokering and interoperability management for Telecom capabilities and services. However, such kind of service interoperability is generally Telecom protocols based, and within these entities, there is no such mechanism for cross application domain brokerage and service description for catering to the convergence requirement.

Consequently, in our research, we attempt to provide a unified service composition model for empowering end user to access IMS and Web services seamlessly through different types of terminals and enabling the service creation and interoperability regardless of the different types of services (IMS services, Web services, or device offered services).

3.1.2 User-centricity

Another key trend for next generation service delivery is user-centric in which user is not only considered as a consumer, but is also encouraged to be a contributor for services or contents. A lot of work has been done for this aim and many solutions and platforms have been proposed by different enterprises or communities, especially after the emergence of Mashup in Web 2.0. Nevertheless, most of the proposed frameworks or systems are static or semi-automatic. For instance, even some famous so-called user-centric solutions like YahooPipe, MashMaker, and MARGMASH still need the human intervention for completing a service creation process. Such human invention is always not friendly enough by considering the actuality that most of the ordinary users do not have any knowledge on Telecom or Internet technology, that hinders the widely popularization of the concept of “service creation by end-user”.

In order to encourage multi-level users to contribute to the service creation, it is necessary to hide the backend complexity and simplify the service creation interface and process. Thus our proposed service composition model should be enforced with a sophisticated mechanism for catering to the user-centricity

requirements, and thus even a user, who does not have any development skill, can easily create her/his personal service.

3.1.3 Automaticity

Automatic service composition is extremely important for realizing the user-centric feature. Since service composition is a really complicated task, and it is beyond the ordinary user's capability to deal with the whole process manually. Such complexity is due to the facts that the dramatically increasing number of services makes it difficult to find out an appropriate one from a huge service repository. The highly dynamic service environment requires that services can be created and updated on the fly. Thus the service composition system needs to detect the update and make the decision at runtime, which is apparently unfeasible or unefficient for being handled manually by users or developers.

In order to reduce the manual operation, improve the efficiency of composite service generation and execution, we propose an alternative mechanism for addressing the automatic composition issue. Our proposal is mainly related to the service creation and update processes.

3.2 An automatic service composition model in converged environment

As we mentioned in the previous section, convergence, user-centricity and automaticity are the most important challenges which are hampering the wide popularity of service composition in the service marketing. These challenges motivate us to devise a unified service composition model, which enables not only the professional, but also the non professional user to create their proper services easily, regardless of the underlying heterogeneities among different kinds of services.

Two main requirements need to be respected while proposing a service composition model:

(1) The impact on the underlying control plane should be as small as possible;

(2) The complexity and heterogeneity should be hidden for both end users and service providers.

3.2.1 Global overview for the service composition model

Figure 3-1 illustrates the functional model of our proposed service composition solution in IMS/Web converged environment.

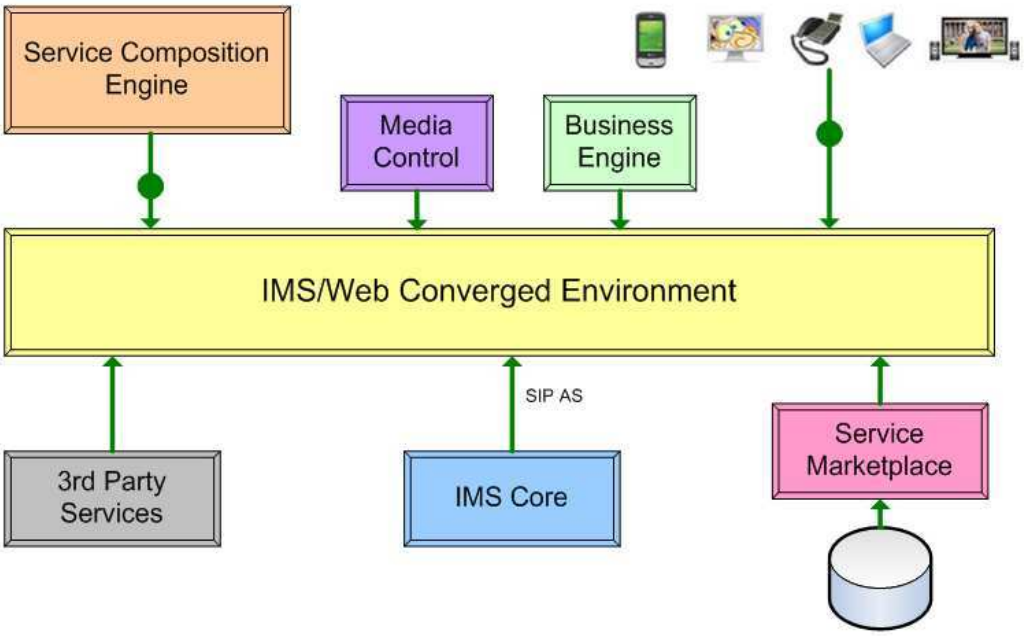


Figure 3-1: Unified service composition model relying on IMS/Web converged environment

The above presented service composition model relies on a unified Service Composition Engine (SCE) and an IMS/Web converged control plane. This converged control plane provides a common set of control, media and service delivery capabilities that are required to allow end user to access converged Web, media and IMS services seamlessly through different types of terminals. To be noted that, our work does not intend to illustrate the IMS/Web converged control plane in details, but focuses on the service composition engine relying on it, which enables users to reuse the existing IMS or Web services for creating their personal services. Thus we reuse the WIMS framework specified by SERVERY project as the IMS/Web converged control plane.

3.2.2 Functional architecture overview

From the functional view, the main role of SCE is to enable the reuse of the existing capabilities, services and network resources, which relates to the actions like service creation, service exposure, service discovery, and service orchestration. Figure 3-2 is a simplified model of SCE, which is linked to the IMS/Web converged environment, the unified Service Exposure Layer, and the user-friendly Service Creation Environment.

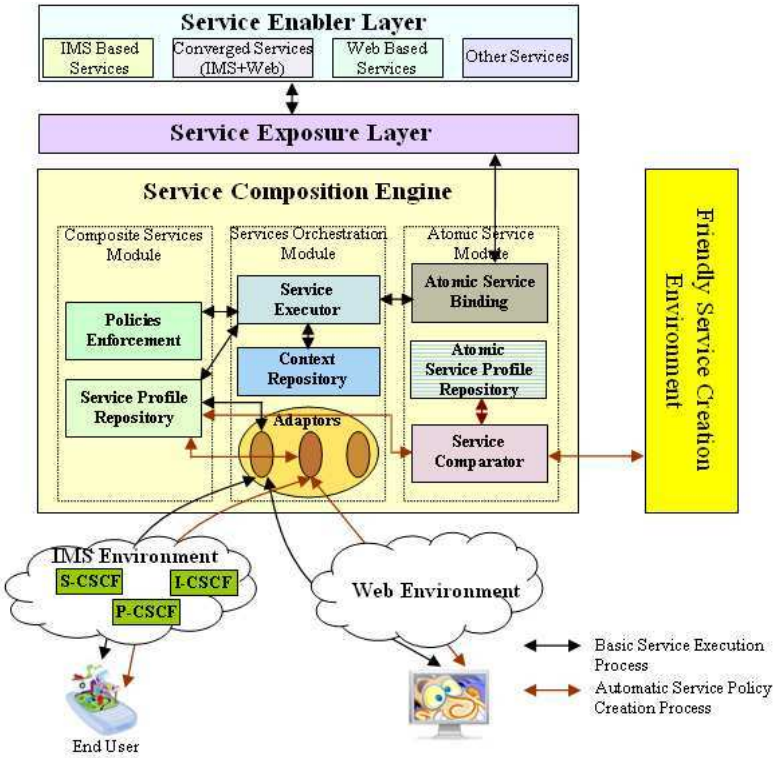


Figure 3-2: Service Composition Engine (SCE) in IMS/Web converged environment

3.2.3 Service profiles

This model uses “Service Profile” to encapsulate the information related to service composition and orchestration. Such service profiles can be considered as the formalisms that are used to express business, engineering or process criteria. These service profiles are different from the User Service Profiles specified in IMS, in which there is one kind of User Service Profile based on iFC, by being divided into several different kinds, such as user service profile, atomic service profile and composite service profile for satisfying the aim of enabling user-centric automatic service composition. They can be represented as a XML file which is similar to the service profile in IMS or a script which is widely used in Web service composition.

More concretely, each user, who has subscribed to the service marketplace, owns a user service profile as shown in Figure 3-3. According to this user service profile, once a user logs into her/his service environment, or open her/his device, she/he can invoke the service she/he has subscribed, according to the service invocation logic and execution conditions specified in iFC.

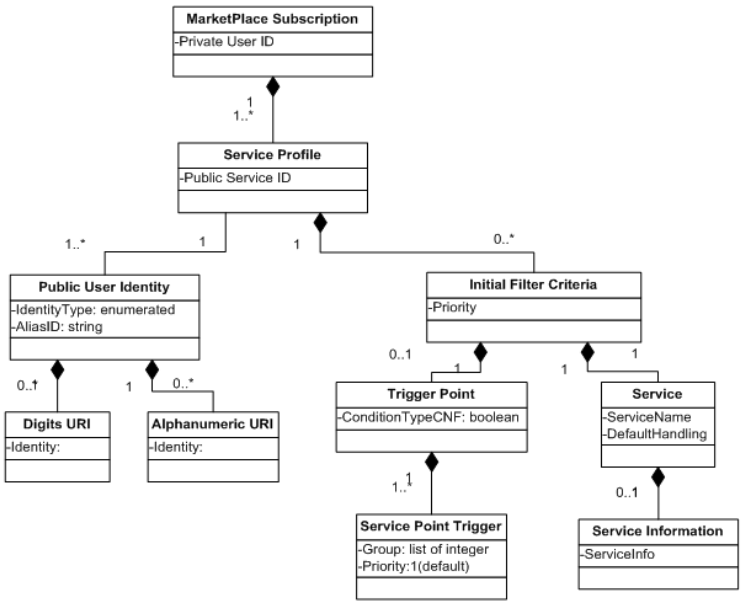


Figure 3-3: An example of a simplified user service profile

An atomic service profile represents the service information of an atomic service which can be reused in a composite service. Such atomic services can be considered as service capabilities in Telecom, or other Web services or workflows. A composite service can also be considered as an atomic service if it is used entirely as a component service in another composite service.

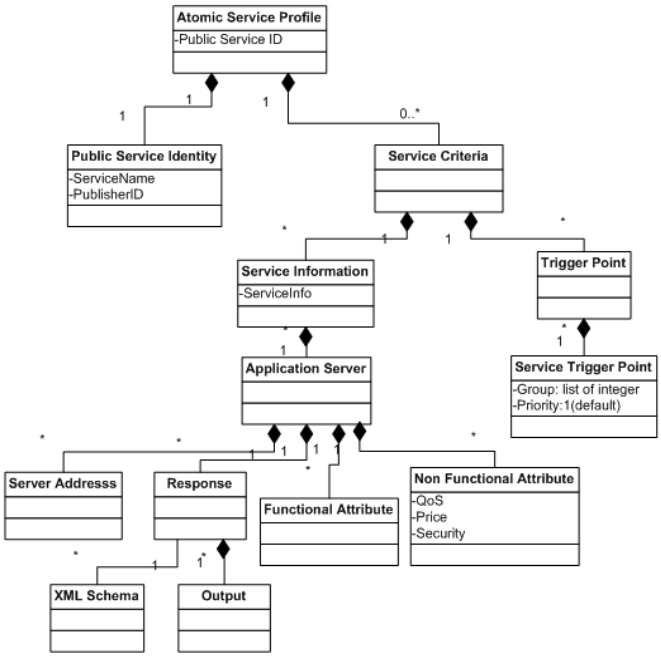


Figure 3-4: An example of a simplified atomic service profile

A composite service profile can be represented by a set of combinations of a condition and an action to be performed if such condition is true. The “action” can be an invocation of a function, script, code, or workflow which covers larger scope than the services defined in iFC for IMS services. The “condition” can be Boolean attribute that yields true or false, or an extension to some other machine readable data. Moreover, the priority attribute is added to each combination for indicating the service invocation order.

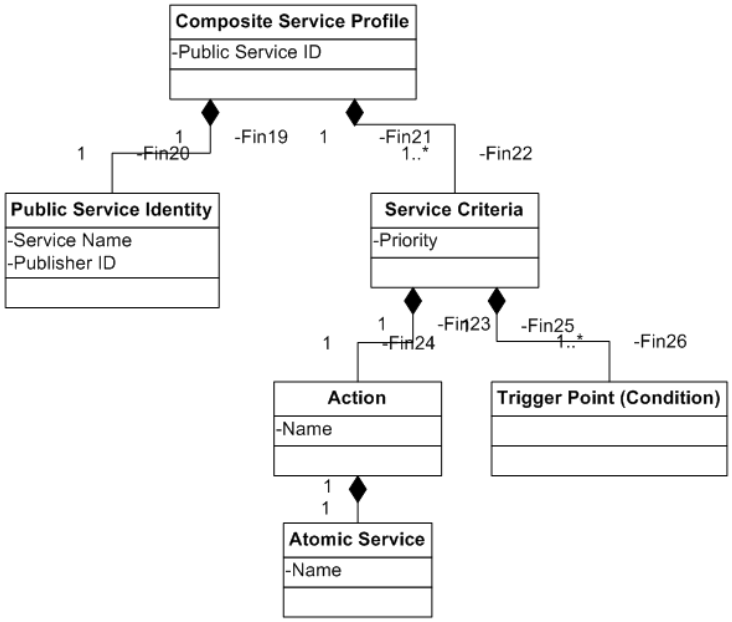


Figure 3-5: An example of a simplified composite service profile

Relying on a set of service profile related components, such as Service Profile Repository and Policies Enforcement, the service composition related policies extracted from the relevant service profiles are applied to the passing requests. For the service invocation process, a service composition policy should go through the following steps: (1) Evaluate the service profiles by using the received messages and other contextual information; (2) Execute the service actions resulting form the positive evaluation of the service conditions; (3) Return a service execution decision to the requestor.

In our proposed service composition model, all of above steps mainly rely on the Service Executor component. When SCE receives a request initiated by an end user, a corresponding composite service profile is retrieved from Service Profile Repository. The Service Executor evaluates it using information extracted from incoming messages or other contextual information retrieved from Context Repository. According to the evaluation result, Service Executor executes an “action” as a consequence of the evaluation result. Such an action can be an invocation of one capability or multiple capabilities in the predefined orders, or termination of the message routing and send the final decision to end user, or even the rejection of the incoming message.

The above mentioned service profiles are mainly related to the composite service execution process. Their intrinsic variety makes them easy to be extended for addressing some specific requirements in service provisioning process, such as the automatic service composition.

3.2.4 Main components

The main components in SCE comprise: Service Profile Repository and Policies Enforcement which reside in *Composite Services Module*; Service Executor, Context Repository and various adaptors which locate in *Service Orchestration Module*; Atomic Service Binding, Atomic Service Profile Repository and Service Comparator in *Atomic Service Module*.

3.2.5 Basic composite service execution process

In order to manage the aggregation and the interoperability among different atomic services, SCE needs service composition information for controlling the invocation of atomic services. Such information is stored in *Service Profile Repository* as a composite service profile. Each profile in *Service Profile Repository* is associated to a unique composite service. It specifies the sequence of atomic services to be invoked for this service, and indicates which atomic services will be invoked in which condition.

Regarding the underlying converged service environment, the incoming service invocation requests may be SIP based (e.g. sent by Telecom terminals) or HTTP based (e.g. sent by Internet terminals, or web pages, or some widgets). In order to ensure a unified user experience, all of those SIP or HTTP based requests should be adapted by SCE. Thus, some additional adaptors are considered as necessary. One of such adaptors is Protocol Adaptor. Consequently, different protocols based requests are sent to Protocol Adaptor firstly. Then according to the predefined rules, Protocol Adaptor extracts the necessary service information from those newly received service invocation requests (e.g., the unified Service IDs), and formalizes them into uniform HTTP based requests for the subsequent steps.

Afterwards, Protocol Adaptor transfers the extracted service information to Service Profile Repository. Depending on the received information, Service Profile Repository searches for the corresponding service profile from its directory for the composite service that the user demands to execute, and then transfers it to the Service Executor component.

After receiving a service profile from *Service Profile Repository*, *Service Executor* performs the services chaining by forwarding the service invocation request to the corresponding atomic services. Thanks to the Service Exposure Layer, even the Telecom service enablers can be invoked as the user friendly web applications (e.g. Widgets, Portable Service Elements [56]), or accessed through the APIs.

This enables the service chaining regardless of the services' types – IMS or Web based. Such Service Exposure Layer can rely on Parlay X Gateway [51], or other service exposure frameworks [52][53][54].

Meanwhile, *Service Executor* updates the service execution information which has been stored in *Context Repository* for memorizing the demanded composite service has been invoked. A composite service may be implemented as a standalone application server or just be represented as a service profile. If such composite service has been implemented as an application server, Service Executor, after receiving the response from the application server, begins to execute the atomic services chaining logic indicated in the retrieved service profile by sequentially routing the service invocation request among multiple atomic services. During the processing of the request, each atomic service performs its own specific role. In the case where the composite service is only represented as a service profile, Service Executor executes the atomic services chaining directly as described above but without passing an application server. After the execution of an atomic service, a response is sent back to SCE with the execution result. According to the received response and by comparing with the service chaining logic indicating in service profile, *Service Executor* decides to forward the request to the next atomic service to continue the services chaining, or terminate the services chaining and reply to user with the execution result.

Moreover, in a complete service chaining procedure, it needs to ensure the seamless transition from one atomic service to the next. The intermediate information such as user information and other contextual information should be maintained during the service chaining process, and such information is also stored in *Context Repository.*

Finally, some policies for security, charging, Service Level Agreements (SLAs), logging, QoS, privacy and so on can be enforced by the *Policies Enforcement* component. Furthermore, it provides a generic mechanism to control access for third party applications to use these atomic services, e.g. whenever a third party is authorized to access a resource in the operator's network, a policy is assigned to it.

3.2.6 Basic service creation and service exposition processes

Other important functionalities we consider in this SCE are the service exposure and service creation processes which rely on the components of *Atomic Service Profile Repository*, *Atomic Service Binding*, *Service Comparator* and several other adaptors. This module should also interact with the Service Creation Environment which contains several service creation tools for specifying and validating the application logic by end-users.

As we mentioned above, in our service composition model, not only the composite services have their service profiles, but also the component services, which we call Atomic Services, have their proper service

profiles which are named as Atomic Service Profile. These atomic service profiles bridge the advantages of service description files (e.g. the WSDL file) for Web services and the iFC based service profiles for IMS services to facilitate the service management and selection for the service creation and execution processes. The existing of such atomic service profiles implies the necessity of an efficient publication and update processes for them.

In our service composition model, once a new atomic service is introduced into the network or an existing atomic service is updated, it advertises its capabilities and requirements to SCE and stores them in SCE as an atomic service profile. This means, the newly introduced or updated atomic service initiates a connection between itself and SCE – we assume the server hosting the atomic service(s) has an additional module, which contains the description of the atomic service's functionality and usage requirements. This additional module is responsible for initiating the connection to SCE. It then embeds an atomic service profile which contains the atomic service's capabilities and requirements information into the exchanged messages that may rely on XML. When SCE receives these messages, it extracts the service profiles from them and stores them at Atomic Services Profile Repository. In order to avoid too many messages exchange, this atomic service profile advertisement can be based on the traditional stateless HTTP messages. That is because this process has no direct relationship with the service execution process which may be stateful or based on SIP for accessing the capabilities located in the IMS environment, that also means we have to add a small module within SCE for receiving the incoming HTTP requests, that is a HTTP interface which intercepts the incoming HTTP requests advertised by atomic services and triggers the corresponding service profile repository for storing the incoming atomic service profiles on it.

A usage example for the atomic service advertisement is depicted in Figure 3-6. Service provider A provides a Presence capability as an atomic service. In order to facilitate end-users to discover and use this capability, they add an additional module in the Presence server with the description of its functionalities and usage requirements (i.e. Capability ID, functional description, Address, Location, Price, Security, QoS, etc.). This additional module embeds the information into the exchanged messages and sends them to SCE as depicted in the upper part. As shown in step 1, at the beginning, in order to encourage end-users to use this Presence capability, provider A configures the usage of this capability as free, but with low security and medium QoS guarantee. After a few months, the security and QoS are ameliorated, and the service provider begins to charge this Presence service, thus the price, security and QoS portions are reconfigured as shown in step 2. Thus an updated capability service profile is re-sent to SCE.

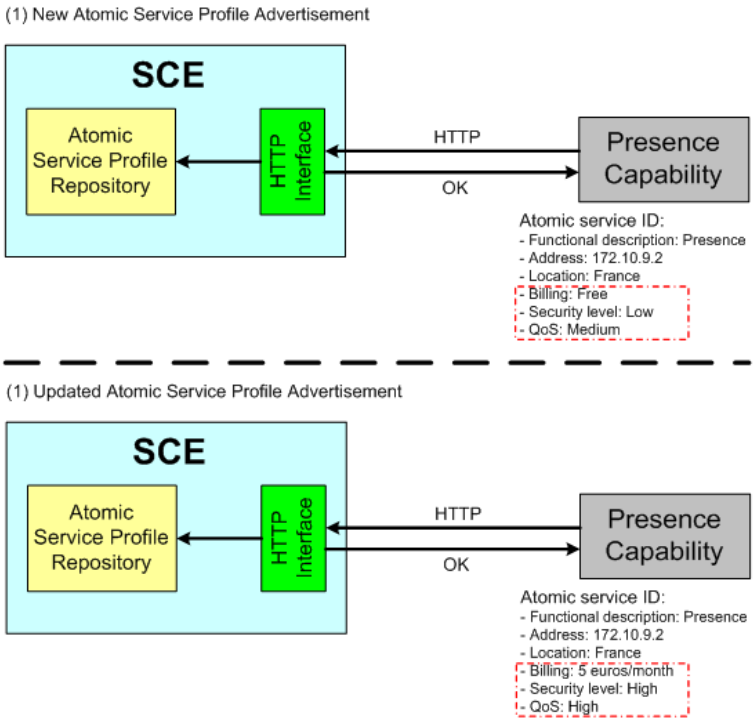


Figure 3-6: A usage example for the atomic service advertisement

The Atomic Service Binding component provides the specific formats to access these actual atomic services according to the information presented by atomic service profiles.

Service Creation Environment is an environment that empowers users to create a new service by facile method (e.g. drawing method: drag-and-drop the capabilities instantiation building blocks). Such a service creation environment can be Natural Language Enabled Service Creation Environment (e.g. SPICE Natural Language Composer), Friendly Graphical Service Creation Environment (e.g. YahooPipe), widget based service creation environment (e.g. EzWeb), or Integrated Development Environments (IDE) (e.g. Eclipes IDE).

After a user validates a new created composite service in a friendly service creation environment, the service creation environment abstracts the requirements for the new service according to the end-user's input and sends it to SCE (more concretely, relying on Service Comparator component) for selecting the appropriate atomic services for this composite service. Then, it generates a service profile for this composite service and stores it in Service Profiles Repository. Finally, Service Profiles Repository sends the newly introduced service profile to an adaptor (e.g. Widget Adaptor) for automatically generating a widget or client code for different types of terminals.

3.2.7 Basic service discovery process

The service discovery process can be divided into two types: static (at design time) and dynamic (at runtime).

- *Static service discovery*: services are identified during the creation of a composite service in order to find services that can be used in a loose coupling way.
- *Dynamic service discovery*: services are identified and bound to a composite service at runtime in order to replace the existing services.

These service discovery processes mainly rely on Service Comparator and relevant repositories. Figure 3-7 is an example of the Service Comparator module:

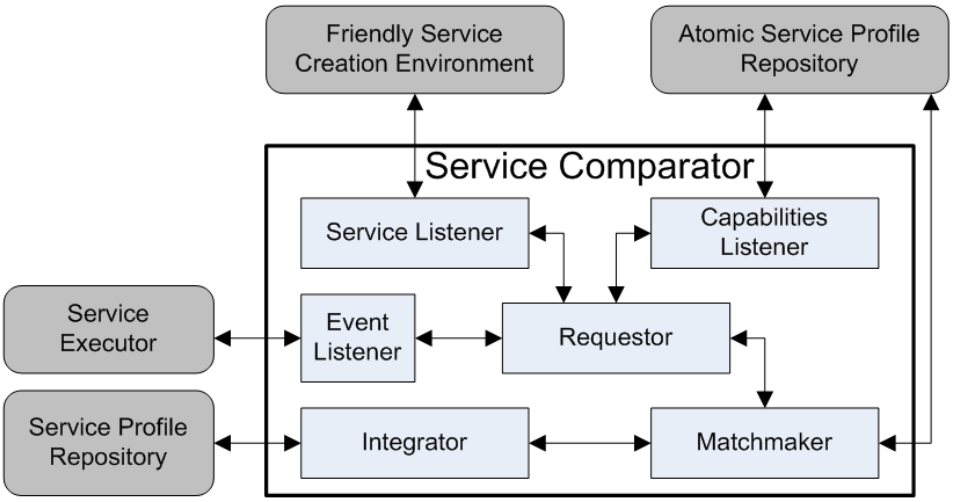


Figure 3-7: An example of Service Comparator

Service Listener: receives the service creation request from end-user.

Capabilities Listener: receives the information about the emergence of a new capability or the update of a previously existing capability.

Event Listener: receives the event notification from the service execution environment for invoking the necessary service update.

Requestor: orchestrates the functionalities of the different components within Service Comparator. (1) Instantiates a service discovery query to Matchmaker once Service Listener, Capabilities Listener or Event Listener detects the changes from service creation environment, capabilities or service execution environment; (2) Receives response from Matchmaker

Matchmaker: parses the constraints of a query and evaluates the constraints against the service creation requirement. According to the predefined selection rules, it selects the appropriate capabilities to a composite service.

Integrator: verifies the behavioral consistency among all the selected capabilities.

Once a modification from the service creation or the invocation environment, or an update form Atomic Service Profile Repository is detected, Service Comparator invokes Matchmaker and applies the specified selection rules on the decision making process, then returns the appropriate service information to Integrator.

For example, when a service composition request comes from end-user, Service Comparator should parse user's requirements and evaluate the constraints against the capabilities' information stored in Atomic Service Profiles Repository, and then provide a list of candidate services for the composite service. Each candidate service within this list should include a distance value which indicates its priority for the composite service. The service with the shortest distance will be the most suitable service for the composite service. Such service selection rules can be represented as several level constraints, and these constraints can be internal or external. The internal constraints can be described in the atomic service profile, e.g. access requirements; for the external constraints, they can be described by OWL, since one of the aims of OWL is to describe the relationship between different services.

3.2.8 Enhanced service profiles for automatic service composition

Above introduced service composition model is easy to be extended for satisfying different requirements without underlying modification. Automatic service composition is an example that is enabled by extending the service profiles used in our previously proposed model.

3.2.8.1 Extended service profiles: abstract service profile and concrete service profile

Firstly, in order to realize the automatic service composition functionality, we reuse and extend the concepts of Abstract Service and Concrete Service, which were originally proposed by Nassim Laga in SERVERY project for the widget composition. We then split the service composition process into two processes: Abstract Composition and Concrete Composition. Abstract composition consists of defining an appropriate combination of the functionality tasks and their data dependency. Concrete composition consists of determining the most appropriate services from a set of functionally equivalent ones for each functionality task. Both processes result in the generation of a service profile: an Abstract Service Profile or

a Concrete Service Profile. In this solution, when a composite service is created at design time, a loose coupling service invocation logic, as shown in Figure 3-8, is generated and represented by an Abstract Composite Service Profile, as shown in Figure 3-9. The Abstract Composite Service Profile defines the service functionalities combination through the high-level functional tasks description for each execution step as shown in Figure 3-8. In Figure 3-8, each node is related to an abstract service which can be mapped to one or several existing concrete capabilities, which are functionally equivalent. In order to facilitate the concrete service discovery, each functional node description is related to a unified abstract function ID.

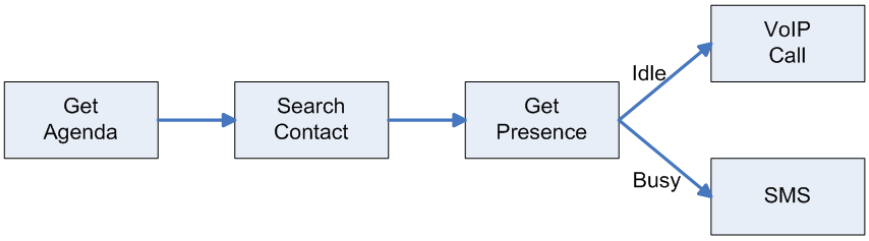


Figure 3-8: Functionalities combination

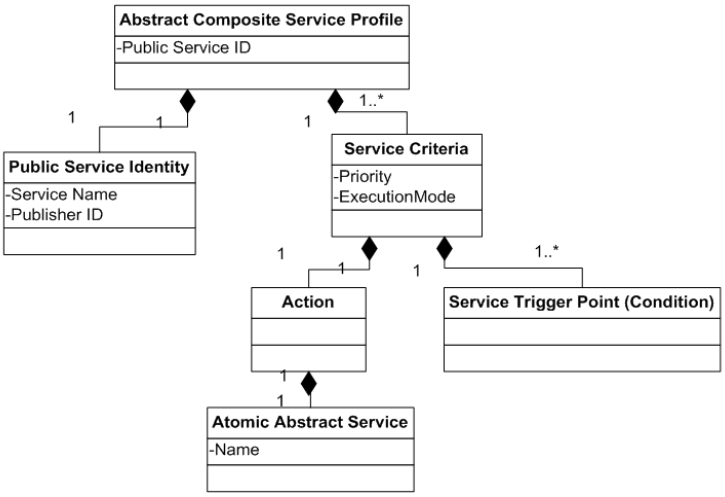


Figure 3-9: Abstract composite service profile

Meanwhile, to enable the automatic selection process, SCE contains a set of abstract atomic service profiles which are specified by the system administration members, as shown in Figure 3-10.

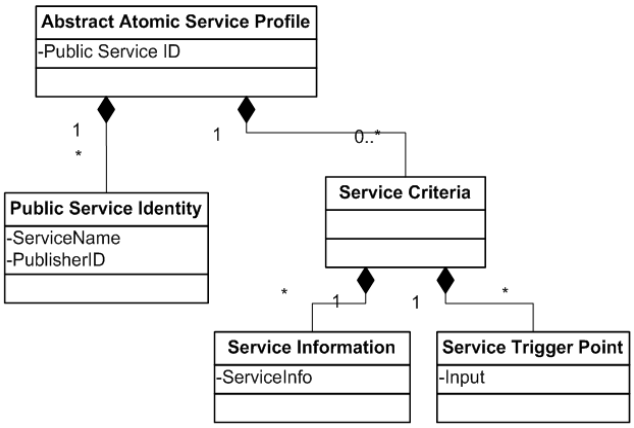


Figure 3-10: Abstract atomic service profile

Each atomic service also advertises an extended atomic service profile which we called Concrete Atomic Service Profile to SCE, as shown in Figure 3-11, when it is introduced to the network or updated as illustrated in Section 3.2.6.

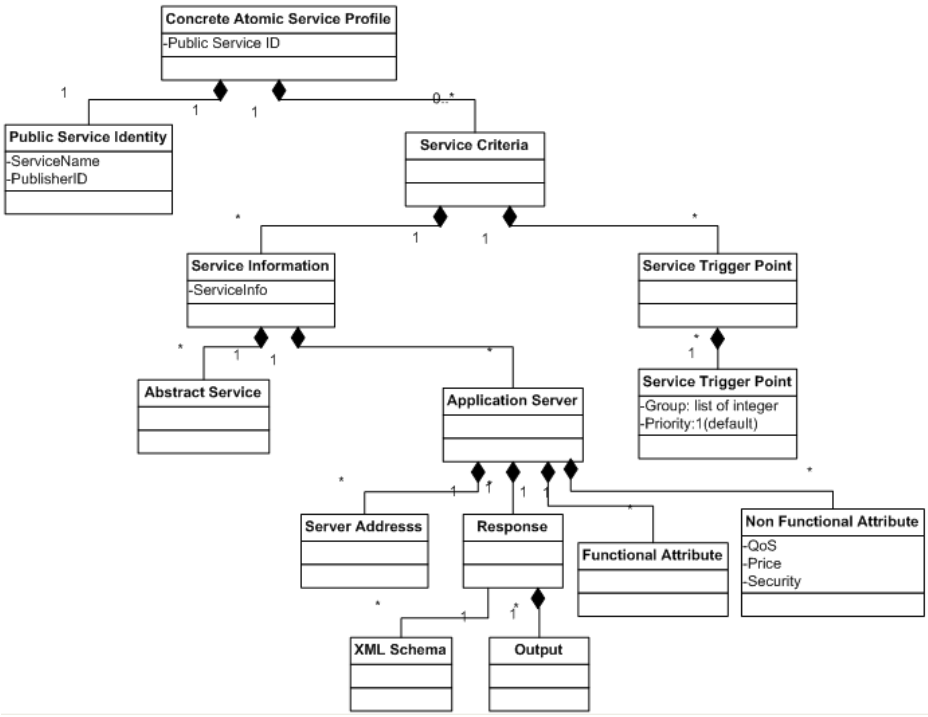


Figure 3-11: Concrete atomic service profile

Multiple capabilities may have similar functionality but with different attributes (e.g. different location/price/QoS/providers) as shown in Table 3-1. And atomic service profiles should also contain at least a unified abstract function ID and a unique capability ID.

CapID: PresenceFr		CapID: PresenceSp		CapID : PresenceGe	
Fonction	Presence	Fonction	Presence	Fonction	Presence
Address	157. 15…	Address	10. 239…	Address	176. 14…
Location	France	Location	Spain	Location	Germany
Billing	Free	Billing	5 euros/M	Billing	Free
Security	High	Security	High	Security	High

Table 3-1: Functionally equivalent atomic service profiles

According to the abstract composite service profile and the concrete atomic service profiles, Service Comparator automatically selects the appropriate concrete atomic services for this composite service, and creates a concrete service instance which is represented as a concrete composite service profile based on some predefined rules and constraints as shown in Figure 3-12. Thus, an abstract composite service profile may be instantiated several times with respect to the individual user's requirements, and several instances may be running concurrently.

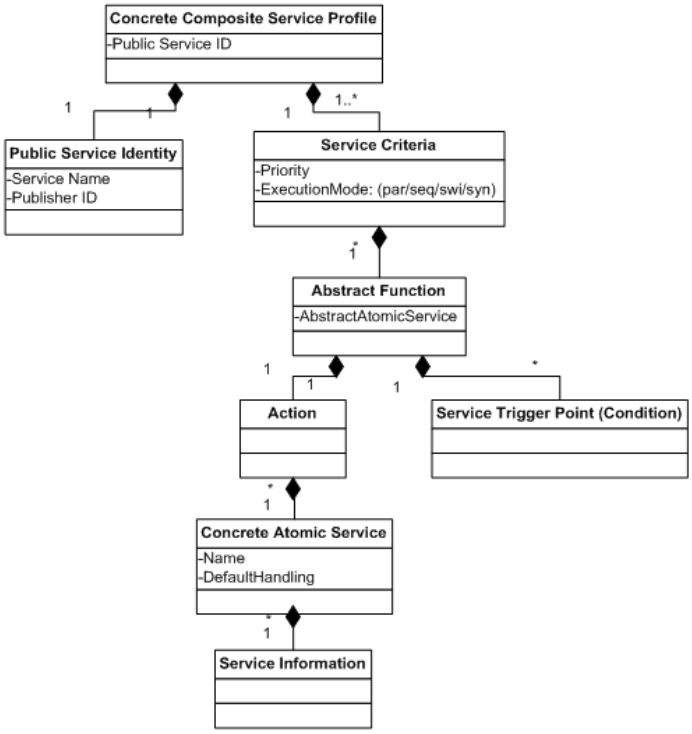


Figure 3-12: Concrete composite service profile

Based on the above improvements for the model, the automatic service composition functionality can be further enhanced. Generally, the service composition process can be divided into three phases: service

creation, service update and service execution. And the automaticity feature is primarily relevant to the service creation and service update phases as introduced hereinafter.

3.2.9 Automatic service creation process

With above mentioned enriched service profiles, our proposed service composition model is enhanced with automatic service creation feature. When a service is created by a professional or non professional user, the user only needs to define the functions to be engaged in this created service. She/he does not need to know or define precisely which existing capabilities or services are actually invoked and how such concrete capabilities and services can be accessed and configured. That will greatly facilitate the service creation process and encourage non-professional users to create more personalized services, since the users are not required to have application development skills and do not need to spend a lot of time to get to know about the concrete capabilities, services, and network environment, which are always very difficult or even impossible for non-professional users to comprehend.

A reference process for utilizing extended service profile in automatic service creation process is shown in figure 3-13: when a non professional end-user wants to create a personal service through some friendly service creation environments, such as a Natural Language Enabled Service Creation Environment or a Friendly Graphical Service Creation Environment, she/he can input her/his requirements by natural language, or wiring some instantiated building blocks. The backend processor (e.g. a Lexical Syntax Engine) parses the information (e.g. user defined functionalities, user's preference, necessary contextual information) that retrieved from service creation environment, and formalizes it into a machine understandable service creation request. This service creation request contains an abstract composite service profile, and the auxiliary information, which is mainly germane to the non-functional properties. Then it is sent to SCE. Depending on the information extracted from the received service creation request, such as the abstract functional IDs included in the abstract composite service profile, Service Comparator retrieves the corresponding capabilities/services profiles. After consulting the predefined capabilities/services selection rules, and analyzing the received service creation information (e.g. abstract functionalities, data dependency), concrete service capabilities requirements (e.g. goal, output and input requirements, constraints), and some other contextual information (e.g. user's preferences, user's location, user's terminal capabilities), Service Comparator selects an appropriate capability/service for each abstract function ID. Eventually, after an additional verification of behavioral consistency among these selected capabilities, the Service Comparator component creates a new concrete service profile which represents the capabilities invocation logic and stores it in Concrete Composite Service Profile Repository. If the result of the behavioral consistency verification is negative, the service capabilities/services selection process is re-

performed by the Service Comparator component to find out a replaceable capability/service with the equivalent functionality.

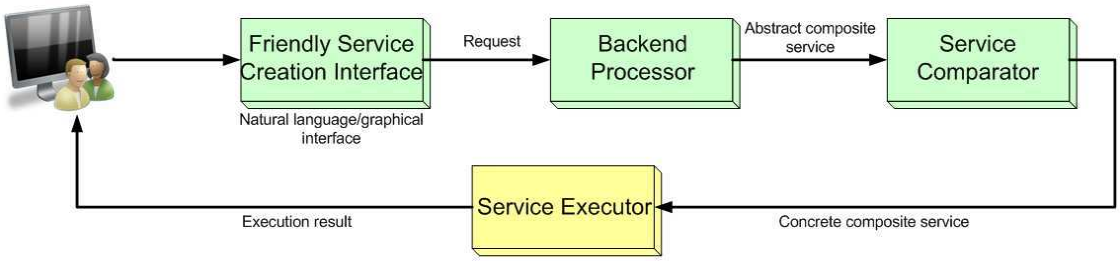


Figure 3-13: Simplified service creation example

Since during the abstract service profile definition process, which may be handled by friendly service creation environment, the service node can not only be an abstract atomic service, but also an abstract composite service. If the node is an abstract composite service, SCE should be able to decompose it into several basic abstract atomic services. On the other hand, in order to balance the time consuming, SCE should be empowered to map an abstract composite service to a concrete composite service directly without the decomposition action.

3.2.10 Automatic service update process

It is common that most capabilities/services are updated regularly, or there are some variations in the highly dynamic service environment such as new services become available, existing services become obsolete, or business rules change. Once a modification occurs, it is necessary to perform some corresponding adjustments in the service invocation logic and the corresponding service binding in order to cope with these exceptional situations with minimal user intervention. This derivates the concept of automatic service update which essentially means: when the capabilities or service execution environment have been modified, the relevant composite service logic and the corresponding configuration are automatically adjusted, either by modifying the corresponding configuration parameters, or by reselecting other more appropriate capabilities for this composite service. From the SCE's point of view, this automatic update process can be "passive" or "active".

3.2.10.1 Passive update process

The "passive" update means SCE is always on the listening state. All the updates for the composite service logic are triggered either by the incoming messages published by capabilities/services, or by the runtime service execution results.

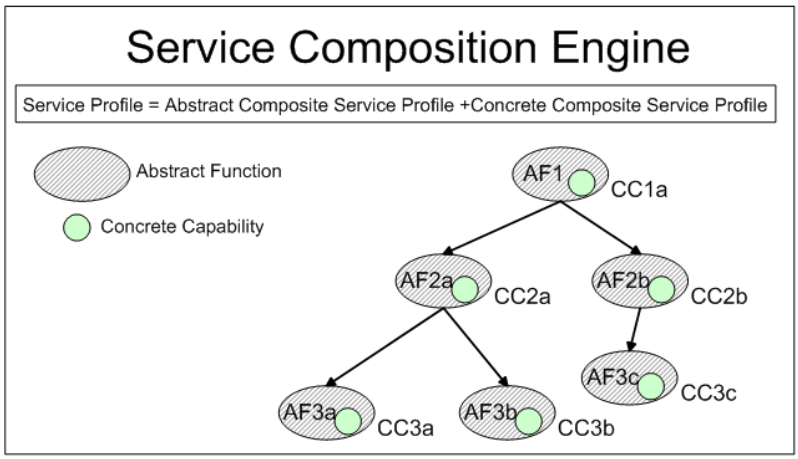


Figure 3-14: Service profile for passive update

As show in Figure 3-14, each composite service is linked to a couple of profiles: abstract composite service profile and concrete composite service profile. The abstract composite profile describes the functionalities engaged in this composite service and their invocation logic. The concrete composite profile indicates the selected concrete capabilities/services, their triggering orders and conditions. Both the abstract composite profile and the concrete composite profile are created at design time and are bound together.

When a new capability is introduced into the network or an existing capability is updated, it advertises its capabilities and requirements to SCE. Such capability update request contains at least an abstract function ID and a unified capability ID. Once SCE receives a capability update request, it searches in Atomic Service Profile Repository for finding out if a previously advertised profile from the same capability has been stored, according to the unified capability IDs indicated in the received profile and the previously stored profiles. Then SCE makes a decision for this incoming message: *Create* or *Update.*

If the decision is *Create*, a new item is added into Atomic Service Profiles Repository for storing the incoming capability profile, and a relevant capability binding item is generated and stored in the Atomic Service Binding component according to the information extracted from this incoming profile. Meanwhile Atomic Service Profiles Repository forwards this capability profile to Service Comparator. According to the ***Abstract Function ID*** in the capability profile, Service Comparator retrieves all the corresponding service profile couples which contain the same Abstract function ID. By analyzing the information stored in the incoming capability profile with the relevant concrete composite service profiles, Service Comparator decides whether the concrete service profile should be updated or not. If the outcome of the decision is positive, the previously selected capability is replaced by the newly introduced capability in the concrete composite service profile.

If the decision is *Update*, Atomic Service Profile Repository replaces the previously stored item by the new one, and the corresponding item in the Service Binding component is updated if necessary. Then, Atomic Service Profile Repository transfers the profile to Service Comparator. According to the ***Concrete Capability ID,*** which is extracted from the newly updated capability profile, Service Comparator retrieves the relevant composite service profile couples from Service Profile Repository. By analyzing these composite service profiles and the updated capability profile, Service Comparator makes a decision whether the existing concrete composite service profile should be adjusted or not according to the requirements indicated in the updated capability profile. If the decision is positive, the corresponding concrete service profile is updated and then restored into Service Profiles Repository.

In the above presented case, we assume that once the capabilities are updated, they send their capability update message to SCE successfully. However, there are some exceptions that such capability profile update process fails, thus SCE always keeps the obsolete information for the capability; or although all the capabilities information stored in SCE are the latest ones, the capabilities accesses or executions are failed due to some external reasons, such as there are some conflicts among the involved capabilities after their updates. In such situations, to ensure the user experience, the automatic update process should be triggered by the runtime service execution result. Thus, the abstract composite service profile is transferred to Service Comparator, and Service Comparator reactivates the service discovery process and re-creates a concrete composite service profile. This new concrete composite service profile is then rebound to previous abstract composite service profile by replacing the invalid concrete composite service profile.

3.2.10.2 Active automatic update process

The “Active” solution means that SCE checks the Atomic Service Profile Repository regularly to make sure all the selected capabilities for the composite service are the optimal ones at any given time.

In this case, as shown in Figure 3-15, Service Profile Repository only needs to store the abstract composite service profiles, and these profiles are enacted by Service Executor. In this case, the responsibility of Service Executor is to process messages by notifying completion status related to an abstract composite service, and subsequently scheduling next abstract function that needs to be activated. Hence, once the user logs in and executes the abstract service profile, the Service Comparator component is invoked simultaneously. Before the execution of each abstract function, according to the user’s information and preference, Service Comparator contacts Atomic Service Profile Repository at runtime for discovering the most appropriate concrete capability/service that fulfils the requirements specified in the current abstract function definition. Eventually it routes the service invocation request to this newly selected

capability/service. After the execution of the selected capability/service, a service completion notification is sent back to the Service Executor with the execution result. After the necessary processing for this response and according to the predefined invocation logic, Service Executor continues to contact Service Comparator for mapping a successor abstract function with a concrete capability/service as described above.

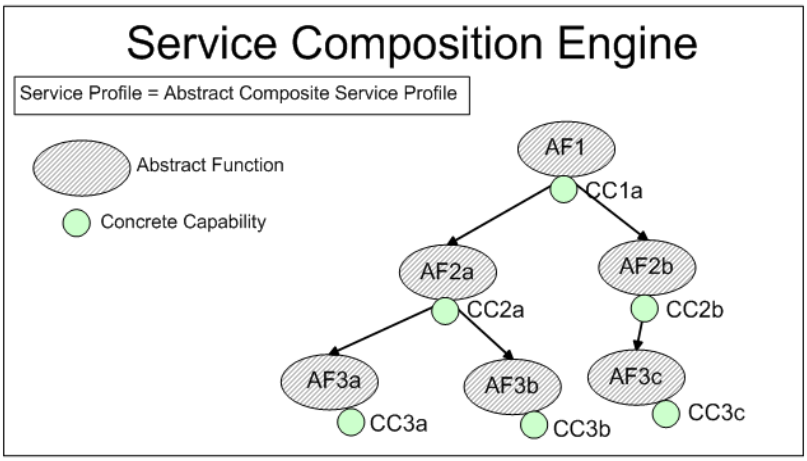


Figure 3-15: Service policy for active update

During the service execution process, Service Executor should ensure the service execution integrity, that is because the runtime service selection can not guarantee that all the abstract functions can find out the appropriate concrete services, and execute them successfully until the termination of the last execution step. Consequently, Service Executor has to provide a compensation mechanism for certain irreversible actions. If Service Comparator can not discover an appropriate capability/service for the current abstract function, then all the running services are aborted and the completed ones are compensated by executing a service-specific compensating action. This process ensures that either all the services are executed or none is. An alarm is sent to the user, and asks her/him to modify the current abstract composite logic, or her/his preference. If the selected capability can not be executed successfully, e.g., the user is not authorized to access to this capability, Service Executor re-contacts Service Comparator to select another capability with the same functionality but with lower priority.

3.2.10.3 Comparison and discussion for passive and active solutions

Compared with the passive update solution, the runtime performance of active update is improved. That is because, in the active update solution, all the selected capabilities are the optimal ones at any time, even for the case when the capabilities are being modified during the composite service execution process. Nevertheless, the possible execution latency will be a problem, and the QoS can not be guaranteed, since Service Comparator needs to interact with Atomic Service Profile Repository, and select a capability before

the execution of each abstract function which is both time and resource consuming, especially when no capability can be discovered during the service execution process or the discovered capability can not be executed successfully.

For the passive update solution, there is no time latency brought by the service discovery process, since a complete set of concrete atomic services have been discovered and selected at design time, and the service profile update process and the service execution process can be performed in parallel by executing the already defined one. However, this incurs the limitation for real-time characteristic: it is possible that when the capability profile advertisement process is failed, or there is no such advertisement mechanism for some capabilities, it would lead to the service execution failure due to the absence of service capability's real-time update. Considering the above comparison, a hybrid solution may be able to balance the time-consuming and the real time characteristic as shown in Figure 3-16.

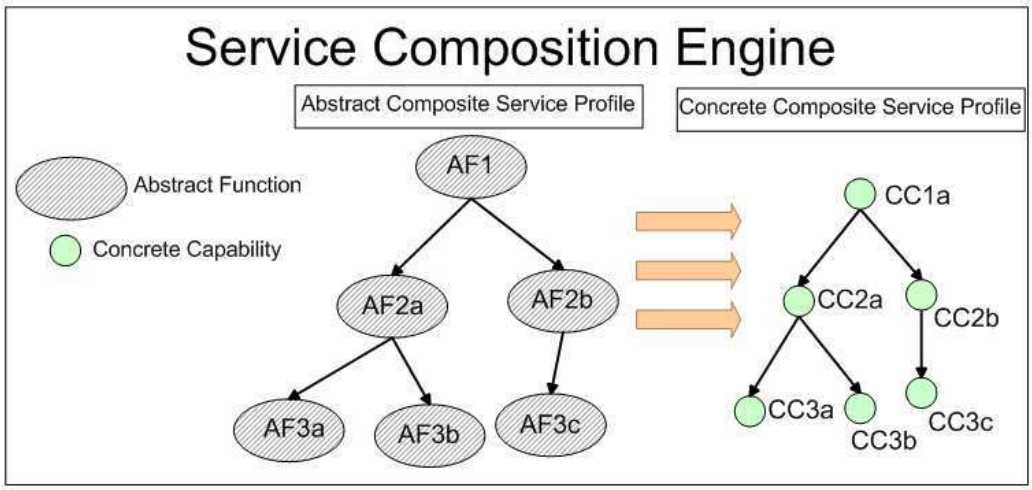


Figure 3-16: Service profile for hybrid solution

In the hybrid solution, Service Profile Repository only stores the abstract composite service profile as the active solution does, and service discovery is performed neither at design time, as the passive solution, nor at runtime, as the active solution, but just after the user logs in and before the execution of the service. Thus, once the user logs in, Service Comparator retrieves an abstract composite service profile, and contacts Atomic Service Profile Repository to define a concrete composite service profile for this composite service, according to user's login information, the runtime capabilities information, and requirements. This is similar to the concrete composite service profile creation at design time for passive solution.

Table 3-2 is a comparison among these three service update strategies:

	Passive Solution	Active Solution	Hybrid Solution
Latency	☺☺☺ All the concrete atomic services have been selected before the execution of a composite service, thus there is no additional latency brought by service discovery process.	☹☹☹ Service Comparator needs to be invoked for discovering a concrete atomic service before the execution of each abstract function. This increases the latency for the service execution.	☺☺☹ A small latency is added during the user's login process for discovering all the atomic concrete services. However, once a user logs in successfully, the executable concrete composite services are created, thus there is no additional latency for the service execution process.
Real-time	☺☹☹ If the capabilities' update process fails, or a capability is being update during the service execution process, the selected concrete atomic services can not be guaranteed to be the up-to-date ones	☺☺☺ All the selected atomic concrete services can be guaranteed to be the up-to-date ones.	☺☺☹ It can assure that each time a user logs in, the loaded composite services in her/his service environment are the up-to-date ones, except the case that the selected capabilities are being updated during the service execution process.
Fidelity	☺☺☹ Even the involved capabilities may not be the same as the ones selected at the design time. However, user's intervention can be added during the service selection process at runtime.	☹☹☹ The involved capabilities may not be exactly what end user or service developer wants, since all the capabilities' selections are automatically handled according to the predefined rules.	☺☹☹ The capabilities selection process is automatically handled by the backend components. However, user can add his intervention for the service selection during his login process by re-setting his preference.
Complexity	☺☺☹ It can add the user intervention for the service selection process.	☹☹☹ The definition of the capabilities selection rules needs enormous work. It is difficult to cover all the possible situations.	☹☹☹ The definition of the capabilities selection rules needs enormous works. It is difficult to cover all the possible situations.

Table 3-2: Comparison of three service update strategies

3.3 Proof of concept prototype

In order to prove our proposed service composition model, a MSN robot like service composer is implemented based on the proposed service composition model as a use case.

As shown in Figure 3-17, a simplified backend service composition platform is instantiated as a MSN robot like contact in Contact List, which enables to reuse MSN authentication mechanism and the existing contact information. It also facilitates the service creation process for users without any additional installation or configuration. Thus, users just need to add the email of such "contact" into their Contact List, and then they can talk with this "contact" by entering their requests, e.g. "Send my next week available time slots to John", for creating composite services. In this example, this user creates a composite service that involves the network agenda service, the network address book service, and the messaging service. Then this created composite service can be executed at runtime.

Figure 3-17: MSN robot like service creation interface

A natural language interpreter, which is linked to this “contact”, extracts the information from the user’s request and structures it into a specified format, which encapsulates a temporary abstract composite service profile, and forwards them to SCE. To achieve this goal, we implemented a simplified Natural Language Interpreter. To be noted, this proposed Natural Language Interpreter acts as an auxiliary tool for our proposed service composition model, thus it only provides very simple natural language processing function, more details about the possible natural language interpreters can refer to other natural language processing tools, e.g. SPATEL Natural Language Composer. When the Interpreter component receives the text entered by user, e.g. “Get my next week time slot, and send it to John”, it extracts the relevant services by stemming the keywords from the request and mapping them to the relevant abstract services as shown at the left side of Figure 3-18. And then according to some predefined selection rules, an abstract composite logic is created as shown at the right side of Figure 3-18. This abstract composite logic is encapsulated into an abstract composite service profile, and sent to SCE.

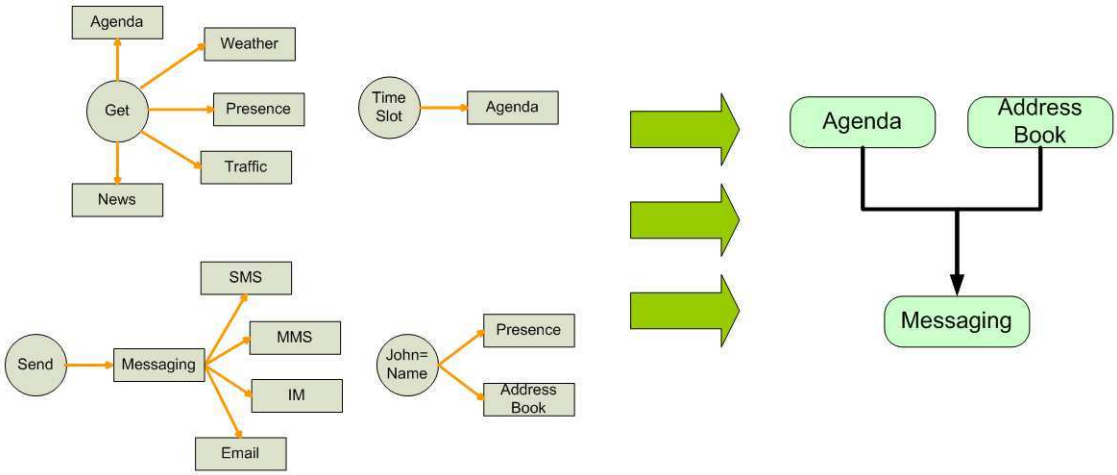


Figure 3-18: Interpretation for natural language based request

According to the received information, Service Comparator selects the relevant service capabilities according to some contextual information and the user's preference, and then refines their invocation logic, which will be represented as a concrete composite service profile. Then this service profile is forwarded to Service Executor. After the execution of the explicitly corresponding service capabilities (e.g. get the agenda information from Network Agenda Capability) and the auxiliary service capabilities (e.g. get the present status of John from MSN Contact List, if John is offline, invoke the SMS capability for sending a SMS with the user's available time slots' information got from Network Agenda capability), as the call flow illustrated in Figure 3-19, a final execution result is sent back to the interpreter. Eventually, this interpreter sends back a response to the user such as "The message has been sent to John".

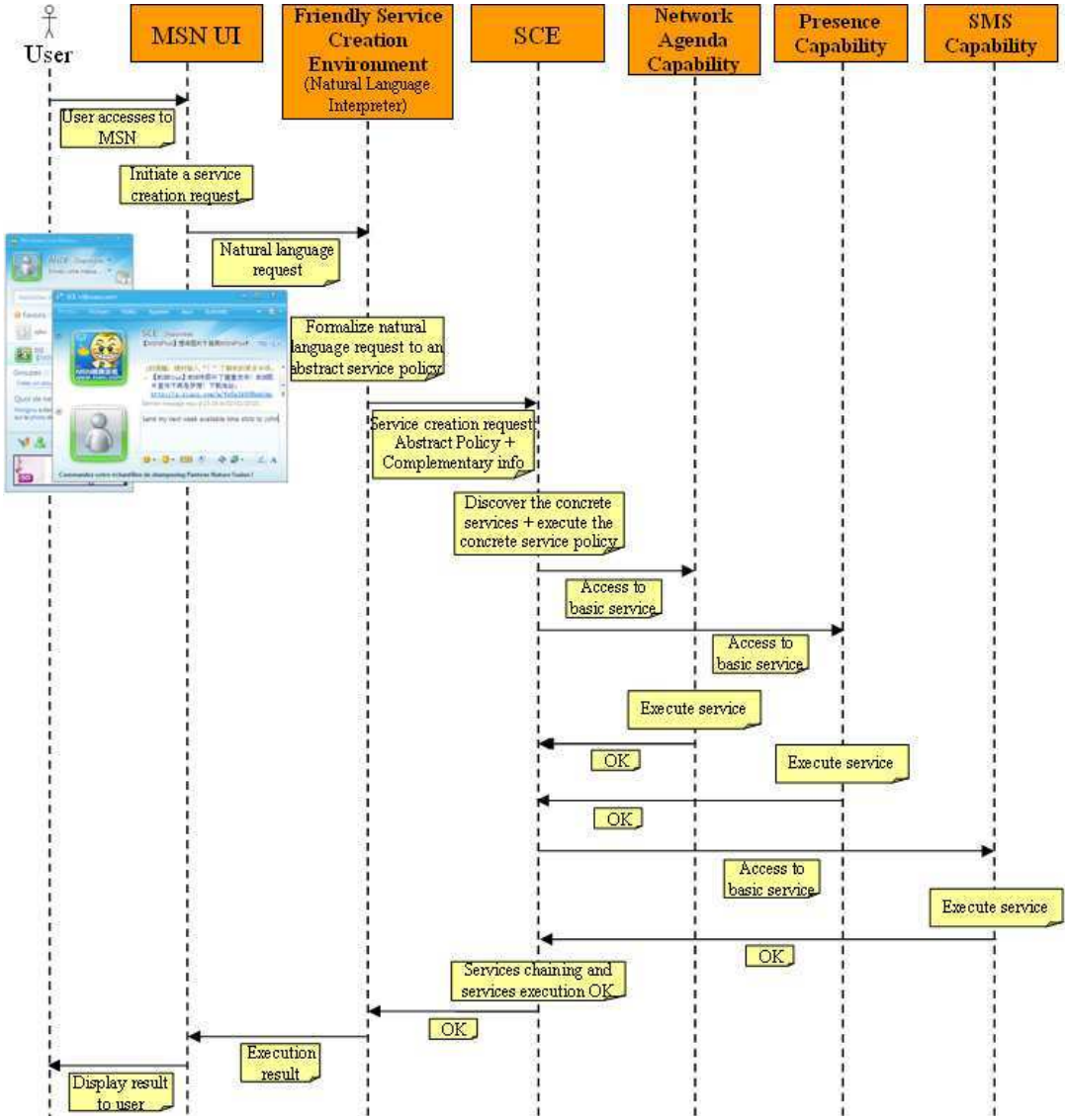
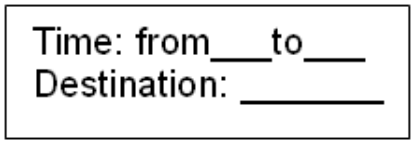


Figure 3-19: Simplified call flow for service creation and execution through SCE

Moreover, the user can store her/his created service by entering the "Save" for replying the corresponding question asked by this MSN robot. Then, the previously defined abstract composite service profile and the concrete composite service profile specified by Service Comparator are stored in Service Profile Repository. This created composite service is also materialized as a contact displayed in the user's Contact List. Thus once the user clicks on this contact, the communication window displays the content below:



After the user enters the time and destination contact name, this service is re-executed as illustrated in upper part. This user created service is also empowered with the automatic update feature due to the automatic update mechanism introduced in Section 3.2.10. For example, if the selected concrete SMS service has been updated (e.g. augment the price for sending a SMS), this service update information invokes Service Comparator for reselecting a replaceable service with the same SMS functionality but with lower price for catering to user's preference requirements (e.g. the SMS price does not exceed certain limitation).

To create such composite services, the user is not required any development skill and does not need to know which exact services (e.g. SMS service provided by Orange, or by Telefonica, or even by a Mobile Virtual Network Operator(MVNO)) are selected form Atomic Service Repository. The service update process is also transparent for the end user and does not require any user intervention.

3.4 Summary

As reflected in this specific use case, the main interest of the proposed approach lies upon the enhancement of the user-centric feature, which has been recognized as a main trend in both Telecom and Internet worlds. Considering the user-centricity, QoE is a crucial factor that has to be taken into account. Many similar service creation platforms have been proposed by different communities. Nevertheless, most of them are still far from being popularized for the ordinary users due to their complexity. For example, MARMITE [64], a graphical composition framework that enables the creation of a mashup service by wiring the basic services, its authors tested their framework on a sample of six persons. The result was only three out of six could build a composite service successfully, and those three persons had knowledge either in development or spreadsheet. On the other hand, to achieve the goal of user-centricity, an automatic service composition mechanism is extremely important, since service composition is a complicated task. It is difficult for the ordinary user's to deal with the whole process manually, even if he/she has some development skill. These are the main reasons that spur us on to propose a service composition model which hides the backend complexity from users, and simplifies the service creation interface, and satisfies the service creation variety requirements. Consequently, a user having no application development skills can create her/his personalized service thanks to the previously introduced automatic creation and update mechanisms.

Evidently, to realize the above service composition model, an intelligent service discovering and selecting mechanism is indispensable by considering that the quality of the service composition result depends so much on the selected concrete services. Some current service discovery solutions mainly rely on technologies like Ontology [65], lexical database (e.g. WordNet [66]), and data mining [67].

The advantages of such enriched service composition model are explicit: it facilitates the service creation process for users, especially for non professional users, by enabling the automatic service selection, composition and monitoring. Thus, both users' experience and users' involvement level in the service creation process will be greatly improved. It reduces the service maintenance cost for service providers due to the automatic update mechanism, since each modification, either from the capabilities side or from service environment, will invoke the adjustment on the corresponding composite service automatically. Moreover, it enhances the cooperation among different parties (e.g. end-users, service providers, operators, etc.). Finally, it optimizes the service lifecycle and enables delivering innovative services at a more rapid pace.

Chapter 4 Service exposure in converged environment

Service exposure plays a significant role in the evolution of service composition, since enabling a service to be reusable means that this service needs to be known and accessible by users and/or developers. Various service exposure solutions have been proposed. However, current solutions are mainly targeting a specific group of developers and are not friendly enough to users. Taking these actualities into account, in this chapter, we propose three service exposure models for catering to the different service environment requirements. These proposed solutions complement the previous introduced service composition model, and enable user-generated applications to be created in a more open and sustainable marketplace. These frameworks enable both professional developers and non-professional users to easily reuse existing services, capabilities and resource to create new services, and thereby profit from a highly-personalized, meaningful communication and interaction experience.

4.1 Service exposure challenges statement

Nowadays, although a great deal of work has been done for service exposure as we introduced in Chapter 2 for the state of the art, service exposure approaches still suffer from a set of limitations. These limitations of service exposure hinder the further evolution of service provisioning. In the following part, we outline some challenges for current service exposure solutions, which we intend to address in our proposed service exposure models. These challenges are: convergence, user-centricity, scalability, controllability and cross-platform service information sharing.

Convergence: Most of current service exposure mechanisms within IT, Telecom and devices worlds are evolving rather independently. They are mainly designed for a specific group of developers and/or specific network. There is still missing such mechanism for cross domains service discovery and publication through a unified environment. Thus when a developer or user wants to create a converged service that involves Telecom, Web or device-based services, she/he has to search for services from among several different service platforms, understand the different service description patterns, as well as their underlying heterogeneities. The need to manipulate different platforms increases the complexity of integrating the heterogeneous services into a unified composite service.

User-centricity: In these years, the service exposure mechanisms have evolved from the traditional custom, proprietary and domain-specific application interfaces based model to the easier over-the-top model, which are attracting thousands of developers due to their simplicity and speed to market. However, most of current service exposure solutions aim at the professional developers, and require the users to have at least certain development skill or network knowledge for using these exposed services. Consequently, even some so-called user-centric service creation platforms, as they rely on these service exposure solutions, they are still too complicated for ordinary users. To encourage user involvement instead of exclusivity and really include non-professional users, the user-centric feature needs to be significantly enhanced from the service exposure aspect.

Scalability: Most of the current service exposure solutions focus on using centralized systems (e.g. UDDI) for storing service information and managing service access. However, when the number of services that are exposed to developers and users increases drastically, it will degrade the service discovery efficiency. In order to improve user experience for the service creation and discovery, the scalability of a service exposure and discovery mechanism needs to be further improved.

Controllability: When the Telecom operators expose their capabilities to external parties, they generally exert the control over the access to their exposed capabilities and network assets. This controllability feature should be extended to other kinds of services (e.g. Web services, user generated services, or device-offered services) for avoiding the possible service outrages caused by unauthorized, as well as unintended usage by 3rd parties, developers or users.

Cross-platform service information sharing: There are many service exposure platforms exiting in the network, these platforms are generally independent from each other. The actuality that most of them adopt the walled-garden paradigm for protecting their revenues from being shared by others, and the lack of a unified service information sharing model, leads to the limited interoperability among different platforms. That inevitably leads to the isolated service information islands, and the redundancy of the service information storage. How to enable the service information sharing among different platforms, to reduce the resource redundancy, and to allow a user to reuse the services held by different service exposure platforms seamlessly, are questionable for the service exposure evolution.

Taking these challenges into account, we start by defining a centralized service exposure model, which mainly addresses the challenges of convergence, user-centricity and controllability. This centralized service exposure model specifies the basic service exposure and discovery processes for the different kinds of services in the convergent environment. In order to improve the system's scalability, enhance the service

discovery efficiency, enable the cross-platform service information sharing, and resolve the shortcomings of the centralized solution, we then propose two P2P based distributed solutions for service exposure.

Table 4-1 summarizes our objectives and contributions of these three service exposure mechanisms.

Solutions	**Objectives (Challenges addressed)**	**Contributions**
Centralized service exposure model	- What a converged service exposure environment looks like? - How to enhance the user centric feature in such converged service environment? - How to enforce the control on the exposed services?	• Enhance the user-centricity feature by providing diverse service publication and discovery tools that are satisfying the different level users' requirements, especially by adopting the widget as the basic service publication and discovery tool for non-professional users. • Enhance the convergence feature by providing a unified access to the services derived from different domains. • Enable the cross-domain service interoperability regardless of the underlying heterogeneities. • Enforce the control on the exposed services. • Specify the basic service publication and discovery processes in a convergent service environment. • Improve the service discovery efficiency and accuracy by adopting unified semantic annotation on both service description files and service discovery requests.
Hierarchical Chord based P2P service exposure model	- How to resolve the limitations in the centralized service exposure solutions?	• Reuse the basic service publication and discovery processes defined in the centralized solution for the different kinds of services. • Improve the scalability and service discovery

	- How to improve the scalability of a service exposure system? - How to enable the cross-platforms service information sharing? - How to enable ambiguous searching in Chord based P2P system?	efficiency of the service exposure system by adopting the structured P2P architecture for the service information storage and sharing. • Respect the diversity and the autonomy of the existing service exposure platforms while enabling the seamless cross-platforms service information sharing. • Avoid the single failure point by introducing redundant storage in the Chord based system. • Enable the ambiguous searching in the system, which is currently indigent in the structured P2P systems, by introducing the abstract service publication and discovery, as well as the hierarchical node and resource organization in a structured P2P system.
Triplex-layer based P2P service exposure model	- How to expose the device-offered service, including both powerful and weak devices, and thus involve them in the service composition paradigm? - How to reuse the device gateways for the device-related service exposure? - How to improve the ambiguous searching in the P2P based environment, meanwhile guarantee the high service discovery efficiency?	• Extend the services that are used in service composition to a larger range – it not only includes the traditional Telecom and Web services, the user generated services, but also includes the devices-offered services, no matter they have or do not have direct IP access to the network. • Further ameliorate the ambiguous searching in P2P network by adopting the unstructured P2P system as the basic service information sharing system, meanwhile, guaranteeing the high service discovery efficiency. • Improve the service discovery efficiency and accuracy by introducing the SON layer and the dependency layer over the underlying

		unstructured P2P layer. • Involve the weak devices, which have no direct IP access to the network, by using the distributed gateways to delegate them to the external parties, and introducing the two phases based service exposure (Local Device Exposure and Global Device Exposure) for the weak devices.

Table 4-1: Objectives and contributions of three service exposure solutions

The relationship among these three service exposure mechanisms, as well as their relationship with the previously introduced service composition management model can be summarized as Figure 4-1 shows.

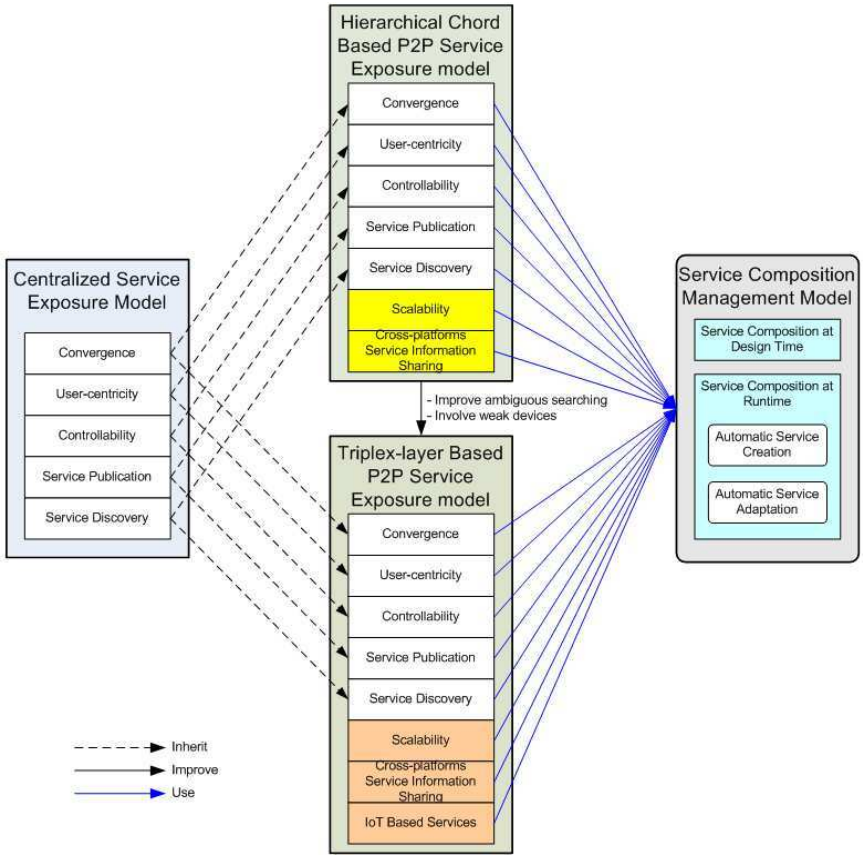


Figure 4-1: Relationships among three service exposure solutions

These three proposed service exposure solutions are illustrated in the following sections with the relevant usage discussion and performance analysis.

4.2 A semantic enriched service exposure model in converged environment

As introduced in upper part, one of our objectives is to propose a converged service exposure framework for a variety of services, including Telecom/Web/device/user-generated services, thereby enabling a new ecosystem for the creation of innovative services through user-centric fashion. Nowadays, most of current solutions (both in Telecom and Web worlds) focus on using the centralized systems for storing the service information and managing the service access. One of the obvious advantages for these centralized solutions is that they are easy to be implemented and enforce the monitoring for the resource held on them. Consequently, we start by proposing a centralized service exposure framework. This framework is intended to be one in which even non-professional users can easily reuse existing services, capabilities and resources for creating new services regardless of the underlying heterogeneities and complexities.

4.2.1 Overall architecture

The reference model is shown in Figure 4-2. This proposed service exposure framework acts as an intermediary for the adaptation and abstraction of the services provided by Telecom, Internet and Smart Space devices, and enables the convergence of the different kinds of services.

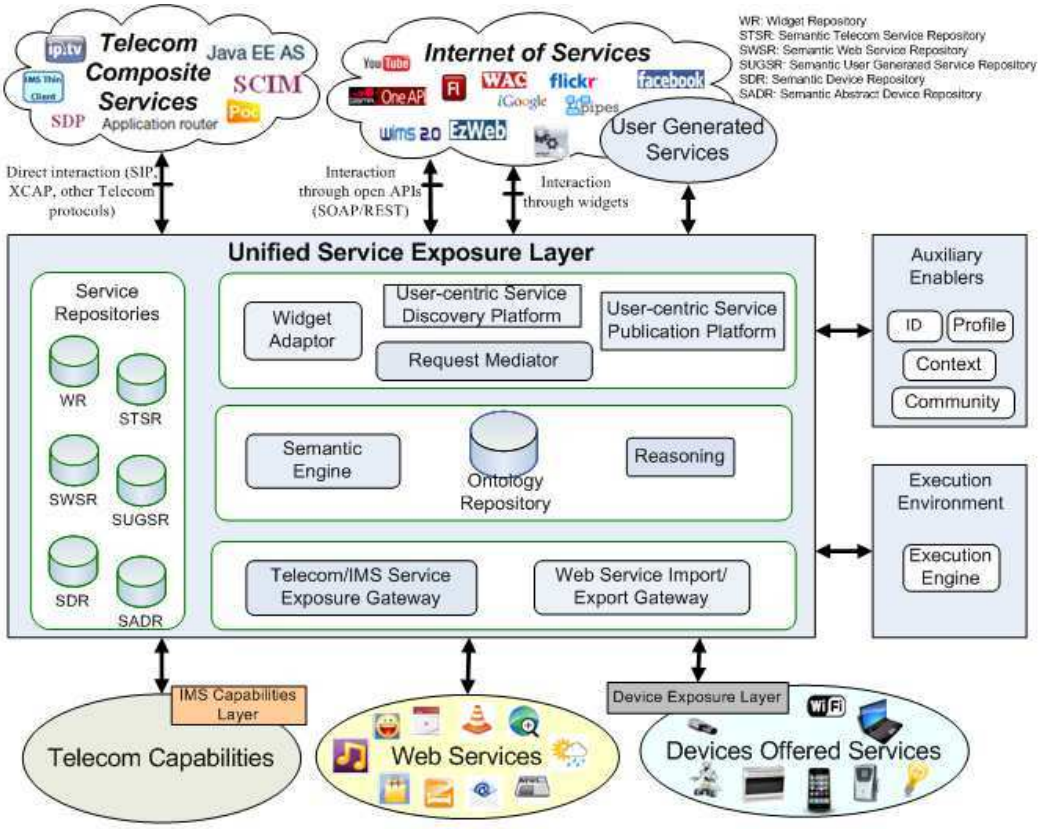


Figure 4-2: Overall architecture for converged service exposure

There are four groups of entities that can be identified:

- User centric adaptation group: User-centric Service Discovery Platform, User-centric Service Publication Platform, Request Mediator and Widget Adaptor;
- Semantic enrichment group: Semantic Engine (SE), Ontology Repository and Reasoning;
- Service exposure and access control group: various exposure gateways;
- Service repositories: a set of repositories for storing the semantic-rich service description files.

There are two types of users are classified for using these exposed services:

- Professional users: the users who have some software development skill and/or have the network related knowledge, thus they can use some complicated integration environments to create new services.
- Non-professional users: the ordinary users who do not have any development skill. Any development and network complexity should be transparent for them, and the service discovery, publication and creation environment should be as simple as possible.

During the service lifecycle, all of the entities inside the service exposure platform collaborate tightly to enable a seamless interaction among the heterogeneous services, and to facilitate their reuse by both professional and non-professional users. In the following sections, the cooperation among these entities is illustrated by considering the different kinds of services' exposure and discovery.

4.2.2 Service exposure processes

4.2.2.1 Service exposure for Telecom services

Generally, three kinds of users/developers access and reuse Telecom services and capabilities:

- Telecom-savvy developers
- Web service familiar developers
- Non-professional users

Accordingly, Telecom services exposure should satisfy the different levels of user requirements by providing different interfaces:

- Native Telecom-based interfaces
- Web-based open APIs (e.g. SOAP, REST based APIs) which incorporate with Web mashups directly
- User-centric interfaces (e.g. widget) which allow non professional users to integrate the Telecom capabilities into Web 2.0 services easily

Figure 4-3 presents the interactions among the different components for the Telecom service exposure. From this figure, it can be noted that the Telecom service exposure mainly relates to two processes: Telecom service invocation (access to services) process and the service publication and discovery process. In order to well explain our Telecom service exposure strategy, these two processes are illustrated in the following subsections respectively.

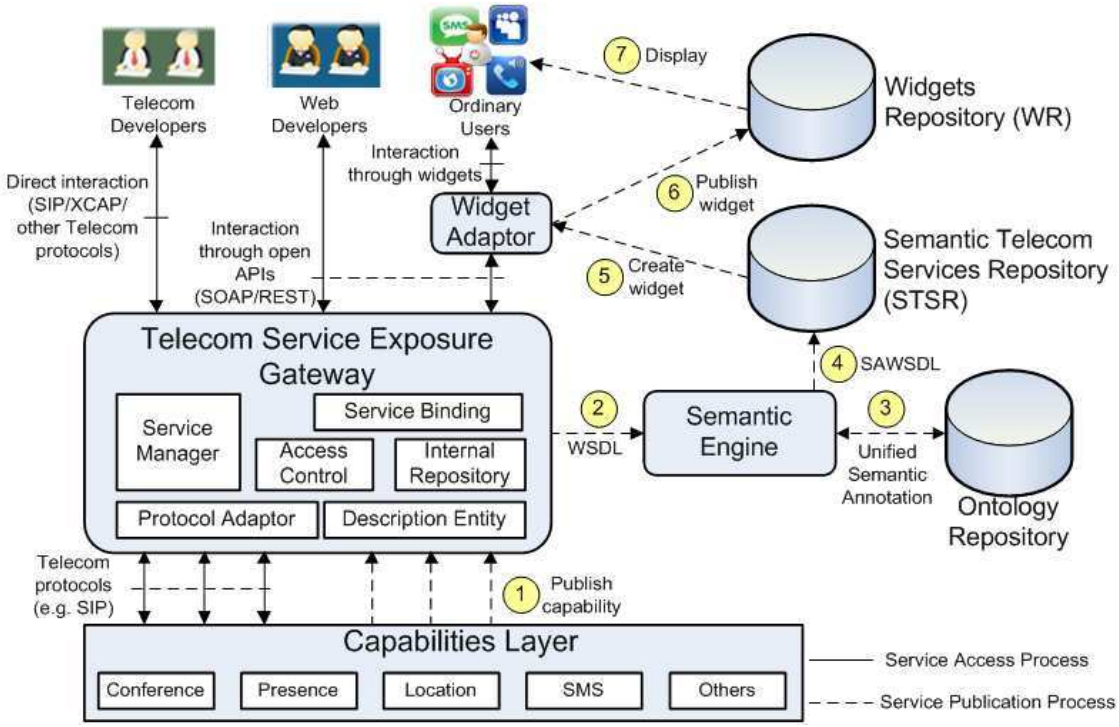


Figure 4-3: Telecom service exposure

4.2.2.1.1 Telecom service access and invocation

As shown in Figure 4-2, Telecom Service Exposure Gateway (TSEG) is used to connect the Telecom capabilities with outer-world applications, which may be developed by professional Telecom or Web developers, or even by some non-professional users. Thus the main functionalities of TSEG are providing a network abstraction function and exposing Telecom capabilities through APIs. Briefly, TSEG firstly intercepts the incoming Web service based requests (e.g. HTTP), and Protocol Adaptor inside TSEG translates the requests into Telecom-specific protocol based messages (e.g. SIP). These translated messages are then forwarded to the corresponding components for the service invocation. Such service exposure gateway can refer to the Parlay X Gateway specification which defines a set of powerful but simple and highly abstracted Telecom capabilities for facilitating both Telecom and IT developers to generate new applications.

Since an operator's APIs is used not only by internal developers, but also by third-party developers and non-professional users, consequently an access control is considered as necessary for guaranteeing the network security. That means when TSEG receives a service invocation request, Service Manager retrieves the corresponding service information from Internal Repository, and simultaneously contacts with Access Control entity in order to verify user's identity or credentials. After this verification, TSEG then performs

the necessary authorization and authentication. According to the different authentication and authorization results, the different users are able to access to the different network capabilities. Moreover, Service Manager is also in charge of routing the Telecom capability invocation requests to the corresponding internal points according to the service information extracted from the Internal Repository.

Protocol adaptation is another responsibility of TSEG. That is because the Telecom capabilities can be reused by other Telecom applications or Web based applications, in this case, the incoming requests can be Telecom protocols based or Internet protocols based. If the incoming request is Telecom protocol based, the request can be understood directly by TSEG and enable it to perform the corresponding actions for the request, however, if the incoming request is Web based (e.g. HTTP), it needs to be adapted by the corresponding Telecom specific protocols such as SIP based message. This requirement is handled by the Protocol Adaptor entity. In the case that the service capabilities access interfaces are already exposed using the Web protocols, this step can be skipped.

Internal Repository in TSEG is used to store the actual location and relevant confidential information for the access control. It thus hides the network topology and details from the outer world. In fact, there are two kinds of service description repositories: the internal repository, and the global repository (Semantic Telecom Service Repository) for storing the general functional and non-functional descriptions with semantic annotations that are retrievable by external users and developers. The two repositories provide a double protection for the underlying Telecom network, since malicious attackers cannot locate the internal servers, and the malicious messages can be intercepted and handled by TSEG.

Taking the non-professional users into account for the service creation, this Telecom service exposure solution should provide a more user-friendly way to access the Telecom capabilities. Widget is one of the solutions, especially after the emergence of some widget based service composition tools such as the EzWeb widget linker and widget communicator. Widget Adaptor and Widget Repository are the relevant components for addressing the aim of providing user-centric solution for service exposure to non professional users. These two entities host and serve the Telecom widgets. They provide the Telecom features by interacting with the interfaces exposed by TSEG. Consequently, if user wants to use a Telecom capability through a widget, the request is firstly forwarded to Widget Adaptor. Widget Adaptor formalizes the request according to the predefined rules, and then sends the request to TSEG for the following execution.

For the Telecom developers, since their applications mainly use the same technologies or protocols as in Telecom functions, thus the relevant messages invoke only Access Control instead of invoking all the functionalities provided by TSEG.

4.2.2.1.2 Telecom service publication and discovery

A service exposure platform should not only allow the user generated or third party generated services to access network capabilities and resources, but also should enable rapid and quality-assured service discovery, and provide supports for service creation, service orchestration and deployment. This requirement calls for the possibility of publishing a Telecom capability in a commonly understandable and comprehensive way, thereby making it discoverable and usable by other services. Such a service publication process can rely on WSDL, for example.

When operators publish their capabilities, they need to publish the functionality and access requirements of their capabilities to Description Entity inside TSEG. This service description file is then split into two description files: internal and external. The external file is transferred to SE. The semantic annotation is added to the external service description file by utilizing the predefined domain ontology stored in Ontology Repository. Finally, the semantic annotated service description file is stored in a central service registry (e.g., a UDDI registry). The mapping information for maneuvering between the internal and the external files is retained in the internal repository. This means that users can use similar methods as Web services for discovering these Telecom services.

4.2.2.2 Web service exposure

For the web service exposure, we reuse the SOA based solution which has been introduced in Chapter 2 as the base. Currently, most of the conventional web service discovery solutions are keywords based, that limits the service discovery veracity and efficiency, especially when there are a huge amount of services are stored in the repository. For addressing these limitations, the semantic annotation mechanisms have been widely applied in service description for providing a new way to represent the knowledge that empowers advanced reasoning mechanisms [68]. However, there is no universal domain ontology that can act as a shared Web service vocabulary dictionary for the annotated terms in service semantic description. Thus it incurs the problems for integrating various existing services provided by different service providers. Obviously, if service instances correspond to the same service provider, the composition and discovery of the relevant services are feasible. However, if a developer or a system tries to compose several services derived from different service providers with different semantic annotations, it would indeed be a burdensome task, especially for the case of automatic service composition by the backend service composition engine, in which it requires all the service description files are machine understandable.

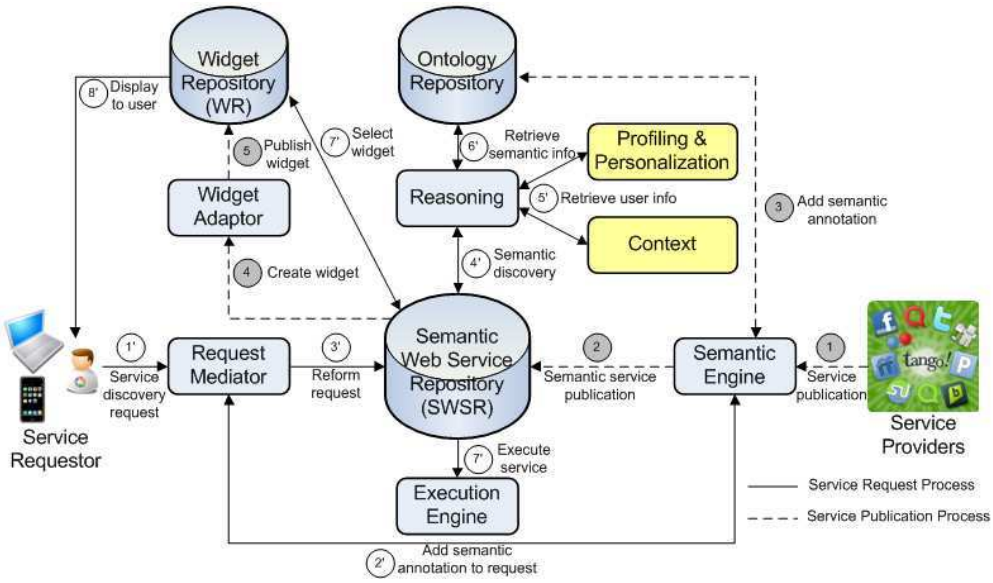


Figure 4-4: Web service exposure

Figure 4-4 illustrates our proposed solution. When service providers publish service, they can use their preferred service publication mechanisms and service description languages (e.g. WSDL 1.X, WSDL 2.0, WADL, etc.). This service description file has to pass through SE that can be aware of the different functionalities and structures described in the service description file. SE then contacts Ontology Repository, which identifies the semantics for describing service's functionalities, workflow and invocation, to translate the received service description file into a unified and semantically-annotated service description file (e.g., SAWSDL). During this process, a user friendly interface may be displayed, which asks the service provider to add some additional information including the non-functional features (e.g. pricing model, QoS, SLA, contact information, etc.) of the service. For the services based on other standards which can not be parsed by SE, a dedicated graphical interface is displayed for asking the publisher to do an additional description step.

If this published service is represented as a widget, the semantic service description file is sent to Widget Adaptor, and then a corresponding widget is generated and stored in Widget Repository.

The semantic service annotation is not only used for the service description, but also used for the service request. Since the users requesting services are likely to possess a wide range of technological competence, they may be the professional developers who are service annotation technologies savvy, or they may be the non professional users who just want to invoke certain services or compose the services in a user-centric way. Consequently, the incoming service request could be in the format of simple natural language text (e.g., "I want to send a SMS") or it could be compliant to the service annotation ontology. It

needs an additional mechanism to make the disparate incoming requests to be understandable regardless of their original format. Therefore, Request Mediator has been added to make incoming requests easy to understand. When Request Mediator receives a service request, it transforms the request into a structure that corresponds to the semantic service annotation mechanism adopted for service publication (e.g. “I want to send a SMS” is transformed to “Send SMS”). In the case that the service providers and the service requestors already use the same domain ontology, the request will pass Request Mediator without invoking the transforming function.

Then Reasoning engine collaborates with Semantic Web Service Repository and Ontology Repository for the discovery and selection of appropriate services in accordance with the relevant information retrieved from Profiling & Personalization and Context enablers. Finally, a list of corresponding services is sent to the user for a further selection, or a selected service is sent to Execution Engine to be executed directly according to user’s configuration and the tools used (e.g., Natural Language Composition Environment). If a user wants to select the relevant widgets, the list of selected services is sent to Widget Repository, and then a list of the corresponding miniature widget icons is displayed to the user.

4.2.2.3 Device service exposure

Along with the popularization of IoT, devices are more and more involved in service composition, providing real-time data about the physical world via gateways or service-enabled interfaces (e.g., SOAP or RESTful APIs) [69]. To make use of these device-offered services, the first step is to make them machine-meaningful. Generally, devices (e.g., intelligent equipment, sensors, actuators, etc.) interconnect with each other in smart space. Their resource-constrained aspects (e.g., limited computation, communication and storage capabilities) make them more constraint than Telecom or Web services. Moreover, compared to Telecom and Web services, devices’ availability may change more frequently, since they can appear and disappear from the network on the fly. This highly dynamic information is impossible to be encapsulated in a service description file stored in a global repository, because this leads to an unbearable level of network traffic. Therefore, some form of gateway is adopted to enable communication between these devices and external applications, and to handle the runtime discovery of devices and their services. In consideration of these scenarios, we divide the device service exposure into two levels: Local Service Exposure relying on local gateways, and Global Service Exposure (GSE) which reports the basic information about the devices and their services to the global repositories via local gateways.

To facilitate the introduction of device service exposure, in this section, we adopt the DIYSE Gateway [70] as an example for local service exposure, as illustrated in the amplified portion of Figure 4-5.

Evidently, one of the functionalities of this local gateway is the device exposure which hides the heterogeneities and complexities of the underlying networks and devices by exposing them in a common way.

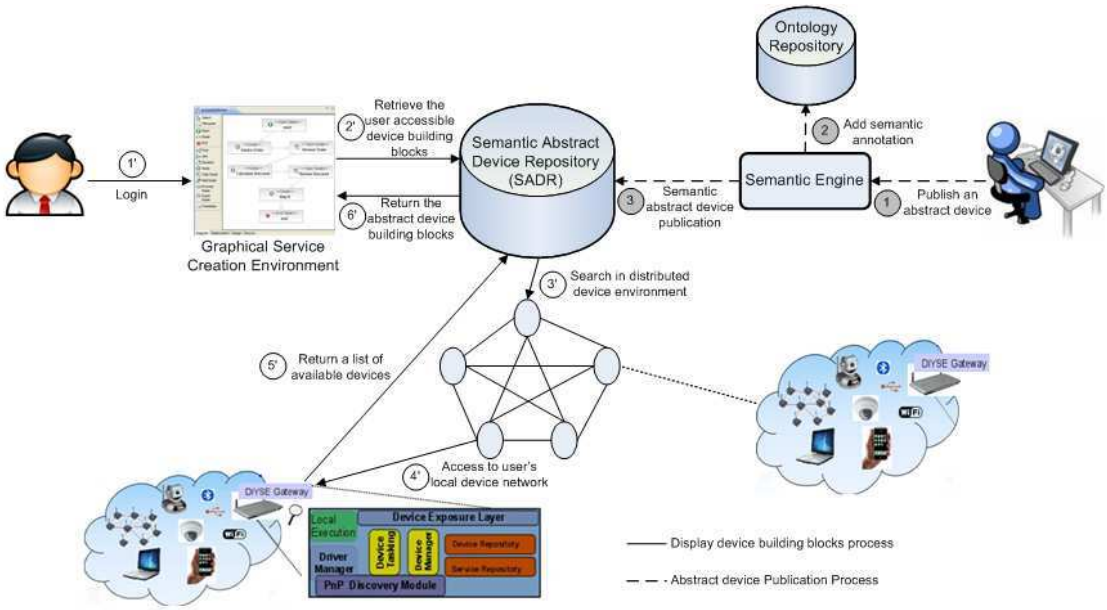


Figure 4-5: Service exposure for devices

From the viewpoint of the global network, there are many local gateways dispersed in the network while managing their own local networks using disparate technologies. Furthermore, when a user or developer is creating a new service, she/he may not be in the same network as devices and is not necessarily familiar with the particular characteristics. This distributed device component information needs to be published in a globally-comprehensible manner to enable devices capabilities to be discovered even when they are located behind the local gateways, executed and composed in a unified manner.

Correspondingly, in term of the service publication process, there are two kinds of device capability publication processes: abstract device capability publication and concrete device capability publication. An abstract device means a virtual device that only contains the generalization information of one kind of devices, and can map to several concrete devices. For example, “camera” is an abstract device, and it can map to several concrete devices provided by different manufacturers such as Sony, Logitech, Canon, and owned by different persons such as Alice and Jon. The impetus for proposing such abstract device publication is that the number of the devices in the network is much larger than Telecom and Web services, while most have the same or similar functionalities. Storing all of these concrete devices in a global repository will make it difficult to be discovered and managed.

Abstract device publication: The abstract device publication is handled by professional developers or service creation platform providers. They send a device description file which only contains the generalization information of one type of devices to SE. SE contacts Ontology Repository to add the necessary semantic annotations. The transformed service description file is then sent to the global Semantic Abstract Device Repository as shown in Figure 4-5.

Concrete device publication: device-offered service generally can be viewed or controlled by user via a Web browser, for example, as specified in Universal Plug and Play (UPnP) for presentation. This makes it easy for users to get the service description file via the same device-monitoring page. Once the device description file is retrieved, SE adds the semantic annotations to the file and sends it to Semantic Concrete Device Repository. Users can also publish their devices as widgets. This situation is actually the publication of a concrete rather than an abstract device. Such concrete device publication process is as shown in Figure 4-6.

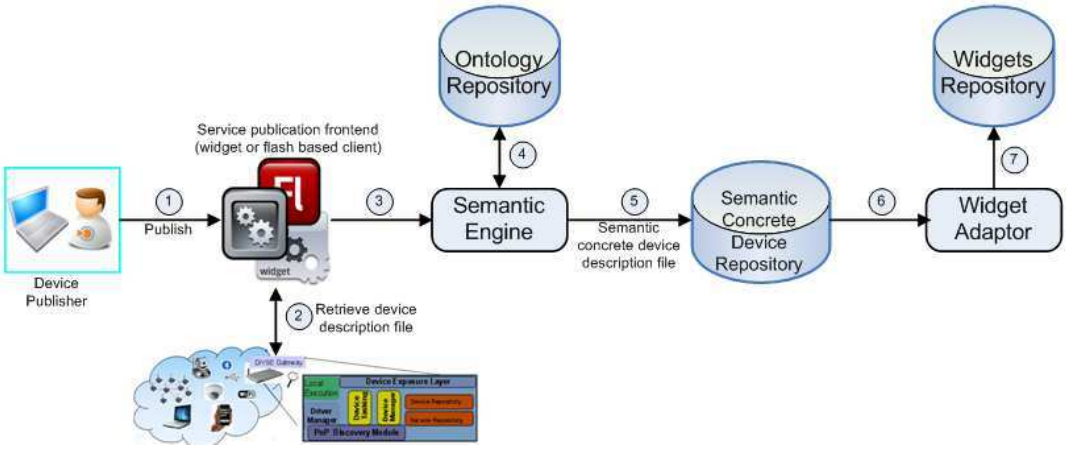


Figure 4-6: Publish a concrete device as a widget

If a user wants to reuse device-offered services to create a service, when she/he logs into a graphical Service Creation Environment (SCE), SCE contacts GSE framework with user's ID. GSE framework can then search the distributed gateways using Flooding method to determine which specific local gateways this user can access and which specific devices behind them can be invoked. The corresponding abstract device building blocks are then displayed in user's SCE. After the user has created a composite service by chaining the different building blocks, the service discovery and selection processes are handled by the corresponding service discovery components: Telecom and Web based services are discovered and selected by GSE framework's components as indicated above, and for device-offered services, the service discovery request is transferred by GSE framework to the corresponding local gateways, and then Device Repository and Device Discovery Module inside a local gateway select the relevant device description files and send

back these selected files to GSE framework. Finally, the Reasoning component cooperates with GSE framework, as well as Profiling & Personalization and the Context enablers to select the most appropriate concrete device(s) to the user.

The above introduced device service exposure mechanism and the adoption of gateways for local device management imply that the execution of a device related composite service is different from the single Telecom or Web composite services' execution. In order to identify such differences, a simple scenario of a composite service, which reuses both the ordinary Telecom services and the device based services, is illustrated as the followings. This service is triggered by a device instead of by a user, which is different than the traditional user generated services.

"Alice creates a service by drag-and-drop method in a graphical service creation environment. This service is that when her house's doorbell is ringing, the camera on the top of the door is turned on, and a MMS embedded with the short video captured by the camera is sent to her immediately."

Figure 4-7: Service composition scenario for device offered services

Different from the Web or Telecom services, the invocation of a device is handled by a local execution engine inside the DIYSE Gateway instead of the global execution engine. Moreover, most of the device activations are event driven, for instance, the activation of a doorbell invokes the execution of a composite service which relates to it, and their presence in the network are taking place only for a short time, which is quite different from the traditional Telecom and Web services that are always in running state. Consequently, in order to reduce the time consuming to execute such a converged composite service, the composite service logic is stored in a local service repository. That is because, if the composite service is only stored in the global service repository, when the doorbell is activated as illustrated in above scenario, the DIYSE Gateway has to contact with the global service repository to find out the corresponding service execution logic. In the case that there are a great number of services containing the doorbell service, it may cause a long delay for the composite service execution.

After the composite service file is retrieved from the local service repository, the local service execution engine analyzes the composite service logic extracted from the composite service file. If it only contains the local devices, this composite service is executed by the local execution engine, if there are also some Web or Telecom services involved, the local execution engine forwards the service execution logic to the global execution engine, and then the global execution engine cooperates with the local engine to

execute this composite service. Just as illustrated in the above mentioned scenario, when global execution engine receives the composite service logic, it knows that the following step is to execute a camera, which is a device based service, thus it sends the camera execution request to the local execution engine. Then the camera device is invoked and a short video is captured, the result is sent back to the local execution engine. Subsequently, this received video is transferred to the global execution engine. After the global execution engine receives the response from the local execution engine, the global execution engine decides the following step for this composite service – send a MMS which is embedding the received video to Alice. The call flow for such device related service execution process is illustrated in Figure 4-8.

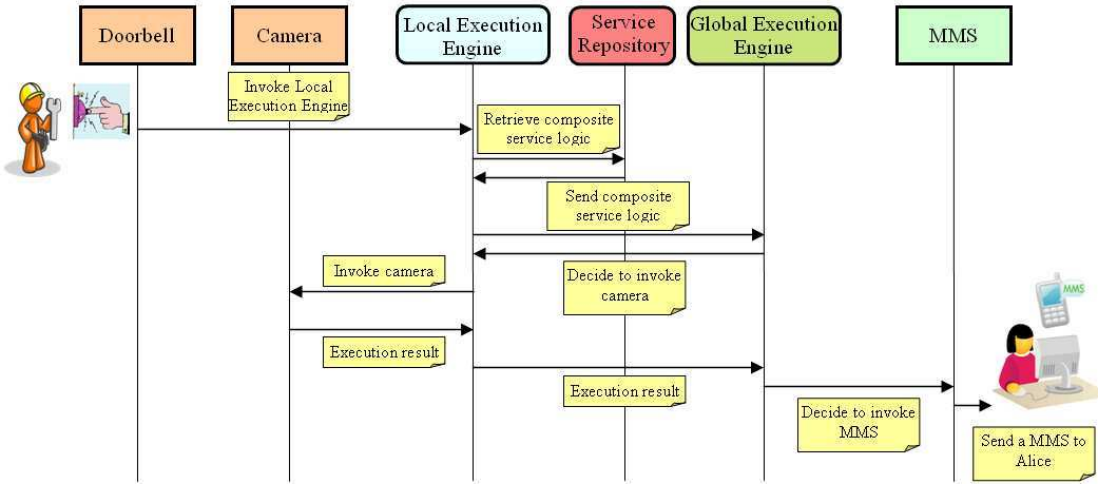


Figure 4-8: Call flow for the execution of a device related composite service

4.2.2.4 User generated service exposure

One goal to propose this service exposure model is to enhance the user-centricity feature. When a user generates a new service, it is normal that she/he would like to share it with others, and/or this newly-created service may be reused by other services.

The main difference between the service exposure of user-generated services and the Telecom/Web/device service exposure is that the targeted actors are the development amateurs or non-professional users. That requires the exposure and discovery processes to be as simple as possible and the underlying mechanisms to be transparent to users.

Figure 4-9 presents our solution for this kind of service publication and discovery processes.

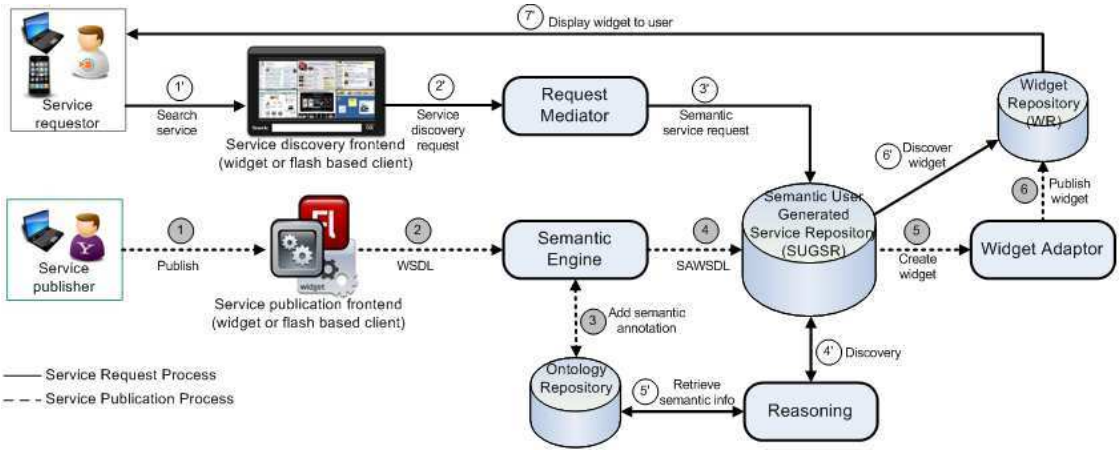


Figure 4-9: User generated service publication and discovery

Service publication process: before the publication process, the user has to be identified by the system, and his/her information (e.g., publisher's name) is automatically added to the semantic service description file.

Next, the user initiates the service publication process by interacting with a user-centric service publication front-end, which can be represented as a widget or as a flash-based client. She/he can provide the URL of the functional description of her/his service to the service publication client if she/he has already created this service description file (e.g. URL of a WSDL file) as shown in Figure 4-10.

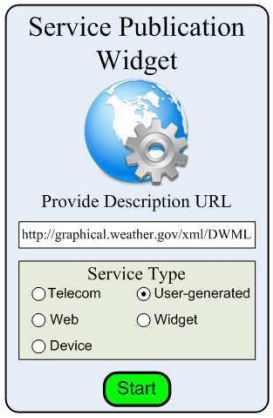


Figure 4-10: A UI example for the service publication front-end

Considering that the creation of a service description file may be difficult for non-professional users, this service publication front-end should also provide an easier way to generate it. For example, a template could ask the user to fill in the necessary information about her/his created service (e.g., service's name, input/output, etc.) as shown in Figure 4-11. This information would then be extracted and completed by the default collocation information to create a formal functional description file. Some non-functional

parameters and business models (e.g., the cost for use of this service) can be added to the service description file.

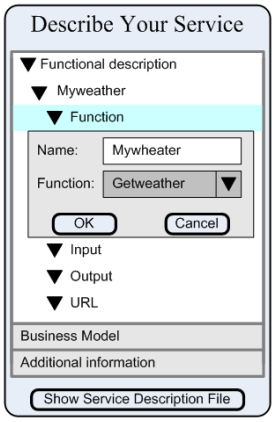


Figure 4-11: A UI example for the service description file generation front-end

The service description file is then transferred to SE for being parsed and added with the unified semantic annotations. Finally, a semantic service description file is generated by linking the functionalities with semantic concepts, and is then sent to Semantic User Generated Service Repository. When a service cannot be parsed automatically (e.g. the service description format is not supported by current SE), a page on which to add the semantic annotations manually will pop-up with a list of recommendation semantic concepts for the different options. If no relevant semantic concept matches the service, the ontology update process, which is out of the scope of our current research, is invoked for adding a new concept. Afterward, the publisher information is added to the semantic enriched service description file automatically according to user's login information. Such information can be modified manually if the user wants to. This information is not only used to identify the ownership of this service, but also can be used for the recommendation based on authorship for the discovery process. For example, when a user wants to install or contract a service, SE may be able to recommend her/him other services which are made or published by the same person.

Service discovery process: This functionality can be invoked through an embedded widget or a flash-based client in user's Personal Service Environment (PSE) or SCE. It supports both natural language and complex searches based on semantics. Therefore, similar to Web service discovery described above, the incoming service request is translated into a unified semantic-based request by Request Mediator, and then sent to Semantic Service Repository. The displayed services vary depending on the users' intent: for users who just want to invoke a service, only the widget-based services are displayed in her/his PSE, since some backend services have no front-end with which to use it. For users who want to create new services, widgets, backend services, and the relevant UI components are represented as building blocks in SCE.

Moreover, some additional information, e.g. the publishers' names and the costs of these services, extracted from the semantic service description files, are also displayed to user.

4.2.3 Usage scenarios

In this section, we present the application of the proposed converged service exposure model to service composition model we introduced in Chapter 3 through several scenarios. In these scenarios, we explain how to create a converged service at design time and how to execute a composite service by collaborating with this service exposure model. These scenarios provide a global view how this converged service exposure platform works and how it provides the support for the service composition model.

4.2.3.1 Converged service creation

Figure 4-12 and Figure 4-13 present an example in which a converged service is created by using the service exposure model.

> *"Bob is a businessman who has to travel frequently. He is living in Paris with his wife. He wants to create a service to get the weather information of Paris, translate it to English, and then send it to his wife Alice by SMS. If the temperature is higher than 20°C, the heater in his house is turned off as well."*

The simplified service composition logic is as shown in Figure 4-12.

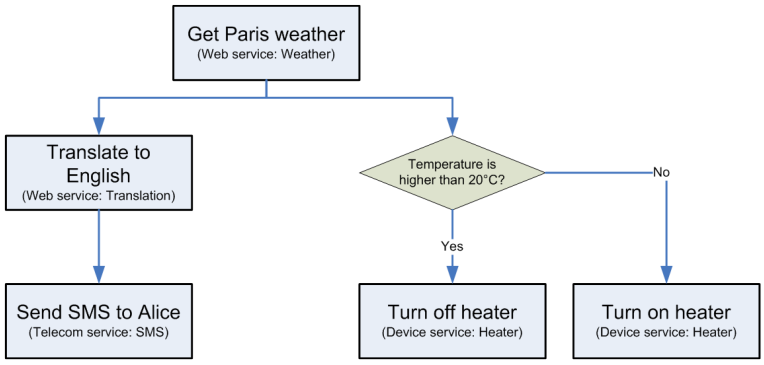


Figure 4-12: A simplified service composition logic

The description of each step for the service creation process is as follows:

1. Bob creates a composite service in a graphical service creation environment through drag-and-drop method.

2. The service creation request is forwarded to Request Mediator.

3. Request Mediator contacts with Semantic Engine for unifying the request to be a system interpretable format.
4. Semantic Engine contacts with Ontology Repository for adding the semantic annotation to the request, and sends back the result to Request Mediator.
5. Request Mediator sends the semantic-enriched service discovery request to the Reasoning component.
6. The Reasoning component extracts the names of the services which need to be discovered, and decides the discovery order for these services. In this scenario, there are four services need to be discovered under the following order: Weather (Web service) -> Translation (Telecom service) -> SMS (Telecom service) -> Heater (Device service). Thus the request for discovering a weather service is sent to Semantic Web Service Repository (SWSR).
7. A set of weather related services are discovered from SWSR, and these services' relevant information is sent back to the Reasoning component.
8. By contacting with Ontology Repository and according to the contextual information embedded in the service discovery request, a most appropriate weather service is selected by the Reasoning component
9. Another request for discovering a translation service is sent to SWSR.
10. A set of translation related services are discovered from SWSR and the relevant information is sent back to the Reasoning component.
11. The Reasoning component selects the most appropriate translation service from the received service list.
12. The Reasoning component notices that the next service needs to be discovered is a Telecom service, thus it sends a request to Semantic Telecom Service Repository (STSR) for discovering SMS related services.
13. A set of SMS related services are discovered from STSR, and the relevant information is sent back to the Reasoning component.
14. The Reasoning component selects the most appropriate SMS service from the received service list.

15. The Reasoning component deduces that the next service to be discovered is a device-offered service, thus it sends a service discovery request to Semantic Device Repository (SDR) for discovering a heater related service.
16. A set of heater related services which Bob can use are discovered from SDR, and the relevant information is sent back to the Reasoning component.
17. The Reasoning component selects the most appropriate heater related service from the received service list.
18. All the selected services' description files are sent back to the service creation front-end.
19. Service creation front-end extracts the necessary information from these service description files and creates an executable composite service by chaining the selected services together.
20. A service description file is generated by the service creation front-end through a service description widget. And this service description file is sent to Semantic Engine for adding the necessary semantic annotation.
21. After the semantic annotation is added to the service description file, this file is sent back to the service creation front-end.
22. Bob publishes the created composite service by sending the semantic-enriched service description file to Semantic User Generated Service Repository (SUGSR).
23. Once SUGSR stores this received service description file, it sends back a notification to the service creation front-end.
24. The service description file is further sent to Widget Adaptor for generating a widget automatically.
25. Once the widget is generated successfully, a script for the widget and a widget description file are added to Widget Repository.
26. A notification for the successful widget publication is sent back to Widget Adaptor.
27. A notification for the successful widget publication is further sent back to the service creation front-end.

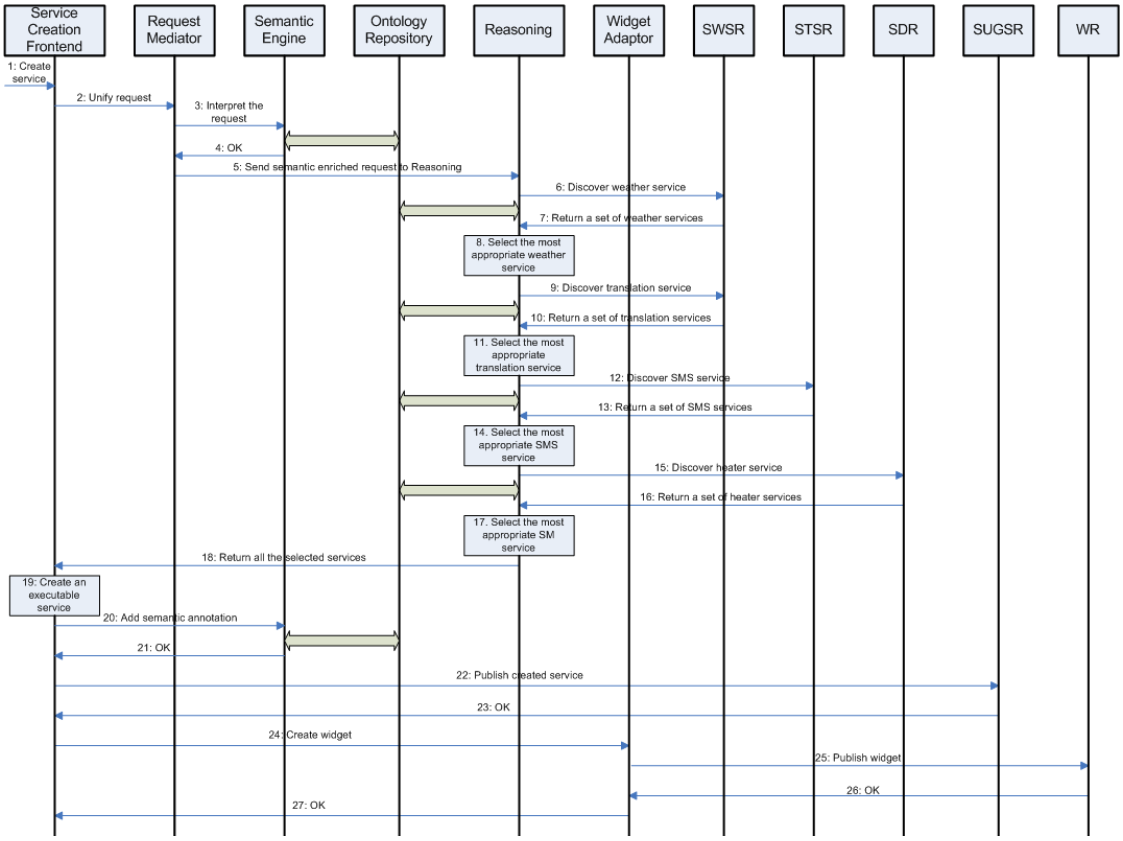


Figure 4-13: Call flow for converged service creation

4.2.3.2 Converged service execution

In this section, we explain the service execution process which is relying on the proposed service exposure model. We reuse the composite service we introduced in Section 4.2.3.1 as the base. That is:

> *"Bob creates a personal service: this service gets the weather information of Paris, translates it to English, and then sends it to Bob's wife Alice by SMS; moreover, if the temperature is higher than 20°C, the heater in his house is turned off as well. Bob creates a widget for this created service and names it as "My Weather Service". He adds this widget to his personal service environment. When Bob wants to invoke this service, he just needs to click the "OK" button in the widget"*

The call flow for this service execution process is illustrated in Figure 4-14, and the description of each step is as follows:

1. When Bob logs in his personal service environment, he clicks the "OK" button in the "My Weather Service" widget.

2. A service invocation request is sent to Global Execution Engine with the target service's ID.
3. Once Global Execution Engine receives a service invocation request, it extracts the service ID from the request, and initiates a service discovery request which embeds this target service ID. Then this service discovery request is forwarded to Semantic User Generated Service Repository (SUGSR).
4. According to the service ID, SUGSR discovers the service profile for the target service, and sends the service file to Global Execution Engine.
5. Global Execution Engine analyzes the received service profile, and determines the execution order for the relevant atomic services.
6. Global Execution Engine initiates a service invocation request, and sends it to the first atomic service (i.e. a weather service). It also memorizes the service execution state for each atomic service, and the contextual information for facilitating the following actions.
7. After the weather service is executed, it sends back the execution result (i.e. the weather information of Paris) to Global Execution Engine.
8. After Global Execution Engine receives the service execution result, it determines next atomic service to be invoked (i.e. a translation service), and sends another service invocation request to the selected translation service, and the weather information it obtained from previous weather service is also embedded in the service invocation request.
9. After the translation service is executed, it sends back the execution result (i.e. the weather information is translated to English) to Global Execution Engine.
10. Global Execution Engine continues the service chaining by sending a service invocation request to a SMS service. As this service is a Telecom service, which is exposed by a Telecom Service Exposure Gateway, so the request is first forwarded to the Telecom Service Exposure Gateway.
11. Once Telecom Service Exposure Gateway receives this request, it does the necessary access control, and determines which Telecom capability this request should be routed to.
12. Telecom Service Exposure Gateway routes the service invocation request to a SMS capability.
13. The SMS capability is invoked. It sends a SMS message with the information extracted from the received service invocation request to Alice, and then sends back a successful execution notification back to Telecom Service Exposure Gateway.

14. Telecom Service Exposure Gateway forwards the execution notification to Global Execution Engine.
15. Global Execution Engine determines the next service to be invoked is a device related service, so it sends a service invocation request to the corresponding Device Gateway.
16. When Device Gateway receives the service invocation request, it does the necessary authentication control according to the user information extracted from the incoming request.
17. If this user is authorized to use the devices behind this Device Gateway, the request is further forwarded to Local Execution Engine.
18. Local Execution Engine retrieves the information related to the target heater and determines the invocation method for it.
19. The heater in Bob's house is turned off.
20. A notification for the successful execution is sent back to Local Execution Engine.
21. The execution notification is further forwarded to Device Gateway.
22. Device Gateway sends back the service execution result to Global Execution Engine.
23. Global Execution Engine collects all the execution results for the relevant atomic services, and determines the composite service execution process is accomplished. It thus determines to terminate the service chaining.
24. A final notification is sent back to Personal Service Environment for indicating Bob that his composite service has been executed successfully.

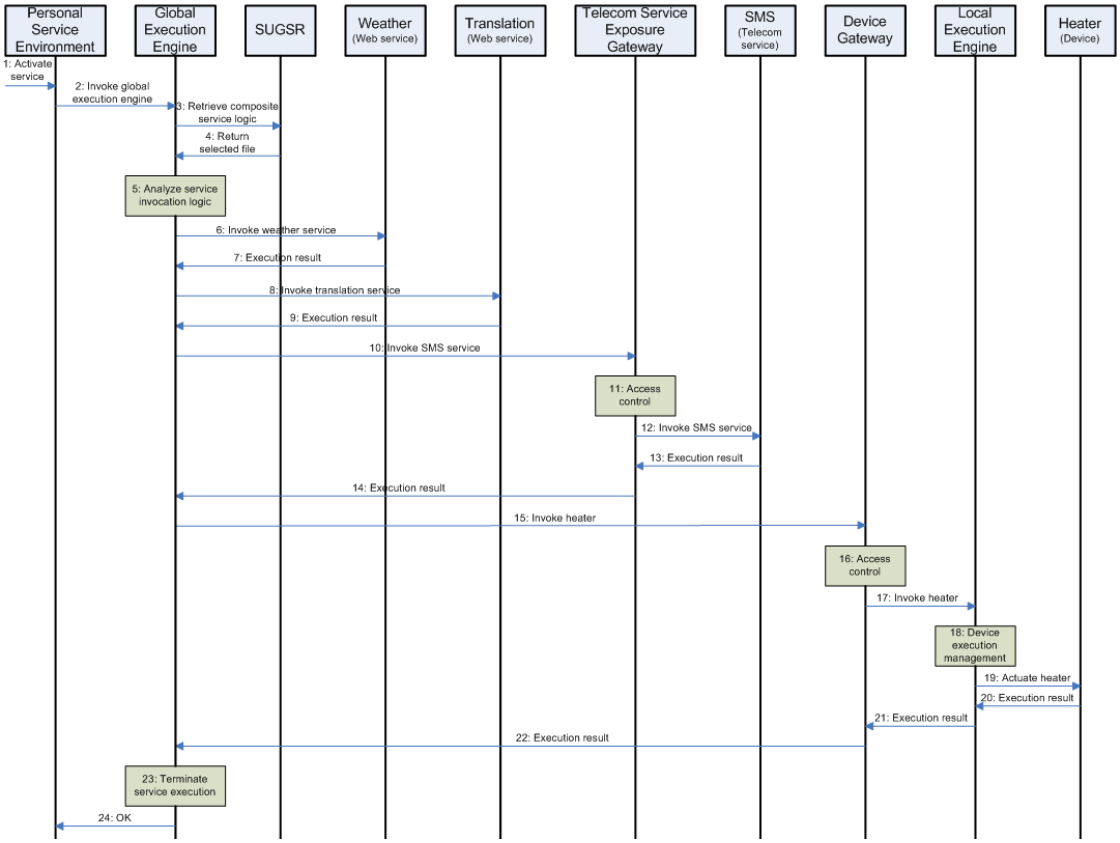


Figure 4-14: Call flow for converged service execution

4.2.4 Discussion

This service exposure model also can act as a complement component for the service creation through MSN robot use case that we introduced in section 3.3. For example, when Bob enters “If the outside temperature is less than 10°C, turn on the heater in my house and send my house current temperature to me”, an abstract service defining the involved functionalities is generated by Service Composer which cooperates with SE and Request Mediator. In this case, Web service (weather), device-based services (heater and temperature sensor), and Telecom service (SMS) are selected. Service Composer then contacts the service exposure platform to discover and select the most appropriate services for the abstract service by collaborating with SE, Ontology Repository, Reasoning, service description repositories and the relevant device gateways. After all the concrete services have been selected, MSN robot validates the concrete service logic and executes this composite service by invoking the corresponding atomic services (e.g., yahoo weather, Bob’s heater and temperature sensor in his house, Orange’s SMS). In this use case, several different kinds of services are invoked; however, the user only needs to use a unified interface to discover

services, instead of having to surf different Websites to discover the different services and spend time learning how to use them.

This proposed service exposure model relies on the enhancement of user-centric and convergence features, as well as on providing a unified access to Telecom/Web/device/user-generated services, meanwhile respecting the diversity of the service market. It thereby enables third-party service providers, developers and users to reuse the existing services, using the tools or technologies they prefer while following SOA, Resource-Oriented-Architecture (ROA) or Widget-Oriented-Architecture (WOA) principles. By separating service discovery and selection functionalities from SCE, this service exposure model not only releases SCE from the cumbersome service discovery task, but also enables the reuse of larger scale services, regardless of their underlying heterogeneities, thus further improving the sharing of different kinds of services among different platforms. It also provides the possibility of applying enhanced controls for service exposure in accordance with different operators' or service providers' requirements via gateways. The user-centric aspect is enhanced by using widgets and flash-based clients for service publication, discovery and composition, thereby greatly facilitating the service generation process which encourages more users to contribute to the service ecosystem.

By comparing with the traditional keyword based service discovery, the adoption of semantic annotations for both service descriptions and requests, improves service discovery efficiency and accuracy. Firstly, a semantic request by itself is more accurate than a simple keyword based search. That is because a semantic search may involve multiple concepts, and each concept relates to certain filter rule, thus that implies multiple filtering rules will be applied on a single request. Secondly, the semantic search is based on the concepts defined in an ontology, in which different concepts may be related to each other. Thus even no exactly matched services can be discovered, some proposals and recommendations for other services that may potentially interest the user, can be provided according to the relationship among the services, which are predefined by the domain ontology.

To fully realize this converged service exposure model, some challenges still require further attention. These challenges include defining a widely-acceptable common data model for all types of service descriptions, and persuading operators and service providers to share their service information. An efficient business model is also indispensable considering that the investments from operators and service providers for the technologies enhancement are revenue-driven. If end users could also profit from their contribution to new services, that would greatly encourage everyone to become involved in this new service ecosystem. The successful Apple App Store reveals this great potential. However, the single access to service exposure

platform may become a bottleneck when the number of services increases dramatically, and be susceptible to malicious attacks.

4.3 A P2P based distributed service exposure model

Most of the current service exposure solutions focus on using centralized systems (e.g. UDDI) for storing service information and managing service access. Such centralized systems are easy to implement and to monitor the resources held on them. However, they are also easy targets for malicious attacks, introduce a single failure point, require a high maintenance cost, and are not scalable and extensible enough to deal with the dynamic service environment. Moreover, centralized systems limit the interoperability among different platforms, which leads to many isolated services islands, and thus inevitably incur a resources redundancy.

Considering the above mentioned limitations of the centralized solutions, we propose a decentralized P2P based architecture to support distributed service exposure on a large scale as an alternative solution. This proposed architecture is hierarchical compared to current pure P2P solutions [80] - [94] with the aim of not being limited to a single specific service domain, and enabling the efficient exchange and interoperability among different kinds of services regardless of their underlying heterogeneities. This new method could lead the service composition paradigm to a new era.

4.3.1 Some backgrounds and related work for P2P

With the rapid evolution of P2P networks, P2P has become a promising solution for the exchange of resources with advantages such as scalability, robustness, and hardware/bandwidth economy. Generally, P2P networks can be divided into two main categories: structured and unstructured.

4.3.1.1 Traditional unstructured P2P based solutions

For unstructured approaches, such as the early Freenet [71] and Gnutella [72] for resource sharing, there is no rule for lodging an item in the network, and the items are distributed randomly over the peers. To find an item, either a central registry for the global indexing or some blind searching algorithms can be used. A central registry can easily become a bottleneck for a system when a great amount of resources have been stored in the P2P network. And the blind searching algorithms mainly can be classified into two types:

Breadth First Search (BFS) based methods and Depth First Search (DFS) based methods [73]. Two typical examples for these two types of searching algorithms are Flooding and Random walk.

In the Flooding-based approaches, which belong to BFS, a query source sends its query messages to all its neighbors. When a node receives a query message, if it does not have the target resource, it forwards the query messages to all its neighbors except for the one the query message comes from. Thus the number of the query messages would grow exponentially with the hop counts. Flooding-based approaches may lead to long search delays and a high amount of signaling traffic, and easy to incur repeated visits on a single node.

Another typical search algorithm for unstructured P2P networks is Random Walk, which belongs to DFS. In Random Walk-based solutions, the query source just sends one query message (walker) to one of its neighbors. If this selected neighbor does not have the target resource, it forwards the received message to one of its neighbors except for the one who sends the message to it. Thus the search cost is reduced since the query messages grow linearly with hops. However, it leads to long search delay and low success rate because each hop only covers one node. Even the walker can be increased for improving the search time and success rate, but the effect is limited due to the link degree and redundant path.

4.3.1.2 Traditional structured P2P based solutions

For the structured approaches, the locations of the items are organized in an overlay using distributed indexing technologies, such as Distributed Hash Table (DHT). The main advantage of DHT-based solutions is that the data placement is systematic and thus search procedures generate less traffic for the network. Briefly, in DHT-based approaches, each node is identified by a unique ID. This ID may be created by hashing the IP address of the node or is allocated randomly. Each resource is also hashed to obtain a key value (called a Resource ID). Both the node IDs and the resources IDs use the same name space. Each resource is then stored in the node with the ID that is closest to its key. Several important protocols, including Chord [74], CAN [75], Pastry [76], Kademlia [77] and Tapestry [78], have been designed for DHT to improve service discovery efficiency. They differ in data management and routing strategies, but essentially follow the paradigm of Consistent Hashing [79] whose essential principle is to let the nodes enter and leave the network with minimal disruption for the network.

However, neither the current unstructured P2P approaches nor the structured P2P approaches can adequately support efficient service discovery and publication directly. Unstructured P2P systems enable complex queries, which is an essential requirement for service discovery. They are easy to implement, however, their underdevelopment routing mechanisms lead to heavy network traffic and the possible repeated visit for certain peers. Moreover, they provide no search guarantees for large-scale P2P networks. The structured P2P networks, like the DHT based ones, theoretically guarantee that matching answers will

be found if they exist in the networks. However, they cannot support complex queries since the resource discovery is highly dependent on deterministic matching between the binary hash IDs, which means when a resource is given a key, the location of the resource can be found with limited cost according to the key and only through this key. It implies that users need to provide at least the exact name information about the resource they want. Nevertheless, when a user wants to discover a service, in most of the cases, she/he does not know which specific service to search for, instead, she/he only knows which functionality she/he wants, or uses some synonymous keywords, which means that an ambiguous search is required for service discovery.

4.3.1.3 Other P2P based solutions

In last decade, relying on either unstructured or structured P2P infrastructures, a variety of advanced P2P based solutions have been exploited by academia and industry to improve service discovery and publication on a large scale. These solutions mainly adapt the existing P2P systems from 3 aspects for satisfying the different requirements:

- Improve peer or registry organization and service management.
- Enrich semantic annotations for services.
- Enhance service discovery efficiency and selection accuracy by applying some auxiliary information (e.g. QoS, context) during the service filtering process.

One trend is to group peers into clusters according to similar interests or if they have semantically similar content. HyperCup [80] is an early example to optimize the position of peers by using concept ontology. It groups peers with similar interests or services into concept clusters according to the predefined concept ontology, and then these concept clusters are organized into a hypercube topology to enable efficient message routing for service discovery. METEOR-S [81] tried to use a classification system based on registry ontology to optimize registry organization. In this approach, each registry maintains only the services that pertain to certain domain(s). The different registries act as peers in an unstructured P2P network, and then they are further grouped into clusters according to the mapping between registries and the domain ontology. Similar solutions that classify either the registries or the service providers which act as peers in an unstructured P2P network, and these peers further form a federation that represents similar interests in a decentralized fashion [82],[83],[84],[85]. As we can observe from these examples, the federation-based solutions are generally related to the unstructured P2P architectures, and they still encounter some common issues for unstructured P2P networks such as high network traffic, long delay, and low hit rate, even if they have improved these issues to a certain extent. Some alternative solutions based

on a structured P2P system have also been proposed. SPiDer [86] employs ontology in a DHT-based P2P infrastructure in which service providers with good resources are assigned as super nodes and organized into a Chord ring to take on the role of indexing and query routing. Chord4S [87] tries to avoid the single failure point in Chord-based P2P systems by distributing functionality-equivalent services to several different successor nodes. PDUS [88] combines Chord technology with UDDI-based service publication and discovery technology to provide a hybrid service discovery approach. Researchers in [89] and [90] introduce solutions based on alternative structured P2P topologies such as Kautz graph-based DHT or Skip Graph. These solutions generally use the peers that form the Chord ring to store all the service information about the functionally similar services, and thus require those peers to have good resources such as high availability and high computing capacity. Another trend in current P2P-based service publication and discovery systems is to combine semantic technology with the underlying P2P infrastructure. The concept of a Semantic Overlay Network (SON) was proposed in [91] to improve the organization of peers and enhance content search by linking semantically-similar contents into clusters. This concept can be combined with unstructured P2P networks as we mentioned above to facilitate the creation of the peer federation, or with structured P2P networks as introduced in ERGOT [92]. Moreover, some auxiliary information such as context information, QoS and service reputation [93] [94] are also applied in P2P systems to improve service discovery efficiency and the user experience. However, these kinds of solutions still inherit most of the limitations associated with the underlying structured or unstructured P2P networks.

Taking the benefits and challenges for P2P networks into account, and comparing the advantages and limitations among existing solutions, we exploit the possibility of using an enhanced Chord-based DHT architecture as the basis for a P2P service exposure platform, benefiting from Chord's intrinsic virtue while mitigating its limitations for practical usage in service discovery. Chord follows the concept of Consistent Hashing by linking all the nodes into a ring. Each node keeps a small finger list containing the information of its succeeding nodes in a distance of 2^i. This finger table enables exponentially growing jumps through the ring, and thus drastically reduces the cost for resource discovery - even if a resource resides in the last node of the ring, there is no need to traverse all the nodes. This enhancement would greatly reduce the network traffic and shorten the time needed to discover the target service(s). To further improve its service discovery efficiency and enable ambiguous searching, which is limited in current Chord architecture, a series of adaptations for resource organization and message routing, including the introduction of hierarchical service publications, are proposed as we illustrate in the following subsections.

4.3.2 Overall architecture

In our proposed solution, not only the traditional Telecom and Web services are exposed, but device-offered services, user-generated services are also accommodated. When different kinds of services expose themselves to the network, they generally use different technologies or go through different service exposure gateways. In order to reuse these existing service exposure technologies and limit the modifications to the existing service publication and discovery platforms, a hierarchical P2P architecture which uses a P2P overlay for sharing diverse service information while respecting and maintaining the underlying heterogeneities is proposed as shown in Figure 4-15.

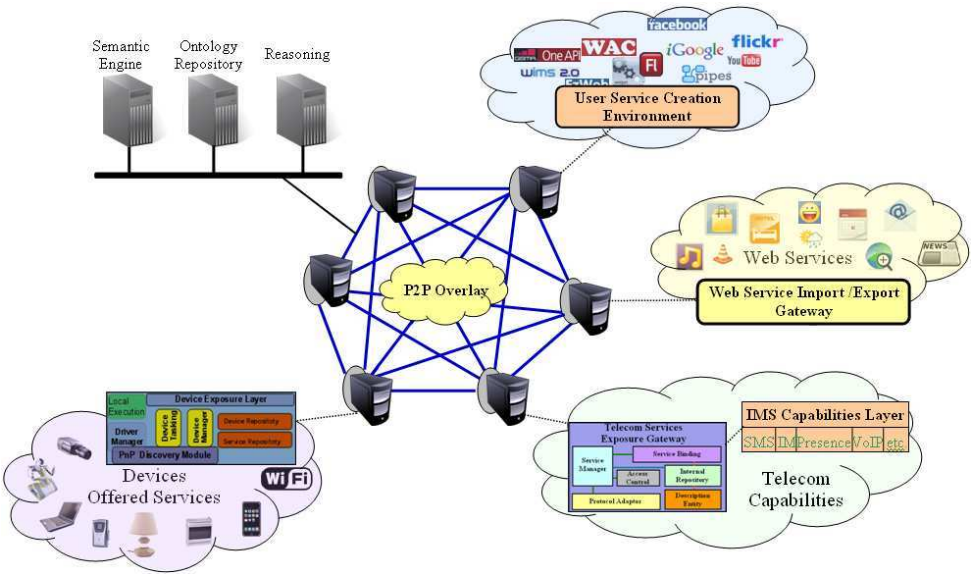


Figure 4-15: Overall architecture for the hierarchical P2P service information sharing system

As depicted in Figure 4-15, this system is composed of several global servers, a number of supper nodes, ordinary nodes and diverse service exposure gateways. The global servers include Semantic Engine, Ontology Repository, and Reasoning, which are in charge of harmonizing the representation forms of the different services' descriptions during the service publication process. These global servers are also responsible for translating the incoming search request, as well as providing a primary filtering during the service discovery process. The super nodes and the ordinary nodes are responsible for the service knowledge storage, retrieval, and service discovery. Behind these super nodes are various different kinds of service exposure gateways for catering to the different kinds of services exposure requirements. Some examples are Telecom Service Exposure Gateway, Web Services Import/Export Gateway, Local Device Exposure and Management Gateway, and even some service creation platforms which embed the functionality of enabling personal service publication by users. These service exposure gateways can be

existing gateways which are used in the traditional UDDI-based centralized solutions, but within this proposed architecture, they are more flexible than when used alone in a centralized environment. They expose their services in a local domain using their own technologies to target a certain group of users or developers; moreover, they also share their service information with users or developers outside their domain by adding some overlay functions to their platforms. For instance, a user inside a local smart device environment can create a personal service not only by combining his devices' offered services discovered by a Local Device Exposure Gateway, but also by integrating the Telecom services provided by an outside Telecom Service Exposure Gateway.

4.3.3 Adaptation of P2P overlay to service publication and discovery

In order to enable the service publication and discovery on the large scale, we propose a hybrid P2P architecture. This architecture is mainly based on Chord and the centralized super node model for the service publication and discovery. The reason behind this proposed solution is that: we would like to benefit from the high service discovery efficiency, meanwhile enabling the ambiguous searching capability, and limiting the index sizes in the nodes. This proposed solution is named as Hybrid Chord in the following parts. Table 4-2 illustrates a brief comparison between our proposed solution and the original Chord based solution, as well as the centralized solution.

Solutions	Index Size	Ambiguous matching
Centralized	A huge index table	Support ambiguous searching
Chord	Small index table	Do not support ambiguous searching
Hybrid Chord	Small index table (bigger than Chord, but by comparing with the centralized solution, it is much smaller)	Support ambiguous searching for the functional equivalent services

Table 4-2: Comparison among centralized solution, Chord based solution and hybrid Chord based solution

We first separate the nodes to be super nodes and ordinary nodes. That is, a certain number of powerful nodes (e.g., the personal computers or servers with fast network connections, high bandwidth and quick processing capabilities, etc.) are selected as super nodes. Then the ordinary nodes connect to the super nodes through the centralized method, to share the responsibilities of service storage and discovery. And a DHT overlay, called Global Super Nodes Overlay, is introduced by using unified APIs: Put, Get and

Remove methods, providing the possibilities of sharing, retrieving and removing resources. This overlay is further divided into four Subset Service Overlays, and each Subset Service Overlay is identified with a unique service domain tag. These Subset Service Overlays are created by chaining the corresponding super nodes together according to the Chord protocol. Consequently, there are three sub-networks in our proposed service discovery and publication model: (1) the network which is used to search the service domain tag; (2) the Subset Service Overlay which is used to locate the target super nodes according to the Chord algorithm; (3) the super node – ordinary nodes cluster which is used to discover the concrete services. Below, we present a more detailed introduction for this architecture.

Since a vast range of services are assumed to be stored in this P2P system, searching for a particular service would be difficult and inefficient if the services are randomly stored in the peers. If each peer is assigned to store a particular type of services, that would improve the service information organization and thus accelerate the service search process. To accomplish this, we use a simplified service classification method to categorize the services, as shown in Figure 4-16. That is, services are first categorized into four basic types: Web services, Telecom services, Device services and User-generated services. Then, according to the different functionalities provided by the services, each service category is further divided into sub-classes, which are called Abstract Services. For example, in the Telecom Services group, the abstract services include: Message, Presence, Charging, IPTV, MMS, SMS, VoIP, etc. Each abstract service can be mapped to several concrete services, for example, SMS can be mapped to the Orange SMS service, the Telefonica SMS service and the BT SMS service.

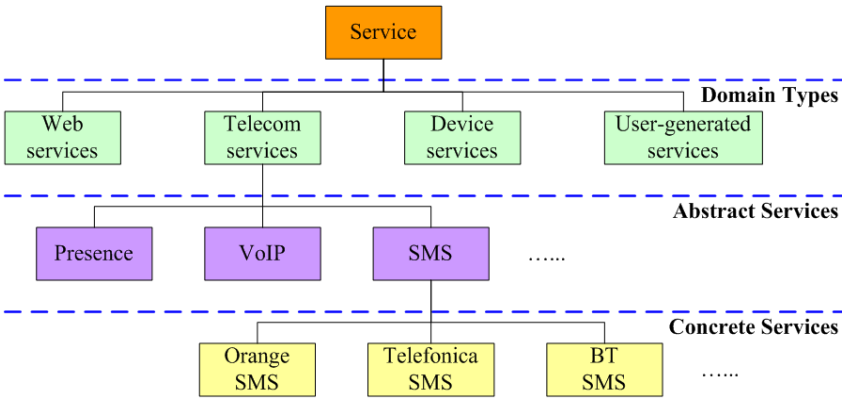


Figure 4-16: Simplified service classification method

Each super node is tagged with one domain type which includes Web services, Telecom services, Device services and User-generated services, as shown in Figure 4-17. As the ordinary nodes are managed and connected to the P2P overlay through the super nodes, they inherit the same service category tag of the super node they contact.

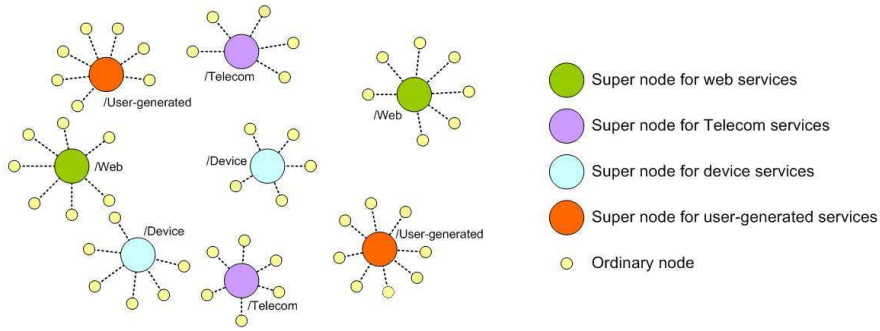


Figure 4-17: Service category tagging for super nodes in Global Service Overlay

Considering the very large number of services published in the system, e.g. in Google's Gadget Repository, there are currently more than 200k published widgets, and in Apple's App Store, there are more than 500k published applications nowadays. Using a single super node to store all the services related to one domain tag can easily lead to a bottleneck and requires maintaining a very powerful server. Consequently, it is reasonable that multiple super nodes are used for storing one kind of service. Therefore, the super nodes storing the same kind of services (i.e., with the same service domain category tag) are grouped to a virtual overlay (called a Subset Service Overlay) as presented in Figure 4-18. To guarantee the service discovery efficiency, inside the same Subset Service Overlay, the super nodes are chained together by using the Chord algorithm.

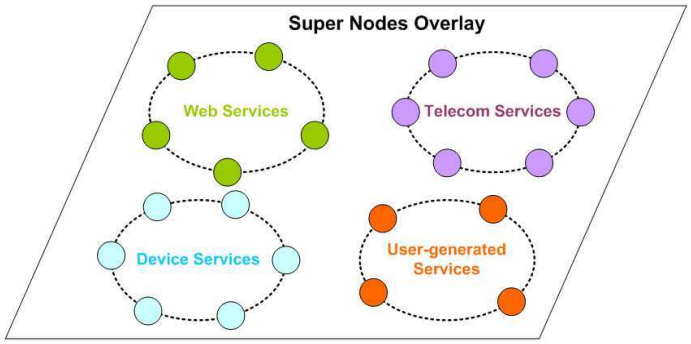


Figure 4-18: Subset service overlays

To adapt above introduced modification, the super nodes inside each Subset Service Overlay have two tables: one is responsible for storing information about the neighboring super nodes inside the same Subnet Service Overlay, called Internal Finger Table, and the other table - External Finger Table, stores the information about the closest super nodes residing in other Subnet Service Overlays. Consequently, no matter through which super node the service discovery request arrives at the global overlay, even this super node resides in other Subset Service Overlay, the request can be forwarded to the corresponding Subset Service Overlay via one hop.

Moreover, each super node in the Subset Service Overlay is assigned to be in charge of certain specific abstract service(s), as shown in Figure 4-19. To avoid the possible single failure point situation and improve the system reliability, one abstract service is assigned to several successive super nodes rather than to a single super node. Consequently, if one super node fails, the same content can be discovered from its successive super nodes. Since assigning a super node for an abstract service relies on the matching between the super node's ID and the abstract service's key, it is possible that a super node could contain several abstract services, if these abstract services' keys are in the same interval of key space for which this super node is responsible.

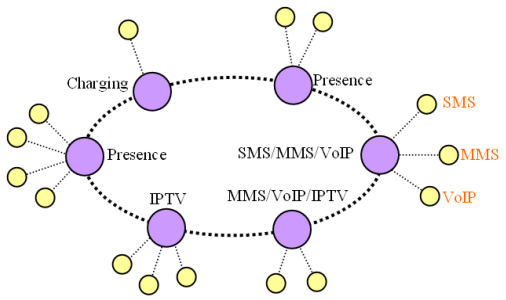


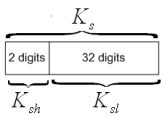
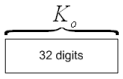
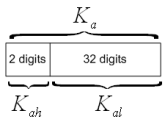
Figure 4-19: An example of abstract service tagging for super nodes in Subset Service Overlay

Finally, the concrete service description files are stored in the ordinary nodes under the control of the corresponding super nodes. And each super node contains another table, named as Local Node Index, to store the extracted information about the ordinary nodes under its control and the relevant services are storing in these ordinary nodes. Similar to the abstract services, one concrete service description file is stored in several ordinary nodes to improve the reliability and avoid the single failure point by redundant storing.

4.3.4 Node ID and resource key assignment

In DHT architecture, the service publication and retrieve actions mainly rely on *put (key, data)* and *get (key)* messages. The key value thus plays a very important role in this architecture. In our solution, the above-mentioned service category tags, nodes' ID, abstract services' names, and the concrete services' names are represented by keys. That is to say, they are calculated using the same algorithm (e.g. SHA-1, MD5, etc.) which produces a specified bit hash value. However, instead of only containing a single part for a key, we use a layered method to represent a key according to the different kinds of nodes and resource requirements.

Table 4-3 summarizes the keys we use in this proposed solution and their basic structures. In this section, we mainly use MD5 as the basic hash algorithm for the node ID and resource key generation. And a more detailed introduction for the assignment of these keys is illustrated in the following part.

Name	Symbol	Structure	Note
Super Node's ID	K_s	$K_s = K_{sh}K_{sl}$ 	K_s: a 34-digit hexadecimal value that contains two parts (K_{sh} and K_{sl}). It is a super node's unique identifier. K_{sh}: a 2-digit hexadecimal value that indicates the service domain tag. K_{sl}: a 32-digit hexadecimal value that represents the IP address of a super node.
Ordinary Node's ID	K_o	K_o 	K_o: a 32-digit hexadecimal value that only contains one part. It represents the ordinary node's IP address. It is an ordinary node's unique identifier.
Abstract service's ID	K_a	$K_a = K_{ah}K_{al}$ 	K_a: a 34-digit hexadecimal value that contains two parts (K_{ah} and K_{al}). It is an abstract service's unique identifier. K_{ah}: a 2-digit hexadecimal value that indicates the service domain tag. K_{al}: a 32-digit hexadecimal value that represents the unique name of an abstract service.
Concrete service's ID	K_c	$K_c = K_{ch}K_{cm}K_{cl}$	K_c: a 66-digit hexadecimal value that

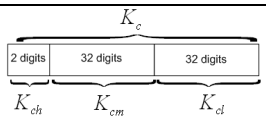
			contains three parts (K_{ch} , K_{cm} and K_{cl}). It is a concrete service's unique identifier K_{ch} : a 2-digit hexadecimal value that indicates the service domain tag. K_{cm} : a 32-digit hexadecimal value that represents the abstract service a concrete service belongs to. K_{cl} : a 32-digit hexadecimal value that represents the unique name of a concrete service.

Table 4-3: Summarization of the keys' structures

For the super node's NodeID K_s, it can be composed of two parts $K_{sh}K_{sl}$. The high key K_{sh} is one of the 2-digit hexadecimal values {01, 02, 03, 04}, which correspond to the four service domain tags {Telecom service, Web service, Device service, User-generated service} respectively. The reason for using a 2-digit hexadecimal value to represent the service domain tag is as follows: first, the low key K_{sl} is represented by a 32-digit hexadecimal value as we introduce in the following part, to be consistent with K_{sl}, the high key K_{sh} also should be represented by a hexadecimal value; second, this system is extensible, that means if a new service category is introduced to the system, a new service domain tag should be added to current service domain tags group, and thus we need to reserve the key space for the possible new service domain tags, that is why we assign 2-digit values instead of 1-digit values to K_{sh}. The low key K_{sl} is created by hashing the node's IP address. For example, if we use the MD5 algorithm, IP address "157.159.100.207" is hashed to be a 128-bit hash value, which is generally represented as a 32-digit hexadecimal number - "7b650a99e4ea2bee1fa841747b2ec8b5". Thus if a super node with this IP address is responsible for Telecom services, its NodeID K_s will be "017b650a99e4ea2bee1fa841747b2ec8b5" in which the first two digits "01" represent the service domain tag – "Telecom service". When a super node is introduced to the network, a Node ID K_s is assigned to it, and according to this node ID, it is added to the corresponding subset overlay directly.

As mentioned above, there are two tables stored in each super node: an Internal Finger Table and an External Finger Table. As we use the Chord protocol for the Subset Service Overlay organization, consequently, the Internal Finger Table is similar to the Finger table defined by Chord. That is to say, each super node, *n*, maintains an Internal Finger table with m entries for the routing information (m may be the number of the bits in the key/node identifiers). In the same Chord circle, the K_{sh} parts of the node IDs and the K_{ah} parts of the abstract service profile identifiers are the same. Consequently, the entries' contents in the finger table are only related to the values of the K_{sl} parts. The ith entry in the table at node *n* contains the identifier, S_i, that succeeds *n* by 2^{i-1} identifiers on the circle, i.e., $s_i = successor(n + 2^{i-1})$, where $1 \le i \le m$, the interval between two entries is [s_i, s_{i+1}) and the node ID stored in this entry is in charge of the identifiers belong to the interval. It also contains the IP address and UDP port for the corresponding node.

Figure 4-20 shows an example of the Internal Finger Table in a small size Chord-based network. In this example, the number of the bits in the node and resource identifiers is 6, and thus the name space is [0, 2^6). Each Internal Finger Table contains 6 entries (i.e. m=6). For node N5, it is as signed an identifier of "5". Each entry in its Internal Finger Table contains a Start Identifier - S_i, which is calculated by $s_i = successor(n + 2^{i-1})$, for example, the first Start Identifier is calculated by $s_1 = successor(N5 + 2^{1-1}) = S6$, and the second Start Identifier is calculated by $s_2 = successor(N5 + 2^{2-1}) = S7$. The second item in the entry is an interval value, which begins with the Start Identifier in current entry, and ends with the Start Identifier in next entry. The third item is the node ID who is the immediate successor node of the Start Identifier.

To discover a target resource, it first needs to find out the most possible location of the target resource. Consequently, when a node *n* receives a resource discovery request, it first checks whether it contains the target resource. If it contains the target resource, it sends back the response to the node that initiates this request. Otherwise, node *n* checks the interval between itself and its immediate successor, if the target resource's ID falls between *n* and its immediate successor, that means the target resource is stored in its immediate successor, thus node *n* forwards the service discovery request to its immediate successor. If the target resource's ID does not belong to the interval of node *n* and its immediate successor, node *n* searches its Internal Finger Table and finds the target resource ID belongs to the interval of which entry, and then the request is routed to the node whose ID is indicated in the same entry (the 3rd column in the figure). If the node that receives this request does not contain the target resource, it repeats the same process for routing the request to a node which is more possible to contain the target resource.

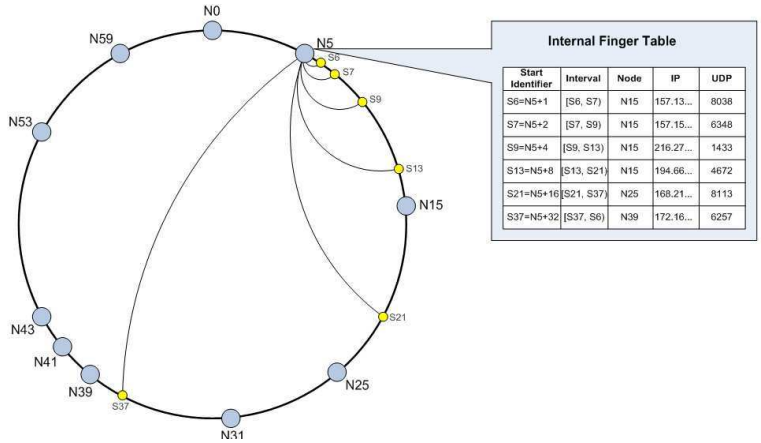


Figure 4-20: Internal finger table in a super node

External Finger Table lists the information of the closest super nodes that belong to other subset service overlays. In our proposed solution, the distance between two super nodes is calculated by comparing their K_{sl} values (i.e. the hash values of their IP addresses). Thus for a super node n_i, its closest super node in the subset service overlay it belongs to is its successor super node, and its closest super nodes in other subset service overlays are the super nodes whose K_{sl} values have the smallest differences with the K_{sl} value of n_i, by comparing with other super nodes in their respective subset service overlays. In our current research, we mainly divide services into four types, and thus there are four subset services overlays. Each node needs to store one of its closest nodes' information for each neighboring subset service overlay. Consequently, it contains three entries in this table, as depicted in Table 4-4. In this table, "subset service overlay" is expressed as "SSO" for short. The first column contains the neighbor subset service overlays' identifiers, which is one of the 2-digit values that correspond to the service domain tags. The second column contains the closest super nodes' IDs in the respective neighbour subset service overlays. The third and fourth columns contain the corresponding IP addresses and UDP ports for the closest super nodes.

External Finger Table

Current SSO Identifier: 01			
Neighbor SSO Identifier	Closest Node ID	IP	UDP
02	027b650a99e4ea2be e1fa841747b2ec8b5	157.159.100.207	8038
03	03ae00ac3171c4e87 0cc43b0523ee11283	91.121.112.154	6348
04	04373d6f74a6fe41c 9c1fd55f94ad43a41	164.2.255.244	1433

Table 4-4: External finger table in a super node

Ordinary nodes are managed by the corresponding super nodes, and each super node contains a table about the ordinary nodes under its control. This implies that all the ordinary nodes under the control of a

specific super node store the same domain-type services. To facilitate service discovery, each ordinary node is assigned a Node ID K_o. The ordinary node ID only contains one part, which is obtained by hashing its IP address.

The abstract service description files are stored in super nodes. They are published to super nodes by using the *put* (K_a, *data)* message, in which K_a is the unique ID for an abstract service description file. The key of abstract service K_a is composed of two parts, $K_{ah}K_{al}$, where the high keyword K_{ah} is an identifier of the service domain name, that is $K_{ah} \in$ {01, 02, 03, 04}. Keyword K_{al} is obtained by hashing the service's name. This hash value generation process must be based on the same hash algorithm for a super node's low key generation, i.e., K_{sl}. That is because in the service publication process, the super node that is selected to store an abstract service is determined by comparing the distance between the abstract service's ID K_a and super nodes' IDs K_s. That is to say, a super node, whose ID is bigger than abstract service's ID, meanwhile having a shortest distance with the abstract service's ID, is selected to be the node that is responsible for this abstract service. An example for an abstract service's key K_a assignment is as follows: SMS service is an abstract Telecom service, so it ID K_a is "014cecb21b44628b17c436739bf6301af2", in which the first two digits "01" is the high key K_{ah} that indicates this service is a Telecom service, and the rest of the digits is the hash value of the abstract service's name - "SMS" - that is generated by using MD5 algorithm.

For concrete service description files, since each concrete service must map to an abstract service, each concrete service's key K_c is composed of the key of the corresponding abstract service K_a and the part hashed from its proper name. K_c should thus be indicated as $K_{ch}K_{cm}K_{cl}$, where the $K_{ch}K_{cm}$ represents the abstract service K_a and K_{cl} represents its proper name. For example, when a SMS service provided by Orange (named as "SMS_Orange") is published to the system, a 66-digit hexadecimal number "014cecb21b44628b17c436739bf6301af25b2ca82f1174532d98ca84d29361443c" is assigned to it as the identification key K_c. In this key, the first two digits "01" (i.e., K_{ch}) represents the service domain tag "Telecom service", the following 32 digits "4cecb21b44628b17c436739bf6301af2" (i.e., K_{cm}) indicate the abstract service (i.e., "SMS") that this concrete service belongs to, and the last 32 digits "5b2ca82f1174532d98ca84d29361443c" (i.e., K_{cl}) is generated by hashing the concrete service' name "SMS_Orange". When a service provider publishes a concrete service through a *put* (K_c, *data)* message, this message is forwarded to the subset service overlay that corresponds with the value of K_{ch} (the first two

digits). According to the value of $K_{ch}K_{cm}$ (the first 34 digits), the message is then sent to the corresponding super node. Finally, by consulting the Local Nodes Index table, the super node determines which ordinary node should hold this concrete service description file.

4.3.5 Node joining and departure processes

4.3.5.1 Super node's join and departure processes

4.3.5.1.1 New super node joins to the network

There are two ways for adding a super node to the network: (1) a new node is introduced to the network and it acts as a super node directly; and (2) when the number of ordinary nodes in a cluster exceeds a predefined value, this cluster is split into two clusters and a powerful ordinary node is selected as a super node to the newly generated cluster.

4.3.5.1.1.1 A new node is added to the network as a super node directly

When a new super node is added to the overlay, it should be queryable by other nodes already in the system. In a DHT based P2P system, the location of a resource is based on the distance between the node ID and the resource key, consequently, the system should first assign this newly introduced super node a node ID. One challenge for this process is how to assign the K_{sh} value to the new super node while assuring that the distribution of resources in the global overlay is as even as possible. To achieve this goal, each super node contains an additional parameter about the abstract service distribution level inside the subset service overlay it belongs to. We denote this parameter as:

$$M = \frac{S_{abstract_service}}{S_{super_node}}$$

- $S_{abstract_service}$: represents the amount of the abstract services stored in this subset service overlay.
- S_{super_node}: indicates the total number of current super nodes in the same subset service overlay.

When a new abstract service is added to the network and stored in a super node, that super node notifies all the other super nodes inside the same subset service overlay to update their parameter *M*. Similarly, when a new super node is added to the network, the corresponding *M* is also adjusted. The values of *M* for the super nodes residing in the same subset service overlay are identical. Relying on this new parameter, the distribution of the super nodes in the four subset service overlays can be guaranteed to be even to some extent.

That is to say, when a new super node n_{new} is added to the network, it communicates with a bootstrap super node n_i it can contact (we assume there are a set of bootstrap super nodes to ensure the basic function in the initial phase, and node n_i then queries the super nodes in other subset service overlays, according to the External Finger Table, and gets the values of their *M*s. Comparing its own M_i with the *M*s retrieved from other subset service overlays, it selects the biggest one and retrieves the value of K_{sh} which corresponds to that selected *M*. When the newly-introduced node receives this K_{sh}, it uses it as its own K_{sh} and hashes its IP address to generate a Node ID K_s.

After the node ID is generated, the new node n_{new} communicates with the bootstrap super node n_i it contacted previously and sends the node joining message *Join (node_ID, info)*. This message with information about the new node (e.g. Node ID, IP Address, UDP Port) is forwarded to the corresponding super node which has the same K_{sh} according to the External Finger Table associated with n_i . That is to say, if the K_{sh} value of n_{new} is the same as n_i 's K_{sh} value, the new node is added to the same subset service overlay as n_i , and this message is kept by n_i ; otherwise, n_i selects the entry whose K_{sh} value is identical to n_{new}'s K_{sh} value from its External Finger Table, and according to the super node information retrieved from this selected entry, the *Join* message is forwarded to the corresponding super node.

When the corresponding super node receives this message, similar to the traditional node join process in Chord, the contacted super node needs to add the new node to the Chord ring. This process mainly contains three phases:

(1) Localize the predecessor node and successor node for the new node;

(2) Create the Internal Finger Table and the External Finger Table;

(3) Notify the new node's presence to other nodes.

Firstly, the contacted super node *n* determines the successor node n_s and predecessor node n_p for the new super node n_{new} by using the Chord's key location algorithm. That is to say, super node *n*, which receive the *Join* message, extracts n_{new}'s ID from the message, and compares this ID with its own ID and its successor's ID. If n_{new}'s ID falls between itself and its successor, that means *n* is the predecessor of the new super node n_{new}, and *n*'s successor is n_{new}'s successor. Otherwise, *n* searches its finger table for the node *n'* whose ID most immediately precedes the new node's ID, and then forwards the request to this selected node *n'*. When n' receives the request, the same process is repeated until the successor and

predecessor of n_{new} are found out. Then the information about n_{new}'s successor and predecessor is sent back to the new node. The new node stores the information of its successor node and predecessor node. Meanwhile, its predecessor node replaces the previous successor node n_s by the new super node n_{new} as its current successor node. And its successor node n_s replaces the previous predecessor n_p by the new super node n_{new} as its current predecessor node. Consequently, n_p, n_{new} and n_p are chained together. Afterward, it begins to initiate its Internal Finger Table. The finger for each entry in the Internal Finger Table (i.e., the node ID indicated in each entry) is calculated by following functions:

$$i_0 = [\log(n_s.id - n_{new}.id)] + 1$$

$$fingers[i] = \begin{cases} n_s (1 \le i \prec i_0) \\ n_s.FindSuccessor(n_{new}.id + 2^{i-1})(i_0 \le i \le m) \end{cases}$$

Once the Internal Finger Table is created, some additional information such as the abstract service distribution level parameter *M* is also stored in it. Meanwhile, the new node needs to notify its presence to other nodes and update the relevant fingers for the existing nodes, as shown in step 2 in Figure 4-21, which is called as *Stabilize*. Meanwhile, a message with the new node's ID is sent to other subset service overlays through the new super node's successor node. The nearest super nodes residing in the target subset overlays, which are outside the new super node's subset service overlay, are determined by using a mechanism similar to that for locating a successor node in the target subset service overlay that we introduced above. Some differences for this process are: (1) when the *join* message is sent to other subset service overlays, the K_{sh} part of the new super node' ID is replaced by the service domain tag identifier of the relevant subset service overlay; (2) after the successors and predecessors have been found out in the corresponding subset service overlay, the successor nodes do not need to perform any update action for their finger tables, and the predecessor nodes only need to update their External Finger Tables by replacing node ID in the corresponding entries (i.e., the entry with the same K_{sh} identifier as the original K_{sh} of the new super node.) with the new super node's original ID; (3) only the successor nodes information is sent back to the new super node for generating an External Finger Table, and these selected nodes are considered as the closest node in the respective subset service overlay for the new super node.

The new super node n_{new} now becomes part of the subset service overlay, and it should also share the storage load for the service description files. The adjusted Chord algorithm will indicate this newly added super node with the relevant abstract service description files it should maintain as shown in step 3 in Figure 4-21. Consequently, the corresponding abstract service description files are moved to the newly

added node from the successor node, and simultaneously, the relevant concrete service description files are transferred to the ordinary nodes monitored by this newly added super node from the ordinary nodes controlled by the successor node.

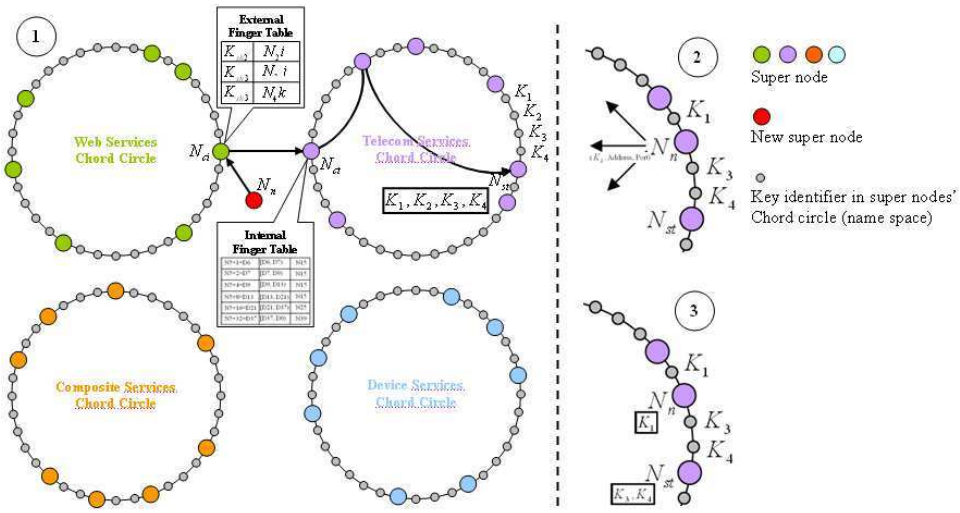


Figure 4-21: A new super node is added to service overlay

4.3.5.1.1.2 An ordinary node is upgraded to a super node

In this proposed model, all the nodes (supper and ordinary nodes) are organized into clusters according to their proximity and function. Inside each cluster, the most powerful node is selected as a super node to monitor the ordinary nodes inside that cluster through the centralized method, as well as to communicate with neighboring super nodes. The super nodes are further separated into four groups according to their K_{sh} values. Inside each group, the super nodes are chained together by using the Chord protocol. To be prepared for the exceptional case of a super node failing or leaving unexpectedly, a candidate super node is selected from the cluster to stores a backup of the information stored in the current super node. The selection of such a candidate super node not only depends on the nodes' capabilities, but also on the distance between it and the super node. This distance is calculated by comparing the hash values of the nodes' IP addresses with the hash value of current super node's IP address. If more than one node's capabilities satisfy the minimal capability requirement, the closest one to current super node is selected as the candidate super node. This strategy enables to limit the impact on the cluster when the super node leaves.

If a super node leaves or fails, the candidate super node takes over its role of super node, contacts all the ordinary nodes inside the cluster according to the backup information, and requests them to register to it, and a K_{sh} value is added to its node ID. The newly upgraded super node then informs the neighboring

super nodes of its new status, and asks them to modify the relevant information. Since the replaced super node was the nearest predecessor super node of this former candidate super node, such replacement has no impact on the position of other super nodes in the Chord ring. However, some ordinary nodes whose IDs belong to the interval of the upgraded super node and its successor super node will be moved to the successor super node, and the abstract services whose resource keys belong to this interval will also be transferred to the successor super node. The corresponding concrete services' description files will be transferred to the ordinary nodes under the control of the successor super node. Finally, another candidate super node is selected from the cluster and stores a backup of the new super node.

If the number of ordinary nodes inside a cluster exceeds the predefined limitation (set to prevent crowding), then to guarantee system service discovery efficiency, the cluster will be split into two clusters and a powerful ordinary node is selected to be upgraded to be a super node for the new cluster. In order to ensure the even distribution of services, the selection of this upgraded super node not only depends on its capabilities, but also on its position in the interval between the existing super node and this super node's successor: the one which is nearer the middle of the interval has higher priority. The corresponding finger tables (External and Internal) are then created and stored in the new super node. The abstract services whose key belongs to the interval of the newly upgraded super node and its predecessor super node are moved to the newly upgraded super node, and the corresponding concrete services are also transferred to the relevant ordinary nodes. Finally, the relevant indexes (Local Nodes Index and Local Concrete Service Index) are generated and stored in the new super node according to the distribution of the abstract service description files and of the concrete service description files.

4.3.5.1.2 A super node leaves the network

When a super node leaves the network, it sends out a *depart (node_ID)* message to its own subset overlay as well as to other subset overlays, and the corresponding finger table adjustments are performed by the relevant nodes when they receive this departure notification. And a candidate super node replaces its place in the Chord ring as we introduced in 4.3.5.1.1.2.

In our current research, we assume that the super nodes for this distributed service publication and discovery system rely on the comparatively stable servers or service publication/discovery/creation platforms, the join or departure of super nodes are not frequent, thus the network traffic invoked by service description file conveying will not bring a severe impact on the system.

4.3.5.2 Ordinary node's join and departure processes

Since ordinary nodes are generally more dynamic than super nodes, a more efficient mechanism is needed to limit the impact on the system during node joining or leaving process. We therefore use a hybrid mechanism for ordinary node management, as shown in Figure 4-22. That is, we use both consistent hashing circle and centralized index for the ordinary node management.

In our solution, all the ordinary nodes under the control of a super node are formed into a consistent hashing circle according to the nodes' IDs. The super node is also included in this circle by comparing the value of its K_{sl} part with the values of other ordinary nodes' IDs K_o. Meanwhile, since the number of ordinary nodes under the control of one super node is limited, the super node contains an index table (i.e., Local Nodes Index) that is storing the relevant information of all the ordinary nodes inside the same cluster, thus enables a centralized control on these ordinary nodes, and the lookup of service information stored in the ordinary nodes only requires one hop. This index contains the ordinary nodes' IDs, IP addresses, UDP ports, and the stored abstract services, as indicated in Table 4-5.

Local Nodes Index				
Node ID	IP Add	UDP Port	Abstract Services	Number of abstract services
K_{o1}	157.15…	7654	SMS, MMS, VoIP	3
K_{o2}	10.29…	2062	MMS, VoIP, Conference	3
K_{o3}	176.14…	1214	SMS, Presence	2
K_{o4}	123.121…	7654	IPTV, MMS, Location	3

Table 4-5: An example of Local Nodes Index

When a super node receives a concrete service publication message, it uses this index to identify which ordinary node to forward the message, i.e., which node would store the concrete service description files related to the assigned abstract service type.

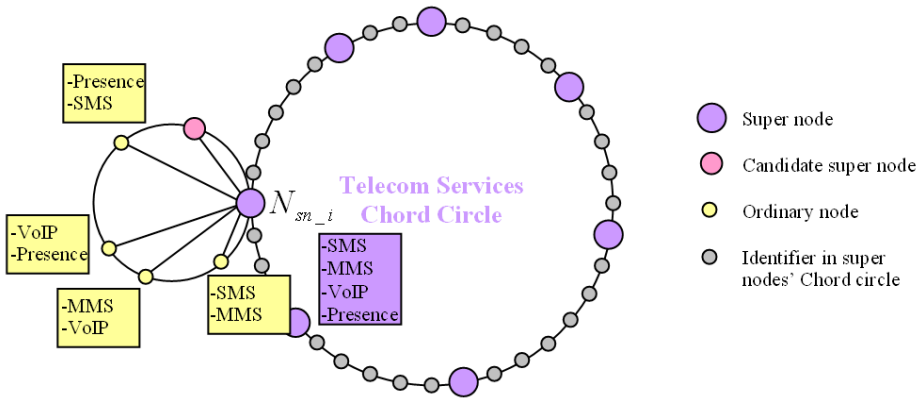


Figure 4-22: Relationship between super node and ordinary nodes

When a new ordinary node is introduced, it joins the corresponding cluster by contacting a bootstrap node. Meanwhile, an ordinary node ID K_o is assigned to it by hashing its IP address. A *Join(*K_o*, info)* message is then sent to the four subset service overlays by the bootstrap node. Each subset overlay selects the closest super node in its ring according to the Chord algorithm, and sends the super node information back to the bootstrap node. After the bootstrap node receives responses from the four subset service overlays, it selects the closest super node and sends this selected super node's information to the new ordinary node. The new ordinary node then registers itself to this super node. Relying on the consistent hashing algorithm, the ordinary node is added to the local nodes' circle. Finally, a new entry for the new ordinary node is added to the Local Nodes Index of this super node.

4.3.6 Service publication process

4.3.6.1 Publish or remove an abstract service from the network

In order to avoid having multiple abstract services with the same functionalities but different names published in the network, we assume that only the network administration groups' members have the right to publish an abstract service. After their administration group membership is authenticated, a developer or system manager can publish an abstract service through their own service publication platform, which acts either as a super node or an ordinary node depending on its capabilities and configurations. If an ordinary user or developer wants to publish a new abstract service to the network, she/he should contact one of the administration group's members, who will be delegated to publish this new abstract service to the network.

When a service publication platform receives an abstract service publication request, it assigns a resource key K_a to the new abstract service by selecting the corresponding service domain tag identifier to K_{ah} part and hashing the abstract service's name (generally, the name is the functional name of that

service, and we assume there is a predefined domain ontology for these abstract services shared among administration members) for the K_{al} part. A $get(K_a)$ message is then sent to the P2P network to determine if this abstract service has already been published in the network. If an abstract service with the same K_a has been published, the publication process is abandoned and the publisher is notified of the existence of this abstract service. If no relevant abstract service is found, then the super node generates a $put(K_a, name, summary, data, publisher_ID)$ message in which K_a represents the resource key, *name* represents the abstract service's name, *summary* is the functional summary for this abstract service, *data* represents the service description file, and *publisher_ID* indicates the publisher's identity.

Next, the first contacted super node compares the K_{ah} of the abstract service with its own K_{sh}: if they are identical, it uses Chord-based key location algorithm to determine which super node inside the same subset service overlay will store this service description file. That is, super node *n*, which receive the *put* message, extracts K_a from the message, and compares this abstract service key with its own ID and its successor node's ID. If the abstract service's key K_a falls between itself and its successor node, its successor node is considered as K_a's successor node and selected to store this abstract service. Otherwise, *n* searches its finger table for the node *n'* whose ID most immediately precedes K_a, and then forwards the request to this selected node *n'*. When *n'* receives the request, the same process is repeated until the successor node of K_a is found out.

Once a super node is selected, the *put* message is forwarded to the selected super node as shown on the left side of Figure 4-23, and then the abstract service description file *data* is extracted from the message and stored in the selected super node's local abstract service repository. Meanwhile, to avoid creating a single failure point for the service discovery process, this selected super node forwards the *put* message to its *n* nearest neighboring super nodes to make backups (step 4 on the left side of Figure 4-23).

If the K_{ah} of the new abstract service is different than the K_{sh} of the current super node, the *put* message is forwarded to the corresponding subset service overlay according to the super node's External Finger Table, as shown on the right side of Figure 4-23. The super node in the corresponding subset service overlay which receives this *put* message performs the same procedure introduced above to store the new abstract service in the most appropriate super node.

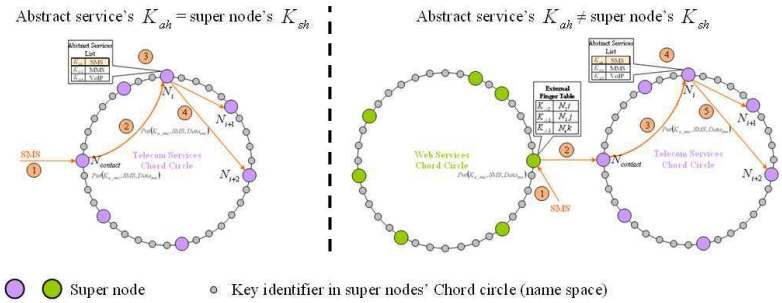


Figure 4-23: Abstract service publication process

When an administration member wants to remove an abstract service description file from the network, the first step is to have her/his membership authenticated. Then, if she/he already knows the key K_a of the target abstract service, she/he only needs to send a *remove(* K_a *,publisher_ID)* message to the network. Otherwise, a *remove(name,domain_tag,publisher_ID)* message must be sent to the network, and then the first contacted super node replaces the domain tag of the abstract service by service domain identifier, and hashes the abstract service's name to generate the corresponding key K_a for this abstract service. The same super node then replaces the attributes – name and *domain_tag* - inside the received message with the newly generated key K_a. The *remove(* K_a *,publisher_ID)* message is then injected to the network. The user's right to remove an abstract service has to be authenticated by the pertinent components according to the *publisher_ID* extracted from the message. In fact, only the publisher and the network administrators have the right to update or remove an existing abstract service. In order to guarantee that all the relevant target abstract service description files are removed from the network, *remove* messages have to go through all the super nodes in the relevant subset service overlay. Moreover, all the concrete service description files related to this abstract service are also removed from the ordinary nodes. Generally, because of its influence on concrete services, once an abstract service is introduced to the network, it will not be removed unless it will no longer be used and there are no concrete services pointing to it.

If an abstract service description file needs to be updated, the administration member can use either the *update(* K_a *,summary,data,publisher_ID)* or the *update(name,domain_tag,summary,data,publisher_ID)* message. If the contacted super node receives the *update(* K_a *,summary,data,publisher_ID)* message, it injects that message directly to the overlay, otherwise the super node that receives the update message must first generates the key K_a of the abstract service according to its name and service domain tag. The locations of the corresponding abstract service description file and its backups are then determined by using the similar abstract service location method which we introduced at the beginning of this subsection, and

next they are replaced by the new abstract service description file which is encapsulated in the *data* and *summary* portions.

Finally, after the abstract service has been added to the super node, it should assign which ordinary node will be responsible for the concrete services that belong to it. To avoid a possible overload for certain ordinary nodes, the super node contains an attribute that indicates the number of abstract services stored in it: $S_{sn_abstract_service}$. The Local Nodes Index table also contains the number of abstract services related to each ordinary node $S_{on_abstract_service}$, and the total number of the ordinary nodes inside the cluster is n_{node}. The responsibility factor for each ordinary node is $R_{ordinary_node}$ and the average responsibility factor is $R_{average}$. Their relationships are as follows:

$$R_{ordinary_node}[i] = \frac{S_{on_abstract_service}[i]}{S_{sn_abstract_service}}$$

$$R_{average} = \frac{(S_{on_abstract_service}[1] + S_{on_abstract_service}[2] + \ldots + S_{on_abstract_service}[n_{node}])}{n_{node} * S_{sn_abstract_service}}$$

Before we assign the abstract service to an ordinary node, we have to make sure that the files stored in the candidate ordinary nodes have not reached their maximum capability. That is achieved by comparing the current stored files' number in each node n_{file} with the maximum number they can store $n_{file_\max}$, which is predefined by the system. If $n_{file} \geq n_{file_\max}$, this ordinary node will not be considered for the assignment of a new abstract service. Otherwise, the following step is performed by comparing the $R_{ordinary_node}$ value of each node with the $R_{average}$ value. If $R_{ordinary_node}[i] \geq R_{average}$, it means this ordinary node's load has reached or is higher than the cluster's average load level, and so it has a low priority to be assigned a new abstract service. If $R_{ordinary_node}[i] < R_{average}$, then the ordinary node's load is lower than the average load level, and it has a high propriety for being assigned a new abstract service. After all the high-priority ordinary nodes have been selected, one ordinary node is selected randomly. Simultaneously, *m* successor nodes of this selected node are assigned to be responsible for this abstract service for the purpose of making backups. Finally, the corresponding parameters in the Local Nodes Index table in the super node will be updated. The following tables illustrate, as an example, what changes as a new abstract service NAB (Network Address Book) is added to the super node:

Before:

Node ID	IP Add	UDP Port	N_{file}	Abstract Services	$S_{on_abstract_service}$	$R_{ordinary_node}$
K_{o1}	157.15…	7654	18	SMS, MMS, VoIP	3	0.429
K_{o2}	10.29…	2062	12	MMS, VoIP, Conference	3	0.429
K_{o3}	176.14…	1214	6	SMS, Presence	2	0.286
K_{o4}	123.121…	7654	10	IPTV, MMS, Location	3	0.429
$S_{sn_abstract_service} = 7$; $n_{node} = 4$; $R_{average} = 0.393$; $m = 2$; $N_{file_max} = 30$						

After:

Node ID	IP Add	UDP Port	N_{file}	Abstract Services	$S_{on_abstract_service}$	$R_{ordinary_node}$
K_{o1}	157.15…	7654	18	SMS, MMS, VoIP	3	0.375
K_{o2}	10.29…	2062	12	MMS, VoIP, Conference	3	0.375
K_{o3} (selected)	176.14…	1214	6	SMS, Presence, NAB	3	0.375
K_{o4} (backup)	123.121…	7654	10	IPTV, MMS, Location, NAB	4	0.500
$S_{sn_abstract_service} = 8$; $n_{node} = 4$; $R_{average} = 0.406$; $m = 2$; $N_{file_max} = 30$						

Table 4-6: Update for local node index table

If too many files have been published in one ordinary node, and this ordinary node is currently assigned more than one abstract service, the abstract service that has the least relevant concrete service description files is assigned to another non saturated node following the same process as when a new abstract service is added to the super node. After the assignment process is finished, the concrete service description files related to this abstract service are transferred to the newly selected ordinary node. This process will be repeated until the oversaturated ordinary node is no longer saturated or when only one abstract service is left on this ordinary node. If an ordinary node that is only responsible for one abstract service, and the number of concrete services stored in it has already reached the ordinary node's predefined service storage limitation, a new ordinary node will be selected to store the additional introduced concreted services.

4.3.6.2 Publish and remove a concrete service from the network

In contrast to the abstract service publication process, not only the administration members can publish a new concrete service to the network, but the users, the professional developers and the services providers are empowered to publish their created service(s) to the overlay.

When a concrete service is published, the service publication platform initiates a service publication request according to the publisher's inputs. During this process, the service publication platform keeps one

copy of the service description file in its local server, just as a traditional service publication platform does. Consequently, when a user searches for a service through this service publication platform, it can only use its own service discovery mechanism and only search in its local service repository. Meanwhile, if a service publication platform wants to share its stored service information with other service publication platforms, service discovery platforms or service creation environments, it only needs to send out the service publication request to the super node it contacts, if it joined the network as an ordinary node. Otherwise, if it acts as a super node, this service publication request is forwarded to Semantic Engine to unify the request as required by the network and then injected into the network.

We assume that each service publication platform maintains a simplified list of the published abstract services which only contains the names of the abstract services in its local repository. Therefore, when a user wants to publish a concrete service, she/he can select an abstract service that this concrete service belongs to from this list. When the super node receives this service publication request, it forwards the service description file extracted from the request to Semantic Engine, and then Semantic Engine unifies the service description file by contacting Ontology Repository. Next, the unified service description file is sent back to the super node, which generates the resource key K_c to this published concrete service. As mentioned before, K_c is composed of K_{ch}, K_{cm} and K_{cl}, where $K_{ch}K_{cm}$ indicates the abstract service it relates to. The K_{cl} value is obtained by hashing the concrete service's name. The super node then generates a $put(K_c, name, summary, data, publisher_ID)$ message in which K_c represents the key assigned to this concrete service, *name* represents the name of the concrete service, and *data* is a unified service description file. Next, the super node compares the K_{ch} of the concrete service with the K_{sh} part of its own node ID. If they are identical, it locates the super node which is responsible for storing the abstract service this concrete service relates to by using the Chord resource location algorithm. After the target super node is located and the relevant information has been sent back to the initially contacted super node, the initially contacted super node forwards the *put* message to the selected super node. This selected super node then forwards the *put* message to its n (e.g. n = 2) nearest neighbor super nodes, which also contain the same abstract service description file for backup, as illustrated in Figure 4-24.

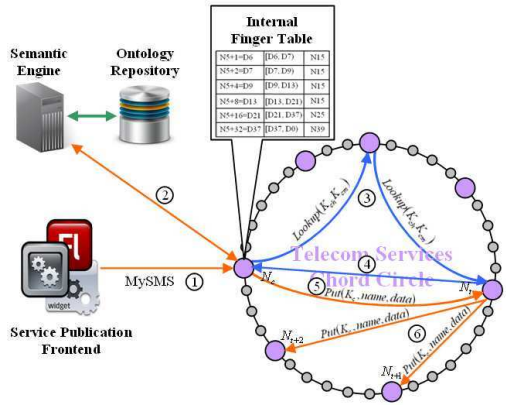


Figure 4-24: Concrete service publication process for $K_{ch} = K_{sh}$

If the K_{ch} of the concrete service is different from the K_{sh} of the super node, as determined by External Finger Table, the super node forwards the *put* message to the corresponding super node that resides in the subset service overlay whose super nodes have the same value of K_{ch} for their K_{sh} parts. This super node then performs the same abstract service location process to select the target super node to receive the *put* message, as shown in Figure 4-25.

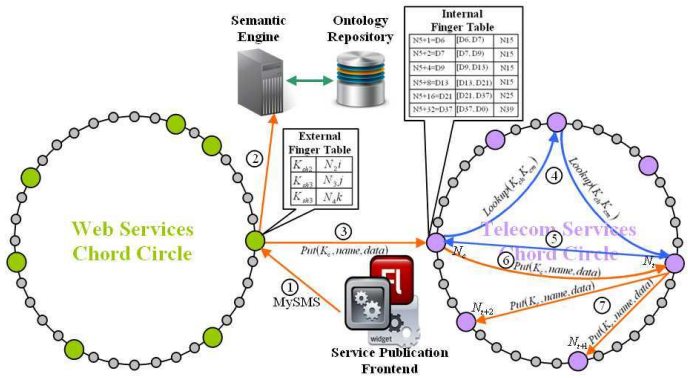


Figure 4-25: Concrete service publication process for $K_{ch} \neq K_{sh}$

After the target super node receives the *put* message, it first looks up its Local Concrete Service Index to verify if an identical key K_c exists in its entries, which would mean that this concrete service has already been published. If such an identical key is found, the incoming message is abandoned, the service publication process is terminated, and a notification of the existence of this concrete service is sent back to the service publication platform. If no identical key is found in the Local Concrete Service Index, the super node extracts the $K_{ch}K_{cm}$ from K_c (i.e., extracts the first 34 digits from K_c), and then converts the $K_{ch}K_{cm}$

to the corresponding abstract service name by consulting the Local Abstract Services List, which contains the names and IDs of the abstract services stored in this super node. The super node then looks up its Local Nodes Index to determine which entries contain this abstract service name. Finally, according to the ordinary nodes' information stored in the selected entries, the *put* message is forwarded to the corresponding ordinary nodes.

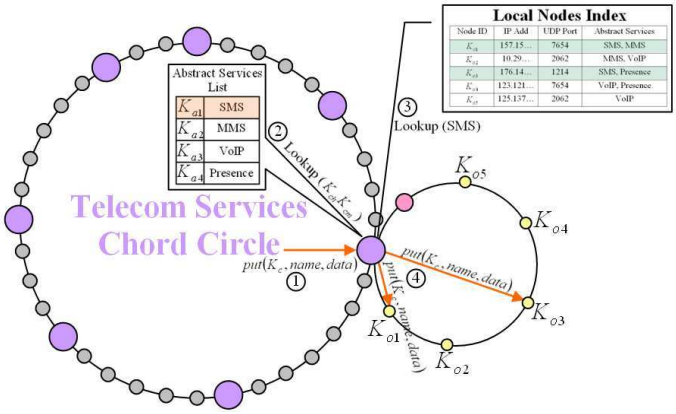


Figure 4-26: Adding a new concrete service to ordinary node

Meanwhile, the concrete service's key K_c, *name*, *summary* and *publisher_ID* extracted from the *put* message, as well as the ordinary node's ID K_o, IP address and UDP port that has been selected to store this concrete service's description file are added to the Local Concrete Service Index table to facilitate future service discovery, as shown in Table 4-7. Moreover, since the cluster of ordinary nodes may change frequently, with some nodes joining and some existing nodes leaving at runtime, there is no need to return the ordinary nodes' information back to the service publication platform. In fact, the selected super nodes' information, which is comparatively more stable, is sent back to the service publication platform, thereby making it possible to update of the service description file.

Local Concrete Service Index table						
Service Key	Service Name	Summary	Publisher_ID	Ordinary nodes	IP address	UDP Port
K_{c1}	MySMS	Function:SMS Provider:Alice	Hsqjkdh512cq	K_{o1}	172.52....	8080
				K_{02}	172.36....	8080
K_{c2}	OrangeSMS	Function:SMS Provider:Orange	Qshqs11chqoe	K_{o1}	172.52....	8080
				K_{02}	172.36....	8080
K_{c3}	OrangeMMS	Function:MMS Provider:Orange	Qshqs11chqoe	K_{o3}	172.17...	2062
				K_{o4}	172.65...	8080

Table 4-7: Local concrete service index table

When a concrete service description file needs to update, only the publisher has the right to do updating. If the service publication platform already stores the corresponding super nodes' information, it can send the $update(K_c, name, summary, data, publisher_ID)$ message directly to the corresponding super nodes. Otherwise, the *update* message is injected to the network similar to the new service publication process. After the target super nodes have been located, they consult the Local Concrete Service Index, and select the entry that contains the same K_c. Next, the *publisher_ID* extracted from the received message is compared with the existing *publisher_ID* in the selected entries. If they are not the same, the update process is abandoned. Otherwise the update process continues by extracting the ordinary nodes' information from the selected entry, the *update* message is forwarded to the corresponding ordinary nodes, and the service description file extracted from *data* parameter replaces the existing one. Finally, the relevant *summary* content is updated in the Local Concrete Service Index, and a notification of the successful update is sent back to the service publication platform.

Similar to the update process, only the publisher has the right to remove an existing concrete service description file. The publisher can send a $remove(K_c, publisher_ID)$ message to the network through her/his contacted super node. Next, if the service publication platform stores the information about the target super nodes, the request is forwarded to the target super nodes directly, otherwise, the contacted super node forwards the message to the target super nodes according to the K_c, which is similar to the new service publication process. After the target super nodes receive the *remove* message, they select the entry form their Local Concrete Service Index that contains the K_c, and then compare the publisher ID extracted from the received message with the publisher ID stored in the selected entry. If there are not identical, the remove process is annulled, and a failure notification is sent back to the publisher. Otherwise, the *remove* message is forwarded to the corresponding ordinary nodes. Once the ordinary nodes receive this request, the relevant service description files are removed from them, and a successful remove notification is sent back to the super node. After all the relevant service description files have been removed from the ordinary nodes, the previously selected entry from the Local Concrete Service Index is also removed from the table, and a final notification is sent back to publisher to indicate that the remove action has been performed successfully.

4.3.7 Service discovery process

When a user wants to discover a service to add it to her/his personal service environment, or to integrate it to a composite service, she/he can send out a service discovery request from a service discovery platform or

from a service creation platform. When a super node receives this service discovery request, it forwards it to Semantic Engine so that the request can be modified to the network interpretable format.

If a user wants to look for a specific service for which she/he already knows the name, the super node will assign the service domain tag identifier, hash the abstract service name, and the concrete service name to generate a concrete service key K_c, and then a $get(K_c)$ message is injected to the network. The concrete service key K_c is composed of three parts: K_{ch}, K_{cm} and K_{cl}, and so, by comparing the K_{ch} with the super node's own K_{sh}, the super node can determine the routing for this *get* message: if K_{ch} is identical to K_{sh}, it means this concrete service description file resides in the same subset service overlay, and by analyzing its $K_{ch}K_{cm}$, which represents the mapped abstract service, the super node can route the *get* message to the nearest super node that is responsible for this abstract service according to the Chord resource location algorithm that we introduced in section 4.3.4. The super node looks up its Local Concrete Service Index to determine which entry contains the key K_c, and then forwards the *get* message to the nearest ordinary node according to the information extracted from the selected entry. After the ordinary node receives the *get* message, it extracts the target service description file from its local repository and sends it to the super node, and the super node sends the service description file back to the service discovery platform or service creation platform that initiated this service discovery request. When the selected ordinary node does not reply in the required time, the super node resends the *get* message to another ordinary node that stores this target service description file.

If K_{ch} is different than K_{sh}, it means that this service description file is stored in another subset service overlay, consequently, the super node looks up its External Finger Table to identify the corresponding nearest super node that resides in the same subset service overlay as the target concrete service description file, and forwards the *get* message to the selected super node. The following steps follow the same pattern as given above, starting from when a super node, which resides in the target subset service overlay, receives the *get* message.

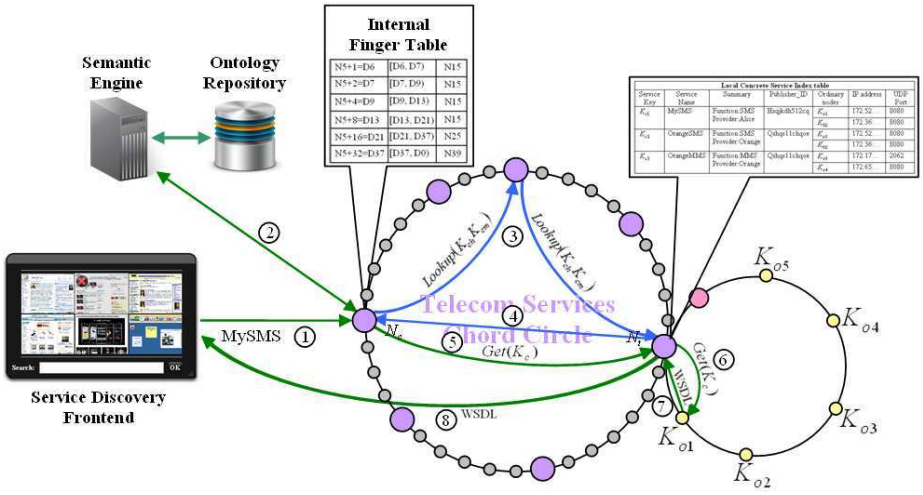


Figure 4-27: Deterministic service discovery process

If a user wants to search for a service satisfying the function she/he needs, but does not know the name of the service, or wants to search a special kind of service, which would mean ambiguous searching, she/he can initiate a service discovery request by using the abstract service's name, e.g. "SMS", or by using natural language, e.g. "send a SMS", through the service discovery front-end. If the request is natural language based, it will be interpreted to a machine understandable format and mapped to the corresponding abstract service(s) by the back-end component of the service discovery platform. This request is sent to a super node. That super node connects with Semantic Engine to unify the request to a network interpretable format. At the same time, it assigns the service domain tag identifier and hashes the abstract service name to generate a key K_a for the target abstract service. A $get(K_a)$ message is generated and injected into the network. According to the K_a, the contacted super node locates the nearest corresponding super node that stores the target abstract service. The *get* message is then forwarded to this super node as explained earlier. The target super node refers to its Local Nodes Index table to discover which ordinary nodes store the concrete services related to the required abstract service. After one of the ordinary nodes is selected, its information is sent back to the service discovery platform.

According to the received ordinary node's information, the service discovery platform resends a *get* message with the auxiliary information (e.g., user identifier, preference, etc.) to the selected ordinary node for enabling a preliminary filtering for the relevant services stored in this ordinary node. And then it establishes a connection with the target ordinary node directly for downloading all the selected concrete service description files that are satisfying user's requirements. If this connection or file download process fails, the service discovery platform sends a request to the target super node for the information about other ordinary nodes that store the relevant service description file. If none of the ordinary nodes under the

control of the target super node can send the service description files to the service discovery platform, the contacted super node will locate another super node that also stores the same abstract service, and the same connection process is performed. These steps will be repeated until the target concrete service description files are successfully downloaded from an ordinary node, or the time-to-live of this request has expired. The service discovery platform must then perform a further filtering according to user's context information and its own selection rules, and finally the most appropriate service(s) is proposed to the user.

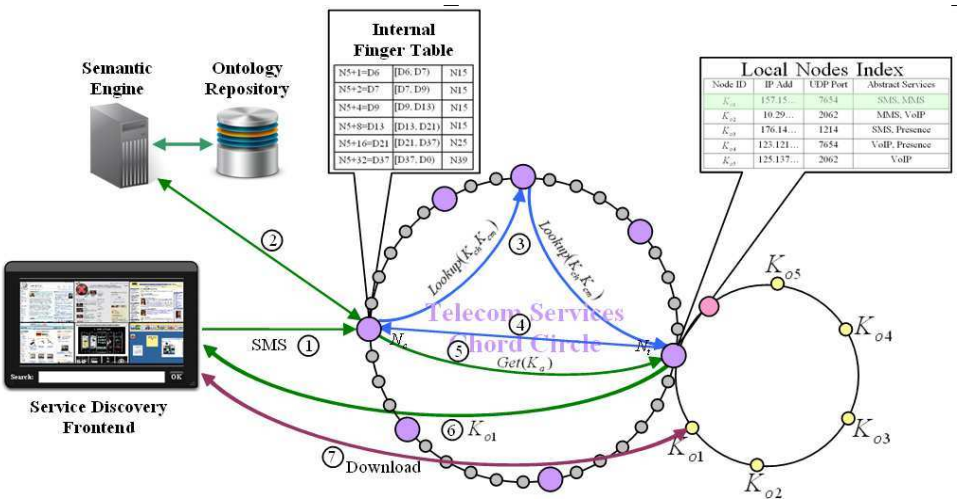


Figure 4-28: Ambiguous service discovery process

4.3.8 Performance analysis

Several performance analyses were conducted in order to prove the proposed Chord based service publication and discovery model, and find out the impact factor(s) for this solution. To simplify the illustration, we name our proposed model as hybrid Chord in this section. We mainly estimate the theoretical average hops, the maximal hops and the index size of the proposed solution, and compare it with the centralized solution and the original Chord-based solution. That is because our proposed solution involves both the centralized control attribute for the super nodes and ordinary nodes cluster, and the Chord-based distributed lookup attribute for the organization of the super nodes inside the subset service overlays.

To simplify the comparison, the average hops is denoted as $E(N)$, the maximal hops is denoted as $M(N)$, and the index size is denoted as $S(N)$.

The relevant parameters are summarized in Table 4-8:

Parameters	Description
N	The number of the nodes in the network.
P	The proportion of super nodes in the hybrid Chord network
m	The number of bits in key/node identifiers
$E_{Centralized}(N)$	The average hops in a centralized system with N nodes.
$E_{Chord}(N)$	The average hops in a Chord based system with N nodes.
$E_{Hybrid_Chord}(N,P)$	The average hops in a hybrid Chord based system with N nodes, and the proportion of super nodes is P.
$M_{Centralized}(N)$	The maximum hops in a centralized system with N nodes.
$M_{Chord}(N)$	The maximum hops in a Chord based system with N nodes.
$M_{Hybrid_Chord}(N,P)$	The maximum hops in a hybrid Chord based system with N nodes, and N*P nodes are the super nodes.
$S_{Centralized_registry}(N)$	The index size of the central registry in a centralized system with N nodes.
$S_{Centralized_node}(N)$	The index size of the nodes in a centralized system with N nodes
$S_{Chord}(N)$	The index size of the nodes in a Chord based system with N nodes.
$S_{Hybrid_Chord_\sup ernode}(N)$	The index size of the super nodes in a hybrid Chord based system with N nodes, and N*P nodes are the super nodes.
$S_{Hybrid_Chord_ordinary}(N)$	The index size of the ordinary nodes in a hybrid Chord based system with N nodes, and N*P nodes are the super nodes.

Table 4-8: Relevant parameters for centralized solution, Chord based solution and hybrid Chord based solution

4.3.8.1 Performance estimation equations

First, we assume that all the resources are distributed evenly in the system, and the four subset service overlays have the same sizes. Theoretically, the averages hops, the maximum hops and the index sizes for the centralized solution, the original Chord based solution and the hybrid Chord based solution can be calculated as follows:

(1) Average hops:

I. Centralized solution:

$$E_{Centralized}(N) = 2$$

For the centralized solution, all the nodes information is indexed in a central registry, consequently to discover a target resource, it just needs to send the request to the central registry, and the central registry searches its local index for discovering the node that contains the target resource. Once the node is discovered, the information about the target node is sent back to the node which initiates the request. Then the original node initiates another request and sends it to the selected node for getting the target resource, and that requires another hop. Consequently, to get the target resource, it only needs two hops, no matter what size the network is.

II. Chord solution:

$$E_{Chord}(N) = \frac{\log_2(N)}{2}$$

The average hops for the original Chord based solution is $\log_2(N)/2$, as introduced in [97]. Its average hops for the discovery mainly depends on the network size, and increases logarithmically with the number of nodes.

III. Hybrid Chord solution:

$$E_{Hybrid_Chord}(N,P) = 25\% * (\frac{\log_2(N*P/4)}{2}) + 75\%(\frac{\log_2(N*P/4)}{2} + 1) + 1$$

As mentioned before, we assume that the resources are distributed evenly in 4 subset service overlays. Consequently, each subset service overlay contains $\frac{N}{4}$ nodes. Among these nodes, $\frac{N \times q}{4}$ are super nodes, which are chained together as a Chord ring, consequently, the average hops inside a Chord subset service overlay is $\frac{\log_2(\frac{N \times q}{4})}{2}$. There is 25% of probability that a resource is discovered in the subset service overlay the access node belongs to. And there is 75% of probability the resource is discovered from other subset service overlay (in this case, it needs one more hop to route the request to the corresponding subset service overlay). Finally, as the nodes are separated to super nodes and ordinary nodes, so it needs one more hop for reaching the ordinary node.

(2) Maximum hops:

I. Centralized solution:

$$M_{Centralized}(N) = 2$$

For the centralized solution, no matter what size the network is, it just needs one hop to reach the central registry for discovering which node holds the target resource, and then other hop to reach the selected node for the target resource, thus the maximum hops for the centralized solution is two hops.

II. Chord solution:

$$M_{Chord}(N) = \log_2(N)$$

The maximum hops for the original Chord based solution is $\log_2(N)$, as introduced in [97].

III. Hybrid Chord solution:

$$M_{Hybrid_Chord}(N,P) = 1 + \log_2(\frac{N*P}{4}) + 1$$

For the hybrid Chord based solution, it needs maximum 1 hop to reach the target subset service overlay according to the service domain tag, then inside a subset service overlay which contains (N*P)/4 super nodes, it needs maximum $\log_2(\frac{N*P}{4})$ hops to reach the target super nodes by using the Chord key location algorithm. Moreover, to route the request from the super node to the ordinary node, it also needs one more hop.

(3) Index sizes:

I. Centralized solution:

$$S_{Centralized_registry}(N) = N$$

$$S_{Centralized_node}(N) = 1$$

For the centralized solution, all the nodes' information should be stored in the central registry. Consequently, for a network with *n* nodes, the central registry contains at least *n* entries.

II. Chord solution:

$$S_{Chord}(N) = m$$

As defined in [97], in the Chord based solution, each node maintains a finger table with up to *m* entries, and *m* is the number of bits in the key/node identifiers.

III. Hybrid Chord solution:

$$S_{Hybrid_Chord_supernode}(N) = 3 + m + (1/P - 1)$$

$$S_{Hybrid_Chord_ordinarynode}(N) = 1$$

In the hybrid Chord based solution, each super node needs 3 entries to store its closest neighbor super nodes in other subset service overlays (i.e., for the service domain tag network). Moreover, it contains an Internal Finger Table with *m* entries (i.e., for the Chord based super node network). Finally, it also should contain table with (1/P-1) entries for storing the ordinary nodes' information (i.e., for centralized super node-ordinary nodes cluster). And for the ordinary nodes, they just need to have one entry to store the super node that it registers to.

4.3.8.2 Example analysis

Based on above introduced equations, we can estimate the average hops, maximum hops and index sizes for some specific cases. For example, there are three networks with 5000 nodes (centralized, Chord, and hybrid Chord), and in the hybrid Chord based network, the proportion of super nodes is 10% (*P=10%*), thus inside each subset service overlay, there are 125 (i.e., 5000*10%/4) super nodes are participating in the organization of the Chord ring.

The average hops $E(N)$ for these three solutions are as follows:

- Centralized: $E_{Centralized}(N) = 2$
- Chord: $E_{Chord}(N) = 6.14$ -> $(\frac{\log_2(5000)}{2})$
- Hybrid Chord: $E_{Hybrid_Chord}(N) = 5.23$-> $(25\% \times (\frac{\log_2(125)}{2}) + 75\%(\frac{\log_2(125)}{2} + 1) + 1)$

The maximum hops $M(N)$ for these three solutions are as follows:

- Centralized solution: $M_{Centralized}(N) = 2$
- Original Chord solution: $M_{Chord}(N) < 13$ -> $(\log_2(5000))$
- Hybrid Chord solution: $M_{Hybrid_Chord}(N) < 9$ -> $(1 + \log_2(125) + 1)$

The index sizes $S(N)$ for these three solutions are as follows:

- Centralized solution: $S_{Centralized}(N) = 5000$
- Original Chord solution: $S_{Chord}(N) = 128$ -> (We use MD5 as the hash algorithm, thus node/resource identifiers contain m=128 bits)

- Hybrid Chord solution: $S_{Hybrid_Chord_supernode}(N) = 140$ -> $(3 + 128 + (\frac{1}{10\%} - 1))$

Figure 4-49 also presents the relationship between the average hops and the number of nodes in the network, in which the number of nodes increases from 5000 to 200000. And in this comparison, we assume in the hybrid Chord solution, there are 10% of the nodes are the super nodes.

From this figure, we can find that, not matter what size the network is, the centralized solution only needs two hops to reach the target resource. And for both Chord and the hybrid Chord based solutions, the average hops increases logarithmically with the number of nodes. Moreover, by comparing with the original Chord based solution, the hybrid Chord based solution can reduce about one hop for the average hops.

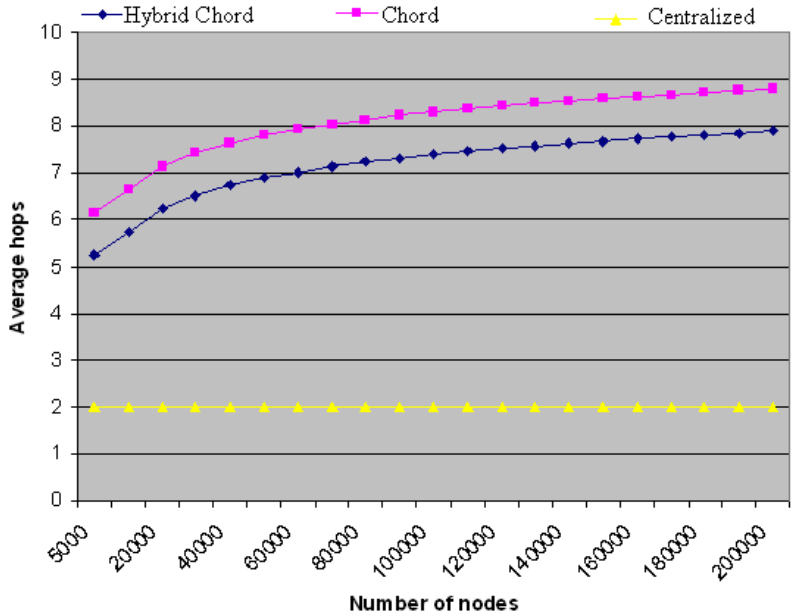


Figure 4-29: Relationship between the average hops and the number of nodes

Figure 4-30 presents the relationship between the maximum hops and the number of nodes in the network according to the mathematical equation we presented above. In this figure, we can find that, not matter what size the network is, the centralized solution only needs two hops to discover the target resource. And for both Chord and the hybrid Chord based solutions, the maximum hops increases logarithmically with the number of nodes. Moreover, by comparing with the original Chord based solution, the hybrid Chord based solution can reduce about 3 or 4 hops for the maximum hops.

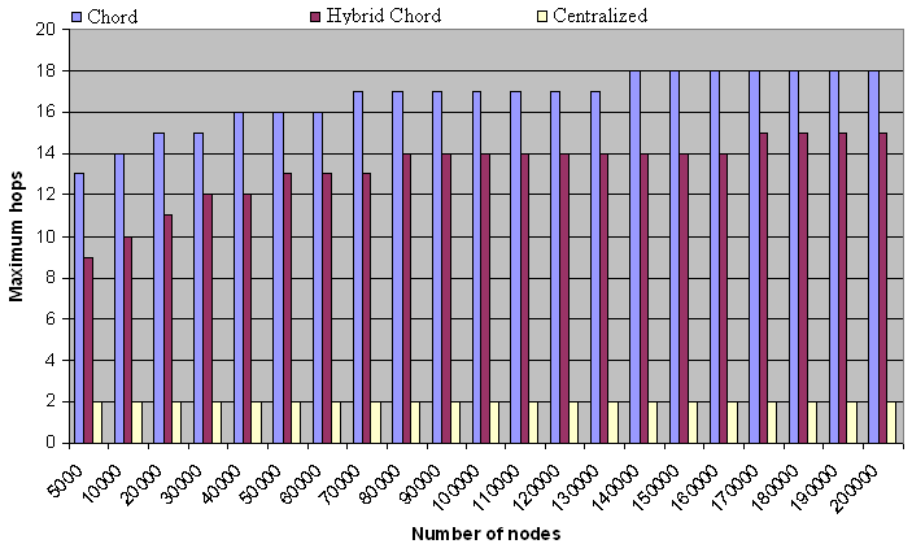


Figure 4-30: Relationship between the maximum hops and the number of nodes

Figure 4-31 presents the relationship between the index size and the number of nodes. For the centralized solution, the index size increases linearly with the number of nodes. And for the original Chord and the hybrid Chord, their index sizes do not change with the number of nodes.

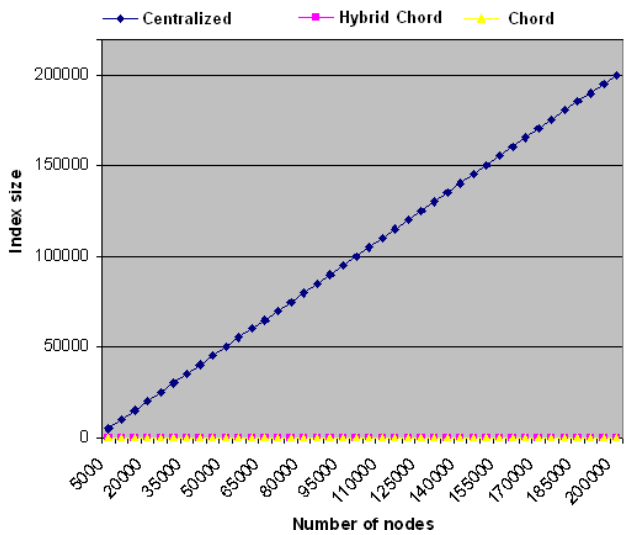


Figure 4-31: Relationship between index size and the number of nodes

In above analysis, we pre-fixed the proportion of the super nodes (10%) in the hybrid Chord for facilitating the comparison with the centralized solution and the original Chord based solution. However, as shown in above theoretical estimation equations for the hybrid Chord's average hops, maximum hops, and the index size, we can find that the proportion of the super nodes also plays a very important role on the performance of the hybrid Chord solution.

Figure 4-32 illustrates the impact on the average hops when the proportion of super nodes increases from 0.08% to 80%. In this analysis, we assume the node number is 5000. From this figure, on the one hand, we can find that the lower proportion of the super nodes (i.e., the higher centralization level for the system), more hops the hybrid Chord solution can reduce, by comparing with the original Chord based solution. When the proportion is reduced to 0.08%, it means there is only 4 super node in the network (5000*0.08%=4). In this case, it can be considered as the centralized solution with 4 individual central registries. In other words, each super node is considered as the central registry for its own subset service overlay. Consequently, if the super node, which initiates the service discovery request (in the Chord and hybrid Chord solutions, it is the nodes that participate in the Chord ring's organization initiate the service discovery request, and in the centralized solution, it is the node connects to the central registry initiates the request), belongs to the same subset service overlay as the target resource (25% of probability), it can discover the ordinary node which holds the target resource in its own index and then route the request to the target ordinary node, in this case, it needs only one hop. Otherwise, if it does not belong to the same subset service overlay (75% of probability), it needs one hop to route the request to the target subset service overlay (another super node), and then it needs another hop to reach the target ordinary node, in this case it needs two hops to reach the target resource. Consequently, the average hops for this specific case is 1.75 hops (1*25%+2*75%=1.75).

On the other hand, we also can find from Figure 4-32 that if the proportion of super node is higher than certain level (here is 35.35%), the hybrid Chord solution not only can not improve the system performance, but also will degrade the system performance by increasing the average hops.

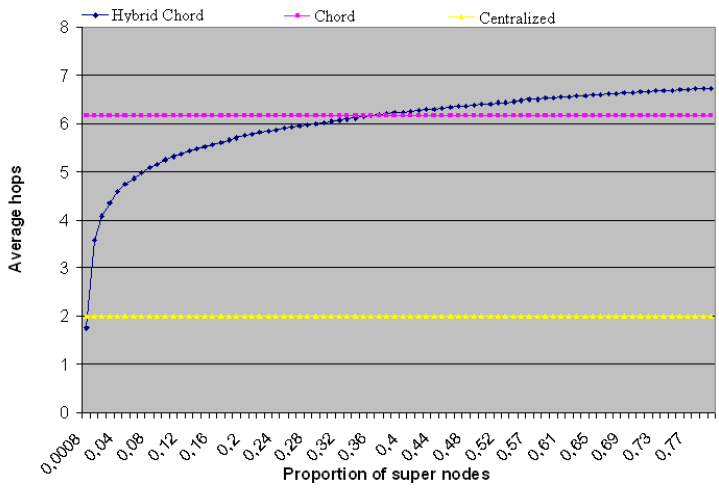


Figure 4-32: Relationship between the proportion of super nodes and the average hops

Figure 4-33 illustrates the relationship between the proportion of super nodes and the maximum hops. It shows that the maximum hops increases logarithmically with the proportion of super nodes. When the proportion of super node is reduced to be 0.08% (only 4 super nodes), it can be considered as the centralized solution, in which the super node acts as the central registry in each subset service overlay, and thus to discover the target node, the maximum hops is 2 hop (one hop for reaching the target super node, one hop for reaching the target ordinary node). When the proportion of super node is higher than 82%, its maximum hops is the same as the original Chord based solution.

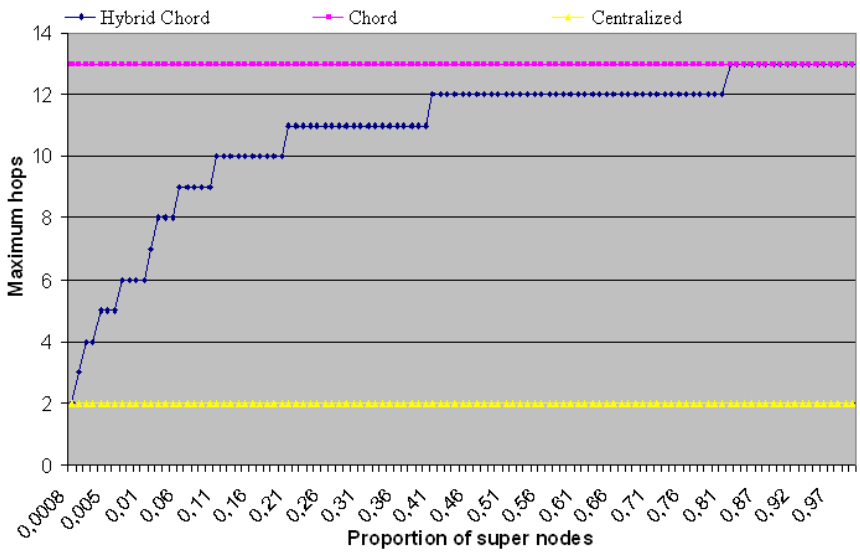


Figure 4-33: Relationship between the maximum hops and the proportion of super nodes

Figure 4-34 depicts the relationship between the proportion of super nodes and the index sizes. From this figure, we can notice that when the proportion of super nodes increases, the index size in the super node reduces. When the proportion of super nodes is 0,02%, it means there is only one super node in the network, so this super node acts as the single central registry in the centralized solution, consequently, it should contains all the ordinary nodes' information, and the index size increases to 4999. If the proportion of super nodes is 0.08%, it means there are 4 super nodes in the network, so each super node acts as the central registry in its own subset service overlay, in this case, it needs to store 3 entries for the neighbor super nodes residing in other subset service overlay, moreover, it also needs to store 1249 (1/0.08%-1) entries for its ordinary nodes. Consequently, it needs to contain 1252 entries in total (in this case, there is no Internal Finger Table for the super nodes). Once the proportion is higher than 0.08%, the super nodes needs to add m more entries (here, m=128, since we use MD5 as the hash algorithm) to its index for storing information of the other super nodes that are inside the same subset service overlay.

If the proportion of super nodes increases to be 100%, it means there is no ordinary node in the system, thus the super nodes just needs to store its Internal Finger Table (128 entries) and its External Finger Table (3 entry).

This figure also implies that if we want to limit the index size in the hybrid Chord solution, the proportion of super nodes should not be too small. For example, if the proportion of super nodes is bigger than 2%, it can guarantee the index size is less than 180; if the proportion of super nodes is bigger than 10%, the index size would be smaller than 140.

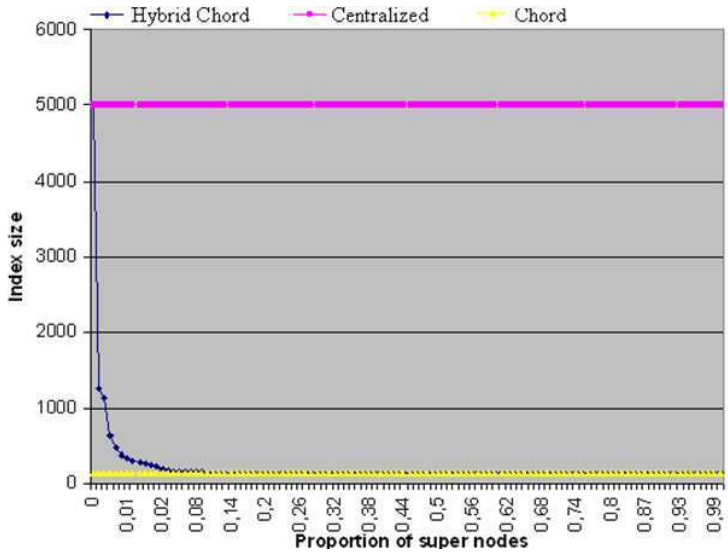


Figure 4-34: Relationship between the index size and the proportion of super nodes

Depending on above analyses for the hybrid Chord solution, and the comparison with the Chord solution and the centralized solution, we can deduce that the hybrid Chord based solution is very scalable, when the number of nodes increases, the average hops and maximum hops increase very slowly (logarithmically). Even for a network whose nodes increase from 5000 to 200000, its average hops only increases from 5.23 to 7.89. And such increase plays no impact on the index size. This proposed solution also can improve the Chord based solution to some extent. For example, it reduces about one hop for the average hops, and reduces 3 to 4 hops for the maximum hops. However, this improvement tightly depends on the appropriate selection of the proportion of the super nodes.

The proportion of super nodes is the key balance value in our proposed solution. As shown above, if the proportion of super nodes is too big, it not only can not improve the system, but also will degrade the system by increasing the average hops. If the proportion of super nodes is too small, the index stored in each super node will be big. Moreover, that also implies the super node – ordinary node cluster will be too centralized, and thus incur the same problems as the centralized solution, such as the node update problem,

and too many query message exchanges, single failure point, and consequently the super nodes are easy to become a bottleneck for service discovery.

Consequently, it is significantly important to select an appropriate proportion value for such a system. According to our previous analyses, we can find that the interval from 0.04 to 0.2 is reasonable for a system. Since it can guarantee to reduce the average hops (from 0.4 to 1.6 hops), and reduce the maximum hops (about 3 or 4 hops), meanwhile, the index size is between [135, 155] which is much smaller than the centralized solution.

4.3.9 Discussion

The solution we introduced in this section improves the service exposure system mainly from the following aspects:

- It respects the diversity and autonomy of the service exposure platforms. Each service platform can use the technologies they prefer to expose their services to users. Meanwhile, it enables the service sharing among different operators, service providers and users regardless of their underlying heterogeneities by introducing a P2P based service information sharing system. That is to say, a user can discover a service provided by her/his service discovery platform provider using its specific technologies, meanwhile, she/he also can discover a service provided by other service discovery platform through the P2P based service information sharing system.

- As there are a great amount of services are published in the P2P based service information sharing system, thus it is necessary to use an efficient service organization and discovery mechanism to ensure user's experience. Thus our proposed solution uses the structured P2P overlay as the distributed service repository network for improving the service publication and discovery system's scalability. The adaptation of Chord protocol as the base for this proposed solution ensures the efficiency of query routing in large-scale P2P networks. To further improve the service discovery efficiency, we separate the nodes in the Chord ring into super nodes and ordinary nodes, thus the average hops needed for service discovery is reduced from $E(D)=\frac{\log_2(N)}{2}$ to $E(D)=\frac{1}{4}(\frac{\log_2(N*P/4)}{2})+\frac{3}{4}(\frac{\log_2(N*P/4)}{2}+1)+1$. Furthermore, these modifications also improve the system by reducing the maximal hops needed for discovering a service, even this service is residing in the last node of the Chord ring. Moreover, these improvements for the Chord based service information system also avoids the irrelevant nodes, which are responsible for services derived from other domains, being disturbed during the service discovery process and

thus they can reserve their resource for other more relevant tasks. Another improvement for the Chord based system by comparing with other traditional Chord based solution is the adaptation of redundant service information storage. That can avoid the single failure point for the system, and thus improve the system's reliability.

- Traditionally, Chord based systems are famous for their high discovery success rate and low average hops needed for the service/resource discovery. Both of that depend on the precondition that it is a deterministic searching, and the current Chord based solutions are lacking of the capability of ambiguous searching. That is because most of the DHT-based solutions locate the resource position by calculating the distance between the resource ID and the node ID. Thus, if a user wants to search an SMS service, but has no idea of the exact name of the concrete service (e.g. Orange SMS), a request with only the keyword of "SMS" will result in a discovery failure, since the hashing of "SMS" will not match any concrete service names stored in the system. However, ambiguous searching is significantly important for a service discovery system. That is also why most of the P2P based service discovery and publication systems are based on unstructured P2P architectures rather than based on the structured P2P architectures, even at the cost of consuming more network traffic and providing lower success rate. To achieve the aim of benefiting from Chord's high efficiency, meanwhile, enabling the ambiguous searching which is indigent in current solutions, we introduce three layers based service classification (Domain Type, Abstract Service and Concrete Service) for the nodes and service description files organization, propose the abstract service publication and discovery, and use the unified semantics and reasoning engines, and thus finally enable the ambiguous service searching in a Chord based system with a high service discovery efficiency.

This proposed solution was originally designed for service discovery that involves Telecom/Web/Device/user generated services. However, it is equally applicable for general purpose P2P resource discovery. In addition, the subset service overlays are not limited to four. More subset overlays can be added if there is a new service category is introduced to the system, or the total amount of the services or the number of the services in some subset overlays has increased to a certain level, and thus provides good scalability and flexibility.

In contrast to the current P2P-based solutions we introduced in Section 4.3.1 which mainly focus on the organization of the registries or peers without the modifying for the content stored in them, our solution not only improves the node organization, but also tries to redistribute the resources. Our system makes use

of the service description files, placing them in more appropriate positions for better resource organization and thereby fundamentally improving the service searching efficiency.

4.4 A P2P based service exposure model in IoT environment

Along with the evolution of service composition, nowadays, not only the traditional Telecom and Web services are involved in this new paradigm for service delivery, but also a great number of small and embedded devices are being included for service composition. In recent years, the concept of IoT has appeared and widely spread in a short time as an important branch for the Future Internet. In the paradigm of IoT, millions of devices or objects are represented in an Internet-like structure. They are empowered to provide and consume data about the real world, be interconnected among each other, and cooperate with other kinds of services (e.g. Web and Telecom) for providing meaningful services to users, who are more and more demanding. This is related to the phenomenon of device mashup.

Device mashup, similar to the service composition concept for Telecom and Web services, implies the capability to integrate everyday objects, devices and online applications in a seamless way, for providing a more meaningful user experience, irrespective of the disparate natures of the technologies being used. In order to enable the functionalities over devices to be composed together, the information about devices (both hardware and software information) must be organized and exposed in a way that allows efficient discovery and appropriate selection. It requires the capability to make the “smart” everyday objects and devices connect themselves to a large computer network, and make them be accessible wherever they are regardless of the underlying complexity and heterogeneity. This requirement is mainly related to the paradigm of device service exposure.

The intrinsic features of these small or embedded devices make their exposure more complicated, and the classical solutions for Telecom and Web service exposure currently are not feasible to be reused directly for the device-based services. Below are some intrinsic features that bring the challenges for the device service exposure:

- One intrinsic feature of devices is that they are more heterogeneous by nature, in terms of hardware (e.g. size, CPU, memory), software (e.g. operating system, runtime platform), usage (e.g. from toys to building automation). These intrinsic heterogeneities present significant challenges for service provisioning and exposure, and make it difficult to use a single service exposure framework to manage all kinds of devices.

- Besides, most of the devices are resource-limited (e.g. varying constraints in terms of mobility, battery and computation, etc.), and some of them cannot access to an IP network directly. For this kind of devices, they generally need gateways to delegate themselves to external parties. In this case, the service exposure is performed by gateways instead of by the devices themselves directly.
- Moreover, devices have a highly dynamic nature. Different from Telecom and Web services, most devices are not required to be always on, and certain devices may have very short wakeup time and long hibernation time, thus the presence information is extremely important. This kind of highly dynamic information is difficult to be included in service description files and then published to the network.
- Furthermore, there are a large number of devices existing in the networks, and their number is much more than Telecom and Web services. That is because the devices with same or similar functionalities can be provided by different manufacturers and owned by different users, and each user may possess several devices. Consequently, the widely used centralized solutions are easy to become a bottleneck, and thus degrade service discovery efficiency, when the number of devices is increased to a certain level.

Taking these challenges into account, a new service exposure model is considered as necessary to cope with the heterogeneity and complexity presented by the underlying device layer. This model should enable the disparate connected devices to expose their different capabilities, resources, data format and communication protocols in an inter-understandable method while respecting their intrinsic heterogeneities.

In fact, in our previous proposed service exposure models (Section 4.2 and Section 4.3), they already cover the device offered services, however, such device exposure within them are under the presupposition that these devices have the capability to publish their built-in functionality directly to the global network. That implies these devices are powerful enough and have at least IP accesses to the network directly. However, in the paradigm of IoT, there are also a huge number of weak devices that cannot access the IP network directly and generally reside in a local network. For this kind of devices, gateways are generally required to monitor them and delegate them for the external world. The exposure for them will be completely different from the traditional Telecom and Web services. Consequently, the solution we present in this section mainly focuses on the devices that need gateways to connect them with external parties, expose them in a global network, and further enable the service information sharing among different users, different service platforms. Moreover, to avoid the single failure point for the centralized UDDI-like solutions and the possible appearance of bottleneck while the number of devices increases, P2P is

considered as an alternative solution for addressing these issues. However, the P2P based solution we introduced in Section 4.3 can not be applied directly to this kind of service exposure. That is because, the device gateways not only contain the concrete devices' description files, but also their runtime information, these runtime information needs to be updated frequently, and thus can not be re-distributed to the P2P network as we introduced in 4.3. Consequently, in this context, device gateways should act as the P2P nodes in the distributed service information sharing system. However, as each gateway may contains several different device-related services, it is impossible to assign a single hash ID to the gateway according to the content it contains, and then chain all the gateways to the Chord ring. Consequently, another improved P2P based service discovery mechanism is considered as necessary to improve service discovery efficiency in such IoT service environment.

4.4.1 Device service exposure

The devices can mainly be divided into two main categories: IP devices and non-IP devices. For the IP devices that do not need to connect the IP network through a gateway, a client can be installed on, and their built-in functions are exposed to the proper network repositories (e.g. RESTful servers, Device Profile for Web Service (DPWS) servers, etc.). In our proposed IoT based service exposure model, this kind of repositories can act as the distributed nodes for a P2P network that enables other nodes to discover the services held on them.

And for devices that connect the large-scale IP network through gateways, service exposure is delegated by the gateways they belong to. As there are a huge number of devices and gateways exist in the network, if all the devices' information is published to a global repository directly, it would weaken the service discovery efficiency, especially considering the fact that there are many devices which are providing the same functionalities with different nonfunctional characteristics and owned by the different users. Taking this challenge into account, we propose to separate the device service exposure process into two phases: Local Device Exposure and Global Device Exposure.

- Local Device Exposure: relies on the local gateways, and targets the devices that do not have direct IP connection to the global network. The local gateway contains the description files and the drivers of all the devices that are residing in the smart space where it controls. It also contains the contextual information of these devices (e.g. the presence information of these devices).
- Global Device Exposure: is realized by the interoperation between the network nodes and several global servers. The device gateways act as the network nodes. They interconnect each

other through the P2P method, and share the device information stored on them among each other.

Through these two device exposure processes, the nodes that have multiple types of devices, different power and processing capabilities can interact with the devices or applications that reside in other network seamlessly.

4.4.1.1 Local device gateway

Figure 4-35 illustrates a reference architecture for Local Device Gateways. From a functional point of view, the main role of a Local Device Gateway is to enable interaction among the devices residing in a local smart environment, and the interoperability between these local devices and the large-scale services derived from Web and Telecom domains. That is generally related to the actions such as device discovery, device exposure and device execution.

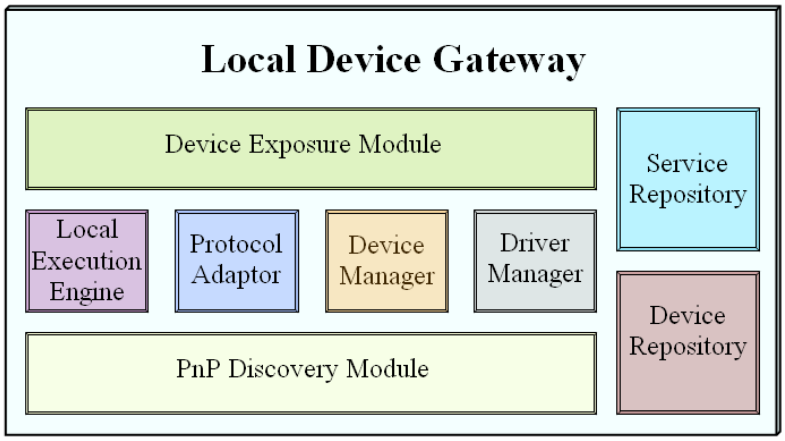


Figure 4-35: An example of the local device gateway

When a device is connected to a smart space, firstly it should be detected by the gateway. This process mainly relies on the PnP (Plug and Play) Discovery Module by using the mechanisms to notify the presence and reachability of the internal devices. Several techniques such as, SIP, DPWS, Bonjour, DNS, REST, UPnP, USB, and DHCP, can be used according to the different techniques adopted by a smart space.

Once a device is discovered, its driver is run by a Driver Manager component, if this driver has already been installed in the gateway. Otherwise the Driver Manager contacts with Global Driver Repository in the large-scale network for finding out the corresponding driver and downloading it to the gateway automatically. If no relevant driver is discovered, it notifies the absence of a driver to the user through a user-friendly device management front-end, and requires the user to install one manually.

The devices are more dynamic than the traditional Telecom and Web services. Some devices can be similar to the Telecom or Web services, which are always on. However, there are also some devices which may have very short wakeup and long hibernation time, or certain devices only be turned on when they are needed. Consequently, it is necessary to track the devices' availability and lifecycle. Such device availability and lifecycle management can be handled by the Device Manager component. In essence, Device Manager needs to detect any change inside its local network related to the devices. For examples, a device connects or disconnects from the local network.

Ideally, a set of devices in an enclosed or local environment (e.g. home or building environment) should be configured in a way that they share standards or common communication protocols. Nevertheless, this statement is not always true, thus a set of translation mechanisms are needed, in order to cope with heterogeneity presented by the underlying device layer. One of such translation functionalities is protocol adaptation. Despite being in the same local environment, different devices may use different communication protocols. In order to make these disparate devices interact with each other, their communication protocols should be adapted to the local environment. Moreover, most of the devices are not designed to be connected in the Internet due to their dependency on local multicast communication method for device discovery, which is not suitable for large-scale networks, such as Internet. However, nowadays one trend for the next generation delivery is convergence, in which not only the traditional Telecom and Web services are involved, but also the device offered services are required to be involved in the paradigm of service composition. In this context, a mechanism is required for translating device communication protocol based messages into large-scale network interoperable format (e.g. from UPnP based messages to REST based messages), thus enabling the interoperation between the local devices and the large-scale network services. Such protocol translation can be implemented at the gateway, as defined in Figure 4-35, relying on the Protocol Adaptor component.

Moreover, certain services in the local smart space are provided by individual devices, and whereas some services are provisioned by combining several devices. For example, a service provides the function "when the temperature is below 20°C, turn on the heater" that involves both the temperature measurer and heater. This kind of service is always related to the paradigm of service composition. Similar to the service composition in Web and Telecom, aggregating different devices into a unified service requires an additional mechanism to chain the disparate services offered by individual devices together. In our proposed gateway model, this is performed by Local Execution Engine. Briefly, when a user wants to execute a device-offered service, the service invocation request is forwarded to the Local Execution Engine firstly. If the service to be executed is related to a single device, the Local Execution Engine contacts with

Device Manager and Device Repository to verify the availability and the invocation method for this device. If the device is available and in the “online” status, the Local Execution Engine invokes the selected device directly by executing the target functionality or retrieving the relevant data; if the device is already installed in the local network, however, its current status is “offline” or hibernation, the Local Execution Engine actuates the device firstly, and then invokes the relevant service. If the target service is a composite service, the Local Execution Engine retrieves a service execution profile that contains the devices’ invocation logic from Device Repository. And then, according to the priority order and the trigger conditions defined in this service description file, the Local Execution Engine chains the relevant devices together for providing a composite service to end-user.

4.4.1.2 Local device exposure

Another important functionality for Local Device Gateways is device exposure. One of our goals to propose this model is to enable the devices residing in a smart environment to be reused by other users or the external applications. In order to accomplish this, the devices should first be made to be understandable and discoverable by other users. As we introduced at the beginning of this section, we separate the device exposure process into two phases: Local Device Exposure and Global Device Exposure. Local Device Exposure relies on the gateway by interoperating Device Exposure Module and relevant repositories.

The repositories inside the local gateway can be separated into two main groups: one is conceived for device’s monitoring and controlling purposes, such as the Device Repository and the Driver Manager. The second group is designed to contain the device description information for enabling the further usage by other users or applications outside the local network. The device description files stored in this repository contain the information related to these devices’ hardware properties (e.g. vendor, Universal Unique Identifier (UUID), hardware nature, etc.), provided functionalities, input/output relevant information, access interface, as well as some nonfunctional information (e.g., privacy, location, owner, etc). Different from the former group of repositories whose information is mainly used in local network for the device execution, the second group of repositories needs to expose their stored device information to the external networks through Device Exposure Module. In other words, once a device is introduced to the local network, if user wants to make this device accessible remotely or to share this device’s functionality with other users or applications, she/he can create a device description file through a user-friendly device management front-end. The Device Exposure Module provides a set of facilities to ease the device description files creation. Generally, device-offered service can be viewed or controlled by users via a Web browser, such as the presentation specified in Universal Plug and Play (UPnP). Once a user wants to publish a device service in her/his local network, she/he can create a device description file or get one from

the device manufacturer, and add this device description file to the personal device monitoring page. After the user confirms her/his action, this device description file is added to the local gateway's Service Repository component. Finally, this device description file can be shared in the distributed service network.

4.4.1.3 Global device exposure

From the viewpoint of the global network, there are many local gateways managing their own local networks using heterogeneous technologies and dispersed in the network. This situation implies a distributed device information base. Furthermore, when a user or developer is creating a new service, he/she can use his/her own devices within his/her gateway, however, he/she also can use some external devices that reside in other gateways. Thus, this distributed device information needs to be published in a globally-comprehensible manner in order to enable all the devices located behind the local gateways to be discovered, executed and composed in a unified manner. Correspondingly, in terms of device service publication process, there are two kinds of publications: abstract device publication and concrete device publication, which is similar to the service publication we introduced in section 4.2.2.3.

An abstract device means a virtual device that only contains the generalization information of one kind of device, and can map to several concrete devices. The abstract device publication is handled by professional developers or service creation platform providers. They send a device description file that only contains the generalization information of one type of device to Semantic Engine. The Semantic Engine contacts an Ontology Repository to add the necessary semantic annotations. The transformed service description file is then sent to the global Semantic Abstract Device Repository.

The concrete device information is published in a Local Device Gateway as we introduced in section 4.4.1.2. In order to make these concrete devices discoverable, each gateway needs to map the concrete devices within it to the relevant abstract services, and create a Local Abstract Service List to store it in Device Exposure Module. These gateways then interconnect each other through an underlying P2P network in which they act as peers. Such process can be considered as Global Device Exposure.

4.4.2 P2P based service information sharing

In our proposed solution, device-based service description files are published in their Local Device Gateways, and these distributed gateways share their service information through the P2P pattern for avoiding the single failure point and improving the service discovery efficiency.

In order to improve the query performance and maintain a high degree of node autonomy, a set of Semantic Overlay Networks (SONs) are created above the distributed gateways. Essentially, each gateway

connects to the global network and acts as a peer in the P2P system. They share the service information stored in their local Service Repository through Device Exposure Module. As these device offered services' description files are semantic enriched, the peers which hold theses semantic description files can be considered as semantic enriched as well, as a result. Then nodes with semantically similar service description files are "clustered" together. Consequently, when a user wants to discover a device or a service provided by the device, the system selects the SON(s), which is (are) better suited to answer. Then the query is sent to one of the nodes in the selected SON(s) and further forwarded to the other members of that SON(s). That will greatly improve the query performance, since queries are only routed in the appropriate SONs, which would increase the chance to find out the matching files quickly with limited cost.

4.4.2.1 Triplex overlay architecture overview

To realize the above mentioned SON based semantic service discovery, we propose a triplex overlay based P2P system as shown in Figure 4-36.

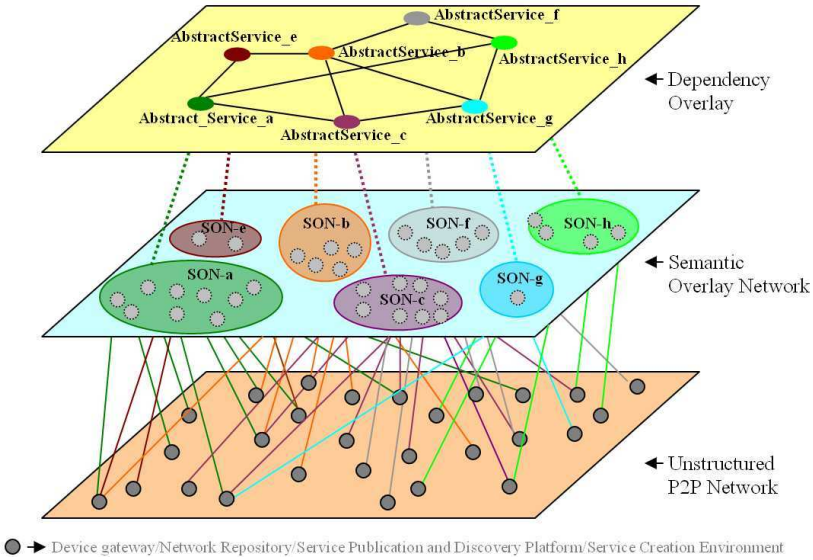


Figure 4-36: Triplex overlay for P2P based service discovery

As shown in this figure, the diverse device gateways, network repositories, service discovery and publication platforms and service creation environments join the P2P based distributed network. They act as nodes in an Unstructured P2P Network layer using blind searching solutions (e.g., Flooding based solutions or RandomWalk based solution) to interact each other.

Then above the underlying P2P system, the nodes providing similar resources are clustered together. We reuse the name SON (Semantic Overlay Network) for this kind of clusters in our current research. The benefits of this strategy include: first, the request is sent to the nodes which have a high probability of

having the target services, thus the request can be answered faster; second, the nodes that have a low probability of having the target services will not receive the request, therefore avoiding a waste of resources on the request transferring, and allowing other more relevant requests to be processed faster.

To further improve the service discovery efficiency, an additional Dependency Overlay is created upon the SON layer. That is because different kinds of devices may be able to interoperate with each other. For example, the output of a device can be the input of another device. Such cooperation among devices allows service providers/developers to provide some more meaningful services to end-users. Thus we can infer that: the devices connected to a gateway have a very high probability of having certain interdependency amongst each other. The service dependency relationship can also be specified according to some social network information, such as service usage frequencies, users' service usage habits, or some network statistic data. Consequently, defining the dependency among the abstract devices, then providing recommendations for the message routing during service discovery process, will improve the success rate.

Dependency Overlay is generated by system administrators. It can be enriched by associating certain auxiliary information for differentiating the different dependency intensity. Such dependency relationship can be either unidirectional or bidirectional. For example, a temperature sensor has unidirectional dependency with a heater, since the output of a temperature sensor can be the input of a heater. The dependency between cameras and doorbells can be bidirectional. That is because, once a doorbell is pressed, it can trigger the camera above it to capture the visitor's face. On the other hand, once a camera captures a person, it can also actuate the doorbell automatically. For the second case, the dependency intensities for two direction dependencies are different: obviously, the former one is more widely accepted by users, thus the dependency intensity for "doorbell->camera", which means the output of a doorbell can be the input of a camera, is higher than the dependency intensity for "camera->doorbell". During the service recommendation process, devices that have higher dependency intensity with the target device are recommended to the user with higher priority. The service dependency definition is not limited to device offered services, but also includes the dependency among the device offered services, Telecom services and Web services, e.g., the dependency of "camera->MMS". As the publication of abstract devices or abstract services is performed by members of the system administration group, consequently once a new abstract device or abstract service is published, the dependency relationship is created simultaneously.

4.4.2.2 SON layer generation

First, we assume there is a set of bootstrap nodes that guarantee the minimal running requirements for the proposed system. These bootstrap nodes form a small scale SON overlay before the running of the system.

That is to say, when a new abstract device or abstract service is introduced to network by a system administration member, this new abstract device or service is assigned to a bootstrap node randomly. In our following illustration, we only consider the abstract services as the abstract devices-offered services, and unify the name as “abstract device”.

Moreover, each gateway contains a table called Local Abstract Service Table. This table is created by mapping concrete device files that are stored in a local gateway, with abstract service profiles that are stored in the Global Abstract Service Profile Repository. In this table, each abstract device entry contains the basic information of the relevant concrete devices (e.g. concrete device’s name), as well as the links (e.g., a URL) to the corresponding concrete device description files as shown in Figure 4-37. Based on Local Abstract Service Table, a device gateway can join the relevant SONs automatically. Since there are several abstract devices are contained in one gateway, thus one gateway can join several SONs according to the different abstract device concepts.

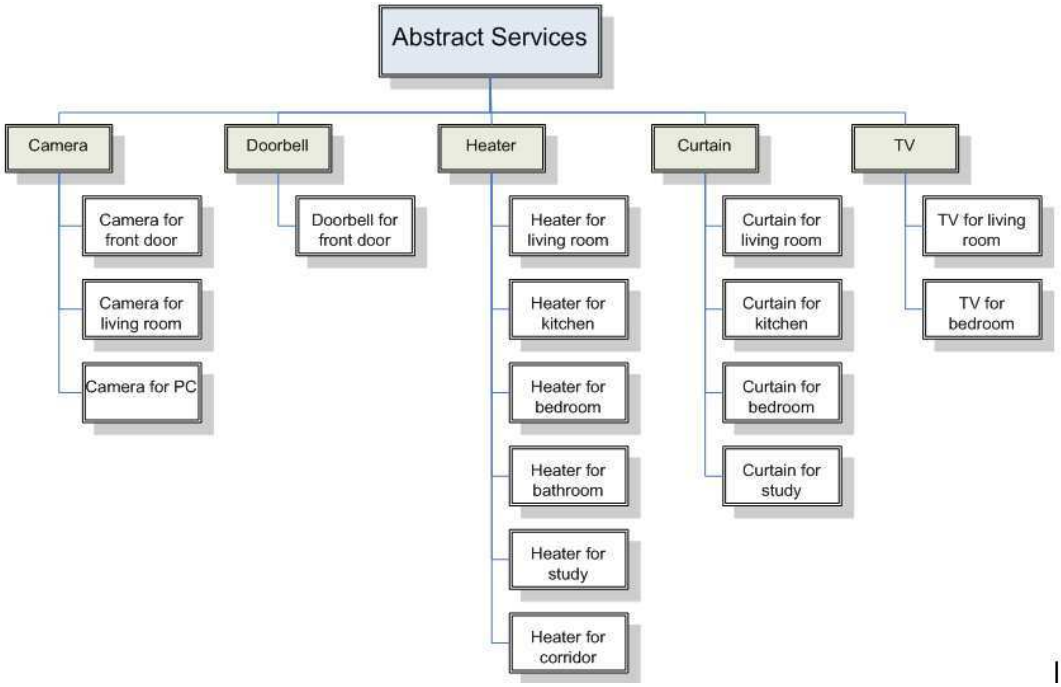


Figure 4-37: An example for Local Abstract Service Table

To clarify the SON generation process and update process, we consider two cases for these processes: (1) a new gateway is added to the network with a list of abstract devices to be exposed; and (2) an existing gateway updates its list of abstract services, which means a new type of device has been introduced to its local network.

When a gateway is introduced to the system, it first joins to the global network through some bootstrap nodes as the ordinary P2P networks do. That is, once the selected bootstrap node receives the *Join* message from a gateway, it broadcasts the *Join* message to the global network. According to certain neighborhood selection rules (e.g., the solutions introduced in [95] and [96]), some nodes are selected as the logic neighbors for this newly introduced gateway and added in the Ordinary Neighbor Node Table (ONNT) as shown in Figure 4-38. In this example, we assume each node contains 5 neighbor nodes' information (e.g., IP and UDP port). This process provides another possible way to search a service: if both the SON Overlay based searching and the Service Dependency Overlay based searching have not found out the relevant services, the system can use the basic RandomWalk or Flooding solution according to the neighbor nodes' information stored in the Ordinary Neighbor Node Table.

Ordinary Neighbor Node Table

Neighbor Node	IP	UDP
1	192.168.20.53	6348
2	157.159.100.41	1433
3	91.168.34.74	8038
4	192.168.22.22	6348
5	82.212.95.78	8038

Figure 4-38: An example of a simplified Ordinary Neighbor Node Table

We assume that each gateway contains another table called SON Linkage Table (SLT) as shown in Figure 4-39, which is used to form the SONs. When the gateway joins to the network for the first time, this SLT is empty as shown in Figure 4-39-a, which means it has not joined any SON yet. After joining to the unstructured P2P network, this new gateway extracts the names of the abstract devices stored in its Local Abstract Service Table, and encapsulates them as an *Update* message. This *Update* message is injected to the network by the blind searching method according to the neighbor nodes' information stored in the Ordinary Neighbor Node Table. In particular, we use the RandomWalk as the basic message routing approach, in which only one neighbor node is selected from the Ordinary Neighbor Node Table for routing the *Update* message. In the *Update* message, the extracted abstract services' names are put into a list, with each entry assigned a time-to-live (TTL) called Time-to-Live for Node (TTL_node). TTL_node is a pre-assigned number *n*, which indicates how many neighbors in the relevant SON have to search. A global time-to-live called Time-to-Live for Hop (TTL-hop) is also assigned to the *Update* message for setting the maximum number of the logic hops for the message.

SON Linkage Table

SON	Neighbor Nodes		
TV			
Heater			
Camera			
……			
Doorbell			

SON Linkage Table

SON	Neighbor Nodes		
TV	Node1	Node3	Node5
Heater	Node2		
Camera	Node1		
……			
Doorbell	Node3		

SON Linkage Table

SON	Neighbor Nodes		
TV	Node1	……	Nodem
Heater	Node2	……	Nodek
Camera	Node1	……	Nodeh
……	……	……	……
Doorbell	Node3	……	Nodeg

Figure 4-39: An example for SON Linkage Table

The generation process for the SON Linkage Table is as shown in Figure 4-40. That is to say, the *Update* message is first forwarded to one of this gateway's neighbors, denoted by *Ni*, and the node that receives this message checks its own SLT. If this table has no entries containing the same name of abstract devices indicated in the *Update* message, this request is directly forwarded to another neighbor of *Ni*. In this case, only the TTL-hop decreases by 1. Otherwise, if one or several entries in this table contain the same name(s) as the abstract devices listed in the *Update* message, both the TTL-node(s) for the relevant abstract device(s) and the TTL-hop are decreased by 1, and the node information (e.g. IP and Port number) is sent back to the original node. The original node then stores the node information in relevant entries of SLT as a neighbor, as shown in Figure 4-39-b. After this, the *Update* message is forwarded to one neighbor of *Ni* node. The above process repeats until either all the TTL-nodes or the TTL_hop are decreased to 0. If all the entries in the SLT have been populated with SON neighbors' information, the join process for SON is regarded as a successful one; otherwise, the original node selects one of its other neighbors to send the same *Update* message. If all the neighbors of this original node have been selected for forwarding this *Update* request, and some entries in the SLT are still not completely filled, then the original node sends the *Update* message to its bootstrap. This connected bootstrap searches among the bootstrap nodes for finding out which bootstrap node(s) is responsible for the corresponding abstract device(s), and sends information of the relevant bootstrap node(s) back to the original node. The original node adds the information into the relevant entries in the SLT and marks it as bootstrap node's information. This process guarantees that if such an abstract device has been predefined in the network, even the relevant SON has not been created yet, the original node can create a new SON, or join to an existing SON by connecting itself to the bootstrap node directly. Once another neighbor in the same SON is found, the information of the relevant bootstrap node is replaced by the newly found neighbor. This process is repeated on a regular basis to make sure that all the information stored in the SLT is up-to-date. This means that once information of a neighbor becomes obsolete or this neighbor is not accessible anymore, a new *Update* message with the names of relevant abstract devices, which need to be updated, is sent out again to the network and starts a new join SON process.

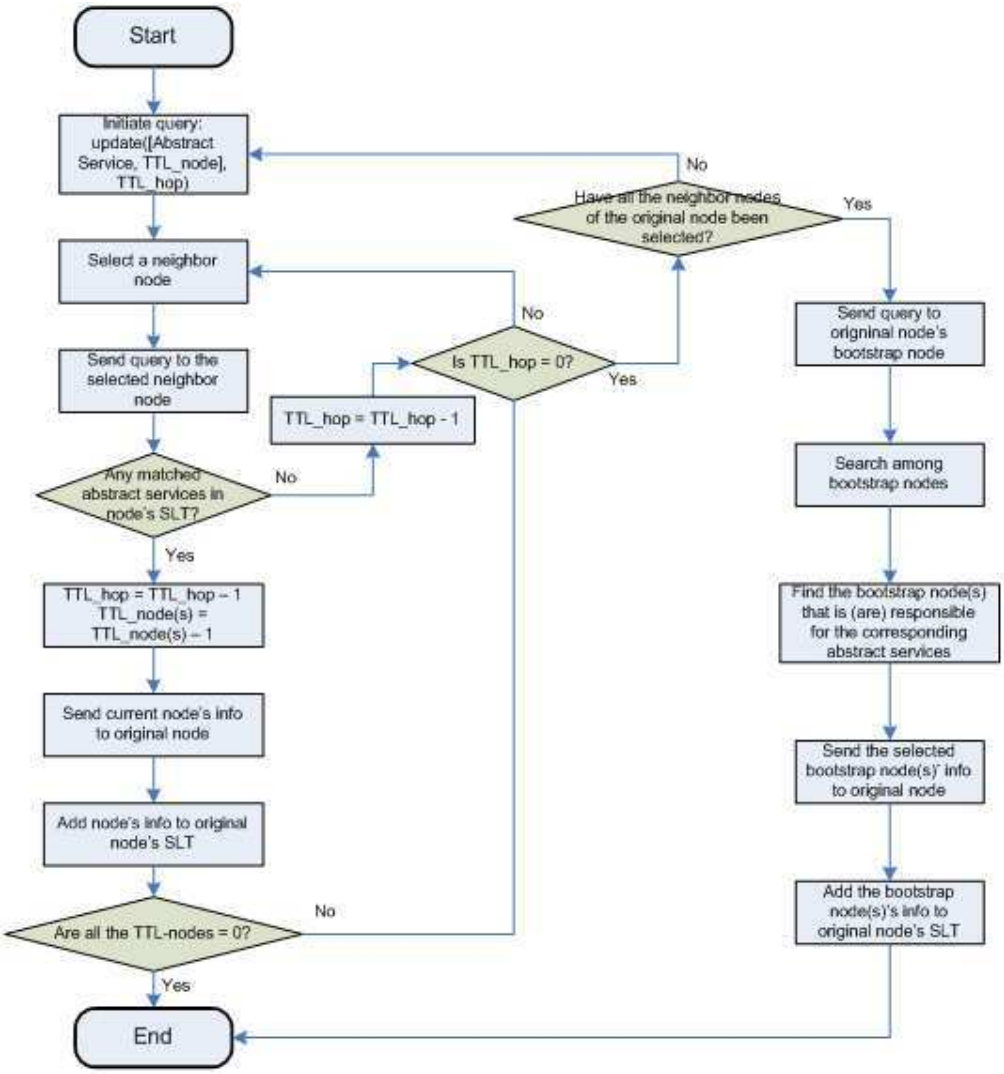


Figure 4-40: Flowchart for SON Linkage Table generation

4.4.2.3 Service discovery based on triplex overlay

Before introducing service discovery based on triplex overlay, we classify the neighbor nodes for a node to be two kinds: the ordinary neighbor nodes and the SON neighbor nodes.

- The ordinary neighbor nodes: are the neighbor nodes that are selected by the unstructured P2P network when node *Ni* joins the P2P network. They are stored in a local table of node *Ni* called Ordinary Neighbor Node Table.
- The SON neighbor nodes: are the neighbor nodes that are stored in *Ni*'s SON Linkage Table. These nodes can be further classified according to the SONs they belong to.

When a user wants to discover a device offered service for the purpose of executing it directly or integrating it into her/his created composite service, she/he can issue a service discovery request through a user-friendly service discovery front-end or through a user-centric service creation environment. We assume these service discovery and creation framework have connected themselves to the global network in advance.

Once the user creates a service discovery request either by entering keywords or natural language based text, or even by selecting the abstract service building blocks, the relevant frameworks formalize a request that contains the target service information (e.g., a *get(target_device, TTL, auxiliary_info)* message). By contacting with the global Semantic Engine, the global Ontology Repository, as well as the global Abstract Device Repository, this newly created request is interpreted into the system recognizable format. For example, if the user has entered "get temperature", this request is mapped to the abstract service "Target device: Temperature sensor". The request is injected into the unstructured P2P network through the node that initiates this request. To facilitate the request interpretation process, the relevant service discovery or service creation environment can install a local Semantic Engine and Ontology Repository, and store the abstract service description files in its local repository. In this case the request can be formalized automatically inside its local framework before being injected into the P2P network.

The *get()* request is injected into the network through the blind searching method. That is to say, the node, which initiates this request, forwards the request to all its ordinary neighbor nodes or one of its ordinary neighbor nodes, and that mainly depends on which blind searching strategy (e.g., Flooding or RandomWalk) the system adopts. To improve the service discovery efficiency, the system first needs to find out the corresponding SON for the message routing. Consequently, as shown in Figure 4-42, the first node(s) that receive the service searching request will verify if they contain the target abstract device, by matching the entries' name stored in their Local Abstract Service Tables with the target device's name indicated in the incoming request. If matched, the entry with the same abstract name in the SLT is selected, and the received request is then forwarded to the SON neighbor nodes whose IP and Port information is stored in this selected entry. Meanwhile, this node extracts the context information (e.g. user identifier, location, preference, etc.) encapsulated in the received message, and forwards this context information, together with the target abstract service's name, to its own Local Service Exposure Module. Local Service Exposure Module contacts with Local Service Repository for checking if any services stored in its local gateway can be used by the user, who issues this service discovery request. It then makes a primary filtering for the discovered result according to the auxiliary context information. For example, if a device is set to be public (e.g. the cameras in a public park), a user can discover this service once the request reaches

this gateway. If a device is set to be private, even it matches the functional requirement for the target device, this gateway, however, still needs to verifier if the user identifier in the request is included in the identifiers that have been granted by the owner of this gateway for the use of her/his devices (e.g. the identifiers for his family members). If the original user identifier is matched, the relevant service description file is sent back to the service discovery framework. Otherwise, no service description file is sent back to the service discovery framework, and this node is marked as no resource matched to user's request.

When the SON neighbor nodes of this node receive the forwarded request, as these nodes are already in the same SON, and this SON is the target SON in which all the nodes contain the service mapped to the target abstract service, they thus invoke Local Service Exposure Module for local service discovery directly without the need to verify if any SON they belong to match the target abstract service. At the same time, they forward the request to their SON neighbor nodes, which are extracted from the entry in their SLT. This process continues until the TTL for this request is reduced to 0, which means the service searching is terminated.

If the first node(s) that receives the service searching request do not belong to the SON for the target service (e.g. Camera), which means that no entry name in Local Abstract Service Table is the same as the target service's abstract name. In this case, this node needs to find out which neighbor nodes are the most possible ones that in turn find out the relevant SON the target service belongs to. We assume each gateway contains a copy of such an abstract service dependency relationship file. In order to select the most possible neighbor to find out the target SON, the node that does not belong to the target SON extracts all the abstract services' names from its Local Abstract Service Table and the target abstract service name from the incoming message. It then analyzes the services dependency relationships among these extracted abstract services relying on the abstract service dependency file, and selects the name of the abstract service whose dependency intensity with the target abstract service is the highest one, as show in Figure 4-41.

Target Abstract Service	Local Abstract Service	Dependency Intensity
Camera	TV	5
Camera	Doorbell	4
Camera	Heater	1
Camera	Curtain	0

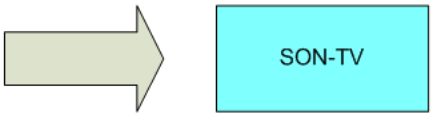


Figure 4-41: Comparison for the abstract services' dependency

According to the selected abstract service's name, the node selects the SON neighbor nodes from an entry in SLT whose entry's name is the same as the selected abstract service's name, and then the service searching request is forwarded to these selected SON neighbor nodes. When these SON neighbor nodes receive the service searching message, they first check their Local Abstract Service Table for finding out if they contain the target abstract service in their local gateways. If a node finds that it contains the target abstract service, it searches its SLT, and selects the SON neighbor nodes from the entry whose name is identical to the target abstract service. It then forwards the service searching request to the corresponding SON, and finally the service searching is limited to this SON that is responsible for the target abstract service. If a node cannot find the target service, it forwards the request to their neighbor nodes that are in the same SON as the first contacted node selected.

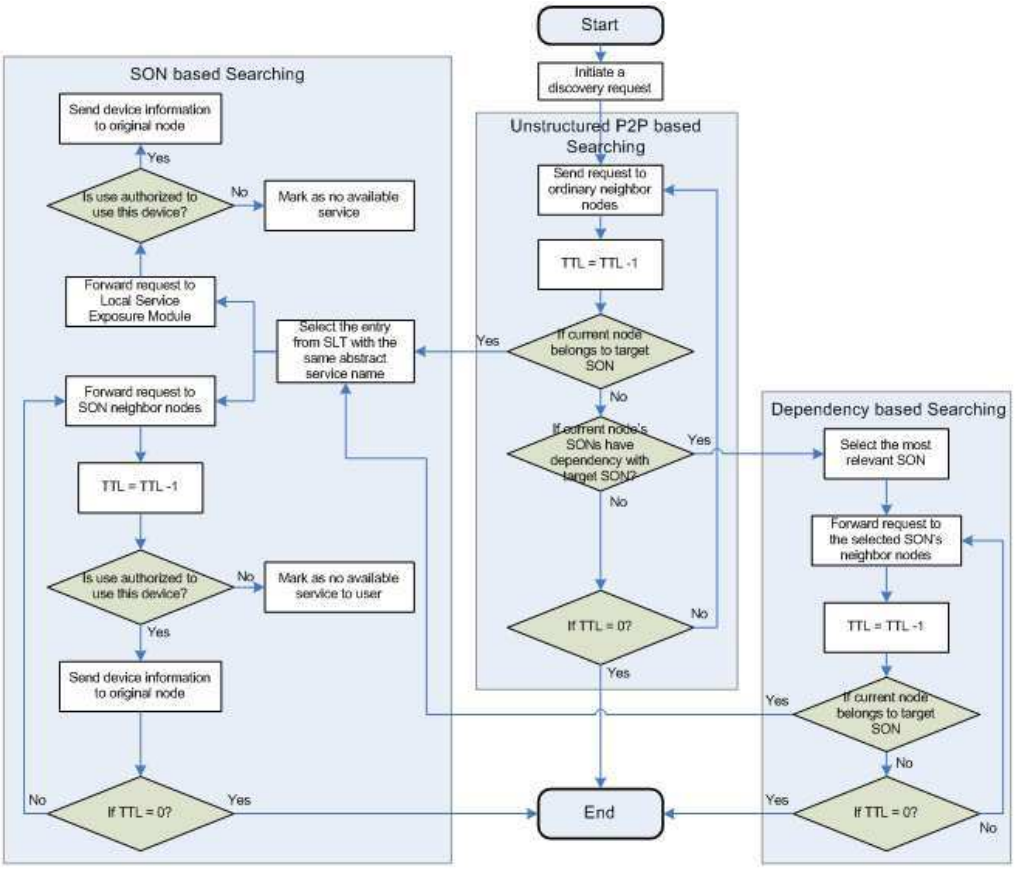


Figure 4-42: Flowchart for service discovery process in triplex layers based service exposure model

If the first contacted node(s) does not belong to the target SON, and it also has no dependency relationship with the target abstract service, the request in the first contacted node(s) is forwarded to its ordinary neighbor node(s) following the blind searching method adopted by the underlying unstructured P2P network. When its ordinary neighbor node(s) receives this service discovery request, it repeats the

service discovery process we introduced above. This process repeats until either the target SON or a dependent SON has been found out, or the TTL is time out.

4.4.3 Performance analysis and discussion

In order to evaluate the performance of this proposed service exposure and information sharing model for the device offered services, the theoretical analysis and several simulations on the success rate and the network traffic impact have been done. We also compare it with the traditional blind searching strategy for the unstructured P2P solutions (which are based on Flooding or RandomWalk), as well as the pure SON based solution.

Considering the fact that, nowadays a lot of work has been done on the unstructured P2P platforms for improving the hit delay by reducing the average hops. However, most of them are at the cost of increasing the network traffic. For example, we have set up a P2P simulation platform based on the Gnutella algorithm. This platform can guarantee a success rate higher than 40% with an average hop of 6.45 in a P2P network containing 5000 nodes. Nevertheless, the involved messages for service discovery are up to 5^6 (15625). This result shows that even if the user experience can be improved by reducing the search delay (hops), the network traffic is easily overloaded if many users use this system simultaneously. Consequently, we evaluate our proposed solution from the network traffic aspect – the total invoked messages for a successful service discovery process, instead of the traditional considered parameter - average hop.

In the following parts, we use the average messages required for discovering a service as a metric for our proposed system. This metric can be regarded as a representation of the invoked network traffic for a service discovery request. In ideal cases, in which only one message exchanges between two neighbor nodes, then the value of this metric is equal to the average number of nodes that are involved for a search process.

To simplify the analysis, we denote the success rate as S(T), and the average message as E(S). We assume that services are distributed in the network evenly and their probabilities being discovered *p* are the same if we use the blind searching solution. The SONs are also assumed to be evenly distributed in the network, and the difference among their sizes is ignored. When a request arrives at a node, the probability that the connected node belongs to the target SON is R. If this node does not belong to the target SON, Dependency Layer provides the recommendation for heightening the probability to find out the target SON in the next hop. This increased probability is denoted as K. In general, we have K>R. If the request arrives at the target SON, its probability to finding out the target service is much higher than the one it searches in the underlying unstructured layer by using blind searching strategy. Even though within the SON, it still

uses the blind searching mechanism. That is because, when a request arrives at a SON, the search space is reduced, but the total number of the user accessible services within this SON is the same, thus the local discovery probability is much higher. Moreover, it can guarantee that all the target resources are clustered into this SON, thereby the nodes outside this SON can be out of consideration.

Theoretically, the success rates for the blind searching solution, the SON based solution, and our proposed triplex solution can be calculated using following equations.

(1) Blind searching solution:

$$S_{Blind}(T_s) = p + (1-p)p + (1-p)^2 p + \ldots + (1-p)^{Ts-1} p$$

$$\Rightarrow S_{Blind}(T_s) = \sum_{i=1}^{Ts} p(1-p)^{i-1}$$

This equation means that: when the first node receives a request, the probability that the node has the target service is *p*. If it does not have this target service, the message is forwarded to one of its neighbors randomly. Thus the probability that the second node contains the target service is *p*(1-p)*. This process repeats until it reaches the predefined maximum messages that can be sent or forwarded.

(2) SON based solution:

$$S_{SON}(T_s) = p + (1-p)R\sum_{i=1}^{Ts-1} q(1-q)^{i-1} + (1-p)(1-R)R\sum_{i=1}^{Ts-1} q(1-q)^{i-1} + (1-p)(1-R)^2 R\sum_{i=1}^{Ts-2} q(1-q)^{i-1} \ldots + (1-p)(1-R)^{Ts-1} Rq$$

$$\Rightarrow S_{SON}(T_s) = p + (1-p)R\sum_{i=1}^{Ts-1} q(1-q)^{i-1} + (1-p)\sum_{i=1}^{Ts-1} R(1-R)^i \sum_{j=1}^{Ts-i} q(1-q)^{j-1}$$

For the SON based solution, even if the first contacted node does not have the target message, it has *R* probability that this node belong to the target SON. If it belongs to the target SON, the probability that the target service can be discovered in a node within this SON is upgraded to *q*, which is much higher that the probability in the unstructured layer. If this node does not belong to target SON, the request will be forwarded to one of its neighbor, and this neighbor has *R* probability that it belong to the target SON. These processes repeat until the number of the forwarded messages reaches the maximum value.

(3) Triplex layer based solution (SON searching + dependency searching):

$$S_{SON+Dependency} = p + (1-p)R\sum_{i=1}^{Ts-1} q(1-q)^{i-1} + (1-p)(1-R)K\sum_{i=1}^{Ts-1} q(1-q)^{i-1} + (1-p)(1-R)(1-K)K\sum_{i=1}^{Ts-2} q(1-q)^{i-1} +$$
$$(1-p)(I-R)(1-K)^2 K\sum_{i=1}^{Ts-3} q(1-q)^{i-1} + \ldots + (1-p)(1-R)(1-K)^{Ts-2} Kq$$

$$\Rightarrow S_{SON+Dependency} = p + (1-p)R\sum_{i=1}^{Ts-1} q(1-q)^{i-1} + (1-p)(1-R)\sum_{i=1}^{Ts-1} K(1-K)^{i-1} \sum_{j=1}^{Ts-i} q(1-q)^{j-1}$$

The difference between triplex layer based solution and the SON based solution is that, the dependency recommendation increases the probability of finding out the target SON. Except for the first node, the probability that other nodes belong to the target SON is increased to K.

Notations:

- *p*: the expected popularity of the resource, which implies the probability that a user can find the target service in a peer using the blind searching method.
- *q*: the probability that a service can be discovered in the target SON.
- *R*: the probability that current node belongs to the target SON.
- *K*: the probability that a request is sent to the target SON form a node that do not belong to the target SON, in the service dependency analysis. To simplify the illustration, we call it as "Dependency Probability".
- T_s: the maximum number of messages that can be generated for one service discovery request. It includes both the message generated by the node that initiates the request, and the messages generated by other peers who receive this request (as they do not have the target services, thus they generate a message for forwarding this request). It also can be considered as the maximum number of nodes the request can be forwarded to before the termination of a search process.
- $S(Ts)$: the success rate if the maximum number of transferring messages is set to T_s.

The simulation parameters are listed in Table 4-9.

Parameters	Values	Description
p	0.1%, 1%, 5%	Probability that a service can be discovered from a randomly selected node
q	10%, 20%	Probability for discovering a service in the target SON
R	5%, 10%	Probability that current node belongs to the target SON
K	10%,20%,30%,40%,50%, 60%,70%,80%,90%,100%	Probability that a request is forwarded to the target SON with dependency recommendation
T_s	5,10, 20, 30, 40, 50,60,70,80,90,100	Maximum number of the nodes the request is forwarded to

Table 4-9: Simulation parameters of the networks

Blows are some of the simulation results by using above mentioned parameters and equations.

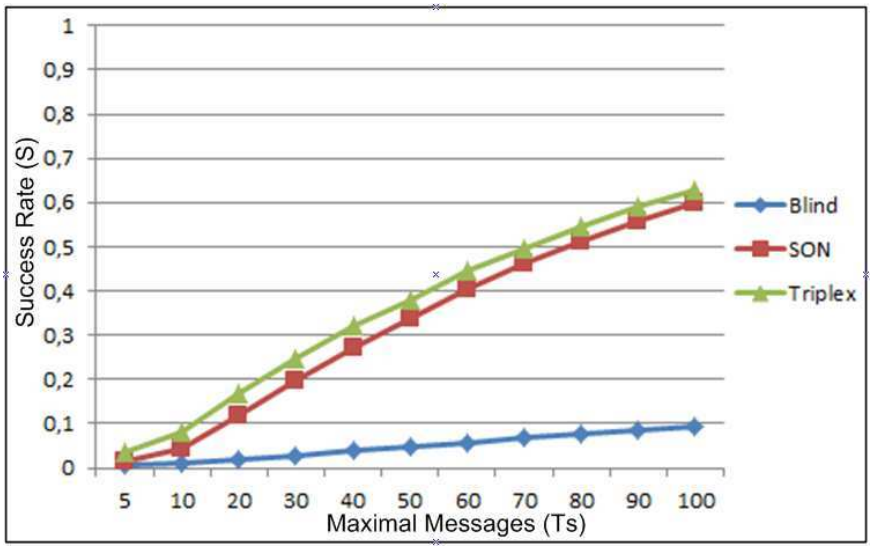


Figure 4-43: Comparison for the success rates among three types of P2P systems (p=0.1%, R=10%, q=10%, k=50%)

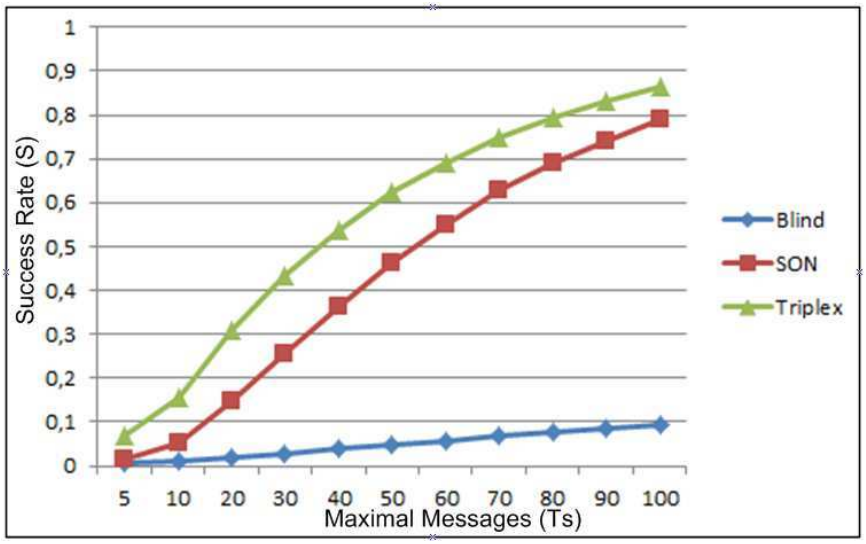


Figure 4-44: Comparison of the success rates among three types of P2P systems (p=0.1%, R=5%, q=20%, k=50%)

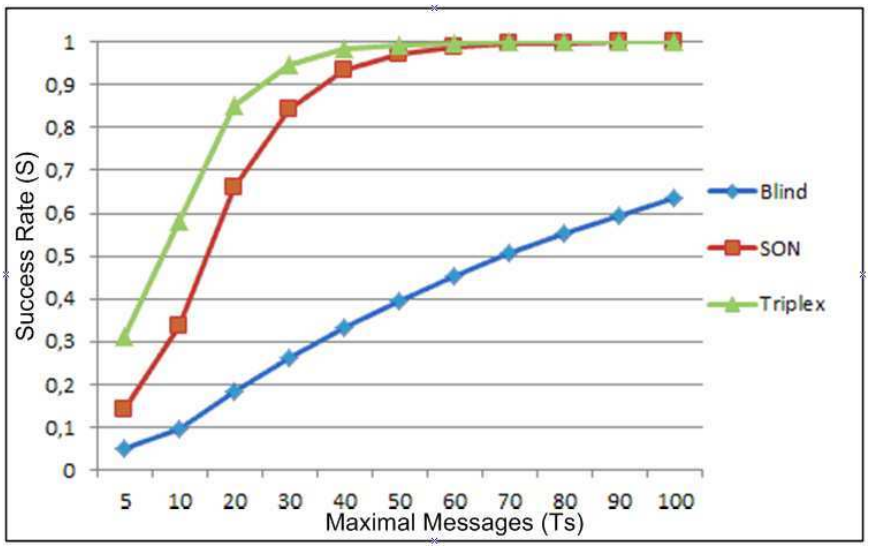


Figure 4-45: Comparison for the success rates among three types of P2P systems (p=1%, R=10%, q=10%, k=50%)

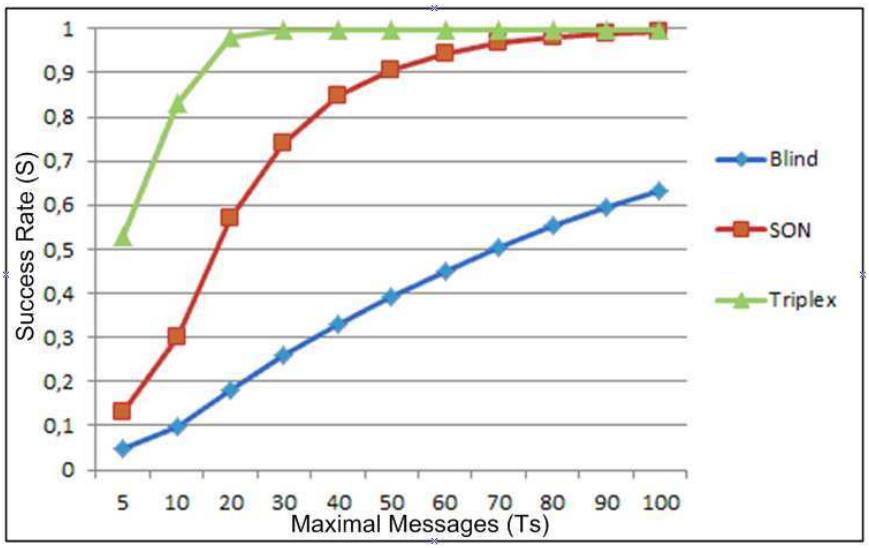


Figure 4-46: Comparison for the success rates among three types of P2P systems (p=1%, R=5%, q=20%, k=50%)

These simulation results show that our proposed solution can greatly improve the success rate with limited messages to be transferred among the nodes. This improvement is more evident when the popularity of services is low in a network.

Figure 4-47 illustrates the influence of the dependency probability *K* for the success rates. The red line is the success rate of ordinary SON based system without dependency recommendation, while the blue one is ones for the SON based system with dependency recommendation.

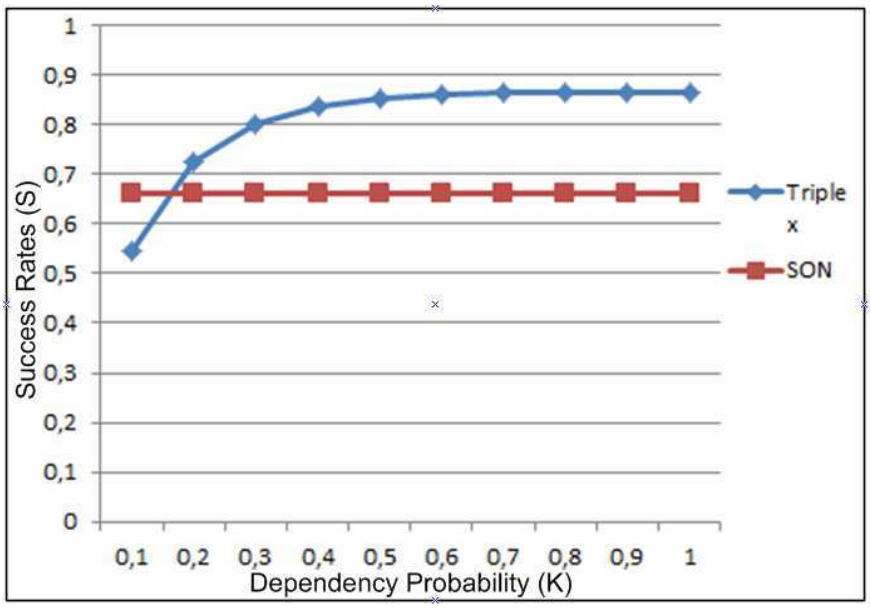


Figure 4-47: Comparison of the success rates (S) by modifying the dependency probability (K)

From the results shown in Figure 4-47, we can observe that as the dependency probability increases, the success rate increases accordingly. However, after K is more than 0.5, this increase rate becomes very slow. Moreover, only when K is bigger than a certain level (in our case, when R=10%, K needs to be bigger than 15.3%), our solution can improve the system's service discovery performance; otherwise it may weaken the system's performance.

We also analyze the performance of this proposed solution using the average messages that are needed for achieving a certain level of success rate. Following equations are used to estimate the average messages.

- Blind searching solution:

$$P(D=j)=\begin{cases} p, j=1 \\ (1-p)^{j-1}p, 1<j\le T_{\max} \end{cases}$$

$$E_{Blind}(S)=\sum_{j=1}^{T_{\max}} j*P(D=j)$$

- SON based solution:

$$P(D=j)\begin{cases} p, j=1; \\ (1-p)R(1-q)^{j-2}q+\sum_{i=1}^{j-1}(1-p)(1-R)^{i}R(1-q)^{j-1-i}q, 1<j\le T_{\max} \end{cases}$$

$$E_{SON}(S) = \sum_{j=1}^{T_{\max}} j * P(D = j)$$

- Triplex layer based solution (SON searching + dependency searching):

$$P(D = j)\begin{cases} p, j = 1; \\ (1-p)R(1-q)^{j-2}q + \sum_{i=1}^{j-1}(1-p)(1-R)K(1-K)^{i-1}(1-q)^{j-1-i}q, 1 < j \le T_{\max} \end{cases}$$

$$E_{SON+Dependency}(S) = \sum_{j=1}^{T_{\max}} j * P(D = j)$$

Where *P(D=j)* is the probability that the query stops at node j, and E(S) represents the average messages needed for obtaining a success rate S.

Figure 4-48 illustrates our simulation results. It shows that our proposed solution can reduce the average messages, meanwhile achieve a high success rate.

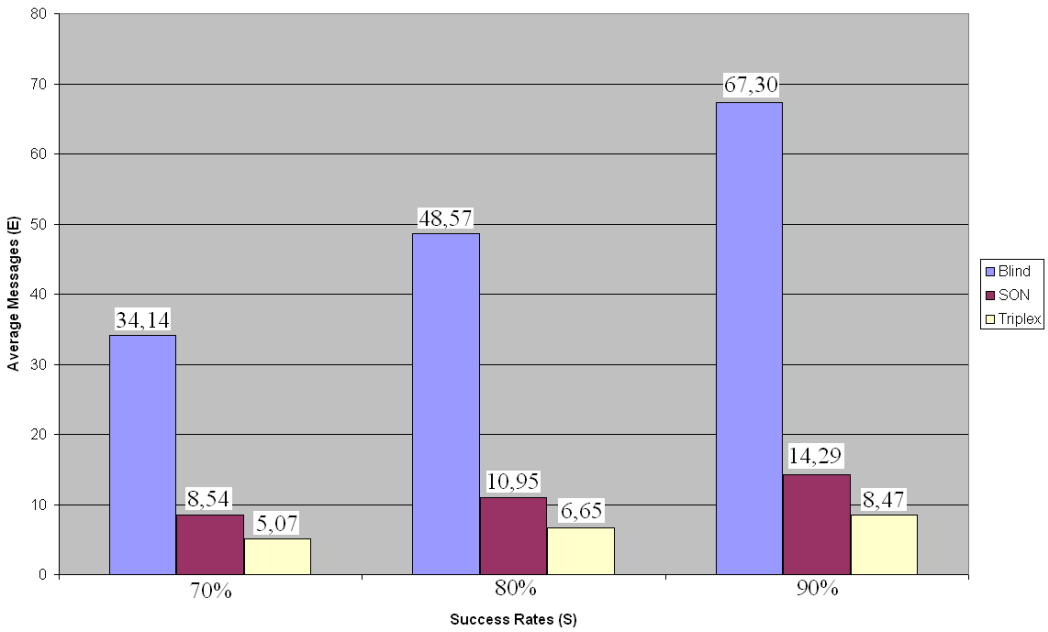


Figure 4-48: Average messages needed for achieving the success rates of 70%, 80% and 90%

The above simulation results show that the adoption of SON layer and dependency layer for routing the service discovery request are able to greatly increase the success rate and to reduce network traffic at the same time. From the simulation results, we also note that the classification for SONs should be appropriate. That is because if too many SONs are in a network, on the one hand, it would reduce the probability of locating the target SON. On the other hand, once the request reaches the target SON, the probability for discovering a service within this SON will be higher than that in the network with few types

of SONs. The reason for this is that more SONs in the network, the searching space inside it will be smaller. Thus we need to find a solution that balances these two parameters. The use of the dependency analysis for the SON selection process is one of the possible solutions. It can keep the high service discovery probability inside a SON, meanwhile it provides the efficient recommendations for the SONs selection. This aims at resolving the problems incurred by the large number of SONs. How to define such dependency rules is challenging in our proposed triplex layer based model. It is also one of our future research topics.

4.5 Summary

In this chapter, we have presented three new strategies for enabling the service information shared in the convergent service environment.

First, we proposed a centralized approach that enhances the user-centric and convergence features, provides a unified access to Telecom/Web/Device/user-generated services, and respects the diversity of the service market. This model thereby enables third-party service providers, developers, and end users to reuse the exiting service using the tools or technologies they prefer following SOA (Service Oriented Architecture), ROA (Resource Oriented Architecture) or WOA (Widget Oriented Architecture) principles. The adaptation of semantic annotation for both service description and request in our approach improve the service discovery accuracy.

Then, overcoming the limitations for the centralized service exposure solutions (e.g. single failure point, resource costing, limitation of scalability, etc.) and considering the fact that the number of the services is increasing drastically and the potential users of a service are diverse, we proposed a hierarchical P2P based service publication and discovery model. This model can facilitate service discovery, improve the discovery efficiency and quality, as well as enhance the system's scalability and interoperability. This new P2P based service publication and discovery model uses a structured P2P overlay as the distributed service repository network, which makes it to be highly scalable. The adoption of Chord protocol as the base ensures the efficiency of query routing in the large-scale P2P networks. The introduction of the abstract service publication and discovery, and the incorporation with the unified semantics and reasoning engines enable the ambiguous service searching, which is indigent in current structured P2P systems. The hierarchical network and resource organization enables the system to quickly identify the target peers, thus greatly improving the service discovery efficiency. This system can act as the base for the converged service information sharing, and enables the rapid discovery of a service among a huge amount of services. The different service exposure platforms and service creation environment thus can be released from the

burdensome and resource costing service storage and discovery tasks. At the same time, they can use the service information for their proper goals, regardless of its headstream.

Finally, as the device offered services are more and more involved in service composition, especially after the popularization of the concept of IoT, we proposed an alternative P2P based service exposure model for targeting the device offered services. This can be considered as a complement for the previous proposed centralized and distributed service exposure model. In our proposed solution, the service exposure is divided into two phases: local service exposure and global service exposure. Both of these two processes mainly rely on the gateways, which are responsible for the monitoring of the local devices and connecting them with the external parties. A triplex overlay has been presented for improving the device offered service discovery in such a large-scale network, and for respecting the intrinsic heterogeneities of devices.

Chapter 5 Conclusion and Future Research Directions

SOA is now acknowledged as a central paradigm for service delivery, acquisition, and consumption in the next generation networking. This includes both NGN in the Telecom domain and NGI in IT domain. Under SOA, services are loosely coupled, so that the interactions among service providers, service consumers and repositories are greatly facilitated. Over the last decade, prodigious research and technology development have been achieved for enhancing this paradigm both in IT and Telecom worlds, resulting in a great amount of supporting technologies, such as the IMS architecture for NGN, the mashup technologies for web service creation. However, there are still some great challenges for network evolutions. From the service point of view, the challenges mainly relates to certain essential functionalities in the SOA paradigm such as service description, discovery, access, and composition management.

Using the IMS/Web/device converged service environment as a basic next generation service environment, in this thesis, we address some open issues such as convergence, automaticity, and user centricity, which are currently hindering the evolution of next generation service. Our contributions mainly include two aspects: (1) the introduction of a user-centric service composition model that enables the automatic service creation and update in the converged environment; (2) the proposition of three different service exposure strategies for catering to the different domains' service exposure and sharing requirements.

5.1 Contributions

5.1.1 Automatic service composition model in IMS/Web converged environments

In order to deal with the service composition management in convergent environments, we first presented the service composition mechanisms in both Telecom and IT domains, as well as the relevant specifications. Although these solutions are able to support the service interoperability among different atomic services, and enhance the service composition capability from certain aspects, the challenges such as convergence, automaticity, and user-centricity, which are considered as some of the essential requirements for next generation service delivery, are still being faced by industry and academia. To

address these challenges, we reused an IMS/Web converged environment as the base for unifying the service composition process for different kinds of services (mainly the Telecom and Web services), and proposed a user-centric service composition model for enabling the automatic service creation and update. This proposed solution improves the service composition paradigm mainly from the following aspects: (i) The introduction of the IMS/Web converged environment and a set of adaptors inside Service Composition Engine enables users to access IMS and Web services seamlessly through different types of terminals, and empowers the service creation and interoperability, regardless of the different types of services (IMS, IN, or Web). (ii) For simplifying the service interoperation management, we defined a set of IMS user profile like service templates called abstract atomic service profiles, concrete atomic service profiles, abstract composite service profiles and concrete composite service profiles for facilitating the service chaining and execution. (iii) We also proposed a mechanism for automatic service selection, and discovery over Service Composition Engine that enables dynamic service composition. The mechanism relies on a Service Comparator component according to user preferences and contextual information. To further improve this automatic service composition feature, three strategies including passive update, active update, and hybrid update were presented and analyzed. (iv) This proposed service composition model can not only be used by professional users, but also by non-professional users. This is achieved by enabling the automatic service selection, composition and monitoring, as well as interoperating with diverse user-centric service creation environments, such as IDEs, Graphical Service Creation Environments, Widget Based Service Creation Environments, and Natural Language Enabled Service Creation Environments. As such, a great freedom to users for selecting their preferred service creation patterns is provided. (v) Finally, we evaluated this service composition model by providing a MSN Robot like Service Composer as a use case for enabling the automatic service creation and update following the introduced service composition model's principles. This proposed use case shows that this service composition model will greatly facilitate the service creation process for users, especially for non-professional users. Both user experience and user's involvement degree will be greatly improved, while the service maintenance cost for service providers is reduced. That is because each modification, either from the service capabilities side or from the service environment, would invoke the adjustment on the corresponding composite service.

5.1.2 Service exposure in converged environment

In this thesis, we also analyzed how services expose themselves to other applications for facilitating the service composition. After presenting an inclusive survey on related work, we perceived that current solutions mainly target a specific group of developers and/or to a specific network, and are not user friendly enough. In order to enable not only professional developers, but also non-professional users to be able to

easily reuse existing services, capabilities, and resources for creating new services, we firstly proposed a centralized service exposure framework for a variety of services, including Telecom/Web/Device/User-generated services. By doing so, it thereby enables a new ecosystem for the creation of innovative services through user-centric fashion. This proposition mainly aims at enhancing the user-centric feature and convergence feature, as well as at providing a unified access to diverse services. The service exposure framework defined in this proposition acts as an intermediary for the adaptation and abstraction of the services provided by Telecom, Internet and Smart Spaced Devices, enabling the convergence of the different kinds of services. To realize these functionalities, four groups of entities have been identified including: user centric adaptation group, semantic enrichment group, service exposure and access control group, and service repositories group. All of the entities inside these four groups collaborate tightly during service lifecycle, so that they enable a seamless interaction among the heterogeneous services and facilitate the reuse of services by both professional and non-professional users.

Considering the challenges generally encountered by the centralized solutions, such as the single access to service exposure platform, which is easy to become a bottleneck when the number of services increases dramatically, the high maintenance cost, and the limitation of scalability and extensibility, we analyzed the feasibility of using a decentralized P2P to support the distributed service exposure on a large scale. A hierarchical P2P architecture was then proposed as a complement for the service exposure model, by providing a distributed service information sharing model in the large-scale network. This proposed solution mainly uses a structured P2P overlay as the distributed service repository network, and adopts the Chord protocol as the base for the service information storing and the query routing. In order to enable the ambiguous searching, the concept of abstract service publication and discovery, which is incorporated with the unified semantic and reasoning engines, was introduced. Furthermore, for improving the service discovery efficiency, an enhanced hierarchical organization strategy for both the resources and nodes has been proposed. The simulation results show that this proposed solution not only achieves the goal of enabling the ambiguous searching, which is considered as necessary for the NGN service delivery, but also improves the discovery efficiency as well as the system scalability and extensibility.

Nowadays, not only the traditional Telecom and Web services are involved in the service composition, but also the small and embedded devices are proclaiming to join this new paradigm. The exposure of the device offered services, as well as the data sharing among these devices, are considered as necessary for realizing the aim of reusing them for service composition. However, some of their intrinsic characteristics require a new approach for the device offered service exposure, which will be different from the solutions used for Telecom and Web services. Consequently, another service exposure model which mainly targets

the device offered service was proposed. In this proposed model, the gateways are used to delegate devices residing in them for service exposure. Moreover, the service exposure process is divided into two main phases: local device exposure, which exposes the local devices to a gateway for the local monitoring, and the global device exposure, which enable the device to be discoverable by external parties. As a great number of gateways are dispersed in the network, we used a P2P based distributed method to enable the interoperation among these gateways. To further improve the service discovery efficiency and accuracy, a triplex overlay based solution was proposed. This triplex overlay architecture includes an underlying unstructured P2P layer, a SON based overlay and a service dependency overlay. The performance analysis shows that our proposed triplex overlay based solution can greatly improve the system performance.

5.2 Perspectives

Nowadays, both in Telecom and Internet worlds, they are evolving rapidly and proclaim to be moving towards a new generation. Regarding their evolution trends, services are considered as a crucial success factor. Service composition, which enables the rapid service creation by reusing the existing services, is considered as the principal service delivery mechanism in the future network environment. Even the tremendous research and technology development have been done over the last decade, there are still some challenges hindering the current solutions from being accepted by industry as a desirable solution, which satisfies the high performance and availability requirements demanded on services when they are deployed in a production environment. We attempted to address some challenges, such as convergence, automaticity, and user-centricity, which have been discussed in this thesis. Besides these, some other challenges, which require further attention and may be our future research directions, are listed below:

- A widely-acceptable common data model is required for enabling the heterogeneous services to be described in a unified manner.
- Using natural languages to create a composite service that will be an ideal approach for the non-professional users. However, this kind of service composition is still far from industrialization due to some unresolved issues, such as natural language ambiguity. Thus it needs a further research work on the Domain Ontology definition and the adoption of Artifact Intelligence for the service creation process.
- Currently, the User Interface (UI) part for the automatic service composition uses the very basic approach, thus the UI of automatically created services are generally very monotonous and unattractive. Adding a UI layer upon the service creation environment will be helpful for improving the user experience for service creation.

- Considering the fact that the investments from operators and service providers for the technology renovation are revenue-driven, an effective business model, which enables operators to profit from providing the service creation facilities and exposing their network assets to 3rd party developers and users, is considered as indispensable. Moreover, if end users also can profit from their contribution to new services, more users would be greatly encouraged to involve in this new service ecosystem. The successful Apple App Store reveals this great potential.
- The use of semantic for the service publication and discovery processes not only enables the better filtering for the discovered services, but also can provide some recommendations for other services which may potentially interest user. Currently, most of such recommendations are semantic tag based. Some other advanced solutions can be investigated or proposed in the future for improving the semantic service discovery process. Moreover, some social network information also can be taken into account by collaborating with the semantic information for improving the recommendation quality.

Chapter 6 Résumé Français

I. Introduction

La dernière décennie témoigne de l'évolution considérable dans les domaines de Télécom et d'Internet. Les avancements de la communication mobile et des technologies concernant des accès à Internet à large bande nous permettent d'être connectés à tout moment, n'importe où par divers moyens (par exemple, l'appel de voix/vidéo, SMS, MMS, messagerie instantanée (IM), e-mail, réseau social, etc.). D'autre part, les attentes des utilisateurs sur les services qu'ils peuvent obtenir de réseaux sont de plus en plus élevées, surtout après l'émergence de NGN (Next Generation Network) et NGI (Next Generation Internet). NGN et NGI sont considérés comme les fondements de la prochaine génération de réseaux et proclament à fournir des soutiens substantiels pour la délivrance de services de prochaine génération. Par conséquent, comment générer efficacement des services compétitifs dans un environnement en évolution rapide, comment réduire à la fois le coût de leur création et leur entretien, comment réduire leur temps de mise en production, ainsi que comment optimiser leur cycle de vie, sont des questions très pertinentes et difficiles pour des opérateurs de réseaux et des fournisseurs de services.

NGN [1] a été introduit pour permettre aux opérateurs de Télécom afin de mieux profiter de leurs compétences et de se préparer à l'environnement des télécommunications émergentes, et donc réaliser les objectifs de réduire le coût de déploiement, optimaliser des investissements pour les infrastructures, faciliter la génération des nouveau services, et par conséquent de garder et/ou renforcer leurs rôles dans le marché de service. L'idée générale derrière NGN est de réutiliser les technologies de l'Internet tels que le Protocole Internet (IP) pour faire face à la concurrence du domaine Internet, il est donc étiqueté comme "tout IP". En outre, NGN fait une séparation claire entre l'accès au réseau et les services, et change l'architecture du réseau de l'approche verticale traditionnelle (i.e., l'accès, le contrôle et les services sont liés étroitement) à une approche horizontale, où des fonctions de contrôle, des fonctions le transport et des fonctions d'application sont séparés en trois couches indépendantes, et chaque couche fournit des éléments réutilisables à d'autres couches [2].

NGI [3], qui est également connu comme Future Génération de l'Internet (FGI) ou l'Internet du futur (FI), est né comme une façon de livraison de nouveaux services sur «Internet 2». Il vise à fournir un ensemble d'améliorations sur Internet, par exemples, l'indépendance de service/technologie, l'évolutivité, la

simplicité, l'adressage, la sécurité, la confidentialité, la mobilité, la fiabilité, la Qualité de Service (QoS)/ Qualité de Expérience (QoE), l'ouverture, et le déploiement rapide de services personnalisés, etc.

En regardant des tendances d'évolution dans les domaines IT et Télécom, on peut remarquer que la distinction entre les réseaux de Télécom et les réseaux d'informatiques est brouillée, surtout du point de vue de service. Convergence devient une caractéristique fondamentale pour des réseaux d'informatique et des réseaux de Télécom. L'environnement de service à venir est généralement considéré comme un environnement de service convergent, dans lequel les différents types de services peuvent interagir les uns des autres à la manière transparente, quels que soient les hétérogénéités sous-jacentes.

Au cours de la dernière décennie, des progrès prodigieux ont été réalisés dans la convergence de divers réseaux câblés, mobiles et sans fil pour fournir des services omniprésents aux utilisateurs. Actuellement, la convergence se déroule entre les mondes de Télécom et d'IT. Un des exemples les plus frappants est l'alliance entre l'IP Multimedia Subsystem (IMS) et le Web 2.0.

En ce qui concerne les caractéristiques et les tendances des évolutions du Web 2.0 et IMS, il est explicite que tous les deux mettent l'accent sur l'amélioration de la livraison de service pour répondre aux exigences de plus en plus complexe. Les deux adoptent le concept de la composition de service comme un mécanisme principal pour permettre la création et l'exécution de services complexes, en intégrant des services basiques. Des travaux de recherche connexes sont effectués depuis des années, et plein de solutions pour la composition de services ont été proposées par différents organismes de normalisations et des sociétés. Néanmoins, certains attributs, tels que la convergence, l'automaticité et l'utilisateur sont toujours centrée sur les questions en suspens, qui sont poursuivis par des industries et des académies. Cette thèse vise à répondre à ces questions pour fournir un modèle de service composition agile qui satisfait les meilleures pratiques dans l'environnement des services convergent.

II. Contexte

Aujourd'hui, le marché de services est axé sur la clientèle, et la concurrence entre des fournisseurs de services est de plus en plus intensive. Pour garder leurs rôles dans le marché de services, les fournisseurs de services et les opérateurs doivent prendre avantage de leurs compétences, trouver les moyens innovants pour fournir des nouveaux services qui satisfont aux exigences des clients, en même temps, réduire le coût opérationnel. Dans ce contexte, la composition de services, qui réutilise les services existants et des ressources de réseaux pour la création d'un nouveau service, est reconnue comme une méthode importante pour la prestation de services de la prochaine génération.

La plupart des solutions de composition de services sont basés sur une architecture orientée services (SOA), qui joue un rôle important pour la prospérité de services Web. Ce concept est compatible avec les réseaux NGN en ce que la couche de service est entièrement séparée de la couche de transport et de la couche de contrôle, et donc il assure l'autonomie de chaque couche et permet leur développement et déploiement indépendant; En outre, cette séparation permet à réutiliser des services existants pour des nouveaux services, quelles que soient les types de services - services d'IMS, services conventionnels ou services Web.

La composition de services est liée à l'interaction positive entre les services, ce qui permet le partage et la réutilisation de services élémentaires par les autres services afin d'offrir des services intégrés pour des utilisateurs. Ces services élémentaires, appelés « Service Capabilities » ou « Enablers » dans certains réseaux NGN, sont les composants modulaires qui peuvent être soit utilisés par la façon indépendante, ou partagées par les services composites afin d'offrir des services enrichis aux utilisateurs.

L'interopérabilité et la coopération entre les composants de service dans la couche de service permet: (1) d'enrichir et de personnaliser les serveurs d'applications basées sur les besoins et les préférences des utilisateurs, et les contextes fournis par les capacités de service; (2) d'éviter le chevauchement et la redondance de la mêmes fonctionnalités offertes par les capacités de services et de mise en œuvre par différents serveurs d'application; (3) de réduire le temps de mise en production pour des serveurs d'applications; (4) de soutenir l'intégration facile et à faible coût pour des serveurs d'applications; (5) de gérer la réutilisation des informations d'utilisateur et de préférence appliquée dans différents serveurs d'application. Pendant le processus de composition de services, toutes sortes de services devrait être indépendant des fonctionnalités de contrôle sous-jacents.

Pour atteindre l'objectif de la composition de services, il a besoin des mécanismes supplémentaires pour la gestion de l'interopérabilité et la coopération des services. Tout d'abord, il faut un mécanisme intelligent pour gérer la composition de services. C'est-à-dire, il faut contrôler l'accès des serveurs d'applications différentes aux capacités de services, et gérer les interactions entre les serveurs d'applications et des capacités de services en respectant la confidentialité de l'utilisateur. En attendant, il devrait être en mesure d'acheminer efficacement le message à la composante la plus appropriée sans intervention humaine. Certaines fonctionnalités non-fonctionnels, tels que la QoS, l'intimité, et la tarification, sont également nécessaires pour être pris en compte, car l'efficacité d'un serveur d'application composite est non seulement dépendante de la combinaison des fonctionnalités, mais aussi dépendante des préférences de l'utilisateur et du contexte. Deuxièmement, la composition de services repose essentiellement sur la condition sine qua non que les capacités de services qui sont réutilisés dans la

composition de services ont été rendus disponibles pour l'application composite à l'avance. Cette condition est liée au paradigme de l'exposition de service, dont le but est de rendre les services qui sortent autonomes pour être compréhensible et interopérable dans le réseau à grande échelle, sans la prise en compte des hétérogénéités sous-jacentes. Cette condition peut être réalisée en réutilisant certaines interfaces (Open API). Cela permettra d'accélérer l'introduction de nouveaux services par les prestataires de services, en donnant plus de contrôle sur le processus d'introduction du service, et de réutiliser des composants applicatifs existants.

III. Composition de services dans un environnement NGN

1. Problématiques

Convergence: En ce qui concerne des évolutions d'informatique et de Télécom, la convergence est largement reconnue comme une tendance inévitable. Un des exemples les plus frappants est l'alliance entre le Web 2.0 et IMS. Web 2.0 apporte un impact significatif sur le provisionnement de services Internet en encourageant la contribution des utilisateurs pour la création des services innovants et/ou des contenus. IMS est largement reconnu par le monde Télécom comme l'architecture de référence pour les réseaux NGN en appliquant le contrôle de service sur les infrastructures basées à IP, permettant des accès indépendants, et garantissant une expérience de service sécurisé et omniprésent pour l'utilisateur. Apparemment, une telle convergence assurément apportera une perspective nouvelle pour la livraison de service de nouvelle génération.

Pendant ce temps, à la fois dans le Web 2.0 et l'environnement IMS, la composition de services est considérée comme un mécanisme essentiel de prestation de services qui permet la création et l'exécution de services complexes, en intégrant les composantes autonomes de services. Cependant, la plupart des mécanismes actuels de la composition de services au sein des mondes IT et Télécom évoluent plutôt indépendamment les uns des autres sur la base des différentes infrastructures sous-jacentes. Par exemple, dans Télécom, IMS a opté pour être IP basé, en particulier sur la base de SIP pour contrôler des communications de voix, données et multimédia en temps réel. Cette considération promet de fournir facilement la convergence entre Télécom et Internet. Le mécanisme basé sur S-CSCF et iFC pour le chaînage de services a été standardisé pour la livraison de service IMS basé. SCIM a été proposé par 3GPP et OSE a été spécifié par OMA pour fournir le courtage des services et la gestion d'interopérabilité pour des capacités de Télécom et des services. Toutefois, ce genre de l'interopérabilité des services est généralement Télécom protocoles axé.

Par conséquent, dans notre recherche, nous essayons de fournir un modèle de la composition de services unifié pour permettre des utilisateurs d'accéder aux services d'IMS et de Web de façon transparente à travers différents types de terminaux, ainsi que permettre la création de services et l'interopérabilité indépendamment des différents types de services (services IMS, services Web, ou des services offert par appareils).

Centrée-sur-utilisateur: Une autre tendance importante pour la livraison de services de prochaine génération est la centrée sur l'utilisateur, dans lequel l'utilisateur est non seulement considéré comme un consommateur, mais est également encouragé à être un contributeur de services ou de contenus. Beaucoup de travail a été fait dans ce but, et de nombreuses solutions et plates-formes ont été proposées par différentes entreprises ou des communautés, surtout après l'émergence de Mashup dans le Web 2.0. Néanmoins, la plupart des plates-formes proposées ou des systèmes sont statiques ou semi-automatique. Par exemple, même certaines solutions qui sont considérées comme centrées sur l'utilisateur, par exemple, YahooPipe, MashMaker, et MARGMASH, elles ont encore besoin de l'intervention humaine pour compléter un processus de création de service. Cette invention humaine est toujours pas assez sympa, en tenant compte de la réalité que: la plupart des utilisateurs ordinaires n'ont pas de connaissances sur la technologie d'Internet ou de Telecom, qui entrave la vulgarisation largement de la notion de "création des services par des utilisateurs".

Afin d'encourager des utilisateurs multi-niveaux à contribuer à la création de services, il est nécessaire de masquer la complexité sous-jacente et de simplifier l'interface et le processus de création de services. Donc notre modèle proposé pour la composition de services doit être appliquées avec un mécanisme sophistiqué pour la restauration aux besoins de centrée sur l'utilisateur, et donc même un utilisateur qui ne dispose d'aucune compétence de développement, peuvent facilement créer son service personnel.

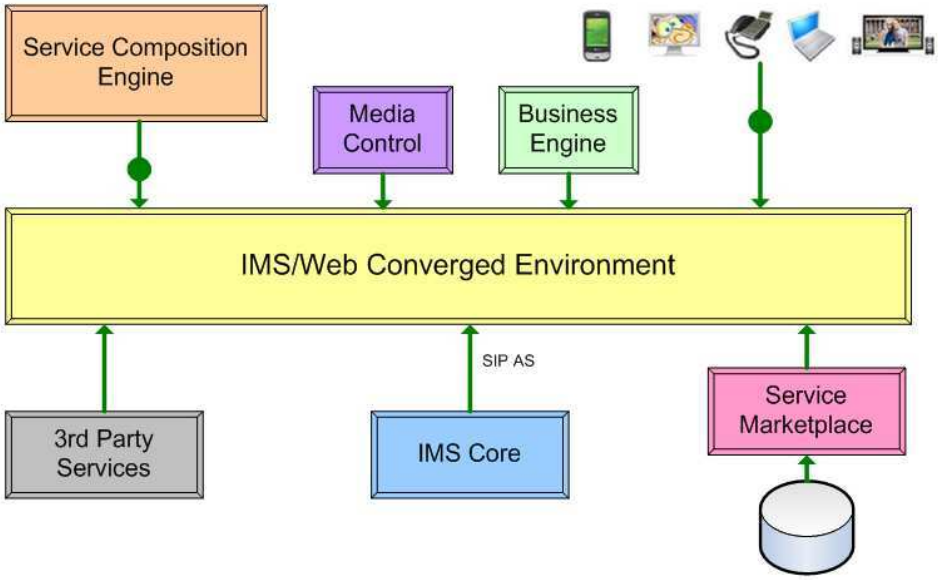
Automaticité: la composition automatique de services est extrêmement importante pour la réalisation de la fonction centrée sur l'utilisateur. Parce que la composition de services est une tâche très compliquée, et il est au-delà de la capacité des utilisateurs ordinaires à gérer l'ensemble du processus manuellement. Une telle complexité est due à des faits que le nombre de services augmente rapidement, qui le rend difficile de trouver celle qui convient à partir d'un grand nombre de services. L'environnement de service est très dynamique, qui nécessite des services peuvent être créés et mis à jour lors de l'exécution. Ainsi, le système de composition de services doit détecter la mise à jour et prendre la décision en temps réel, ce qui est apparemment impossible ou pas efficace pour être manipulé manuellement par les utilisateurs ou les développeurs.

Afin de réduire la manipulation manuelle, d'améliorer l'efficacité de la production et l'exécution d'un service composite, nous proposons un mécanisme alternatif pour traiter de la question de la composition automatique. Notre proposition est principalement liée aux processus de la création de services et de la mise à jour.

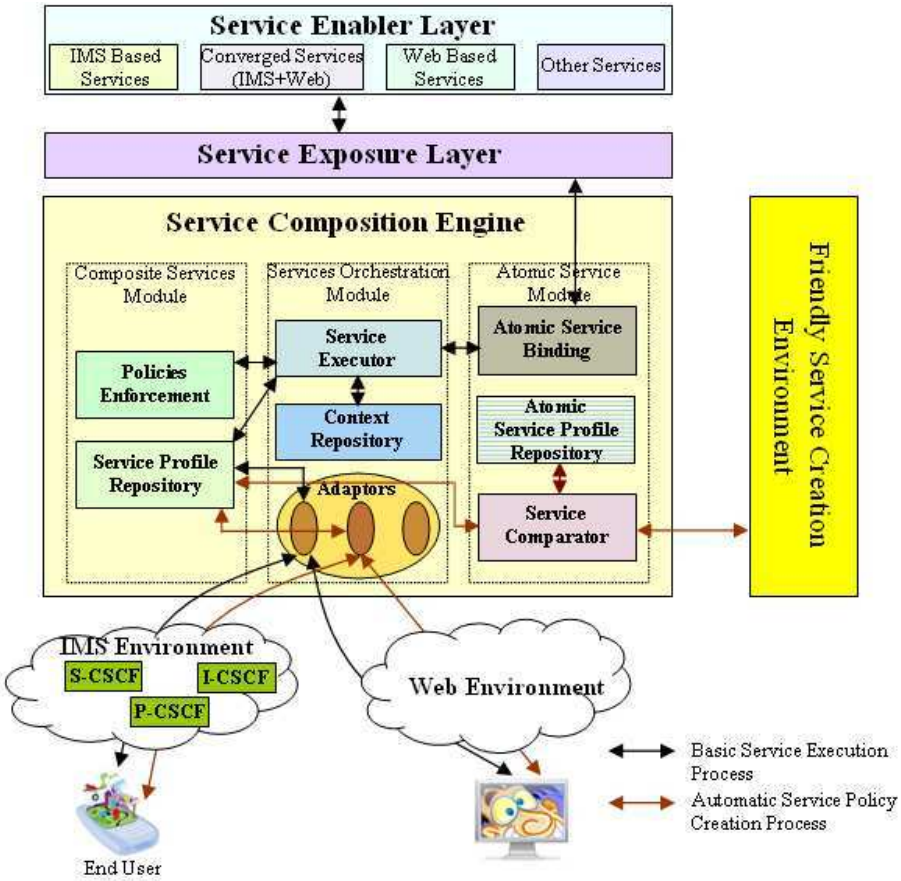
2. Contributions

Alors que la convergence, la centrée sur l'utilisateur et l'automaticité sont les challenges les plus importants qui entravent la grande popularité de la composition des services dans le marché de services. Ces challenges nous motivent à proposer un modèle unifié de la composition de services, ce qui permet non seulement à l'utilisateur professionnel, mais également à l'utilisateur non professionnel de créer leurs propres services facilement, quelles que soient les hétérogénéités sous-jacentes entre les différents types de services.

La figure suivante illustre le modèle fonctionnel de notre solution de composition de services dans l'environnement IMS/Web convergent.



Du point de vue fonctionnel, le rôle principal de SCE est de permettre la réutilisation des capacités existantes, ce qui a trait aux actions comme la création de services, l'exposition de services, la découverte de services, et l'orchestration de services. La figure suivante est un modèle simplifié de la SCE, qui est liée à l'environnement IMS/Web convergent, la couche d'exposition de services, et de l'environnement convivial de création de service.



Ce modèle de composition de services est facile d'être étendu pour satisfaire des exigences différentes, sans modification sous-jacente. Composition automatique de services est un exemple qui est réalisé par l'extension des profils de service utilisés dans notre modèle proposé précédemment.

Tout d'abord, afin de réaliser la composition automatique de services, nous réutilisons les concepts de « service abstrait » et de « service concret », qui ont été initialement proposé par Nassim Laga dans le projet SERVERY pour la composition de gadgets. Nous avons ensuite divisé le processus de composition de services en deux processus: la composition abstraite et la composition concrète. Composition abstraite consiste à définir une combinaison des tâches de fonctionnalités et leur dépendance de données. Composition concrète consiste à déterminer les services les plus appropriées à partir d'un ensemble de fonctionnellement équivalentes ceux pour chaque tâche fonctionnalité. Les deux procédés conduisent à la génération d'un profil de service: un profil de service abstrait ou un profil de service concret. Dans cette solution, quand un service composite est créé au moment de la conception, une logique qui est couplage lâche pour l'invocation du service est générée et représentée par un profil de service composite abstrait. Le profil de service composite abstrait définit la combinaison de services fonctionnalités, chaque nœud est lié à un service abstrait qui peut être mappé à un ou plusieurs capacités existantes concrètes, qui sont

fonctionnellement équivalents. Afin de faciliter la découverte service concret, chaque description de nœud fonctionnel est liée à une ID unifiée de fonction abstraite.

Selon le profil de service composite abstrait et les profils de services atomiques concrets, Comparateur de Service sélectionne automatiquement les services atomiques concrets pour ce service composite, et crée une instance de service concret qui est représenté comme un profil de service composite concret, fondé sur des règles prédéfinies et des contraintes.

Sur la base de ces améliorations pour ce modèle, la composition automatique de services peut être encore améliorée. En règle générale, le processus de composition de services peut être divisé en trois phases: la création de service, la mise à jour de service et l'exécution de service. Et la fonction de l'automaticité est surtout pertinente pour les phases de la création de service et la mise à jour de service.

Quand un service est créé par un utilisateur professionnel ou non professionnel, l'utilisateur a seulement besoin de définir les fonctions à être engagés dans ce service créé. Elle/il n'a pas besoin de connaître ou de définir précisément quels services existants sont invoquées, et comment un tel service peut être accédé et configuré. Cela va grandement faciliter le processus de création de services et encourager les utilisateurs non-professionnels à créer plus services personnalisés, car les utilisateurs ne sont pas tenus d'avoir des compétences de développement d'applications, et n'ont pas besoin de passer beaucoup de temps pour apprendre à connaître les capacités, les services et l'environnement réseau, qui sont toujours très difficile, parfois impossible pour les utilisateurs non professionnels à comprendre.

C’est normal que la plupart des services sont mis à jour régulièrement, ou il y a des variations dans l'environnement dynamique de services, tels que des nouveaux services deviennent disponibles, des services existants deviennent obsolètes, ou des règles métier changement. Une fois une modification se produit, il est nécessaire d'effectuer quelques ajustements correspondants, afin de faire face à ces situations exceptionnelles avec une intervention minimale de l'utilisateur. Ça dérive de la notion de mise à jour automatique de service, ce qui signifie essentiellement: lorsque les capacités ou l'environnement d'exécution de service ont été modifiées, la logique de service composite et la configuration correspondante sont automatiquement ajustées. Du point de vue de SCE, ce processus de mise à jour automatique peut être «passive», «active», ou même «hybride».

La mise à jour «passif» signifie que SCE est toujours à l'état d'écoute. Toutes les mises à jour pour la logique de service composite sont déclenchées soit par les messages publiées par les services, ou par les résultats d'exécution de services. Dans ce cas, chaque service composite est relié à un couple de profils: un profil de service composite abstrait et un profil de service composite concret.

La mise à jour «Actif» signifie que SCE vérifie le dépositaire de profil de service atomique régulièrement pour s'assurer de toutes les capacités sélectionnés pour le service composite sont les meilleures à un moment donné. Dans ce cas, le dépositaire de profil de service composite ne stocke que les profils de services composites abstraits, et ces profils sont édictés par Exécuteur de Service. Dans ce cas, la responsabilité de l'Exécuteur de Service est de traiter les messages de notification d'état d'achèvement lié à un service composite abstrait, et ensuite décider la prochaine fonction abstraite qui a besoin d'être activé.

Dans la solution hybride, le dépositaire de profil de service ne stocke que le profil de service composite abstrait comme la solution active fait, et la découverte de services est exercée, ni au moment du design, comme la solution passive, ni à l'exécution, comme la solution active, mais seulement après que l'utilisateur entre le système et avant l'exécution du service. C'est à dire, une fois l'utilisateur se connecte, Comparateur de Services récupère un profil de service composite abstraite, et contact avec le dépositaire de profil de service atomique à définir un profil de service composite concret pour ce service composite, selon les informations de connexion de l'utilisateur, les informations et exigence d'exécution de capacités. Ceci est similaire à la création de service composite profil concret au moment du design pour une solution passive.

3. Preuve de concept

Afin de prouver notre modèle de composition de services, un MSN robot est mise en œuvre comme un compositeur de service qui est sur la base du modèle de composition de service proposé comme un cas d'utilisation. Une plate-forme simplifiée de composition de services est instancié comme un MSN robot, et il est représenté comme un contact dans la liste de contacts, ce qui permet de réutiliser mécanisme d'authentification de MSN et les informations de contact existant. Il facilite également le processus de création de service pour les utilisateurs sans aucune installation ou configuration supplémentaire. Ainsi, les utilisateurs doivent simplement d'ajouter l'e-mail d'un tel "contact" dans leur liste de contacts, et alors ils peuvent parler avec ce "contact" en entrant leurs demandes, par exemple, "Envoyer mes fentes disponible dans la semaine prochaine à John", pour créer des services composites. Dans cet exemple, cet utilisateur crée un service composite qui comprend le calendrier de réseau, le carnet d'adresse de réseau, et la messagerie. Puis ce service composite est créé et peut être exécuté à temps réel.

Les avantages de ce modèle de composition de services sont explicites: il facilite le processus de création de service pour les utilisateurs, en particulier pour les utilisateurs non professionnels, en permettant la sélection automatique de service, la composition automatique de service et la manipulation automatique de service. Donc, à la fois l'expérience d'utilisateur et le niveau d'implication d'utilisateur dans le processus de création de service seront grandement améliorés. Il réduit le coût d'entretien de service pour les fournisseurs de services en raison du mécanisme de mise à jour automatique, parce que chaque modification, que ce soit du côté des capacités ou de l'environnement de service, invoquera l'ajustement sur le service composite correspondant automatiquement. En outre, il améliore la coopération entre les différentes parties (par exemple, les utilisateurs, les fournisseurs de services, et les opérateurs, etc.) Enfin, il optimise le cycle de vie de service et permet la prestation de services innovants à un rythme plus rapide.

IV. Exposition de services dans un environnement convergent

L'exposition de service joue un rôle important dans l'évolution de la composition de services, parce que permettant un service d'être réutilisable cela signifie que ce service doit être connu et accessible par les utilisateurs et/ou développeurs. Diverses solutions d'exposition de service ont été proposées. Toutefois, les solutions actuelles sont principalement ciblées un groupe spécifique de développeurs, et ils ne sont pas suffisamment convivial pour les utilisateurs. Compte tenu de ces réalités, nous proposons trois modèles

d'exposition de service pour compléter le modèle précédent pour la composition de services, et permettre à l'utilisateur de créer des applications dans un marché plus ouvert et plus durable. Ces plates-formes permettent aux développeurs professionnels et utilisateurs non professionnels de réutiliser facilement les services existants pour créer de nouveaux services, et profiter ainsi d'une expérience très personnalisé et d'une communication significative.

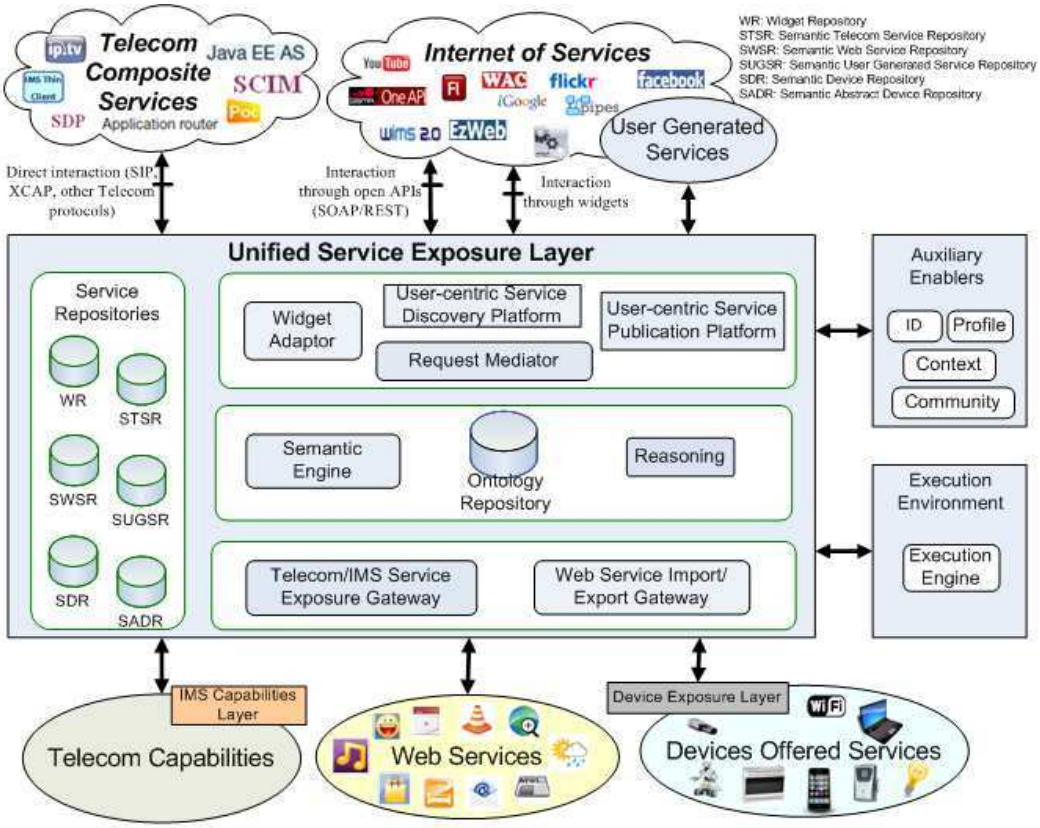
1. Un modèle sémantique enrichi pour l'exposition de service dans un environnement convergent

Dans ces années, un grand nombre de mécanismes d'exposition de services ont été proposés pour satisfaire aux exigences de réseaux différents. Cependant, la plupart des mécanismes d'exposition de services au sein les domaines d'IT, Telecom et appareil évoluent plutôt indépendamment. Ils sont principalement conçus pour un groupe spécifique de développeurs et/ou réseau spécifique. Un tel mécanisme pour la découverte de services et la publication de services dans différent domaines et par un environnement unifié toujours manque. Ainsi, quand un développeur ou un utilisateur souhaite créer un service convergent qui implique des services de Telecom, Web ou appareil, il/elle doit rechercher des services parmi plusieurs plates-formes de services différents, comprendre les différents modèles de description de service, ainsi que leur sous-jacente hétérogénéité. Le besoin de manipuler différentes plates-formes accroît la complexité de l'intégration des services hétérogènes dans un service composite. En outre, la plupart des solutions actuelles visent aux développeurs professionnels, et même pour des plates-formes de création de services qui sont considérée comme centrée sur l'utilisateur. En s'appuyant sur ces solutions de services d'exposition, ces plates-formes de création de services sont encore trop compliqué pour les utilisateurs ordinaires. Afin d'encourager la participation des utilisateurs au lieu de l'exclusivité et vraiment inclure des utilisateurs non professionnels, des caractéristiques de centrée sur l'utilisateur doivent être considérablement améliorées.

Un de nos objectifs est de proposer une plate-forme sémantique améliorée et convergent d'exposition de service pour une variété de services, y compris des services d'Telecom/Web/appareil et des services générés par les utilisateurs, permettant ainsi un nouvel écosystème pour la création de services innovants via la mode de centrée sur l'utilisateur. Aujourd'hui, la plupart des solutions actuelles (à la fois dans les mondes de Télécom et de Web) mettent l'accent sur l'utilisation du système centralisé pour stocker les informations de service et la gestion de l'accès au service. L'un des avantages évidents pour ces solutions centralisées, c'est qu'ils sont faciles à mettre en œuvre et faire appliquer la manipulation des ressources détenues sur eux. Par conséquent, nous avons d'abord proposé une plate-forme centralisée de service d'exposition. Cette plate-forme est destinée à être celui dans lequel même les utilisateurs non professionnels

peuvent facilement réutiliser les services existants pour la création de nouveaux services sans tenir compte des hétérogénéités et des complexités sous-jacentes.

Le modèle de référence est indiqué dans la figure suivante. Cette plate-forme de l'exposition de service agit comme un intermédiaire pour l'adaptation et l'abstraction des services fournis par Telecom, Internet et les appareils, et permet la convergence des différents types de services.



Au cours de cycle de vie du service, toutes ces entités collaborent étroitement afin de permettre une interaction sans faille entre les services hétérogènes, et de faciliter des réutilizations par les utilisateurs professionnels et non professionnels.

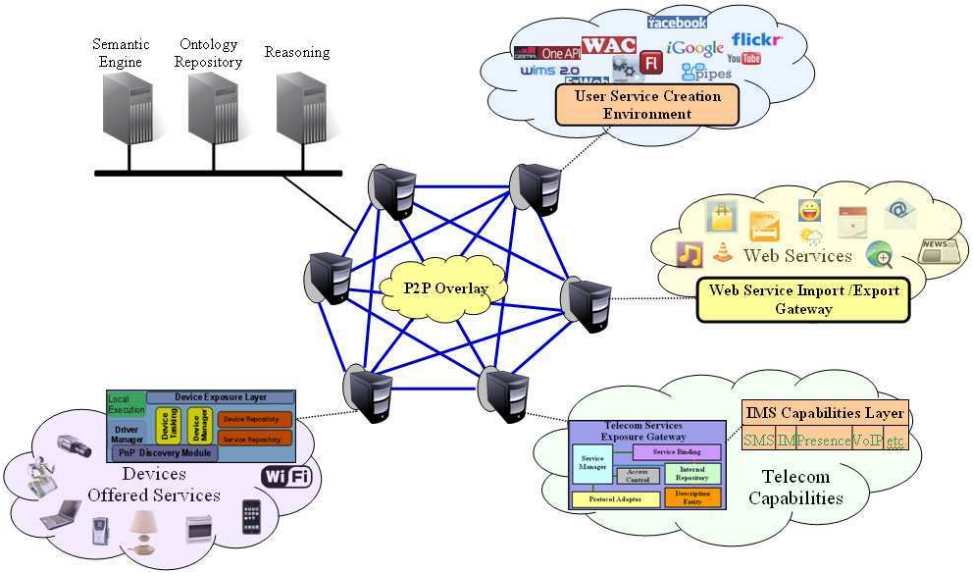
Le modèle de l'exposition de service repose sur l'amélioration des caractéristiques de le centrée sur l'utilisateur et la convergence, ainsi que sur la fourniture d'un accès unifié aux services de Telecom/Web/appareil et des services générés par les utilisateurs, en même temps, il respect la diversité du marché des services. Il permet ainsi les fournisseurs de services tiers, les développeurs et les utilisateurs de réutiliser les services existants, en utilisant les outils ou de technologies, ils préfèrent en suivant des principes de SOA, ROA (Resource Oriented Architecture) ou WOA (Widget Oriented Architecture). En séparant la découverte de services et de fonctionnalités de sélection de SCE, ce modèle d'exposition de

service non seulement libère SCE de la tâche de découverte de service, mais permet également la réutilisation des services à plus grande échelle, indépendamment de leurs hétérogénéités sous-jacents, ce qui améliore encore le partage de différents types de services entre les différentes plates-formes. Il fournit également la possibilité d'appliquer des contrôles renforcés pour l'exposition de services en conformité avec des exigences des opérateurs différents ou des fournisseurs de services via des passerelles. L'aspect centré sur l'utilisateur est renforcée par l'utilisation de gadget et à base de flash clients pour la publication de services, la découverte et la composition, ce qui facilite grandement le processus de génération de services qui encourage plus d'utilisateurs à contribuer à l'écosystème de services.

2. Un modèle distribué basé sur P2P pour l'exposition de service

Nous proposons également une architecture décentralisée basée sur P2P pour l'exposition de services distribués sur une grande échelle comme une solution alternative. Cette architecture proposée est hiérarchique par rapport à courant pures solutions de P2P dans le but de ne pas être limitée à un domaine de service spécifique unique, et permet l'échange efficace et l'interopérabilité entre les différents types de services, indépendamment de leurs hétérogénéités sous-jacentes. Cette nouvelle méthode pourrait conduire le paradigme de composition des services à une nouvelle ère.

Dans notre solution proposée, non seulement les services traditionnelle de Telecom et les services de Web sont exposés, mais des services offerts par appareils, et des services générés par les utilisateurs sont également logés. Lorsque différents types de services de s'exposer au réseau, ils utilisent généralement des technologies différentes ou passent différentes passerelles. Afin de réutiliser ces technologies existantes d'exposition de services et de limiter les modifications aux plates-formes existantes de la découverte et de la publication, une architecture hiérarchique P2P qui utilise une superposition de P2P pour le partage des informations de service diversifiée tout en respectant et conservant les hétérogénéités sous-jacentes est proposé comme le montre la figure ci-dessous.



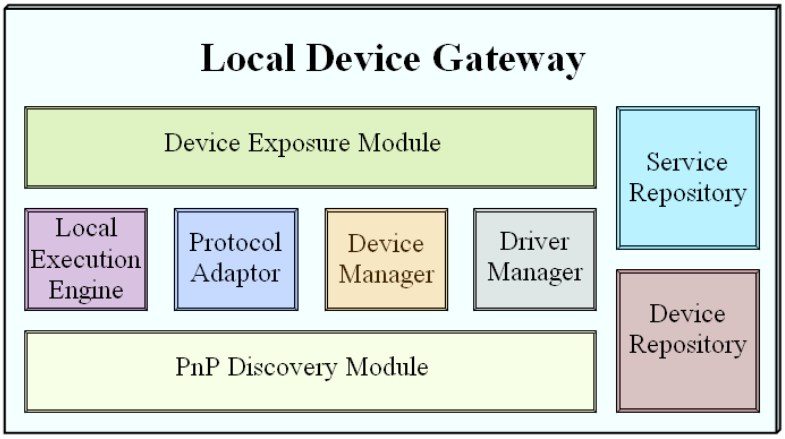
L'utilisation de la superposition structuré P2P comme le réseau référentiel de services distribués, cette système de publication de découverte est très évolutive. L'adoption du protocole Chord comme une base pour la solution proposée garantit l'efficacité de routage pour des requêtes aux réseaux P2P à grande échelle, tandis que l'introduction de la publication de service abstrait, la découverte de service abstrait, l'incorporation de la sémantique unifiée, et les moteurs de raisonnement permettent la recherche de service ambiguë qui est indigent dans les systèmes P2P structurés actuels. Le réseau hiérarchique et l'organisation des ressources permettent au système d'identifier rapidement les pairs qui plus susceptibles de contenir des services répondant aux besoins des utilisateurs et donc d'améliorer grandement l'efficacité de découverte de service. Cette solution proposée a été initialement conçu pour la découverte de service qui porte sur des services de télécommunications/Web/appareil et des services généré par l'utilisateur. Cependant, il est aussi applicable pour la découverte de ressources générales.

3. Un modèle basé sur P2P pour l'exposition de service dans l'environnement d'IoT

Dans le paradigme de l'IoT, il y a aussi un grand nombre de appareils légers qui ne peuvent accéder au réseau IP directement et résident généralement dans un réseau local. Pour ce type de dispositifs, des passerelles sont généralement nécessaires pour les surveiller et de les déléguer pour le monde extérieur. L'exposition pour eux sera complètement différente des services traditionnelle de Telecom et de Web. Par conséquent, la solution que nous présentons dans cette section se concentre principalement sur les appareils qui ont besoin de passerelles pour les relier avec des parties externes, de les exposer dans un réseau mondial, et en outre permettre le partage d'informations entre les différents utilisateurs du service, les

plates-formes de services différents. En outre, pour éviter le point de défaillance unique pour les centralisées solutions et l'apparition éventuelle de goulots d'étranglement alors que le nombre de appareils augmente, un mécanisme basé sur le P2P est proposé pour améliorer l'efficacité de la découverte de service.

La figure suivante illustre une architecture de référence pour les passerelles. D'un point de vue fonctionnel, le rôle principal d'une passerelle local est de permettre l'interaction entre les appareils résidant dans un environnement local, et l'interopérabilité entre ces appareils locaux et les services à grande échelle provenant de domaines Web et Télécom. Cela est généralement lié à des actions telles que la détection des appareils, l'exposition des appareils et l'exécution des appareils.

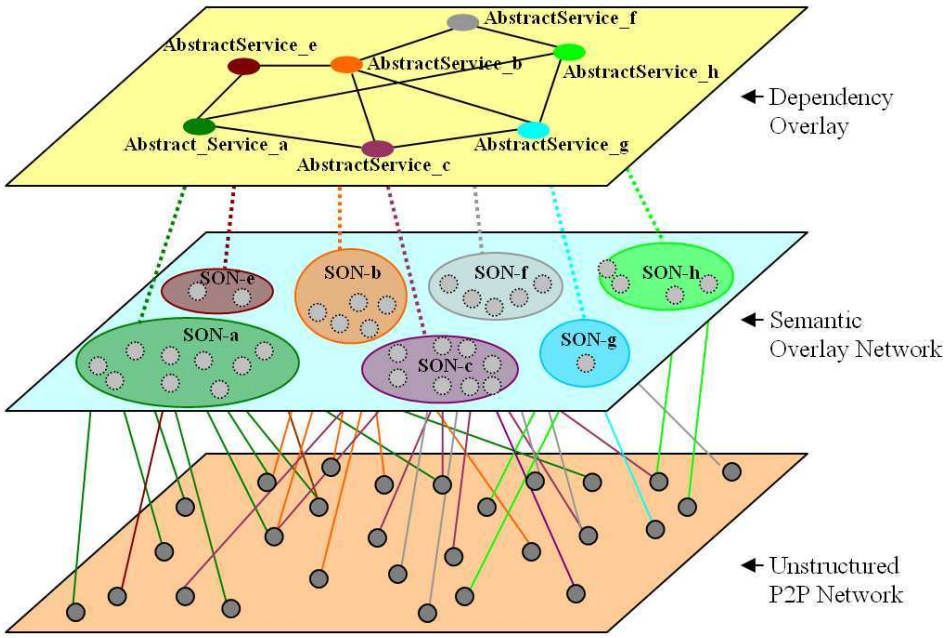


Dans notre solution proposée, des fichiers de description pour des services offerts par des appareils sont publiés dans leurs passerelles, et ces passerelles distribuées partagent leurs informations de service à travers le modèle P2P pour éviter le point de défaillance unique et améliorer de l'efficacité de découverte de service.

Afin d'améliorer les performances des requêtes et de maintenir un haut degré d'autonomie pour des noeuds, un ensemble de SON (Semantic Overlay Network), sont créés au-dessus des passerelles distribués. Essentiellement, chaque passerelle se connecte au réseau mondial et agit comme un pair dans le système P2P. Ils partagent les informations de service stockées dans leur répertoire de service local. Comme ces fichiers de description des services sont sémantique enrichi, les pairs qui détiennent des fichiers de description sémantiques peuvent être considérés comme sémantique enrichi ainsi, comme un résultat. Puis pairs avec sémantiquement similaires fichiers de description de service sont «regroupés» ensemble. Par conséquent, quand un utilisateur veut découvrir un appareil ou un service fourni par le appareil, le système sélectionne le SON(s), qui est (sont) mieux adapté pour répondre. Ensuite, la requête est envoyée à l'un des nœuds dans le SON sélectionné (s) et en outre transmis aux autres membres de ce SON. Cela permettra

d'améliorer grandement les performances des requêtes, parce que les requêtes ne sont acheminées que dans les SONs appropriés, ce qui augmenterait la chance de trouver les fichiers correspondants rapidement avec un coût limité.

Pour réaliser la découverte de service sur la base de SON mentionné précédemment, nous proposons une superposition de triplex sur la base du système P2P, comme indiqué dans la figure suivante.



Comme cette figure montre, les passerelles, des répertoires de réseau, des plates-formes de publication et de découvert, et des environnements de création de services rejoignent le réseau distribué basé sur P2P. Ils agissent comme des nœuds dans la couche de réseau P2P non structuré en utilisant des solutions basées sur les inondations à interagir les uns des autres. Puis au-dessus du système P2P sous-jacent, les nœuds qui fournissent des ressources similaires sont regroupés qui sont appelés SON. Pour améliorer encore l'efficacité du découverte de service, d'une superposition de dépendance supplémentaire est créée sur la couche SON.

Les résultats des simulations montrent que l'adoption de la couche de SON et la couche de dépendance pour acheminer la demande de découverte de service augmentent considérablement le taux de réussite et réduisent le trafic de réseau en même temps. A partir des résultats de la simulation, nous notons aussi que la classification pour SONs doit être discrète. Parce que s’il y a trop de SONs dans un réseau, d'une part, il permettrait de réduire la probabilité de localisation de la cible SON. D'autre part, une fois l’requête atteint la cible SON, la probabilité de découvrir un service au sein de ce SON sera plus élevé que dans le réseau avec peu SONs. La raison est que plus de SONs dans le réseau, l'espace intérieur de la recherche sera plus

petit. Donc nous avons besoin de trouver une solution qui concilie ces deux paramètres. L'utilisation de l'analyse de dépendance pour le processus de sélection de SON est une des solutions possibles. Il peut garder la probabilité de découverte de service élevé dans un SON, en attendant, il fournit des recommandations efficaces pour la sélection de SON. Celle-ci vise à résoudre les problèmes occasionnés par le grand nombre de SON.

V. Conclusion

SOA est aujourd'hui reconnu comme un paradigme central pour la prestation de services, l'acquisition et la consommation dans les réseaux de prochaine génération. En vertu de la SOA, les services sont faiblement couplés, donc les interactions entre les fournisseurs de services, les consommateurs de services et les répertoires sont grandement facilitées. Au cours de la dernière décennie, beaucoup de recherche et de développement technologique ont été faites pour améliorer ce paradigme dans les mondes d'informatique et de Télécom, ce qui entraîne une grande quantité de technologies de soutien, tels que l'architecture IMS pour réseaux NGN, les technologies de mashup pour la création de services Web. Cependant, il y a encore quelques grands challenges pour les évolutions du réseau. Du point de vue du service, les challenges se rapportent principalement à certaines fonctionnalités essentielles dans le paradigme SOA tels que la description de service, la découverte de service, l'accès de service et la gestion de composition.

Utilisant de l'IMS/Web/appareil convergent environnement comme un environnement de base pour des services de prochaine génération, dans cette thèse, nous abordons certaines questions ouvertes telles que la convergence, l'automaticité, et centrée sur l'utilisateur, qui sont actuellement des obstacles à l'évolution du service de prochaine génération. Nos contributions comprennent principalement deux aspects: (1) l'introduction d'un modèle centré sur l'utilisateur pour la composition des services qui permet la création automatique de service et la mise à jour automatique dans le environnement convergent; (2) la proposition de trois différentes stratégies d'exposition des services pour satisfaire des exigences de différents domaines.

Après une enquête approfondie pour la composition des services à la fois dans les domaines IT et Télécom, nous proposons un modèle de composition automatique des services dans un environnement IMS/Web convergent. Ce modèle proposé est destinée à être celui dans lequel même les utilisateurs non-professionnels peuvent réutiliser des services et des ressources existants pour créer des nouveaux services facilement. Pour améliorer encore des fonctionnalités de la composition automatique des services, trois stratégies, y compris mise à jour passive, mise à jour active et mise à jour hybride sont proposées et analysées. Ces améliorations permettent aux utilisateurs de profiter d'une expérience de communication et interaction très personnalisé et significative.

Considérant le rôle de l'exposition des services est très importante dans le paradigme de la composition de services, nous introduisons ensuite trois différentes stratégies d'exposition de services pour satisfaire des exigences de service des domaines différents. Nous proposons tout d'abord une plateforme centralisée d'exposition de services pour une variété de services, y compris des services de Telecom / Web / appareil et des services générés par les utilisateurs. Cette plateforme vise principalement à renforcer la fonctionnalité de centré sur l'utilisateur et la fonctionnalité de convergence, ainsi que la fourniture d'accès unifié à différents services, permettant ainsi une interaction sans faille entre les services hétérogènes et de faciliter la réutilisation de ces services existants à la fois par professionnel et non professionnel utilisateurs. Par la suite, considérant les challenges généralement rencontrés par des solutions centralisées telles que l'accès unique, coût d'entretien élevé, et facile à conduire à goulot d'étranglement de performance, nous analysons la faisabilité de l'utilisation de P2P décentralisé pour soutenir l'exposition de service distribué à grande échelle. Puis, deux modèles de partage de information de services basés sur P2P sont conçus pour compléter le modèle centralisé d'exposition des services: i) Un modèle hiérarchique basé sur le P2P, qui réutilise Chord pour garantir l'efficacité de découverte de services, en même temps adopte le concept de la publication et la découverte des services abstrait pour permettre à la recherche de service ambiguë qui est indigent dans les systèmes P2P structurés actuels. ii) Un modèle basé sur P2P avec triplex strates, qui cible principalement des services offert par des appareils. Dans ce modèle, nous utilisons les passerelles de déléguer des appareils résidant en eux pour l'exposition globale de services, et utilisons une architecture basée sur triplex strates, qui comprend une couche P2P non-structuré, un couche de SONs, et une couche de dépendance de service, pour le partage de l'information de service et le découverte des services. Ces architectures introduites et les mécanismes de l'exposition de service sont analysés et évalués par des implémentations et des simulations.

References

[1] ITU, "General Overview of Next Generation Network", *ITU-T Recommendation Y. 2001*, Geneva, Switzerland, Dec. 2004

[2] E. Bertin, I. Fodil, and N. Crespi, "A Business View for NGN Service Usage", in *Proc. 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks (BcN'07)*, pp.1-5, Munich, Germany, May 21-21, 2007

[3] E. Mikoczy, I. Kotuliak, and O.V. Daventer, "Evolution of the Converged NGN Service Platforms towards Future Networks", *Future Internet Journal*, vol. 3, no. 1, pp. 67-86, Mar. 2011

[4] J.L. Yoon, "Telco 2.0: A New Role and Business Model", *IEEE Communication Magazine*, vol. 45, no. 1, pp. 10-12, Jan. 2007

[5] E. Bertin, N. Crespi and M.L. Hostis, "A Few Myths About Telco and OTT Models", in *Proc. 15th International Conference on Intelligence in Next Generation Networks (ICIN'11)*, pp. 6-10, Berlin, Germany, Oct. 4-7, 2011

[6] K. Knightson, "Basic NGN Architecture Principles and Issues", *ITU-T Workshop on NGN*, Geneva, May 1-2, 2005

[7] 3GPP TS 23.228, "IP Multimedia Subsystem (IMS) – Stage 2", available at http://www.3gpp.org/ftp/Specs/html-info/23228.htm, accessed on Feb. 20th, 2012

[8] Cable Television Laboratories, "PacketCable Architecture Framework Technical Report", PKT-TR-ARCH-FRM-V03-070925, available at http://www.cablelabs.com/specifications/PKT-TR-ARCH-FRM-V03-070925.pdf, accessed on Feb. 20th, 2012

[9] IETF, "SIP: Session Initiation Protocol", IETF RFC 3261, available at http://www.ietf.org/rfc/rfc3261.txt, accessed on Feb. 20th, 2012

[10] 3GPP, "Service and Service Capability", 3GPP TS 22.105, available at http://www.3gpp.org/ftp/Specs/html-info/22105.htm, accessed on Feb. 20th, 2012

[11] 3GPP, "Presence Service Using the IP Multimedia (IM) Core Network (CN) Subsystem; Stage 3", 3GPP TS 24.141, available at http://www.3gpp.org/ftp/Specs/html-info/24141.htm, accessed on Feb. 20th, 2012.

[12] 3GPP, "Conferencing Using the IP Multimedia (IM) Core Network (CN) Subsystem", TS 24.147, available at http://www.3gpp.org/ftp/Specs/html-info/24147.htm, accessed on Feb. 20th, 2012

[13] 3GPP, "Messaging Using the IP Multimedia (IM) Core Network (CN) Subsystem", TS 24.247, available at http://www.3gpp2.org/public_html/specs/X.S0029-0_v1.0_070515.pdf, accessed on Feb. 20th, 2012.

[14] Global Environment for Network Innovations (GENI), available at http://www.geni.net, accessed on Feb. 20th, 2012.

[15] Future Internet Research & Experimentation, available at http://cordis.europa.eu/fp7/ict/fire/, accessed on Feb. 20th, 2012.

[16] Architecture Design Project for New Generation Network (AKARI), available at http://akari-project.nict.go.jp/eng/index2.htm, accessed on Feb. 20th, 2012

[17] European Future Internet Assembly, available at http://cordis.europa.eu/fp7/ict/ssai/ future-internet-assembly_en.html, accessed on Feb. 20th, 2012

[18] European Future Internet Portal, available at http://www.future-internet.eu/, accessed on Feb. 20th, 2012.

[19] EC Portal — Future of the Internet, available at http://ec.europa.eu/information_society/ activities/foi/index_en.htm, accessed on Feb. 20th, 2012.

[20] J. Roberts, "The Clean Slate Approach to Future Internet Design: A Survey of Research Initiatives", *Annals of Telecommunications*, vol. 64, no. 5, pp. 271-276, Jun. 2009.

[21] The FP4 4WARD project, available at http://www.4ward-project.eu/, accessed on Feb. 20th, 2012.

[22] E. Mikoczy, I. Kotuliak, and O.V. Deventer, "Evolution of the Converged NGN Service Platforms Towards Future Networks", *Future Internet Journal*, vol. 3, no.1, pp. 67-86, Mar. 2011.

[23] ITU-T Focus Group on Future Networks, "Future Networks: Vision, Concept, and Requirements", Draft Deliverable, Geneva, Switzerland, Apr. 2010.

[24] J.C. Crimi, "Next Generation Network (NGN) Services", a Telcordia Technologies White Paper, available at http://www.mobilein.com/NGN_Svcs_WP.pdf, accessed on Feb. 20th, 2012.

[25] FP8 Expert Group, "Services in the Future Internet", Workshop Report, Brussels, Feb. 28th, 2011

[26] T. Erl, "SOA Principles – An Introduction to the Service-Orientation Paradigm", available at http://soaprinciples.com/, accessed on Feb. 20th, 2012

[27] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. dissertation, Univ. of California, 2000

[28] E. Silva, L.F. Pires, and F.V. Sinderen, "An Algorithm for Automatic Service Composition", in *Proc. 1st International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing, (ICSOFT'07)*, pp. 65-74, Barclona, Spain, Jul. 22nd, 2007

[29] A. Maaradji, H, Hacid, J. Daigremont, and N. Crespu, "Social Composer: A Social-Aware Mashup Creation Environment", in *Proc. ACM Conference on Computer Supported Cooperative Work (CSCW'10)*, pp. 549-550, Savannah, USA, Feb. 6-10, 2010.

[30] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan, "Adaptive and Dynamic Service Composition in eFlow", in *Proc. 12th International Conference on Advanced Information System Engineering (CAiSE'00)*, pp.13-31, London, UK, Mar. 2000

[31] W3C, "Widgets 1.0 Requirements", Draft Feb. 2007, available at http://www.w3.org/TR/2007/WD-widgets-reqs-20070209/, accessed on Feb. 20th, 2012

[32] EzWeb Widget Wiring, available at http://ezweb.tid.es/ezweb/videos/wiring/wiring.htm, accessed on Feb. 20th, 2012

[33] N. Laga, E. Bertin, and N. Crespi, "Building a User Friendly Service Dashboard: Automatic and Non-intrusive Chaining Widgets", in *Proc. 2009 World Conference on Services (SERVICES'09)*, pp. 484-491, Los Angeles, CA, Jul. 6-10, 2009

[34] OMA, "OMA Service Environment", Approved version 1.0.4, OMA-AD-Service-Environment-V1_0_4-20070201-A, Feb. 2007

[35] 3GPP, "Network Architecture-Release 11", 3GPP TS 23.002, available at http://www.3gpp.org/ftp/Specs/html-info/23002.htm, accessed on Feb. 20th, 2012

[36] C. Buliton, "An introduction to the Service Capability Interaction Manager (SCIM)", *Avaya* White Paper, Sept. 2007, available at https://udn.devconnectprogram.com/products/whitepapers/finalwp_scim-092007-lb3700.pdf, accessed on Feb. 20th, 2012

[37] T. Zoller and F. Deza, "SCIM for IMS and Legacy Networks", available at http://convergence1.netcentrex.net/index.php?option=com_content&task=view&id=186&Itemid=20, accessed on Feb. 20th, 2012

[38] A. Gouya, N. Crespi, and E. Bertin, "SCIM (Service Capability Interaction Manager) Implementation Issues in IMS Service Architecture", in *Proc. IEEE International Conference on Communications (ICC'06)*, pp. 1748-1753, Istanbul, Turkey, Jun. 11-15, 2006.

[39] A. Maaradji, C. Huang, and N. Crespi, "Towards Personalized Services Composition on IMS: A Basic Approach", in *Proc. International Conference on Advanced Infocomm Technology (ICAIT'09)*, Xi'an, China, Jul. 7-9, 2009

[40] C. Huang and N. Crespi, "Extended Policies for Automatic Service Composition in IMS", In *Proc. Asia-Pacific Services Computing Conference (APSCC'09)*, pp.254-259, Singapore, Dec.7-11, 2009

[41] T. Dinsing, G.AP. Eriksson, I. Fikouras, K. Gronowski, R. Levenshteyn, P. Pettersson and P. Wiss, "Service Composition in IMS Using Java EE SIP Servlet Containers", Ericsson Review, no. 3, pp 92-96, 2007

[42] W3C, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", W3C Recommendation, Jun. 26, 2007, available at http://www.W3.org/TR/wsdl20/, accessed on Feb. 20th, 2012

[43] OASIS, "Organization for the Advancement of Structured Information Standards", available at http://www.oasis-open.org/home/index.php, accessed on Feb. 20th, 2012

[44] X. Dong, A.Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity Search for Web Services", In *Proc. 30th International Conference on Very Large Data Bases (VLDB'04)*, pp.372-383, Toronto, Canada, Aug. 29 – Sept. 3, 2004

[45] A. Sajjanhar, J. Hou, and Y. Zhang, "Algorithm for Web Services Matching", in *Proc. 6th Asia-Pacific Web Conference (APWeb'04)*, vol. 3007, pp.665-670, Hangzhou, China, Apr. 14-17, 2004

[46] W3C, "W3C Semantic Web Activity", available at http://www.w3.org/2001/sw/, accessed on Feb. 20th, 2012

[47] G. Spanoudakis, G. Mahbub and K. Zisman, "A Platform for Context Aware Runtime Web Service Discovery", in *Proc. IEEE International Conference on Web Services (ICWS'07)*, pp. 233-240, Salt Lake City, UT, Jul. 9-13, 2007

[48] T. Yu, Y. Zhang, and K. Lin, "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints", *ACM Transaction on the Web (TWEB)*, vol. 1, no.1, May 2007.

[49] C. Rolland, R.S. Kaabi, and N. Kraiem, "On ISOA: Intentional Services Oriented Architecture", in *Proc. 19th International Conference on Advanced Information Systems Engineering (CAiSE'07)*, pp. 158-172, Trondheim, Norway, Jun. 11-15, 2007

[50] P. Wang, K. Chao, C. Lo, R. Farmer, and P. Kuo, "A Reputation-Based Service Selection Scheme", in *Proc. IEEE international Conference on E-Business Engineering (ICEBE'09)*, pp. 501-506, Washington, USA, Oct. 21-23, 2009.

[51] ETSI, OSA Parlay X 3.0 Specifications, available at http://portal.etsi.org/docbox/TISPAN/Open/OSA/ParlayX30.html, accessed on Feb 20th, 2009

[52] OMA, "Enabler Release Definition for RESTful Bindings for Parlay X Web Services", Draft Version 2.0, OMA-ERELD-ParlayREST-V2_0-20101129-D, Nov. 29th, 2010, available at http://member.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/, accessed on Feb. 20th, 2012

[53] OneAPI, available at http://www.gsmworld.com/oneapi/, accessed on Feb. 20th, 2012

[54] J. Aitken, "Exploiting OneAPI", available at http://oneapi.gsmworld.com/wp-content/uploads/2011/09/ExploitingOneAPI1.0.pdf, accessed on Feb. 20th, 2012

[55] WAC, available at http://www.wacapps.net/web/jil;jsessionid=738475C407832A48DF22B6893511CBBD.jilweb1, accessed on Feb. 20th, 2012

[56] D. Moro, D. Lozano, and M. Macias, "WIMS 2.0: Enabling Telecom Networks Assets in Future Internet of Services", in *Proc. 1st European Conference on Towards a Service-based Internet (ServiceWave'08)*, pp.74-85, Madrid, Spain, Dec. 11-13, 2008

[57] K. Lee, "Monetizing Networks with Service Exposure", an Oracle White Paper, Nov. 2009, available http://www.oracle.com/us/industries/communications/045473.pdf, accessed on Feb. 20th, 2012

[58] N. Blum, T. Magedanz, and F. Schreiner, "Service, Enablers and Architectures: Definition of a Connected Web 2.0 Telco Service Broker to Enable New Flexible Service Exposure Models", in *Proc. International Conference on Intelligence in Next Generation Networks (ICIN'08)*, Bordeaux, France, Oct. 20-23, 2008.

[59] British Telecom, Web 21c SDK, available at http://web21c.bt.com/, accessed on Feb. 20th, 2012

[60] Orange, Orange Partner program, available at http://www.orangepartner.com/, accessed on Feb. 20th, 2012

[61] Deutsche Telekom, Developer Garden, available at http://www.developergarden.com/, accessed on Feb. 20th, 2012

[62] Telefonica, BlueVia, available at https://bluevia.com/en/, accessed on Feb. 20th, 2012

[63] Vodafone, Betavine, available at http://www.betavine.net/bvportal/resources/api/betavine, accessed on Feb. 20th, 2012

[64] J. Wong and J.I. Hong, "Making Mashups with Marmite: Towards End-user Programming for the Web", in *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*, pp. 1435-1444, San Jose, USA, Apr. 28- May 3, 2007

[65] U. Keller, R. Lara, H. Lausen, A. Polleres, L. Predoiu, and I. Toma, "Semantic Web Service Discovery", WSMX Working Draft, Oct. 2005, available at http://www.wsmo.org/TR/d10/v0.2/d10.pdf, accessed on Feb. 20th, 2012

[66] R. Karimpour and F. Taghiyareh, "Conceptual Discovery of Web Services Using WordNet", in *Proc. IEEE Asia-Pacific Service Computing Conference (APSCC'09)*, pp. 440-444, Singapore, Dec. 7-11, 2009

[67] X. Zhang, Y. Yin, M. Zhang, and B. Zhang, "A Composite Web services Discovery Technique Based on Community Mining", in *Proc. IEEE Asia-Pacific Service Computing Conference (APSCC'09)*, pp.445-450, Singapore, Dec. 7-11, 2009

[68] S. Amit, "Semantic scales up: beyond search in Web 3.0", *IEEE Internet Computing Journal*, vol. 15, no. 6, pp. 3-6, Nov. 2011

[69] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services", *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp.223-235, Jul. 2010

[70] DiYSE project, available at http://dyse.org:8080/pages/viewrecentblogposts.action?key=hometest, accessed on Feb. 20th, 2012

[71] I. Clarke, O. Sandberg, B. Wiley and T.W. Hong, "Freenet: A distributed Anonymous Information Storage and Retrieval System", in *Proc. International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pp. 46-66, Berkeley, USA, Jul. 25-26, 2000

[72] Gnutella, available at http://www.gnutella.com, accessed on Feb. 20th, 2012

[73] X. Li and J. Wu, "Searching Techniques in Peer-to-Peer Networks", *Handbook on Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks*, New York: Auerbach, 2005

[74] I. Stoica, R. Morris, D. Liben-Nowell, D.Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Application", *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17-32, Feb. 2003

[75] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shender, "A Scalable Content-addressable Network", in *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pp. 161-172, San Diego, USA, Aug. 27-31, 2001.

[76] A. Rowstronl and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems", in *Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01)*, pp.329-350, Heidelberg, Germany, Nov. 12-16, 2001

[77] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric", in *Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, pp. 53-65, Cambridge, USA, Mar. 7-8, 2002

[78] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz, "Tapestry: A Resilient Global-Scale Overlay for Service Deployment", *IEEE Journal on Selected Areas in Communications*, vol. 22, no.1, pp. 41-53, Jan. 2004

[79] D. Karger, E. Lehman, T. Leighton, T. Leighton, R. Panigraphy, M. Levine, and D. Lewin, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web", *in Proc. 29th Annual ACM Symposium on Theory of Computing (STOC'97)*, pp. 654-663, El Paso, USA, May 4-6, 1997.

[80] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "HyperCup – Hypercube, Ontologies and Efficient Search on Peer-to-peer Network", in *Proc. 1st International Conference on Agents and Peer-to-peer Computing (AP2PC'02)*, pp.112-124, Bologna, Italy, Jul. 2002

[81] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services", *Information Technology and Management Journal*, vol. 6, no.1, pp, 17-39, Jan. 2005

[82] F.B. Kashani, C. Chen, and C. Shahabi, "WSPDS: Web Services Peer-to-Peer Discovery Service", in *Proc. 5th International Conference on Internet Computing (IC'04)*, pp.733-743, Las Vegas, USA, Jun. 21-24, 2004

[83] G.D. Modica, "Resource and Service Discovery in SOAs: A P2P Oriented Semantic Approach", *International Journal of Applied Mathematics and Computer Science*, vol. 21, no. 2, pp. 285-294, Jun. 2011

[84] R. Li, Z. Zhang, Z. Wang, W. Song, and Z. Lu, "WebPeer: A P2P-based System fpr Publishing and Discovering Web Services", in *Proc. IEEE International Conference on Service Computing (SCC'05)*, pp. 149-156, Orlando, USA, Jul. 11-15, 2005

[85] M.P. Papazoglou, B.J. Krämer, and J. Yang, "Leveraging Web Services and Peer-to-Peer Network", in *Proc. 15th International Conference on Advanced Information Systems Engineering (CAiSE'03)*, pp 485-501, Klagenfurt/Velden, Austria, Jun. 16-20, 2003

[86] O.D. Sahin, C.E. Gerede, D. Agrawal, A.E. Abbadi, O.H. Ibarra, and J. Su, "SPiDeR: P2P-based Web Service Discovery", in *Proc. 3rd International Conference on Service Oriented Computing (ICSOC'05)*, pp 157-169, Amsterdam, Holland, Dec.12-15, 2005

[87] Q. He, J. Yan, Y. Yang, R. Kowalczyk, and H. Jin, "Chord4S: A P2P-based Decentralised Service Discovery Approach", in *Proc. IEEE International Conference on Service Computing (SCC'08)*, pp 221-228, Honolulu, USA, Jul. 7-11, 2008

[88] Y. Ni, H. Si, W. Li and Z. Chen, "PDUS: P2P-base Distributed UDDI Service Discovery Approach", in *Proc. International Conference on Service Sciences (ICSS'10)*, pp3-8, Hangzhou, China, May 13-14, 2010

[89] Y. Zhang, L. Liu, D. Li, F. Liu, and X. Lu, "DHT-based Range Query Processing for Web Service Discovery", in *Proc. IEEE International Conference on Web Services (ICWS'09)*, pp. 477-484, Los Angeles, USA, Jul. 6-10, 2009.

[90] G. Zhou, J. Yu, R. Chen, and H. Zhang, "Scalable Web Service Discovery on P2P Overlay Network", in *Proc. IEEE International Conference on Services Computing (SCC'07)*, p122-129, Salt Lake, USA, Jul. 9-13, 2007

[91] A. Crespo and H.G. Molina, "Semantic Overlay Networks for P2P Systems", Standord Univ., Dept. of Computer Science, Technical Report, Oct. 2002.

[92] G. Pirro, P. Missier, P. Trunfio, and D. Talia, "ERGOT: Combining DHTs and SONs for Semantic-based Service Discovery on the Grid", in *Proc. 9th International Conference on Intelligent Systems Design and Applications (ISDA'09)*, pp. 902-907, Pisa, Italy, Nov. 30 – Dec. 2, 2009

[93] V.L. Hung, H. Manfred, and A, Karl, "Towards P2P-based Semantic Web Service Discovery with QoS Support", in *Proc. 3rd International Conference on Business Process Management (BPM'05)*, pp. 18-31, Nancy, France, Sept. 5-7, 2005.

[94] E. Fatih, O.D. Sahin, A. Divyakant, and E.A. Amr, “A Peer-to-peer Framework for Web Service Discovery with Ranking”, in *Proc. IEEE International Conference on Web Services (ICWS’04)*, pp. 192-199, San Diego, USA, Jul. 6-9, 2004

[95] R. Beverly and M. Afergan, “Machine Learning for Efficient Neighbor Selection in Unstructured P2P Networks”, in *Proc. Second Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML’07)*, Cambridge, MA, Apr. 10, 2007

[96] H. Liu, A. Abraham, and Y. Badr, “Neighbor Selection in Peer-to-Peer Overlay Network: A Swarm Intelligence Approach”, in *Pervasive Computing, Computer Communications and Networks*, pp 405-431, 2010

[97] I. Stocia, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Faashoek, F. Dabek, and H. Balakrishnan, “Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications”, *IEEE/ACM Transaction on Networking*, pp. 17-32, vol. 11, Feb. 2003

Abbreviations

3GPP	3rd Generation Partnership Project
API	Application Programming Interface
AR	Application Router
AS	Application Server
BPEL	Business Process Execution Language
B2BUA	Back-to-Back User Agent
CORBA	Common Object Requestor Broker Architecture
DHT	Distributed Hash Table
ETSI	European Telecommunications Standards Institute
FGI	Future Generation Internet
FI	Future Internet
FN	Future Networks
GSM	Global System for Mobile Communications
GSMA	GSM Association
GPRS	General Packet Radio Service
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Mark-up Language
HSS	Home Subscriber Server
I-CSCF	Interrogating-Call Session Control Function
IDE	Integrated Development Environment
iFC	Initial Filter Criteria
IETF	Internet Engineering Task Force
IR	Information Retrieval

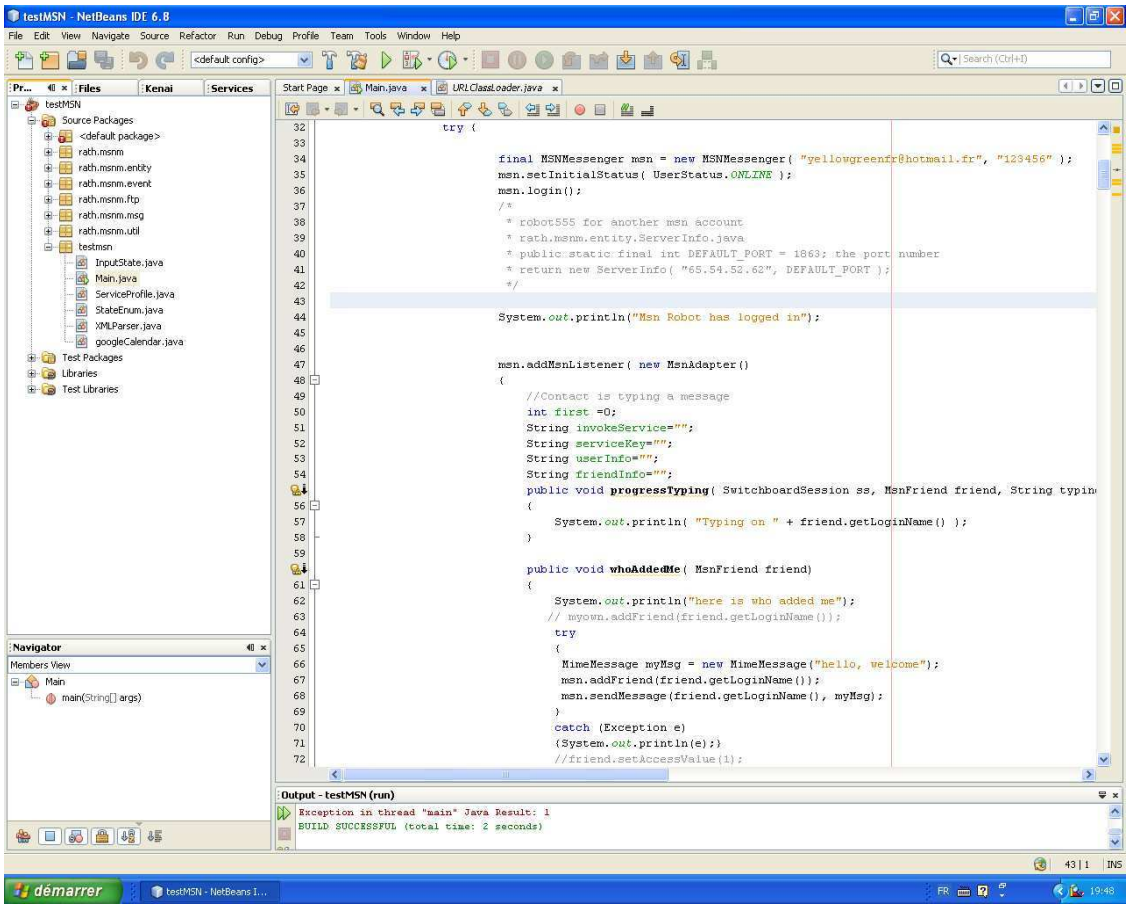
IoT	Internet of Things
IP	Internet Protocol
IMS	IP Multimedia Subsystem
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
jABC	Java Application Building Center
JBPM	Jave Business Process Management
JPDL	JBPM Process Definition Language
LTE	Long Term Evolution
OASIS	Organization for the Advancement of Structured Information Standards
OSA	Open Service Architecture
OSE	OMA Service Environment
OMA	Open Mobile Alliance
OWL-S	Semantic Markup for Web Services
OWL	Web Ontology Language
P-CSCF	Proxy-Call Session Control Function
PoC	Push-to-talk Over Cellular
PSE	Personal Service Environment
PnP	Plug and Play
QoE	Quality of Experience
QoS	Quality of Service
RDF	Resource Description Framework
REST	Representational State Transfer
ROA	Resource Oriented Architecture
RPC	Remote Procedure Call

SAWSDL	Semantic Annotations for WSDL and XML Schema
SAWSD	Semantic Annotation for WSDL
SAE	System Architecture Evolution
SCE	Service Composition Engine
SCIM	Service Capability Interaction Management
S-CSCF	Serving-Call Session Control Function
SDH	Synchronous Digital Hierarchy
SE	Semantic Engine
SIP	Session Initiation Protocol
SLA	Service Level Agreements
SLG	Service Logic Graphs
SOA	Service Oriented Architecture
SoAT	state of the art
SOAP	Simple Object Access Protocol
SON	Semantic Overlay Network
SS7	Signalling System #7
SMS	Short Message Service
TISPAN	Telecoms & Internet Converged Services & Protocols for Advanced Networks
TSEG	Telecom Service Exposure Gateway
TTL	Time-to- Live
M2M	Machine-to-Machine
MMS	Multimedia Messaging Service
MVNO	Mobile Virtual Network Operator
NGN	Next Generation Network
NGI	Next Generation Internet

NGSI	Next Generation Service Interfaces
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over IP
WAC	Wholesale Application Community
WADL	Web Application Description Language
WSDL	Web Service Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WOA	Widget Oriented Architecture
WLAN	Wireless Local Area Network
WMS	Web Multimedia Subsystem
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XML-RPC	Extensible Markup Language – Remote Procedure Call

Appendix

1. MSN Robot Likes Service Invocation and Creation Environment:




```
public static void main(String[] args) {
        try {
        final MSNMessenger msn = new MSNMessenger( "yellowgreenfr@hotmail.fr", "123456" );
         msn.setInitialStatus( UserStatus.ONLINE );
         msn.login();
        System.out.println("Msn Robot has logged in");
        msn.addMsnListener( new MsnAdapter()
        {
                //Contact is typing a message
                int first =0;
                String invokeService="";
```

```
String serviceKey="";
String userInfo="";
String friendInfo="";
public void progressTyping( SwitchboardSession ss, MsnFriend friend, String
typingUser )
{
System.out.println( "Typing on " + friend.getLoginName() );
}
        public void whoAddedMe( MsnFriend friend)
        {
            System.out.println("here is who added me");
            try
            {
             MimeMessage myMsg = new MimeMessage("hello, welcome");
             msn.addFriend(friend.getLoginName());
             msn.sendMessage(friend.getLoginName(), myMsg);
            }
            catch (Exception e)
            {System.out.println(e);}
            //friend.setAccessValue(1);
        }
public void instantMessageReceived( SwitchboardSession ss, MsnFriend friend,
MimeMessage mime ){
            System.out.println( ">>>>Message from " + friend.getLoginName() );
            String Msg_Rec = mime.getMessage();
            System.out.println( Msg_Rec );
            System.out.println( "******************************" );
            if (first==0){
            MimeMessage myMSG = new MimeMessage("Dear Sir, do you want to
            execute an atomic service? (Yes/No)");
                    try{
                        ss.sendInstantMessage(myMSG);
                        first = 1;}
```

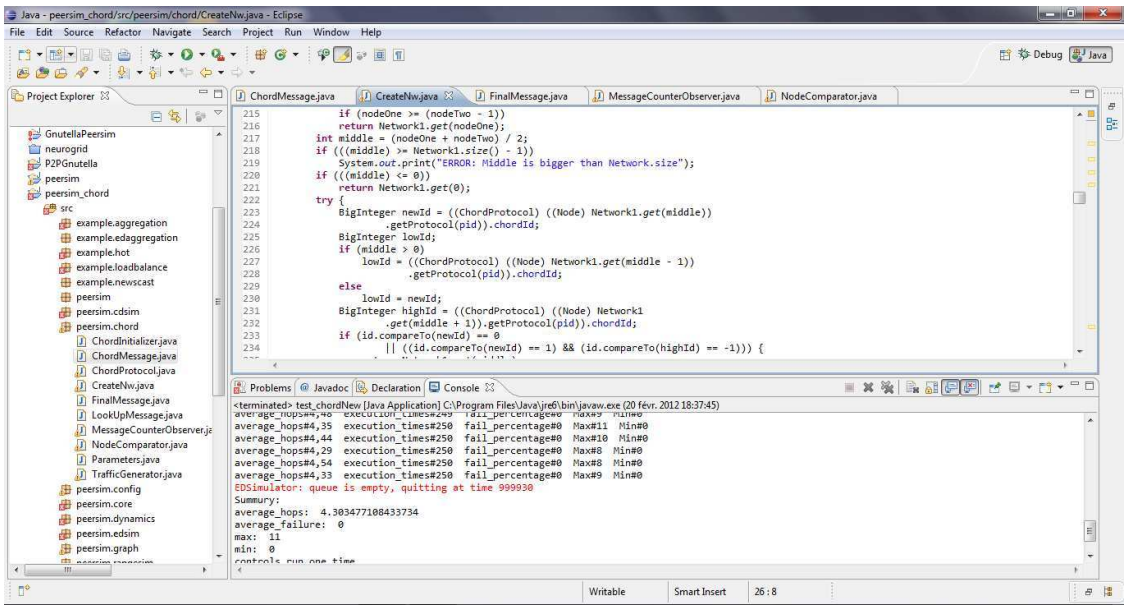
```
        catch (Exception e)
        {
            e.printStackTrace();}
 }
else if
(first==1)&(Msg_Rec.equals("Yes")||Msg_Rec.equals("yes")||Msg_Rec.equals
("Modify")||Msg_Rec.equals("modify"))){
    MimeMessage myMSG = new MimeMessage("Please select the atomic
    services from below list " +"under the following format ('Execute:name of
    service' or 'E:name of service'):\n" +"( 1 ) Weather\n( 2 ) Map\n( 3 )
    Message\n( 4 ) Translation\n( 5 ) Calendar\n( 6 ) News");
        try {
            ss.sendInstantMessage(myMSG);
            }
        catch (Exception e){
            e.printStackTrace();
        }
 }
else if
((Msg_Rec.startsWith("execute:")||Msg_Rec.startsWith("Execute:")||Msg_Rec
.startsWith("E:")||Msg_Rec.startsWith("e:")))
 {
serviceKey = "";
userInfo="";
friendInfo="";
System.out.println(Msg_Rec);
Msg_Rec=Msg_Rec.replaceAll(" ", "");
System.out.println(Msg_Rec);
StringTokenizer ms = new StringTokenizer(Msg_Rec, ":");
int mspart;
mspart = ms.countTokens(); // include the place for "execute"
 string[] servicesList= new String[mspart-1];
```

```
string mytoken = ms.nextToken(); //get "create" info
for (int i=0; i<servicesList.length;i++) // add the selected services into a list,
the first place servicelist[0]is for "create"
    {
      mytoken = ms.nextToken();
      if (mytoken.equals("weather")||mytoken.equals("Weather")){
         servicesList[i]="Weather";
      }
      else if (mytoken.equals("map")||mytoken.equals("Map")){
         servicesList[i]="Map";
      }
      else if (mytoken.equals("message")||mytoken.equals("Message")){
         servicesList[i]="Message";
      }
      else if (mytoken.equals("translation")||mytoken.equals("Translation")){
         servicesList[i]="Translation";
      }
      else if (mytoken.equals("calendar")||mytoken.equals("Calendar")){
         servicesList[i]="Calendar";
      }
      else if (mytoken.equals("news")||mytoken.equals("News")){
         servicesList[i]="News";
      }
 else if
(mytoken.equals("news+translation")||mytoken.equals("News+translation")){
         servicesList[i]="News+Transaltion";
      }
else if
(mytoken.equals("weather+translation")||mytoken.equals("Weather+translation
")){
         servicesList[i]="Weather+Transaltion";
      }
    }
```

```
    invokeService=servicesList[0];
    String confirmMSG = "";
    for (int i=0; i<servicesList.length;i++)
    {
        //System.out.println(servicesList[i]+"***");
        if (i<(servicesList.length-1))
        confirmMSG = confirmMSG+servicesList[i]+", ";
        if (i==(servicesList.length-1))
        confirmMSG = confirmMSG+servicesList[i]+".";
    }
confirmMSG = "Do you want to create a service contains following
functionalities: " +confirmMSG + "(Confirm/Modify/Cancel)";
MimeMessage myMSG = new MimeMessage(confirmMSG);
        try
        {
            ss.sendInstantMessage(myMSG);
            first=2;
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
 }
else if
((((first==1)||(first==2)))&(!(Msg_Rec.equals("Yes")||Msg_Rec.equals("yes")||
Msg_Rec.startsWith("Create,")||Msg_Rec.equals("create,")||
Msg_Rec.equals("Confirm")||Msg_Rec.equals("confirm")||Msg_Rec.equals("
Modify")||Msg_Rec.equals("modify")))){
MimeMessage myMSG = new MimeMessage("What do you want to do? " +
            "If you want to create a service, please type 'Yes';\n " +
             "If you want to teach me, please enter the content under following
             format: 'study: question content + answer content;\n'" +
            "Otherwise please contact my developper: hhh@hotmail.com");
```

```
        try
        {
            ss.sendInstantMessage(myMSG);
            first=1;
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
```

2. Simulation for hierarchical P2P based service exposure model:



PEERSIM CHORD

random.seed 1234567890

simulation.endtime 10^6

simulation.logtime 10^6

simulation.experiments 1

Network.size 1000

Network1.ratio 1

Network2.ratio 1

Network3.ratio 1

Network4.ratio 1

Network1.size 0

Network2.size 0

Network3.size 0

Network4.size 0

protocol.tr UniformRandomTransport { mindelay 0

```
                                        maxdelay 10 }
protocol.my ChordProtocol               { transport tr }
control.traffic TrafficGenerator        { protocol my
                                        step 100 }
init.create CreateNw                    { protocol my
                                        idLength 32
                                        succListSize 12 }
control.observer MessageCounterObserver { protocol my
                                        step 25000 }
control.dnet DynamicNetwork             { add1 0
                                        add2 0
                                        add3 0
                                        add4 0
                                        minsize1 0.5
                                        maxsize1 1.5
                                        minsize2 0.5
                                        maxsize2 1.5
                                        minsize3 0.5
                                        maxsize3 1.5
                                        minsize4 0.5
                                        maxsize4 1.5
                                        step 100000
        init.0 ChordInitializer         { protocol my }
        }
public void processEvent(Node node, int pid, Object event) {
```

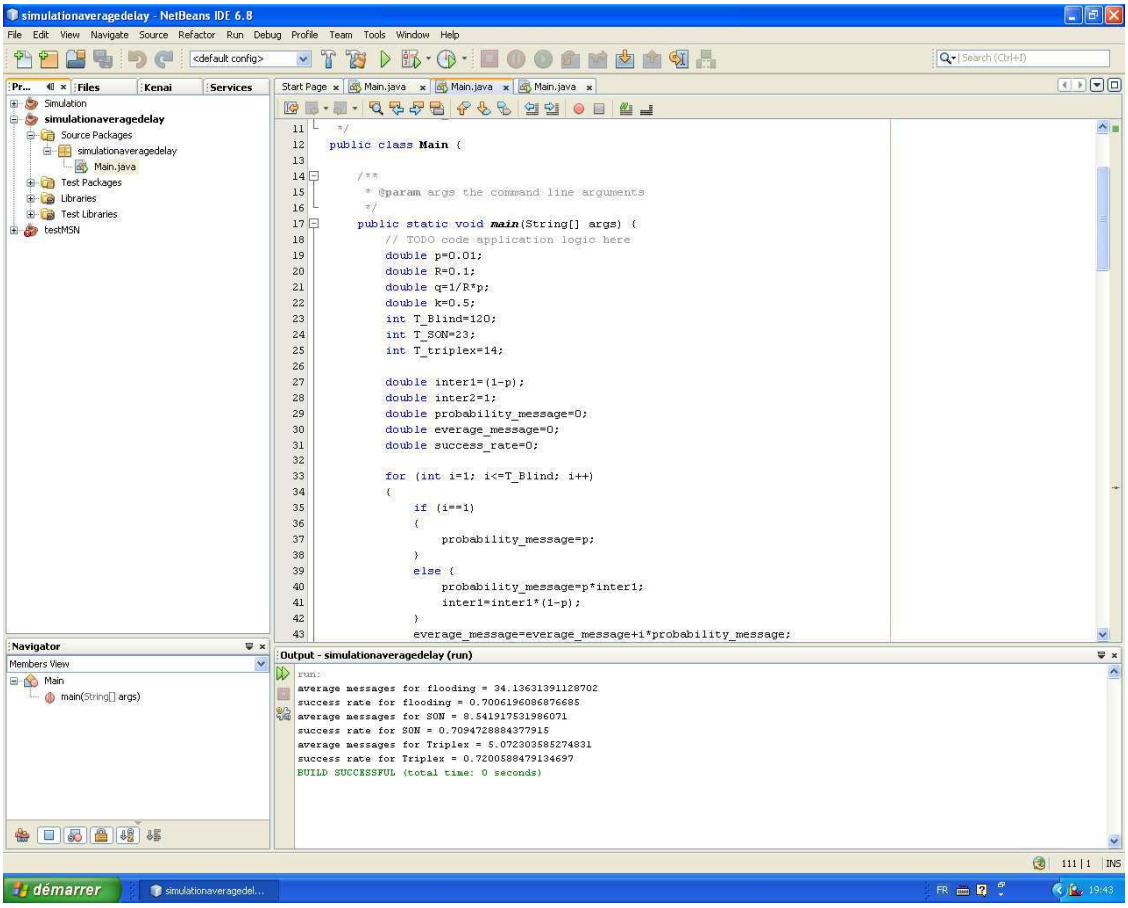
```
p.pid = pid;
currentNode = node.getIndex();
if (event.getClass() == LookUpMessage.class) {
        LookUpMessage message = (LookUpMessage) event;
        message.increaseHopCounter();
        BigInteger target = message.getTarget();
        Transport t = (Transport) node.getProtocol(p.tid);
        Node n = message.getSender();
        if (target == ((ChordProtocol) node.getProtocol(pid)).chordId) {
                t.send(node, n, new FinalMessage(message.getHopCounter()), pid);}
        if (target != ((ChordProtocol) node.getProtocol(pid)).chordId) {
                if(message.getTargetNetid()==this.getNetid()){
                        switch(message.getTargetNetid()){
                        case 1:{  Node dest = find_successor1(target);
                                if (dest.isUp() == false) {
                                        do {      varSuccList1 = 0;
                                                  stabilize1(node);
                                                  stabilizations++;
                                                  fixFingers1();
                                                  dest = find_successor1(target);
                                        } while (dest.isUp() == false);}
                                if (dest.getID() == successorList1[0].getID() &&
                                (target.compareTo(((ChordProtocol)
                                dest.getProtocol(p.pid)).chordId) < 0)) {fails++;}
                                else { t.send(message.getSender(), dest, message, pid);}
                                };
```

```
                    break;
            case 2 :{......};
            case 3 :{......};
            case 4 :{......};
        else {  switch(message.getTargetNetid()){
            case 1:  t.send(message.getSender(), externalPoint1, message, pid);
                     break;
            case 2:  t.send(message.getSender(), externalPoint2, message, pid);
                     break;
            case 3:  t.send(message.getSender(), externalPoint3, message, pid);
                     break;
            case 4:  t.send(message.getSender(), externalPoint4, message, pid);
                     break;
            default:System.out.println("error:netId is not configurated
            correctlly");}}}}
    if (event.getClass() == FinalMessage.class) {
        FinalMessage message = (FinalMessage) event;
        lookupMessage = new int[index + 1];
        lookupMessage[index] = message.getHopCounter();
        index++; }}
……
public void stabilize1(Node myNode) {
    try {   Node node = ((ChordProtocol) successorList1[0].getProtocol(p.pid)).predecessor;
            if (node != null) {
                if (this.chordId == ((ChordProtocol) node.getProtocol(p.pid)).chordId) return;
                BigInteger remoteID = ((ChordProtocol) node.getProtocol(p.pid)).chordId;
```

```
                    if (idInab(remoteID, chordId, ((ChordProtocol) successorList1[0]
                    .getProtocol(p.pid)).chordId)) successorList1[0] = node;
                    ((ChordProtocol) successorList1[0].getProtocol(p.pid)).notify(myNode);}
            updateSuccessorList1();
            } catch (Exception e1) {    e1.printStackTrace();
                                        updateSuccessor1();}}
……
private void updateSuccessorList1() throws Exception {
        try {
        while (successorList1[0] == null || successorList1[0].isUp() == false) {updateSuccessor1();}
        System.arraycopy(((ChordProtocol) successorList1[0].getProtocol(p.pid)).successorList1, 0,
        successorList1, 1, succLSize - 2);
        } catch (Exception e) { e.printStackTrace();}
}
……
public void notify(Node node) throws Exception {
        BigInteger nodeId = ((ChordProtocol) node.getProtocol(p.pid)).chordId;
        if ((predecessor == null)|| (idInab(nodeId, ((ChordProtocol) this.predecessor.getProtocol(p.pid))
        .chordId, this.chordId))) {predecessor = node;}
}
……
public Node find_successor1(BigInteger id) {
        try {if (successorList1[0] == null || successorList1[0].isUp() == false) {updateSuccessor1();}
            if (idInab(id, this.chordId, ((ChordProtocol) successorList1[0].getProtocol(p.pid)).chordId)) {
                return successorList1[0];}
            else { Node tmp = closest_preceding_node1(id);
```

```
                return tmp;}
        } catch (Exception e) { e.printStackTrace();}
        return successorList1[0];
}
……
private Node closest_preceding_node1(BigInteger id) {
        for (int i = m; i > 0; i--) {
                try {if (fingerTable[i - 1] == null || fingerTable[i - 1].isUp() == false) {continue;}
                BigInteger fingerId = ((ChordProtocol) (fingerTable[i - 1].getProtocol(p.pid))).chordId;
                if ((idInab(fingerId, this.chordId, id)) || (id.compareTo(fingerId) == 0)) {
                        return fingerTable[i - 1]; }
                if (fingerId.compareTo(this.chordId) == -1) {
                        if (idInab(id, fingerId, this.chordId)) { return fingerTable[i - 1];}}
                if ((id.compareTo(fingerId) == -1)&& (id.compareTo(this.chordId) == -1)) {
                        if (i == 1) return successorList1[0];
                        BigInteger lowId = ((ChordProtocol) fingerTable[i - 2].getProtocol(p.pid)).
                        chordId;
                        if (idInab(id, lowId, fingerId)) return fingerTable[i - 2];
                        else if (fingerId.compareTo(this.chordId) == -1) continue;
                        else if (fingerId.compareTo(this.chordId) == 1) return fingerTable[i - 1];}
                } catch (Exception e) { e.printStackTrace();}}
        if (fingerTable[m - 1] == null)
                return successorList1[0];
}
```

3. Theoretical analysis for triplex layers based service exposure model in IoT service environment




```
public static void main(String[] args) {
    double p=0.01;
    double R=0.1;
    double q=1/R*p;
    double k=0.5;
    int T_Blind=120;
    int T_SON=23;
    int T_triplex=14;
    double inter1=(1-p);
```

```
double inter2=1;
double probability_message=0;
double everage_message=0;
double success_rate=0;
for (int i=1; i<=T_Blind; i++)
{   if (i==1)
    {probability_message=p;}
    else {        probability_message=p*inter1;
                  inter1=inter1*(1-p);}
    everage_message=everage_message+i*probability_message;
    success_rate=success_rate+probability_message;}
System.out.println("average messages for flooding = "+everage_message);
System.out.println("success rate for flooding = "+success_rate);
everage_message=0;
success_rate=0;
inter2=1;
double inter3=(1-R)/(1-q);
for (int i=1;i<=T_SON;i++)
{  inter2=inter2*(1-q);
    if(i==1)
    {probability_message=p;}
    else{         probability_message=(1-p)*R*inter2*q/(1-q)/(1-q);
                  for(int j=1;j<=i-1;j++)
                  {           probability_message=probability_message+(1-p)*R*inter2*q*inter3/(1-q);
                              inter3=inter3*(1-R)/(1-q);}}
```

```
    everage_message=everage_message+i*probability_message;
    success_rate=success_rate+probability_message;}
System.out.println("average messages for SON = "+everage_message);
System.out.println("success rate for SON = "+success_rate);
everage_message=0;
success_rate=0;
inter2=1;
double inter7=1;
double inter8=1;
double probability_inter=0;
for (int i=1;i<=T_triplex;i++)
{     inter2=inter2*(1-q);
      if(i==1){ probability_message=p;}
    else{         probability_message=(1-p)*R*inter2*q/((1-q)*(1-q));
                  inter8=inter2/(1-q)/(1-q);
                  inter7=1;
                  probability_inter=0;
                  for(int j=1;j<=i-1;j++){
                              probability_inter=probability_inter+(1-p)*(1-R)*k*inter8*inter7*q;
                              inter7=inter7*(1-k)/(1-q);}
                  probability_message=probability_message+probability_inter;}
                   everage_message=everage_message+i*probability_message;
                  success_rate=success_rate+probability_message;}
System.out.println("average messages for Triplex = "+everage_message);
System.out.println("success rate for Triplex = "+success_rate);
```

```
    }
}
```